

©Copyright 2026
Gian Marco Visani

Rational protein modeling and design with local and rotation-equivariant neural networks

Gian Marco Visani

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2026

Reading Committee:

Armita Nourmohammad, Chair

William Stafford Noble

David Beck

Program Authorized to Offer Degree:

Paul G. Allen School of Computer Science & Engineering

University of Washington

Abstract

Rational protein modeling and design with local and rotation-equivariant neural networks

Gian Marco Visani

Chair of the Supervisory Committee:

Armita Nourmohammad

Department of Physics

Proteins carry out their diverse biological functions through specific interactions with other molecules, mediated by their three-dimensional structure. Therefore, predicting how a protein’s amino-acid sequence gives rise to its structure and how both determine interaction specificity is central to understanding biological processes and designing therapeutics. Machine Learning (ML) has emerged as a powerful paradigm to learn these relationships from data, resulting in effective models that can predict high-level properties quickly, without needing to resolve individual details. In this thesis, I argue that building ML models that are both fast and physically meaningful is best achieved by letting first principles guide design choices. Specifically, the fact that protein function emerges from local inter-atomic interactions, both within and between proteins, motivates modeling *residue-centered local atomic environments* with *rotational equivariance* — forming a unified $SO(3)$ -equivariant neural network framework described in Chapter 2. I then apply this framework to develop three models spanning diverse aspects of protein structure and function. In Chapter 3, I introduce H-(V)AE, a rotationally equivariant (Variational) AutoEncoder that learns compact embeddings of local atomic environments, useful as general-purpose representations for downstream tasks. In Chapter 4, I present H-Packer, which predicts physically correct side-chain conformations consistent with a given backbone structure and amino-acid sequence. Finally, in Chapters 5 and 6, I introduce HERMES, a structure-based model pre-trained to predict amino-acid propensities from local atomic context. I show that HERMES can be used to predict mutational effects on protein stability and protein-protein binding affinity, highlighting validated applications to structure-based vaccine antigen design as well as T-Cell Receptor (TCR) specificity prediction and TCR-specific peptide design. This thesis demonstrates that physical locality and rotational equivariance are effective inductive biases for machine learning models of proteins, yielding a fast, principled, and broadly applicable framework for protein modeling and design.

ACKNOWLEDGEMENTS

First of all, I'd like to thank my advisor, Armita Nourmohammad, for answering my cold email at the end of 2021, giving me a chance to join the lab, and for supporting me ever since. At the time, all I knew was that Geometric Deep Learning sounded super cool, and that I wanted to work on applications of it to biology. Armita's lab was a natural place to work on this given the work that Mike was doing, but over the years I got so much more out of the lab. At first, I was too narrowly focused on the gamified idea of "success" I had created: publish in the big-3 ML conferences, beat the baselines, be recognized, and I wanted all of these quickly, because I saw other people get them and I wanted to be like them. But over time, Armita showed me the joy of working on problems simply because you find them interesting and fun, and the benefit of learning diverse techniques, and seeing connections between them. What's interesting to me is that I believe I always possessed this child-like excitement for breadth of knowledge in other areas of my life, like cooking and diving, and even in learning mathematical and engineering curiosities. But somehow on the professional level, my sense of pride and competitiveness got the better of me. I credit you, Armita, and your very diverse lab, for showing me that I can make my child-like curiosity a central part of my research drive. That being said, I am also a very engineering and planning-driven person, I think it's safe to say much more than you, but despite this difference I am grateful to have received your support in developing this side of me through mentoring younger students.

Thank you Mike Pun, for being a patient mentor to me in the first few years of my PhD. I know I was not the easiest mentee to deal with: hot-headed, ML-focused, stubborn, anxious, feisty. I owe much of this thesis to you, not only because you literally started the work I then carried on, but also because your level-headed-ness was an inspiration to me, because you taught me that what mattered in the PhD process was learning and personal development, not publishing papers.

Thank you Zach Montague, for always being there to support me and everyone else, both scientifically and emotionally, and for adding your dark humor, which I personally enjoy, to our days. While others benefited from your presence more than me on the scientific level (simply due to timing and to our lines of work being quite different) I was nonetheless grateful for your presence, and had no doubts you would achieve the success I am now seeing you achieve.

Thank you William Galvin, for being just an amazing person to work with. For being an honest and thoughtful communicator, and for being excited to learn alongside me.

Thank you Kevin Borisiak, for always being patiently available to brainstorm with me, and to teach me maths and physics notions that I was not able to easily grasp on my own.

Thank you Obinna Ukogu, for inspiring me to be a more careful scientist: grounding myself in the meaningful biological questions, and only then thinking about the tools and data I can use to solve them.

Thank you Ella Carlander, for your thoughtful help and comments on my many presentations to the group.

Thank you Jakub Otwinowski, for putting a foot in the door of industry for me, for believing in me, and for insightful conversations in the last few months.

Thank you Will DeWitt, for hosting me in your lab in the final months of my PhD, and for being a patient, thorough, extremely pleasant collaborator on our response paper adventure.

Thank you Zac Jones, for being an amazing collaborator. For always being available to answer my many questions about biology and sharing excitement with me.

Thank you Asya Minervina for believing in my work enough to take the time to test my peptide designs, and for insightful conversations.

Thank you Phil Bradley, for being always available and excited to share your knowledge and suggestions with me.

I also want to thank all the other members, past and present, of the Nourmohammad and DeWitt labs, for the help, support and camaraderie they have offered me throughout my PhD journey. Also, I want to thank everyone I have had the pleasure of engaging in scientific conversations with, who have taught me something or who have believed in my work. In this bunch, Sebastian Ols from the King Lab, and Rohith Krishna from the Baker lab stand out.

Still within the professional sphere, but outside research, I want to thank the admins of CSE for always being extremely extremely helpful and supportive. You made my stay in the department absolutely frictionless: shout out in particular to Chris Nagle, Joe Eckert, Elide Dorough.

Outside of the professional sphere, I want to thank my friends in Seattle, past and present, for enjoying the good times with me and supporting me through bad times. Thank you for coming over for dinner and eating the food I cooked, for you enabled one of my greatest passions. Thank you for sharing your passions with me and letting me enrich myself through you. In particular, thank you Artidoro Pagnoni for being a thoughtful friend, someone I know I can rely on if I am ever in need; thank you Lancelot Wathieu for being my emotional support throughout these years and someone I can truly be myself with; thank you Kevin Borisiak for both joining me in, and most importantly for inviting me in, great adventures under the seas.

Thank you to my italian friends - to TMIS - who have cherished my friendship throughout the 10 long years that I have (so far) been abroad. Thank you for treating me like I never left, for being excited to see me, for sharing who you're becoming with me. Thank you in particular to Pietro Ortolani (aka the hunter), Francesco Dragoni (aka the dragon), and Michele Magnani (aka the grey cat) for coming to Seattle for my defense: it was a dream come true.

Thank you Alison Ciling, my partner in life and crime, for being both my first supporter and most honest critic. For sacrificing yourself to support me. For loving me like you do. For nerding out with me and for joining me in adventures. For sharing your love of cuisine and cooking with me. Most of all, thank you for being the creative muse that adds spice to my life. I love you very much.

Thank you mom and dad for believing in me all these years and supporting me in whatever endeavor I set myself out to, even when it is far away from you... Thank you for using the knowledge you have of my character, to help me navigate life even with such distances and in contexts that are not familiar to you. Thank you for being extremely loving parents, always. Thank you dad for giving me many of the passions I cherish and nurture. Thank you mom for giving me many of the character traits and strengths I credit to my success.

Thank you to my grandparents Francesca, Giovanni and Giuliana, for helping raise me. Thank you Nonna Frenci for your continued love and honest approach to life. Thank you Nonno Giovanni for all your stories, and your lessons about the importance of hard work.

Thank you to my siblings, Matilde and Matteo, for being supportive of me. I know I haven't been the most present sibling. I struggle with the selfishness I feel for having lived most of my life away from my little sister, whom I have always loved and cared for immensely, sometimes even assuming a misplaced parental role, that one time Matilde promptly pointed out. I would still give my life for you, I hope you know that.

The absence from you, my family, has been the greatest price I have paid throughout my PhD and my twenties, and I know it has been a price you have paid as well. I hope to give back more in the future. I love you very much.

CONTENTS

Abstract	ii
Acknowledgements	iii
Contents	v
List of Figures	vi
List of Tables	x
Chapter I: Introduction	1
1.1 Proteins and their functions	1
1.2 Modeling proteins	7
1.3 Scope of this thesis	12
1.4 Beyond this thesis	12
Chapter II: SO(3)-Equivariant Framework	14
2.1 Mathematical background	14
2.2 Holographic Encoding of the Data	15
2.3 SO(3)-Equivariant Layers	17
2.4 Why Equivariance?	19
Chapter III: H-(V)AE: Holographic (Variational) Autoencoder	20
3.1 Methods	21
3.2 Results	22
3.3 Summary	29
Chapter IV: H-Packer: Holographic Packer	31
4.1 Methods	32
4.2 Results	35
4.3 Summary	38
Chapter V: HERMES	40
5.1 Methods	41
5.2 Results	43
5.3 Summary	57
Chapter VI: Mapping the T-cell receptor specificity landscape with HERMES	59
6.1 Model	61
6.2 Results	63
6.3 Summary	72
Chapter VII: Conclusion and Outlook	74
References	78
Appendix A: Supplement for chapter 3	100
A.1 Modeling details	100
A.2 Using a non-equivariant decoder	101
A.3 Experimental details	102
A.4 Protein-Ligand Binding Affinity Prediction	106
Appendix B: Supplement for chapter 4	135
B.1 Proof of equivalence between MSE and cosine loss in predicting Sine and Cosine of χ angles	135
B.2 Training Details	135

Appendix C: Supplement for chapter 5	144
C.1 Extended methods	144
Appendix D: Supplement for chapter 6	182
D.1 TCR-pMHC binding affinity and T-cell activity prediction	182
D.2 De-novo peptide design	183
D.3 Alternative computational methods for scoring peptides and design	188
D.4 Experimental protocol to measure peptide activities	195
D.5 Experimental activities of the designed peptides	196
D.6 T-cell and MHC specificity as the entropy of the associated peptide distribution	198
D.7 Predicting MHC-I peptide presentation preferences with HERMES.	198

LIST OF FIGURES

<i>Number</i>		<i>Page</i>
1.1	A protein’s life cycle: from DNA to function.	2
1.2	Canonical amino-acids and their chemistries.	3
1.3	Gibbs free energy landscapes of protein folding.	5
1.4	Examples of machine learning tasks on proteins.	10
1.5	Schematic overview of modeling framework and specific tasks addressed in this thesis.	13
2.1	Invariance and Equivariance to 3D rotations, explained using a water molecule.	15
2.2	Visualization of the Holographic Encoding considered in this thesis.	17
3.1	Visualization of H-(V)AE’s architecture.	23
3.2	H-VAE on MNIST-on-the-sphere.	24
3.3	Visual evidence of the disentanglement in the latent space of MNIST-on-the-sphere.	24
3.4	Structural embeddings to predict protein-ligand binding affinities with H-(V)AE.	26
4.1	Overview of H-Packer.	33
4.2	Test MAE for the simple task of predicting χ angles from atomic conformation.	35
5.1	Overview of HERMES.	42
5.2	Predicting mutational effects on thermodynamic folding stability.	44
5.3	Impact of amino acid size changes on predicting mutational effects on protein stability.	46
5.4	Uncovering mutation preferences via model-averaged substitution matrices.	47
5.5	Identifying model biases with structure-conditioned reversibility	49
5.6	Overview of structure-based vaccine design against enveloped viruses.	50
5.7	Recovering previously-identified antigen-stabilizing mutations with HERMES.	52
5.8	Identifying new antigen-stabilizing mutations with HERMES for RSV-F.	55
6.1	HERMES model for predicting and designing TCR-pMHC interactions.	60
6.2	Predicting binding affinity of TCR-pMHC complexes.	61
6.3	Predicting T-cell activity in response to peptide variants.	66
6.4	De novo peptide design to elicit a T-cell response.	67
6.5	Evaluating T-cell specificity from de novo designed peptides.	69
A.4.1	Training loss trace of H-AE, with and without Batch Norm, on MNIST-on-the-sphere.	113
A.4.2	UMAP visualization of the invariant latent space learned by selected H-(V)AE models trained on MNIST-on-the-sphere.	114
A.4.3	UMAP visualization of the invariant latent space learned by selected <i>conditional</i> H-VAE models trained on MNIST-on-the-sphere.	115
A.4.4	Correlation between Cosine loss and MSE values between pairs of random tensors.	116
A.4.5	Trade-off across regularization strength β between quality (top) and diversity (bottom) of MNIST-on-the-sphere samples generated by H-VAE with digit conditioning.	117
A.4.6	Ablation in regularization strength β on MNIST-on-the-sphere test set.	118
A.4.7	Reconstruction loss across β for both unconditional and conditional H-VAE models on MNIST-on-the-sphere.	118
A.4.8	Reconstructions of Random Test Images by the (NR/R; $z = 16$; AE) MNIST-on-the-sphere model.	119
A.4.9	Reconstructions of Random Test Images by the (NR/R; $z = 16$; VAE; $\beta = 0.6$) MNIST-on-the-sphere model.	120

A.4.10	Reconstructions of Random Test Images by the (NR/R; $z = 120$; VAE; $\beta = 2.8$) MNIST-on-the-sphere model.	121
A.4.11	Random samples generated by the (NR/R; $z = 16$; VAE; $\beta = 0.6$) MNIST-on-the-sphere model.	122
A.4.12	Random samples generated by the (NR/R; $z = 120$; VAE; $\beta = 2.8$) MNIST-on-the-sphere model.	122
A.4.13	Random samples generated by the (NR/R; $z = 16$; VAE; $\beta = 2.0$) MNIST-on-the-sphere model.	123
A.4.14	Random conditional samples generated by the (NR/R; $z = 16$; VAE; $\beta = 0.6$; conditional) MNIST-on-the-sphere model.	124
A.4.15	Random conditional samples generated by the (NR/R; $z = 120$; VAE; $\beta = 2.8$; conditional) MNIST-on-the-sphere model.	125
A.4.16	Random conditional samples generated by the (NR/R; $z = 16$; VAE; $\beta = 2.0$; conditional) MNIST-on-the-sphere model.	126
A.4.17	Trajectories across the latent space for the (NR/R; $z = 16$; AE) MNIST-on-the-sphere model.	127
A.4.18	Trajectories across the latent space for the (NR/R; $z = 16$; VAE; $\beta = 0.6$) MNIST-on-the-sphere model.	128
A.4.19	Trajectories across the latent space for the (NR/R; $z = 120$; VAE; $\beta = 2.8$) MNIST-on-the-sphere model.	129
A.4.20	H-(V)AE implicitly learns to maximally overlap training images on MNIST-on-the-sphere.	130
A.4.21	2D visualization via UMAP of the invariant latent embeddings of Shrec17 test data learned by H-(V)AE.	130
A.4.22	Amino acid latent space learned by H-AE.	131
A.4.23	Amino acid latent space learned by H-VAE.	132
A.4.24	Reconstruction loss as a function of feature degree ℓ	132
A.4.25	2D UMAP visualization of the 128-dimensional invariant latent space learned by H-AE trained on the protein structure neighborhoods with $L = 6$, annotated by secondary structure.	133
A.4.26	2D UMAP visualization of the 128-dimensional invariant latent space learned by H-AE trained on the protein structure neighborhoods with $L = 6$, annotated by number of atoms.	134
B.2.1	Null Reconstruction Error on Test data by using data-derived values as redundant internal coordinates.	136
B.2.2	Difference in Null Reconstruction Error on Test data by using a smaller set of structures for reference.	136
B.2.3	χ angles error trace during training for the simple task of predicting χ angles from true amino-acid conformations.	137
B.2.4	Test MAE for the simple task of predicting χ angles from atomic conformation, with loss weighted by frequency.	137
B.2.5	Angle Error distribution on CASP13 for H-Packer ₅ ^{$\ell_{\max}=5$} , part 1.	140
B.2.6	Angle Error distribution on CASP13 for H-Packer ₅ ^{$\ell_{\max}=5$} , part 2.	141
B.2.7	Angle Error distribution on CASP14 for H-Packer ₅ ^{$\ell_{\max}=5$} , part 1.	142
B.2.8	Angle Error distribution on CASP14 for H-Packer ₅ ^{$\ell_{\max}=5$} , part 2.	143
C.1.1	Comparison of zero-shot model predictions on the Megascale test set.	160
C.1.2	Predicting mutational effects on thermodynamic folding stability.	161
C.1.3	Statistical significance of model performance differences on stabilizing mutation identification.	162

C.1.4	Statistical significance of within-model performance differences in identifying stabilizing mutations across mutational size classes.	163
C.1.5	Statistical significance of model performance differences in identifying stabilizing mutations on the full test set.	164
C.1.6	Pearson correlation between model predictions and experimental stability effects on the RaSP test set (8 proteins) [20].	165
C.1.7	Pearson correlation between model predictions and experimental stability effects on the T2837 dataset and its subsets.	165
C.1.8	Predicting mutational effects on thermodynamic stability using ESMfold predicted structures for fine-tuning or testing.	166
C.1.9	Significance testing results for Fig. 5.4B.	167
C.1.10	Spearman correlations between pairs of model-average substitution matrices M^{model}	168
C.1.11	Model-averaged substitution matrices stratified by protein core and surface residues for zero-shot models.	169
C.1.12	Model-averaged substitution matrices stratified by protein core and surface residues for stability fine-tuned models.	170
C.1.13	Model reversibility scores on the Ssym dataset.	171
C.1.14	Model predictions for amino acid preferences at sites with known antigen-stabilizing mutations.	172
C.1.15	Statistical significance for the number of retrieved antigen-stabilizing mutations.	173
C.1.16	Model ability to recover different mutation patterns that stabilize antigens.	174
C.1.17	Identifying new antigen-stabilizing mutations with HERMES for DENV-E.	175
C.1.18	Comparison of PyRosetta and Biopython Pre-processing pipelines for predicting mutation stability effects on T2837.	176
C.1.19	Ablation of PyRosetta fastrelax parameters for HERMES-relaxed 0.50 on the cDNA117k dataset.	177
C.1.20	Structure of the wild-type pre-fusion RSV-F antigen highlighting observed and candidate antigen-stabilizing mutations.	178
C.1.21	Structure of the wild-type pre-fusion Universal-HA antigen, with highlighted observed and candidate antigen-stabilizing mutations.	179
C.1.22	Structure of the wild-type pre-fusion hMPV-F antigen, with highlighted observed and candidate antigen-stabilizing mutations.	180
C.1.23	Structure of wild-type pre-fusion DENV-E antigen, with highlighted observed and candidate mutations.	181
D.7.1	Robustness of ProteinMPNN performance across scoring schemes	206
D.7.2	Context-dependent mutational effects on T-cell activity.	207
D.7.3	Mutational effect on T-cell activities	208
D.7.4	HERMES and ProteinMPNN prediction accuracies for AlphaFold3-predicted structures.	209
D.7.5	Comparison of experimentally determined structures with AlphaFold3 predicted structures.	210
D.7.6	Peptide design energy profiles across systems and HERMES models.	211
D.7.7	Measurements of GFP levels by flow cytometry and fluorescence microscopy.	211
D.7.8	TCRdock and AlphaFold3 PAEs for HERMES-designed peptides	212
D.7.9	Experimental T-cell activation by HERMES-designed peptides in the NY-ESO system	213
D.7.10	Experimental T-cell activation by HERMES-designed peptides in the EBV system.	213
D.7.11	Experimental T-cell activation by HERMES-designed peptides in the MAGE system.	214
D.7.12	Composition of designed peptides using experimental versus AlphaFold3 structures as design templates.	215
D.7.13	Benchmarking of models for peptide design in the NY-ESO system.	216

D.7.14	Benchmarking of models for peptide design in the EBV system.	217
D.7.15	Benchmarking of models for peptide design in the MAGE system.	218
D.7.16	Discrimination ability of different scoring schemes as design filters.	219
D.7.17	Predicting MHC-I peptide presentation preferences with HERMES.	220

LIST OF TABLES

<i>Number</i>		<i>Page</i>
3.1	Performance metrics of H-(V)AE on MNIST-on-the-sphere and Shrec17.	25
3.2	Results on the Ligand Binding Affinity task.	28
4.1	Comparative assessment of H-Packer on CASP13 and CASP14.	36
4.2	Atom RMSD ($\text{\AA}$) for H-Packer across all residues with different treatments of the true structure.	37
4.3	Ablation in $\ell_{\max}$ of H-Packer.	37
4.4	Relative times to undertake full atomic reconstruction.	38
5.1	Benchmark on predicting mutational effects on protein-protein binding in SKEMPI. . .	56
6.1	Benchmarking of models for predicting TCR-pMHC binding affinities and T-cell activities. .	62
A.2.1	Performance comparison between our H-(V)AE and a H-(V)AE with ref. [252]’s non-equivariant decoder formulation, on the MNIST-on-the-sphere dataset.	102
A.3.1	Model architecture hyperparameters and training time for each H-AE trained on protein neighborhoods.	105
A.4.1	Evaluation of network performances for MNIST-on-the-sphere using a K-Nearest Neighbors (KNN) classifier instead of linear classifier in the latent space.	109
A.4.2	Evaluation of network performances for Shrec17 using a KNN classifier instead of linear classifier in the latent space.	109
A.4.3	Quantitative data ablation results on the Toy amino acids dataset.	109
A.4.4	Quantitative AE vs. VAE ablation results on the Toy amino acids dataset.	109
A.4.5	Test MSE and Cosine loss for H-(V)AE models trained on MNIST, Shrec17 and Protein Neighborhoods.	110
A.4.6	Training speed and reconstruction ablations of H-(V)AE models with different Tensor Product rules.	110
A.4.7	Mean equivariance error for some of our trained H-(V)AE models.	111
A.4.8	Comprehensive benchmarking results on the Ligand Binding Affinity task with Atom3D’s 30% similarity split.	112
B.2.1	List of conformational symmetries that we take into consideration.	136
B.2.2	Ablation in $\ell_{\max}$ of H-Packer, comprehensive of all metrics.	138
B.2.3	Reconstruction RMSD when considering the additional (non-natural) symmetries used by AttnPacker.	138
B.2.4	Reconstruction RMSD when standardizing the true structure by reconstructing it using our data-derived constant redundant internal coordinates.	139
B.2.5	Reconstruction RMSD when considering both the additional (non-natural) symmetries used by AttnPacker, and when standardizing the true structure by reconstructing it using our data-derived constant redundant internal coordinates.	139
C.1.1	Accuracy of HERMES models on wildtype amino-acid classification on all sites across 40 CASP12 test proteins.	155
C.1.2	Inference speed of HERMES models on the T2837 dataset.	155
C.1.3	Table of amino-acid properties used for comparison with substitution matrices.	156
C.1.4	Predicting antigen-stabilizing mutations with HERMES: extended results.	157
C.1.5	Execution times for saturation mutagenesis predictions on the viral antigens considered in this study.	158
C.1.6	Characteristics of antigen-stabilizing mutations.	159
D.7.1	Structure templates used for the binding affinity and T-cell activity analyses.	200

D.7.2	Benchmarking of models for predicting TCR-pMHC binding affinities.	201
D.7.3	Benchmarking of models for predicting peptide-induced T-cell activities.	202
D.7.4	AlphaFold3 folding metrics for structures used in our analyses.	203
D.7.5	Structure templates used for peptide design.	203
D.7.6	Structure templates to compute TCR-MHC specificities.	204
D.7.7	Structure templates to compute PWMs of peptide presentation on MHC.	205

Chapter 1

INTRODUCTION

If DNA is the recipe book of life - as it is the primary source of information about how a living organism is "built" [4] - then proteins are its machinery, its fundamental functional units. Indeed, many of the key players across all areas of biology are proteins: ion channels, antibodies, enzymes - names familiar to most computer scientists - represent the remarkably different functions that proteins can carry, and in fact *have to* carry in order to support life.

Despite their functional diversity, all proteins carry their respective functions according to shared fundamental principles. Thus, making accurate predictive models of the fundamental principles underlying protein function can provide insight into many biological processes, with broad applications in biotechnology and medicine [147, 244, 89]. Making accurate predictions, however, is not enough. Indeed, the speed at which predictions are made also matters, as low speeds can bottleneck discovery, to the point of being downright unfeasible for very slow predictions. In recent years, machine learning has emerged as a tool for creating models of proteins that are both fast in generating predictions, and accurate enough to significantly aid discovery and engineering, with AlphaFold2 being perhaps the most famous example [107].

Throughout my PhD I have followed this line of work. Specifically, this thesis explores the capabilities of a specific class of machine learning models that my colleagues and I have developed.

In this introductory chapter, I will first give a brief primer on proteins: how they originate, how they are composed, and the physical basis of protein function. I will then discuss traditional ways in which proteins and their functions are modeled, using explicit physics models as the fundamental basis. Then, I will discuss machine learning as a paradigm for modeling proteins via learning effective models. And finally, I will introduce the key design choices of the machine learning framework that is the subject of this thesis.

1.1 Proteins and their functions

1.1.1 How proteins are made

Within cells, information of how proteins are built is encoded directly into the DNA. At a high level, the protein creation process can be seen as a sequence-to-sequence translation mechanism: a specific sub-sequence of the DNA - composed of 4 possible *nucleotide* molecules serving as its alphabet - gets translated deterministically - except for errors - into a protein's sequence, composed of 20 possible *amino-acid* molecules.

More specifically, the process follows what is known as the central dogma of biology [4]. The above-mentioned sub-sequences of DNA - referred to as "coding regions" - get first *transcribed* into sequences of RNA - composed of 4 nucleotides just like DNA, with the exception of Thymine being converted into Uracil - then each RNA segment gets *translated* into a *chain* (or sequence) of covalently-bonded *residues*, where each residue is an instance of one of the 20 amino-acid types; see Figure 1.1.

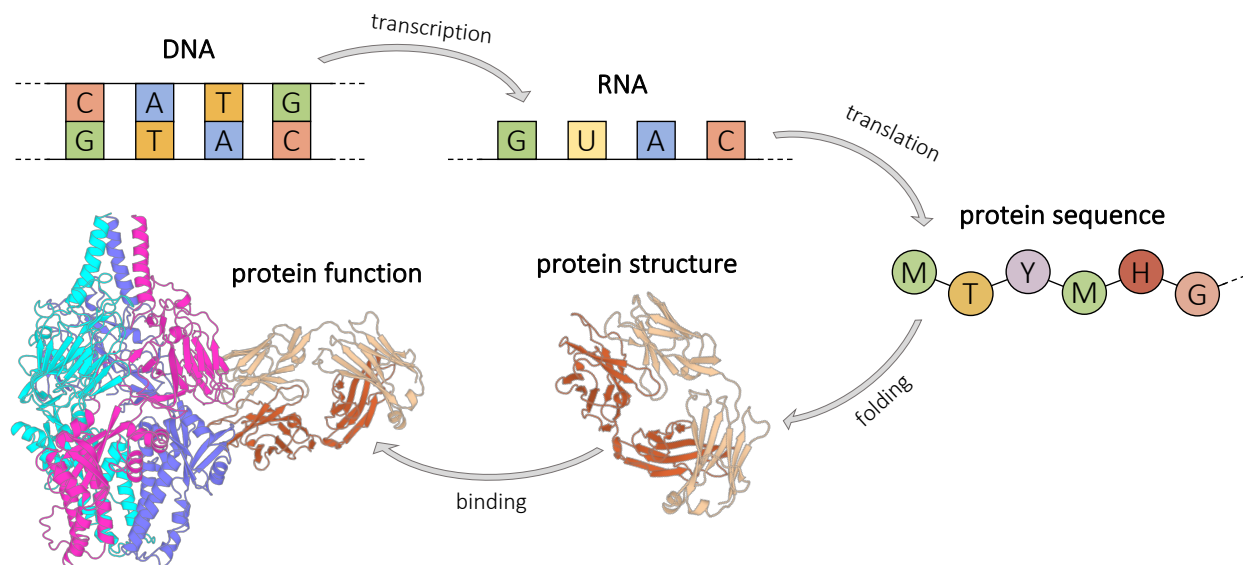


Figure 1.1: A protein's life cycle: from DNA to function. Highly-simplified schematic of the central dogma of biology (from DNA to protein sequence) tied to proteins' sequence-structure-function paradigm. Nucleotides are represented as squares, and amino-acid residues as circles. The protein represented is an antibody, whose function is to bind to the Respiratory Syncytial Virus Fusion antigen (also a protein), which then elicits an immune response against the virus; adapted from PDB ID 4jhw.

1.1.2 Amino-acids

Amino-acids are the building blocks of proteins, and the diversity of protein function stems ultimately from the diversity in the amino-acids. There are 20 possible canonical amino-acids ¹, with size spanning between the 5 heavy atoms of Glycine (G) to the 14 of Tryptophan (W). All amino-acids share a substructure called the *backbone*, composed of a central carbon - the $C\alpha$ - bonded to a carboxylic group ($-COOH$) and an amino group ($-NH_2$) (with the exception of proline, more on it later). Along a protein chain, subsequent amino-acids are linked together along their backbones via *peptide bonds*, whereby the carboxyl group of one reacts with the amino group of the other, releasing water; the sub-section of a protein denoted by all of the bonded amino-acid backbones is commonly referred to as the protein's backbone. What distinguishes each amino-acid is then its variable substructure: this is called the *side-chain*, and it is connected to the backbone via the $C\alpha$. Side-chains span a wide range of chemistries, but are generally categorized as follows: hydrophobic, polar uncharged, positively charged, negatively charged, and special cases, like prolines (P), which can only adopt certain backbone conformations, cysteines (C) that can form the very stable disulfide bonds, and glycines (G) which have no side-chain. See Figure 1.2 for the amino-acids graphs and classification into physico-chemical families. The high diversity of chemistries in the amino-acids, coupled with the extremely high number of possible sequences that can be created with them ², generates enormous complexity that is at the core of proteins' functional diversity.

1.1.3 The Sequence-Structure-Function paradigm

The extreme diversity in amino-acid sequence composition is at the core of the diversity in functional roles that can be carried out by proteins. The relationship between sequence and function is highly complex,

¹There are also other so-called non-canonical amino-acids, but they are rare and mostly get created after the RNA-protein translation event. For simplicity, we do not consider them in this work

²for a typical length of 400 residues, that is 20^{400} possible protein sequences, which dwarfs the estimated number of atoms in the universe - 10^{80}

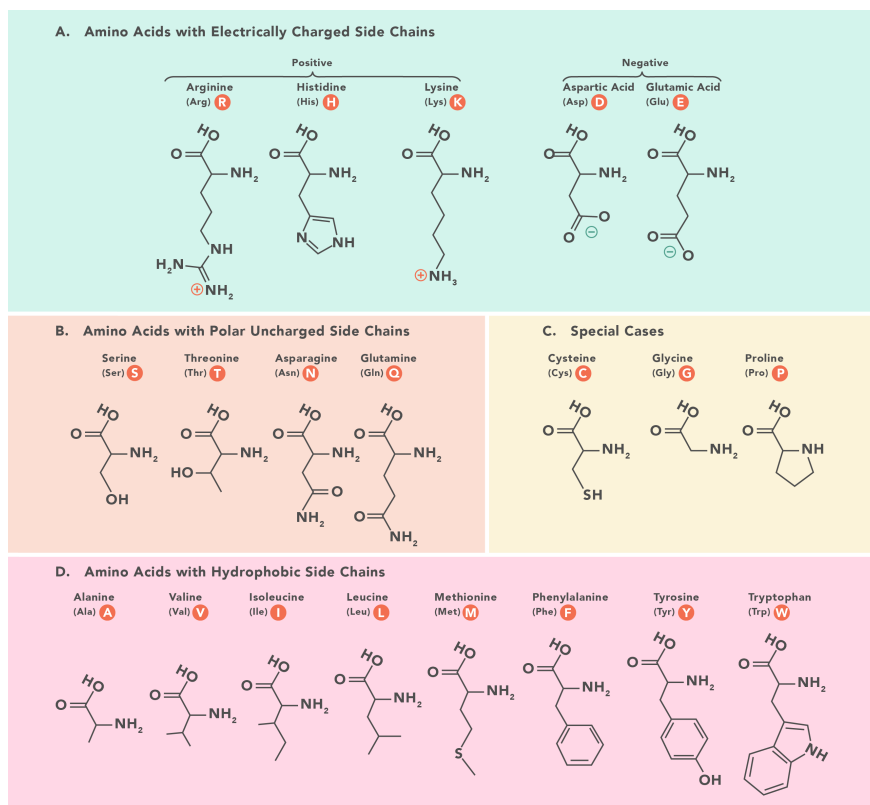


Figure 1.2: **Canonical amino-acids and their chemistries.** Taken from <https://www.technologynetworks.com/applied-sciences/articles/essential-amino-acids-chart-abbreviations-and-structure-324357>. This very image was my best friend throughout the PhD.

and indeed it is either directly or indirectly the focus of many areas of biological research. To unpack this, let us start by asking: what exactly do we mean by "function"? At a fundamental - and seemingly trivial - level, a protein carries out its function through engaging in specific interactions with other molecules in its environment: an ion channel interacts with ions, an antibody interacts with its antigen (often another protein), an enzyme interacts with its *substrate* (often a small molecule), and so on. What, then, determines the specific interactions that a protein engages in? A popular, simplifying description is provided by the "sequence-structure-function" paradigm, which states that the protein's engagement in specific interactions is largely driven by the specific 3D structure that the protein adopts, which in turn is determined by the amino-acid sequence of the protein. A canonical example is ion channels: to serve the purpose their name suggests, they are shaped with a "channel" in the center, with a specific size and shape such that they can let specific ions through at the right time. Enzymes are another example: they are shaped with a "pocket" that can bind to their substrate with high specificity.

Over the course of my PhD, I have learned that this description is useful, but its details as well as its exceptions must be noted with care. In what follows, I will attempt to unpack this paradigm carefully.

sequence → structure

We start by noting that proteins do not generally adopt a static 3D structure: the truth is, as always, more nuanced. As mentioned above, the protein's amino-acid composition (i.e. its sequence) determines the interactions the protein has with itself and with the watery environment it is in. Sometimes, the amino-acid composition is such that the protein never adopts a stable conformation: these proteins are referred to as

"intrinsically disordered", they make up around ~30% of the human proteome [246], and carry important functions [55]. Most times, however, these interactions cause the protein to "fold" onto itself into a mostly-stable 3D structure. We also note that the 3D structure of a single protein chain as a whole is commonly referred to as *tertiary* structure, but often what are colloquially referred to as "functional" proteins are actually composed of multiple non-covalently but tightly bound chains each with their own tertiary structure, forming the *quaternary* structure; for our purposes we may group together both types of structures without loss of generality.

There are several types of interactions and effects that contribute to a protein's fold: these include electronic interactions, which occur between charged side-chains, disulfide bonds, and perhaps the two most important ones, hydrogen bonds and hydrophobic effects. A hydrogen bond is a mostly electrostatic interaction, forming when a hydrogen atom covalently bound to an electronegative atom (called the "donor"), interacts with another electronegative atom (the "acceptor") that has a lone pair of electrons. Hydrogen bonds occur frequently in proteins, and are the primary drivers for the formation of *secondary* structure elements. Secondary structure is a local, regular folding pattern of the protein - in between the sequence (aka primary structure) and the overall 3D fold (tertiary and quaternary structure). There are three main secondary structure elements: α -helices, a right-handed helix held together by hydrogen bonds; β -sheets, linear stretches of protein held together side-by-side via hydrogen bonds; and finally loops, which do not have a regular shape - but can be both flexible and well-ordered - and serve as connecting regions between the other elements. Finally, the effect that is often cited as the primary driver of a protein's 3D fold is the hydrophobic effect. This is the name for the tendency of hydrophobic residues - such as alanines, valines, and leucines - to "hide" from water molecules which, since they are polar, "want" to engage with polar counterparts: other water molecules and polar residues. This effect gives rise to a particular 3D fold by forcing hydrophobic residues to be close to each other, "burying" themselves into the core of a 3D structure that they themselves contribute to create.

So far we have implicitly assumed protein chains are like flexible "strands" that can bend and fold at the sole will of the above-mentioned effects. This flexibility does however have its limits, which are worth mentioning. The protein chain can "bend" along two dihedral angles that form along the backbone: ϕ and ψ . Crucially, the range of values that these angles can take varies among the amino-acids. All amino-acids have roughly the same admissible range, except for two: glycines and prolines. Glycines are the most flexible: the lack of a side-chain allows for certain conformations that in other amino-acids are barred by steric clashes. Prolines instead are the opposite: their unique backbone geometry - with the amino group substituted by the pyrrolidine ring - restricts the possible range of dihedral angles, making Prolines notably unsuitable for both α -helices and β -strands.

The interplay of the interactions and effects mentioned above is complex, and it can be highly dynamic. Luckily, thermodynamics offers a conceptual framing that ties together these complex effects into quantities that ultimately allow us to reason about the global "stability" of a given protein's structural configuration, intended intuitively as the propensity of the protein to hold that specific structural configuration.

Specifically, every protein configuration can be assigned what is called a "Gibbs free energy" (denoted by G , and free energy thereof). Without going into details, this quantity is derived from the Second Law of Thermodynamics, which states that, within a closed environment, the *entropy* of a system (the protein) and its environment (its watery solvent) must spontaneously increase in the absence of external work, i.e. $\Delta S_{tot} > 0$ spontaneously over time [34]. Then, G is derived as a property *of only the system* (the protein) such that G spontaneously decreases over time. In other words, the minimization of G is a substitute for the Second Law that allows us to determine whether a reaction (a protein folding event in this case) would occur spontaneously, by focusing only on properties of the system [178].

What are then the properties of the system that compose G ? In short, G is largely derived from the interactions

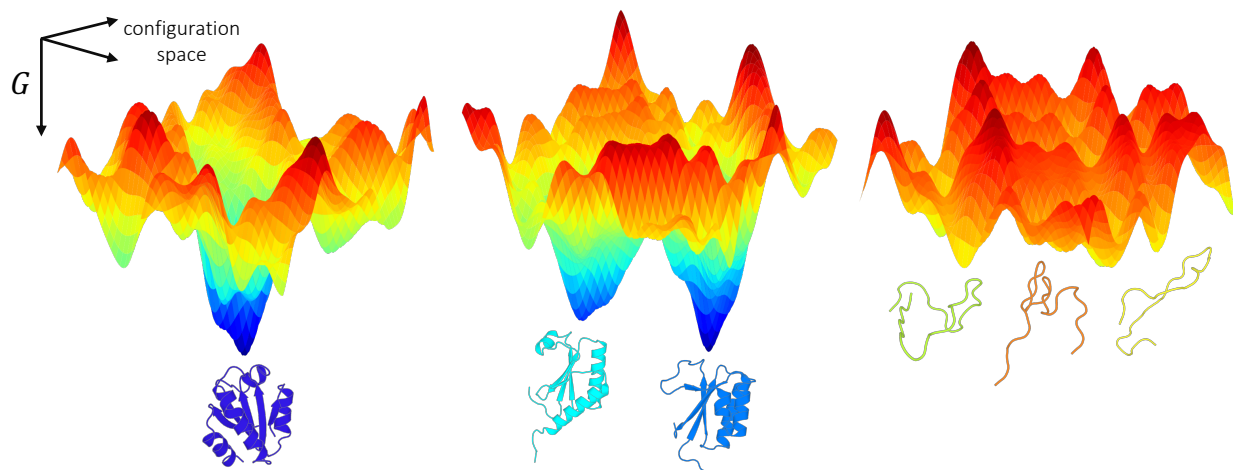


Figure 1.3: Gibbs free energy landscapes of protein folding. Simplified illustrations of three canonical folding free energy landscapes. *left*: landscape with a single deep well, corresponding to a single stable conformation. *center*: landscape with two deep wells, corresponding to a fold-switching protein. *right*: landscape without distinctly deep wells, characteristic of an intrinsically-disordered protein.

and effects mentioned earlier. Specifically, G is defined as:

$$G = H - TS \quad (1.1)$$

where T is the absolute temperature of the system, H is called the enthalpy, and S is the entropy of the system. Enthalpy is essentially the internal energy of the protein, determined by molecular interactions such as hydrogen bonds and electrostatic interactions (it also formally includes a term with pressure and volume, but we assume them to be constant here). Entropy instead quantifies - in loose terms - the amount of "conformational freedom" of the protein: how many specific conformations are "easily accessible" from the current one.

Since nature tends to spontaneously minimize the free energy, the protein folding process can be seen as nature's attempt at minimizing this quantity by searching through configuration space, often visualized with the "folding funnel" [168]. Note that this conceptual framing captures the full complexity of the protein sequence-structure relationship: the key is the structure of the free energy landscape. For example, a landscape with a deep, easy to reach global minimum represents a protein always found in a single, stable fold (Figure 1.3, left); a landscape with many deep local minima indicates a protein with many stable conformations (Figure 1.3, center); and a landscape without recognizable minima is characteristic of an intrinsically disordered protein (Figure 1.3, right). Last, we note that the protein inhabits this landscape *dynamically*: it is constantly being "kicked" around the landscape, moving the structure, by so-called thermal fluctuations, which scale in magnitude following temperature. A useful analogy is that of a "random walk biased by a gradient": the gradient is determined by the free energy landscape, and the amount of randomness is determined by temperature: higher temperatures cause higher exploration against the gradient.

In this thesis, we concern ourselves primarily with proteins that adopt a single, stable conformation, and occasionally with proteins that can adopt meta-stable conformations (i.e. sitting in local minima of the energy landscape, see Chapter 5). For a given stable conformation, we quantify its folding stability as

$$\Delta G_{folding} = G_{folded} - G_{unfolded} \quad (1.2)$$

where G_{folded} and $G_{unfolded}$ are the free energies of the folded and unfolded states, respectively. If $\Delta G_{folding} < 0$ then the protein is stable in the folded conformation, meaning it is more likely to be found in the folded conformation than in the unfolded one, and comparatively more so the lower $\Delta G_{folding}$ is.

Quantifying the relative stability of two meta-stable conformations can be done analogously by considering their relative energy: the conformation with lowest energy should be the most likely to find the protein in. However, there is a key subtlety: since both conformations sit at minima of the energy landscape, there is an energy barrier between them. This barrier may be insurmountable by thermal fluctuations alone, or it may be surmounted on timescales longer than the characteristic timescale of certain biological processes involving the protein. Therefore, coming out with the likelihood of finding a protein in a given meta-stable conformation over another is often more complex than simply considering the energy value at the landscape's minima.

A major breakthrough in protein science has been the ability to experimentally determine the structure of a protein in its lowest-energy state, using techniques like X-Ray Crystallography [200], and deposited in the Protein Data Bank [18]. These structures can be seen as "snapshots" of the protein around its lowest-energy state, so they do not capture their inherent thermodynamic fluctuations, but are usually sufficient representations of the structure we expect the protein to have as it carries its function. When thermodynamical fluctuations are strong enough, or otherwise relevant to the carrying out of the function, the single-snapshot representation is not sufficient, and instead an *ensemble* of snapshots would be a more appropriate representation of the protein's structure as it carries its function; we do not consider this case in this thesis.

structure (and sequence) → function

Having unpacked some of the complexity of the protein sequence-structure relationship, we move on to talk about how both relate to function-specific interactions.

The way in which sequence and structure prime a protein to engage in specific interactions can be understood at **two** levels of abstraction. At the highest level, the protein's general 3D fold determines the "class" of interactions carried out by the protein: e.g. a channel-like fold is necessary for a protein to function as an ion channel, and a specific pocket is necessary for a protein to function as a substrate-specific enzyme. More generally, the 3D fold of the protein can determine the shape and location of the "binding site" of the protein, i.e. the surface "patch" with which the protein interacts with other molecules in pursuit of its function.

At the lower level of abstraction then, is the precise amino-acid composition of the binding site, which determines the exact molecules - e.g. the exact ion or antigen(s) - an individual protein interacts with. Notably, the amino-acid composition of the binding site is often largely independent of the overall fold of the protein: for example, immune receptors explicitly evolved to have a hyper-variable binding site while keeping the overall fold mostly constant, allowing for receptors that bind a very broad range of antigens and thus can guarantee coverage from a broad range of pathogens. The local structure of the binding site can determine interaction (i.e. binding) specificity, but two proteins with the same local structure at the binding site can have different binding specificities if their amino-acid compositions are different enough.

The very same physical forces and effects that determine the folding of a protein - hydrophobic effects, hydrogen bonds, etc. - also determine the binding of the protein with its binding targets in carrying out its function. In particular, the overlap in the types of effects is exact if the target is also a protein. Therefore, the same thermodynamic framing that guided our understanding of folding can also be applied to binding. Specifically, we can speak of the energy of binding as

$$\Delta G_{binding} = G_{bound} - G_{unbound} \quad (1.3)$$

where, like for folding, a negative energy difference ($\Delta G_{binding} < 0$) implies that the system is more likely to be found in the bound conformation than in the unbound one. Specifically, $\Delta G_{binding}$ is related to the state of the system *at equilibrium*. More formally, if we consider the binding reaction between two proteins P_1 and P_2 (or a protein and another molecule), forming the bound complex P_{12} , then equilibrium is reached when

the relative concentrations of bound and unbound states are steady through time:

$$P_1 + P_2 \rightleftharpoons P_{12} \quad (1.4)$$

Then, the binding energy is specifically related to the *binding affinity* or *dissociation constant* K_d :

$$\Delta G_{\text{binding}} = RT \ln K_d \quad (1.5)$$

where K_d is the ratio of concentration between the unbound and bound state $K_d = [P_1][P_2]/[P_{12}]$ (T is absolute temperature, R is the ideal gas constant). Then, a lower K_d - and hence a lower $\Delta G_{\text{binding}}$ - implies a higher likelihood of finding the system in the bound state at equilibrium, a likelihood that is driven by the "tighter binding" granted in turn by the enthalpic and entropic effects of the interatomic interactions previously mentioned.

Binding affinity can be measured experimentally via (mostly expensive) methods like Surface Plasmon Resonance (SPR) [110].

1.2 Modeling proteins

There is a rich and ongoing history of experimental techniques for measuring key properties of proteins. I already mentioned X-Ray Crystallography to determine a protein's structure and SPR for binding affinity, but there are also methods to measure a protein's thermodynamic stability [232], complex functional phenotypes that depend upon stability and multiple binding events [227], and even new techniques for measuring local un-folding events [70].

Each of these types of measurements can give us different insight into how the protein's amino-acid sequence results in its function: resolving the lowest-energy structure tells us the protein's overall function; measuring changes to its stability as a result of mutations can tell us how the protein can be rendered non-functional; measuring its binding affinity to a target can reveal details about its function, and measuring it against a panel of molecules can tell us how to stop the protein from carrying out its function by using a drug.

However, while experimental techniques allow us to have a close look at what happens in nature, they are - and likely always will be - expensive and laborious. The alternative - as in all of science - is to predict desired properties using theoretical and computational models of proteins. Indeed, the mere fact that we seem to have a grasp of all the fundamental physico-chemical forces at play, has made researchers feel that highly-predictive models are within reach.

Being within reach, however, does not mean being simple. In what follows, I will outline some of the different paradigms that are used to make predictive models of proteins. I will do so in decreasing order of explicit physico-chemical detail included in the models.

1.2.1 Molecular Dynamics force fields: pure physics and dynamics

At its core, Molecular Dynamics (MD) consists in constructing a parameterized potential energy function at the atomic level - Lennard-Jones, Coulomb, etc. - whose parameters are fit to reproduce quantum chemical calculations on small molecules and experimental thermodynamic data [93, 138] (with the implicit assumption that these parameters apply broadly across all chemical domains, including proteins). This potential function considers as input the full atomic configuration - including the solvent (usually water for proteins) - and its output is then differentiated with respect to the atoms' coordinates to generate a force field. The force field is then integrated through time with very short (femtoseconds) time steps to generate a trajectory. In essence, the MD trajectory seeks to be a simulation of the reaction dynamics through time (e.g. folding, binding...).

Therefore, in principle and given enough time, one can map out the energy landscape for folding and binding, and compute Gibbs free energy differences such as $\Delta G_{folding}$ and $\Delta G_{binding}$ [171].

While MD appears to be a natural, first-principles way of modeling protein function, it is heavily bottlenecked by speed and computational requirements. Indeed, running a single molecular dynamics simulation at the timescales required to estimate relevant quantities takes enormous computational resources, making it prohibitive for medium and large systems.

The speed of an MD simulation decreases approximately linearly with the number of atoms present in the simulation (see [49] for a theoretical bound on the most expensive operation, and [213] for an empirical benchmark). For example, for atomic systems the size of a medium-size protein (92K atoms, 234 residues), the simulation speed is on the order of $1 \mu s/day$ using commercially-available hardware (a single A100 GPU), and in the order of 100s of $\mu s/day$ if using specialized hardware [213]. By comparison, the characteristic timescale of folding for proteins of similar size is on the order of *seconds* [163]. Thus, molecular dynamics on its own is limited in the types of molecular processes that it can model within a reasonable amount of time - and even then its accuracy is unclear. This bottleneck is even more problematic when considering that much of biological research and engineering involve predicting the properties of multiple proteins or protein-drug combinations, for example in efforts to find optimal drug candidates or finding protein variants that modify a protein's property of interest (e.g. its stability or its binding affinity to a desired target).

Despite these limitations, Molecular Dynamics remains the de-facto standard tool for modeling the dynamics of molecular events, albeit currently limited to events that happen at short-enough timescales.

1.2.2 Physics-based scoring functions: static physics and statistics

Scoring functions, among which the most notable example is Rosetta [7], are related to molecular dynamics in that they also are largely grounded in first-principles physical modeling, and they also construct a potential energy function. However, while the potential in MD is crafted to provide accurate dynamics, the potential in Rosetta is designed only to provide a landscape in which the global minimum corresponds to static, physically plausible conformations akin to native ones found in the Protein Data Bank. Alongside terms that approximate physical interactions (Lennard-Jones, hydrogen bonding...) the Rosetta scoring function also contains terms that encode statistical regularities extracted from experimentally-determined protein structures. An implicit model of solvent is also used, thus avoiding the need of explicitly placing water model molecules in the structure for scoring. Thus, Rosetta is conceived for scoring the "correctness" and "native-ness" of given static conformations, assuming they are at their lowest-energy state.

That being said, the scoring function can also be optimized, in a procedure known within Rosetta as "relaxation". Relaxation is not used to predict accurate molecular dynamics, but rather to locally adjust a protein's conformation, for example after applying mutations to the protein's amino-acid sequence.

Scoring functions like Rosetta are widely used in protein design and engineering pipelines, as they do not require specialized hardware to deliver useful insights. They cannot - at least not always easily - be used for all protein modeling tasks: for example, they are not suitable for predicting a protein's tertiary structure from its sequence, but they are useful for relaxation of local environments and scoring candidate structures. However, their limitations in accuracy and in speed still leave space for alternative approaches.

1.2.3 Machine learning for proteins

In recent years, machine learning (ML) [19], and in particular the sub-field of deep learning [81], has emerged as a new paradigm for modeling proteins at large.

At a high level, machine learning is the building of algorithms that *learn* function maps between different data domains (supervised learning) or that learn data distributions (unsupervised learning, generative modeling, often learning a function from data to likelihood value), such that the learned function *generalizes* to data that was not used to learn it (i.e. during *training*). Here, by generalization we mean specifically the ability of the model to be approximately as predictive on unseen data as it is on training data.

As such, ML has much broader scope than molecular dynamics and physics-based scoring functions: in fact, it can be used both to parameterize MD simulations [161], and for scoring protein conformations [198, 197].

More broadly, in the context of protein modeling, the role of machine learning has been to substitute explicit physics-based modeling with learned maps capturing complex statistical patterns present in data. Furthermore, while explicit physics-based modeling must operate at the atomic level - since the most robust laws that physics has worked out operate at this scale, with machine learning we have more freedom in choosing the scale at which we want to learn the maps, and also the "subjects" of the maps. For example, one could in principle learn a low-level map like the force applied on a single atom [161] just like they could learn a very high-level map like one that outputs the lowest-energy structure of a protein from its amino-acid sequence [107].

Whether a given function map gets learned successfully (i.e. it *generalizes*), is primarily a combination of two things: the representation capacity of the model, and the availability of data.

The model's representation capacity must be large enough to be able to capture the complexity of the task; another way to conceptualize representation capacity is via the *function class* that can be represented by a model: in a successful model, its function class must admit a good-enough approximation of the underlying target map. Luckily, having large-enough representation capacity is not an issue: neural networks, at the basis of deep learning, can in principle represent arbitrarily complex maps [47, 95].

The second necessary ingredient then is data. Unfortunately, the amount of data required to successfully learn a model scales with the complexity (i.e. representation capacity) of the model itself, and thus generally with the complexity of the target map.

Thus, in a complex and multi-scale domain like that of proteins, machine learning research is often concerned with carefully balancing representation capacity and data.

This can be done in several ways. An obvious one is simply to collect more relevant data for the desired task. However, this option is limited by high costs - in terms of the time, financial resource and expertise required for wet-lab experiments. Therefore, careful modeling strategies are required to make the most out of the available data.

Overview of modeling approaches

There are many decisions to be made when constructing machine learning models of proteins. The decisions I have encountered in my research can be placed within three distinct approaches.

The first one is reducing the complexity of the model, and hence its representation capacity, so that it requires less data to generalize. Indeed, in machine learning we often accept the compromise of having a model that makes a more crude approximation of the target map, to the benefit of generalization. However, in some cases the reduction in representation capacity can be done in such a way as to avoid any losses in the resulting model's ability to recover the target map. Specifically, if the target map is known to contain certain *symmetries*, then we can design the model such that any map it can learn must respect that symmetry. For example, say we wish to learn a scoring function - akin to a Gibbs Free Energy - for a protein's structural conformation. We know that the output of such target map (structure→energy) should be *invariant* (i.e. not change) with roto-translations of the structure. Therefore, if we constrain the model's function class in the right way, we can reduce the need for data to learn the map. This approach is central in this thesis and is widely used in machine learning research, belonging to the sub-field known as Geometric Deep Learning [30].

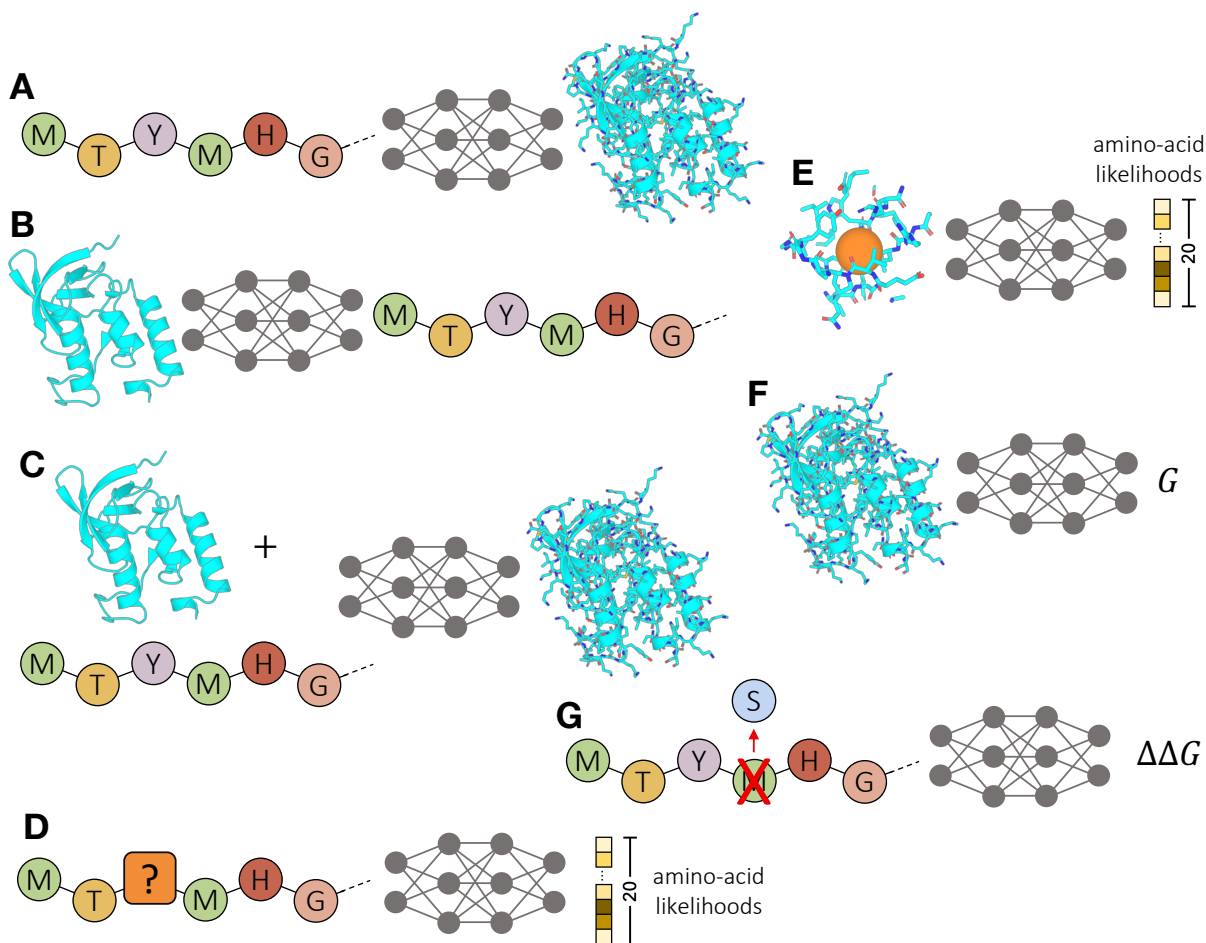


Figure 1.4: **Examples of machine learning tasks on proteins.** A: protein structure prediction from single sequence. B: inverse protein folding. C: rotamer packing. D: masked amino-acid modeling in sequence: predicting the probability distribution of amino-acids that fit in a sequence context. E: masked amino-acid modeling in structure: predicting the probability distribution of amino-acids that fit in a structural context. F: free energy scoring of a protein conformation. G: predicting the effect of an amino-acid mutation on a protein's stability, using the protein's sequence.

The second approach for careful modeling of proteins with machine learning, is to carefully craft the *task*, by specifying the "subjects" of the learned map, such that it is learnable with available data. This is quite abstract, so let me explain by means of examples. AlphaFold2 is cited as the first machine learning algorithm with sufficiently high performance in predicting protein structures to be considered broadly useful [107]. However, the map that the AlphaFold2 neural network learns is actually not from sequence to structure, but rather from Multiple Sequence Alignment (MSA) to structure. In short, an MSA is a stacking of protein sequences that highlights which residues and sections of the protein have been preserved throughout evolution, and which ones have varied; proteins in the MSA are referred to as *homologs* of the query protein, a term indicating a close common evolutionary ancestor which causes sequence similarity. Since co-evolutionary signal in MSAs is indicative of structural features [78], the mapping MSA→structure is much easier to learn (i.e. requires less data examples) than the mapping sequence→structure. Since an MSA can be constructed for any protein (except for *orphan* proteins, which have no detected homologs), at the level of abstraction of the user, the mapping appears to be from sequence to structure, but the machine learning map is really MSA→structure (in fact, it is notoriously hard to accurately predict the structure of orphan proteins [43]). Another example that is more relevant to this thesis, is that of predicting mutation effects. Specifically, say we wish to predict

the effect that a certain mutation (i.e. a single amino-acid substitution in the protein) has on its stability $\Delta G_{folding}$. We could conceivably construct a map from the amino-acid sequences to the energetic difference (sequences $\rightarrow \Delta\Delta G$). However, we know from first-principle protein science that the forces and effects holding a protein together are acting locally throughout the structure (e.g. hydrogen bonds, electrostatics, hydrophobic bulk). Therefore, including in the mapping the structure of the starting (usually called *wild-type*) protein directly provides information of the types of interactions and effects that the mutation could create or destroy. This very approach is considered in this thesis, even going one step further and assuming knowledge of only the local structure surrounding the mutating residue. Overall, the diversity of tasks that can be built and “machine-learned” is astounding; some examples are shown in Figure 1.4.

Finally, the third approach, related in spirit to the second one, is to craft a task that can be used to make *zero-shot* predictions for other tasks that have less available data and thus cannot be learned as robustly. A model’s prediction for a given task is termed “zero-shot” if the model was not explicitly trained to make such prediction. These predictions only work if there is meaningful semantic overlap in the target maps of the two tasks [122, 170]. A notable example of this in protein modeling is the predicted confidence scores of a structure prediction model like AlphaFold2 [107]. Confidence scores are trained to recapitulate the expected error in the structure predicted by the algorithm. However, it has been repeatedly found that confidence scores can serve as scoring functions of both protein sequences and structures that partially recapitulate protein stability [198] and protein-protein binding affinity [26]. We make use of this in chapter 6 and I have co-authored a perspective article on this phenomenon [167]. Another example is the prediction of the effect of amino-acid mutations on several phenotypes (e.g. stability) via probabilistic models. It has been observed time and time again that learning the probability distribution over possible amino-acids conditioned on sequence and/or structure contexts can partially recapitulate mutational effects. Crucially, carefully choosing the context can result in predictions that are more indicative of certain phenotypes over others. Chapters 5 and 6 adopt and discuss this approach.

Machine learning models as effective models

The modeling approaches discussed above highlight the conceptual freedom in design choices that can be made when modeling proteins with machine learning. Scales, tasks, parameterizations: these are all degrees of freedom in the design space of protein ML models, yielding models with varying accuracy, use cases, and speed. In my view, this can be summarized in the notion that *machine learning models learn effective models from data*.

In physics, an effective model is a description of a system that is predictive of its behavior within specified limits - often a specific scale, without needing the full underlying details of the system. While often conflated, we distinguish an effective *model* from an effective *theory* in that the latter claims to correctly describe the causal structure of the system within the specified limits, while the former only claims to be predictive - at least enough to be useful - of the system’s behavior. Under this definition, a machine learning model can be seen as learning an effective model of the system from statistical patterns present in data.

Note that, to enable prediction speeds much greater than those from first-principles physics-based methods like MD and Rosetta, learning effective models is paramount: we already know considering all the details is too computationally expensive!

The fundamental question to ask then becomes: have we learned a useful effective model, or are the “shortcuts” it takes not predictive of real-world phenomena? I have mentioned *generalization* before, but reducing the model to its average performance is hardly insightful. Instead, digging deeper into the model, identifying the scenarios in which it excels and those in which it fails, understanding the principles behind the model - in simpler words, why it makes the prediction it makes - can be useful on many fronts. From a modeling perspective, these *interpretability* analyses can reveal failure points and promising avenues of research.

Also, from the user’s perspective (in our case mostly biologists), these analyses can indicate the precise circumstances in which to trust a given model, akin to a function’s contract in software development. The question of interpretability arises in Chapter 5.

1.3 Scope of this thesis

The bulk of my PhD work has been on developing a set of machine learning models - or tools - all tied in a common framework with key design choices, to model different aspects of proteins. Specifically, all the modeling efforts focus on protein structure, and its relationship to sequence, stability, and binding affinity. Even more specifically, all the models make predictions at the level of single protein sites.

An obvious product of this thesis are the models themselves - freely available and currently used by the research community - and the problems they solve, which I have elected to describe at the beginning of each dedicated chapter. However, I also want this thesis to be read as a thorough attempt to apply a specific set of design choices to model proteins. There are three main design choices, all driven by considerations rooted in physics, and first introduced in my former labmate’s work [185].

- *Locality*. Since the physical forces that hold proteins and protein-protein complexes together act mostly locally in space, we consider as input to the models only the local atomic environment surrounding a protein site, centered at the focal residue’s $C\alpha$ and with a radius of 10 or 12.5 Å, depending on the model. We call these local environments *neighborhoods*.
- *Atom-level representation*. Since the chemical properties of amino-acids - and the interactions they can partake in - are trivially described by their atomic composition - we consider as input the atomic point cloud itself, instead of a coarse-grained representation of structure and sequence like other works [52]. We also often augment it with other physico-chemical properties like partial charge and Solvent Accessible Surface Area (SASA).
- *Rotation-equivariance*. Since all the tasks considered in this work present rotational symmetries about each neighborhood’s center, we build our models with rotation-equivariant neural networks, whose output provably respects the rotational symmetry of the target map.

In the next chapter, I will formally introduce this framework. Then, each of the following 4 chapters will be devoted to a particular first-author publication: the first three are focused on modeling and introduce each a new machine learning model for a specific high-level task; the last one instead is focused on using one of the models to tackle a longstanding problem in theoretical immunology. A high-level visual summary of the models and tasks discussed in this thesis can be found in Figure 1.5.

1.4 Beyond this thesis

Throughout my tenure at UW, I have also participated in other projects and co-authored papers both in line with and tangential to my line of research. These works are:

1. Nourmohammad, Pun, and Visani [167], a viewpoint article on this paper [198].
2. Borisiak, Visani, and Nourmohammad [23], which I helped conceive and in which I acted as a mentor.
3. Liu et al. [135], where I contributed code and discussions for predicting binding specificity.
4. Savioli et al. [204], where I acted as machine learning expert as mentor.

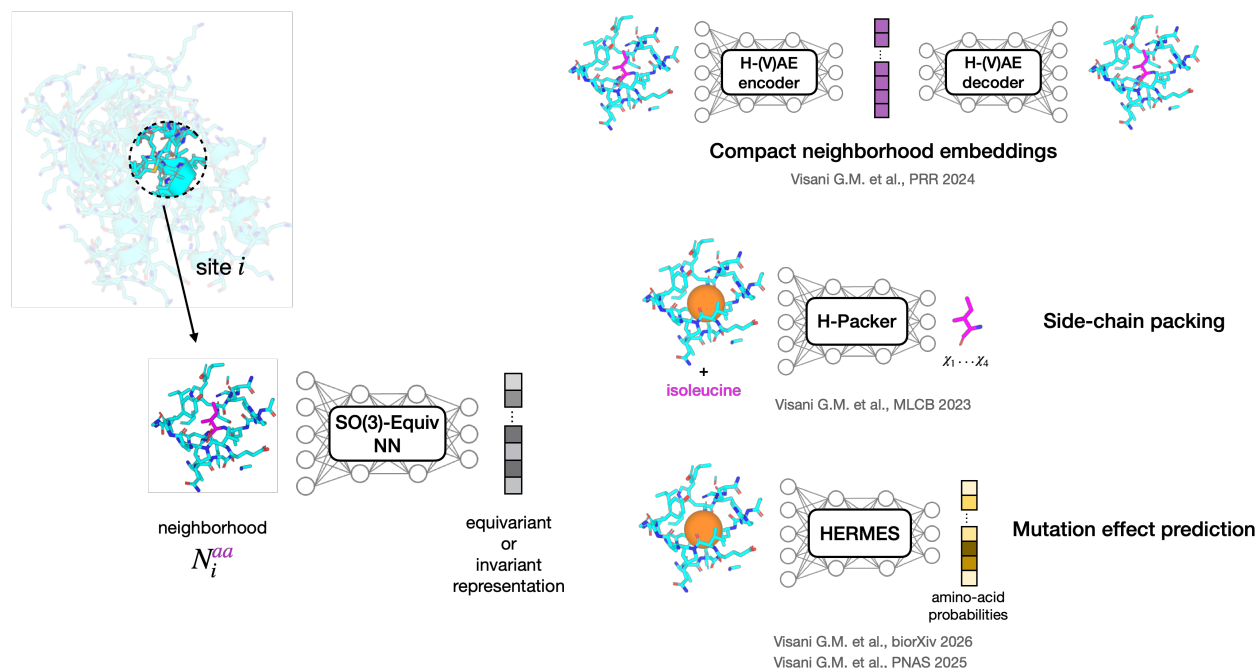


Figure 1.5: **Schematic overview of modeling framework and specific tasks addressed in this thesis.** The left-hand side illustrates this thesis' framework's focus on building equivariant and invariant models of site-centered local atomic environments (neighborhoods). The right-hand side shows the three high-level tasks tackled using the framework, with associated publications.

5. Richardson et al. [189], which describes the results of a Kaggle competition in which I entered HERMES, one of the models described in this work.
6. Visani, Verma, and DeWitt [236], a critical analysis of a high-profile machine learning model, born out of discussion with UW Genome Sciences Professor William DeWitt while I was being hosted in his lab.

Chapter 2

SO(3)-EQUIVARIANT FRAMEWORK

We describe here our general framework for building neural networks that are equivariant to 3D rotations of their input. 3D-rotations are described by the Special Orthogonal of degree 3, aka SO(3), hence we speak of SO(3)-equivariant neural networks. This framework is applicable to any input data within a spherical volume that has a semantically-meaningful center, such as spherical images [240], directional acoustic signals [155] and finally atomic point clouds, the main subject of this thesis.

We first introduce group equivariance, discuss why and how to incorporate it in machine learning, and finally introduce SO(3)-equivariant neural networks. The contents of this chapter are largely adapted from previous works, including [185, 230, 117, 30].

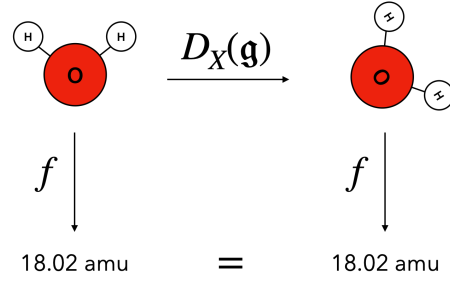
2.1 Mathematical background

Group Invariance and Equivariance. Starting with an intuitive explanation, a *function* is said to be *invariant* to a certain group of transformations (e.g. 3D rotations) if applying one such transformation to the function’s input does not change its output. *Equivariance* is then a generalization of invariance whereby, when the input is transformed by the action of a certain group element (or rather by a matrix representation parameterized by the group element), the output of the function is transformed by the same group element (i.e., by a matrix representation parameterized by the same group element, but that can be different from the input’s representation). In short, equivariant functions transform the input in the same way regardless of its coordinate frame, but do not necessarily discard the coordinate frame information, whereas invariant functions also do the latter. We note that both of these concepts can be extended to *properties* as well, since the property of an object can be seen as the output of a function applied to the object. For example, the mass of a molecule remains constant when rotating it (i.e. it is invariant to 3D rotations), whereas its dipole moment rotates alongside it (i.e. it is equivariant to 3D rotations). Switching now to a more formal treatment, a function between two vector spaces $f : X \rightarrow Y$ is said to be equivariant to a group of transformations $\mathfrak{G}$ iff applying any group transformation to the input space of f corresponds to applying the same transformation to the output space (i.e., via a *representation* parametrized by the same group element). Formally: $f(\mathbf{D}_X(\mathbf{g})\mathbf{x}) = \mathbf{D}_Y(\mathbf{g})f(\mathbf{x})$, $\forall \mathbf{x} \in X \wedge \forall \mathbf{g} \in \mathfrak{G}$. The group acts on the input and output vector spaces with via matrix multiplication with *representations* that are specific to the space (e.g. $\mathbf{D}_X$ and $\mathbf{D}_Y$) and are parameterized by the same group element $\mathbf{g}$. A group may have different representations, and a special one is the one that always maps to the identity: $\mathbf{D}_Y(\mathbf{g}) = \mathbf{I}$, $\forall \mathbf{g} \in \mathfrak{G}$; a function on whose output space $\mathfrak{G}$ acts with the identity representation is said to be *invariant* to $\mathfrak{G}$. See Figure 2.1 for a visual reference of these concepts.

Irreducible representations: the key to building equivariant neural networks. How are equivariant layers generally achieved? The key is to look at the group’s *irreducible representations* (irreps). These are the group’s *smallest* representations, so that any possible representation can be provably decomposed into a direct sum of irreps. Therefore, the group’s irreps can be used to describe how the group elements act on any vector space. Consequently, data itself can be seen as being composed (via direct sum, i.e. concatenation) of features of different "types", where having a particular type means transforming - under the group’s action - according to the group’s corresponding irrep. It turns out that, if data is decomposed in such a way, it is then generally easier to work out the conditions for constructing equivariant functions acting on it. This is easier said than done, but it has been worked out for SO(3), the group describing 3D rotations about a fixed point [76, 116,

Invariance:

$$f(D_X(\mathbf{g})x) = f(x)$$



Equivariance:

$$f(D_X(\mathbf{g})x) = D_Y(\mathbf{g})f(x)$$

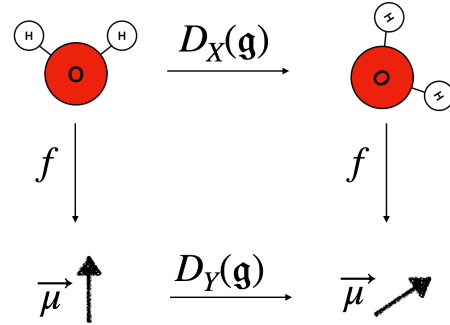


Figure 2.1: **Invariance and Equivariance to 3D rotations, explained using a water molecule.** The mass of a water molecule does not change if the molecule is rotated - it is said to be *invariant* to rotations of the molecule. Instead, its dipole moment rotates alongside the molecule - it is said to be *equivariant* to rotations of the molecule. See text for a more formal mathematical definition. Slide credit: Michael Pun.

230]. The irreps of $SO(3)$ are the Wigner-D matrices:

$$D^\ell(R) \in \mathbb{C}^{(2\ell+1) \times (2\ell+1)} \quad \text{where } \ell = 0, 1, 2, \dots \text{ and } R \in SO(3) \quad (2.1)$$

Notably, $D^0(R)$ describes how *invariant* features transform under rotation, and thus it is 1 for every rotation. Thus, to construct $SO(3)$ -Equivariant models, two things are needed: 1) a way to encode the data into a set of features that "rotate" according to these irreps, and 2) a way to process the resulting data equivariantly.

2.2 Holographic Encoding of the Data

To perform the encoding we leverage the Spherical Harmonics. This set of functions has the desired property that each of them "rotates" according to the Wigner-D matrices:

$$Y_{\ell m}(\theta, \phi) \xrightarrow{\text{rotation: } R} \sum_{m'=-\ell}^{\ell} D_{m'm}^\ell(R) Y_{\ell m'}(\theta, \phi) \quad (2.2)$$

where $Y_{\ell m}(\theta, \phi)$ is the spherical harmonic of degree ℓ and order m defined as

$$Y_{\ell m}(\theta, \phi) = \sqrt{\frac{2\ell+1}{4\pi} \frac{(\ell-m)!}{(\ell+m)!}} e^{im\phi} P_\ell^m(\cos \theta) \quad (2.3)$$

where ℓ is a non-negative integer ($0 \leq \ell$), and m is an integer within the interval $-\ell \leq m \leq \ell$. $P_\ell^m(\cos \theta)$ is the Legendre polynomial of degree ℓ and order m . The spherical harmonics also form a complete and orthonormal basis for signals on a sphere. Thus, a signal over a sphere of radius r - $\rho(\theta, \phi) : S^2(r) \rightarrow \mathbb{R}$ -

can be encoded into a set of Fourier coefficients $\hat{F}_{\ell m}$ via the spherical Fourier transform (SFT):

$$\hat{F}_{\ell m} = \int_0^{2\pi} \int_0^\pi \rho(\theta, \phi) Y_{\ell m}(\theta, \phi) \sin \theta \, d\theta \, d\phi \quad (2.4)$$

Not only do these coefficients efficiently represent the signal without loss of information (up to desired maximum resolution $\ell_{\max}$), but they also transform under $\text{SO}(3)$ following the Wigner-D matrices (analogously to 2.2).

The original spherical signal $\rho(\theta, \phi)$ can then be recovered (up to resolution $\ell_{\max}$) via the Inverse SFT (ISFT):

$$\tilde{\rho}(\theta, \phi) = \sum_{\ell=0}^{\ell_{\max}} \sum_{m=-\ell}^{\ell} \hat{F}_{\ell m} Y_{\ell m}(\theta, \phi) \quad (2.5)$$

To encode 3D data about a predefined center ($\rho(r, \theta, \phi) : \mathbb{R}^3 \rightarrow \mathbb{R}$) one can simply add a complete and orthonormal radial basis function that will capture the radial distribution of the data around the center. Following [184], we use the Zernike radial functions $R_\ell^n(r)$, leading to the Zernike Fourier Transform (ZFT):

$$\hat{Z}_{\ell m}^n = \int_0^1 \int_0^{2\pi} \int_0^\pi \rho(r, \theta, \phi) Y_{\ell m}(\theta, \phi) R_\ell^n(r) r^2 \sin \theta \, dr \, d\theta \, d\phi \quad (2.6)$$

where n represents radial resolution and is capped to a maximum value $n_{\max}$ in practice.

To represent point clouds, a common choice for $\rho(\mathbf{r}) \equiv \rho(r, \theta, \phi)$ is a sum of Dirac- δ functions centered at each point:

$$\rho(\mathbf{r}) = \sum_{i \in \text{points}} \omega_i \delta(\rho(\mathbf{r}_i) - \rho(\mathbf{r})) \quad (2.7)$$

Where ω_i can be used to represent point-specific features. This choice is powerful because the forward transform has a closed-form solution that does not require a discretization of 3D space for numerical computation. Specifically, the ZFT of a point cloud follows:

$$\hat{Z}_{\ell m}^n = \sum_{i \in \text{points}} \omega_i R_\ell^n(r_i) Y_{\ell m}(\theta_i, \phi_i) \quad (2.8)$$

And again, the data can be reconstructed up to resolution defined by $\ell_{\max}$ and $n_{\max}$ using the corresponding inverse transform (IZFT):

$$\tilde{\rho}(r, \theta, \phi) = \sum_{\ell=0}^L \sum_{m=-\ell}^{\ell} \sum_n^N \hat{Z}_{\ell m}^n R_\ell^n(r) Y_{\ell m}(\theta, \phi) \quad (2.9)$$

As mentioned above, we can use ω_i to introduce atom-level features (such as atom type, partial charge, etc.). If ω_i is a *vector* of multiple features (e.g. a one-hot encoding of atom type) indexed by its own index α , the resulting Zernike coefficients will also have an associated index α . This is equivalent to "splitting" the point cloud into multiple *channels* where each point has scalar feature value ω_i , and concatenating features of the same type ℓ , as shown in Figure 2.2B. For the purposes of constructing equivariant operations, however, the indices α and n are combined into a single index c . The number of indices c for a particular type ℓ is called the *multiplicity* of ℓ (C_ℓ), following the e3nn nomenclature [76]; the multiplicity C_ℓ is often the same for all ℓ and just denoted by C .

In this document, we refer to a collection of Zernike coefficients representing a point cloud as a *hologram*, due to the fact that Zernike projections in spherical Fourier space can be understood as a superposition of

spherical holograms of an input point cloud.

Instead, we refer to a generic collection of features of type ℓ (which can be a hologram or the output of one of the equivariant layers described in the next section) as a *steerable tensor*, following the nomenclature of ref. [27]. "Tensor" indicates the multi-dimensional nature of the representation, and "steerable" indicates that it can be transformed ("steered") - under the action of the group of interest, $SO(3)$ in our case - via matrix multiplication.

On the Choice of Zernike. The use of other radial bases is possible within this framework, as long as they are complete. Orthonormality is also desirable as it ensures that each basis encodes different information, resulting in a more efficient encoding of the coefficients. ref. [25] demonstrates that encoding with Zernike polynomials result in a faster convergence compared to the radial basis functions localized at different radii, as well as most other orthogonal harmonic bases, with the exception of Logan-Shepp. "Faster convergence" indicates that fewer frequencies are required to encode the same information. ref. [234] also uses Zernike to construct invariant descriptors of atomic environments. However, this choice is not unique and other special functions such as Bessel functions can be used to encode the data [161].

2.3 $SO(3)$ -Equivariant Layers

Once the data is encoded in the irreps basis, one can leverage a suite of pre-existing rules to build learnable functions (layers) without breaking rotational equivariance, and transform the input tensor into a new tensor composed of features within the same range of possible types. The most important consideration is that coefficients sharing the same ℓ and n , but having different m are *coupled*, meaning that they cannot be transformed independently of each other in arbitrary fashion or else equivariance gets broken. However, invariant properties (such as $\ell = 0$ coefficients, and magnitudes $\sum_m |\hat{Z}_{\ell m}^n|^2$) can be transformed arbitrarily. Arguably the most important equivariant operation is the Clebsch-Gordan (CG) Tensor Product [233]. It is

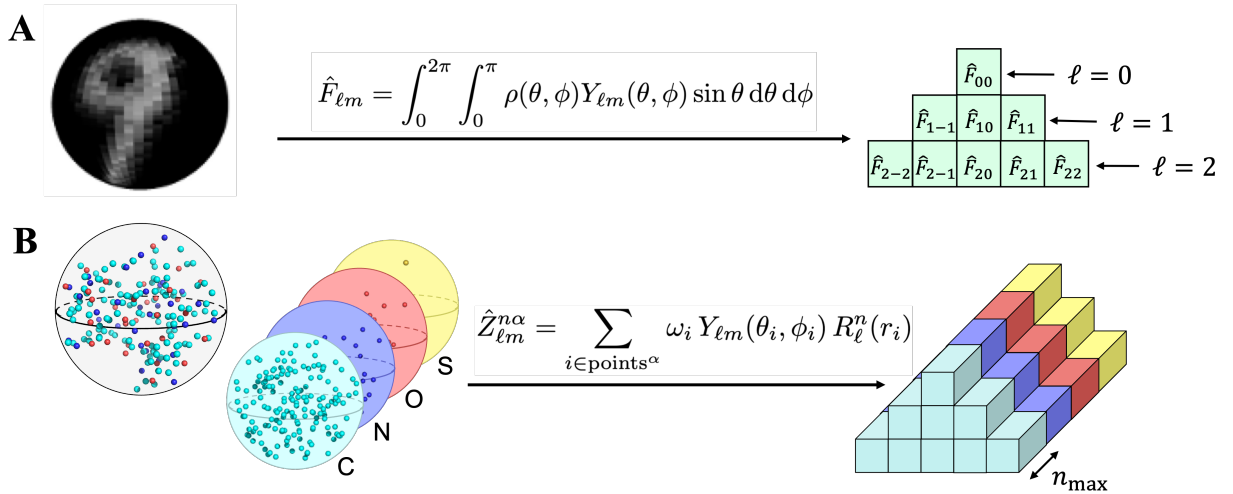


Figure 2.2: **Visualization of the Holographic Encoding considered in this thesis.**

(A): Encoding of a spherical signal. The signal is encoded via the Spherical Fourier Transform into Fourier coefficients indexed by degree ℓ and order m , up to a maximum type (aka degree) $\ell_{\max}$ ($\ell_{\max} = 2$ in this example). The coefficients are most naturally represented with a pyramidal structure, as the order m lies in the range $[-\ell, \ell]$. **(B):** Encoding of a centered point cloud, specifically an atomic environment. In addition to ℓ and m , coefficients are indexed by radial coefficient n as well as channel α , where the latter is indicative of atom-level features (e.g. atom type); in essence, the ZFT (Eg. 2.8) is performed once per channel. The radial and channel indices are conceptually combined in the *multiplicity* of a given type ℓ .

the only known operation capable of non-linearly mixing information between features of different type ℓ and with different momentum index m . The CG tensor product combines two features of degrees ℓ_1 and ℓ_2 to produce another feature of degree $|\ell_2 - \ell_1| \leq \ell_3 \leq |\ell_1 + \ell_2|$. Let $\mathbf{h}_\ell \in \mathbb{R}^{2\ell+1}$ be a generic degree ℓ tensor, with individual components $\hat{h}_{\ell m}$ for $-\ell \leq m \leq \ell$. The CG tensor product is given by,

$$\begin{aligned}\hat{h}_{\ell_3 m_3} &= (\mathbf{h}_{\ell_1} \otimes_{cg} \mathbf{h}_{\ell_2})_{\ell_3 m_3} \\ &= \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} C_{(\ell_1 m_1)(\ell_2 m_2)}^{(\ell_3 m_3)} \hat{h}_{\ell_1 m_1} \hat{h}_{\ell_2 m_2}\end{aligned}\quad (2.10)$$

where $C_{(\ell_1 m_1)(\ell_2 m_2)}^{(\ell_3 m_3)}$ are the Clebsch-Gordan coefficients.

These rules and operations can then be used to build neural network layers that respect SO(3)-Equivariance.

SO(3)-Equivariant Linearity. The equivariant linear layer consists of a set of linear projections each acting on the set of features sharing the same type ℓ . It practically equates to $\ell_{\max} + 1$ standard linear layers with no bias except for the $\ell = 0$ case, and with the consideration that all $2\ell + 1$ moments of the same feature are processed together. Formally, let $\mathbf{h}_\ell \in \mathbb{R}^{C \times (2\ell+1)}$ be a set of C features of type ℓ . Then, we learn weight matrix $\mathbf{W}_\ell \in \mathbb{R}^{C \times K}$ that linearly maps $\mathbf{h}_\ell$ to $\bar{\mathbf{h}}_\ell \in \mathbb{R}^{K \times (2\ell+1)}$; if $\ell = 0$, $\mathbf{b}_\ell \in \mathbb{R}^K$ is also learned:

$$\bar{\mathbf{h}}_\ell = \mathbf{W}_\ell^T \mathbf{h}_\ell (+ \mathbf{b}_\ell) \quad (2.11)$$

This layer was introduced by [116], and within it reside most of the learnable parameters.

Efficient Tensor Product Nonlinearity. The CG Tensor Product is used as the primary nonlinear activation within this framework - as originally prescribed by [116] - by applying it between features within the same tensor. Crucially, this is also the only operation in this framework that can transfer information across features of different types ℓ . [116] and [184] compute this tensor product naively, by processing every possible pair of features of types ℓ_1 and ℓ_2 within the tensor (dubbed Full Tensor Product). Instead, we consider two modifications proposed by [44] that significantly reduce its computation time while sacrificing little in terms of expressive power. The first modification, which we refer to as "channel-wise computation", consists in conceptually splitting the steerable tensor along the channel dimension, yielding C tensors of effective channel-size 1; then, the Tensor Product is computed within each tensor, and the output concatenated along the channel dimension. This modification reduces the time and space complexity with respect to the channel size C from $O(C^2)$ to $O(C)$. The second modification, which we refer to as "sparsification", consists in dropping some of the individual Tensor Products. Specifically, if there already exist a Tensor Product within (ℓ_1, ℓ_2) , and within (ℓ_2, ℓ_3) , then the Tensor Product within (ℓ_1, ℓ_3) is dropped. This modification reduces the time and space complexity with respect to $\ell_{\max}$ from $O(\ell_{\max}^2)$ to $O(\ell_{\max})$. Throughout this work, either both modifications or only the "channel-wise computation" modification are considered. An ablation study on the efficacy of these modifications - specific to H-(V)AE - can be found in Table A.4.6. See ref. [44] for further details.

SO(3)-Equivariant Batch Norm. Introduced by [116]. Normalizes a tensor's features equivariantly by their respective batch-averaged magnitude:

$$N_\ell^c = \frac{1}{B} \sum_{b=1}^B \frac{1}{2\ell+1} \sum_{m=-\ell}^{\ell} (h_{\ell m}^{cb})^2 \quad (2.12)$$

where h is the tensor, indexed by ℓ , m , c and batch b .

Signal Norm. We devise a simple normalization strategy to help confine the activations within the networks, which have a tendency to explode due to the tensor product being akin to a square non-linearity. It consist of

simply normalizing a tensor h by its total magnitude. Specifically, the total magnitude is defined as:

$$N_{tot} = \sum_{\ell} \frac{\sum_c \sum_{m=-\ell}^{\ell} (\hat{h}_{\ell m}^c)^2}{2\ell + 1} \quad (2.13)$$

Then, the tensor features are updated as $\overline{\hat{h}_{\ell m}^c} = \hat{h}_{\ell m}^c w_{\ell} / \sqrt{N_{tot}}$, where w_{ℓ} is a degree-specific affine transformation for added flexibility

Layer Norm Nonlinearity. This consists of applying a standard Layer Norm [14] to each of the *norms* of type- ℓ features (which are invariant), and then feeding the normalized norm into a standard nonlinear activation function. This layer effectively combines the equivariant layer norm in e3nn [76] with the Norm Nonlinearity originally used in Harmonic Networks [255], and then adapted to the SO(3) domain by Tensor Field Networks [230]. It is also used in the SE(3)-Transformer [72, 209].

Implementation. These SO(3)-equivariant layers, as well as the holographic encoding, were implemented with the help of e3nn primitives [76].

2.4 Why Equivariance?

In deep learning, the creation of a model that is invariant or equivariant to a desired group of transformations has been traditionally achieved by training the network with group-transformed copies of the input-output pairs, a practice known as data augmentation. However, for high-dimensional groups, the amount of data augmentation required to learn the desired symmetry can be prohibitive (~ 500 fold for SO(3) [76]). Therefore, encoding the desired symmetry structure into the neural network structure itself, rather than learning it via data augmentation, can significantly reduce training time while guaranteeing exactness.

When the target function is specifically invariant, another alternative is possible. Namely, the input may be represented using exclusively invariant features, which any un-constrained neural network would process into an invariant output. For example, an SO(3)-invariant representation of 3 points may be constituted by their relative distances and angles. This approach is widely used for representing atomic point clouds due to its implementation and conceptual simplicity [52, 207]. However, its success relies upon constructing a representation that contains all possible invariant information that is relevant for the task, avoiding ambiguities. For example, considering only relative distances in the above example of 3 points would ignore key information (e.g., the angles) that is required to represent the 3 points unambiguously. Instead, using an equivariant neural network to learn an invariant output allows for the effective learning of invariant features from the raw input, maximizing expressivity, and is recommended in [30]. Last, in the specific domain of predicting quantum properties of atomic systems, it has been empirically observed that equivariant neural networks have more favorable scaling laws than invariant networks [16].

Chapter 3

H-(V)AE: HOLOGRAPHIC (VARIATIONAL) AUTOENCODER

The contents of this chapter are published in Visani et al. [240], under the title *Holographic-(V)AE: An end-to-end $SO(3)$ -equivariant (variational) autoencoder in Fourier space*.

The appeal of group-equivariant neural networks is straightforward in the supervised learning context, in which the objective is to fit a function to provided input-output pairs that may contain known group symmetries. Indeed, neural networks equivariant to $SO(3)$ and other euclidean transformations have advanced the state-of-the-art in many supervised tasks [251, 230, 72, 27, 16, 161, 203, 129].

In conceiving this work, we hypothesized that applying the group-equivariant paradigm to unsupervised learning could map out compact representations of data that are agnostic to a specified symmetry transformation. In the world of unsupervised learning, auto-encoders (AE's) - and their probabilistic version, variational auto-encoders (VAE's) - are among the most popular paradigms, as they can learn robust and compact representations of high-dimensional data [114].

Therefore, in this chapter we apply our $SO(3)$ -equivariant modeling framework to the domain of autoencoders and develop Holographic-(V)AE (H-(V)AE), an end-to-end $SO(3)$ -equivariant (variational) autoencoder trained to reconstruct the holographic encoding of data. H-(V)AE learns an $SO(3)$ -equivariant latent space composed of a *maximally informative* set of invariants and an equivariant frame describing the orientation of the data.

We extensively test the performance and properties of H-(V)AE in two domains. First, we focus on spherical images, demonstrating high accuracy in unsupervised classification and clustering tasks. Second, we focus on proteins, and demonstrate that H-(V)AE can be effectively used to construct compact, informative, and symmetry-aware representations of protein structures, which can be used for downstream tasks. Specifically, we leverage H-(V)AE trained on a large corpus of protein structure neighborhoods to construct local representations of protein-ligand binding pockets that are both rotationally and translationally equivariant (i.e., $SE(3)$ equivariant). When combined with a simple Random Forest Regressor, we achieve state-of-the-art accuracy on the task of predicting the binding affinity between a protein and a ligand in complex. Code and pre-trained models are freely available at <https://github.com/gvisani/Holographic-VAE>.

This work can be placed within an increasing literature of group-equivariant representation and unsupervised learning.

Disentangled from neural networks and data-driven learning, there is a diverse body of literature on constructing representations of atomic systems that are invariant or equivariant to euclidean symmetries, leveraging spherical fourier transforms and CG tensor products [62, 162]. Notably, Uhrin [234] constructs $SO(3)$ -invariant representations of point clouds using the ZFT and CG tensor product iterations. H-(V)AE can be seen as a data-driven instance of this framework.

On the data-driven side, several works attempt to learn representations that are invariant to certain classes of transformations, but avoid equivariant neural networks. Shu et al. [215] and Koneripalli et al. [118] learn general "shape" embeddings by learning a separate "deformation" embedding. However, their networks are not explicitly equivariant to the transformations of interest. Other work proposes to learn an exactly invariant embedding alongside an approximate (but not equivariant) group action to align the input and the reconstructed data. For example, Mehr et al. [149] learns in quotient space by sampling the group's orbit

and computing the reconstruction loss by taking the infimum over the group. This approach is best suited for discrete and finite groups, and it is computationally expensive as it is akin to data augmentation. Lohit and Trivedi [137] construct an $SO(3)$ -invariant autoencoder for spherical signals by learning an invariant latent space and minimizing a loss which first finds the rotation that best aligns the true and reconstructed signals, introducing an added optimization step - potentially very expensive for 3D data - and reconstructing the signals only up to a global rotation.

A small body of work, which H-(V)AE is most closely related to, focuses on developing equivariant autoencoders. Several methods construct data and group-specific architectures to auto-encode data equivariantly, learning an equivariant representation in the process [92, 119]. Others use supervision to extract class-invariant and class-equivariant representations [69]. A concurrent theoretical work proposed to train an encoder that encodes elements into an invariant embedding and an equivariant group action, then using a standard decoder that uses the invariants to reconstruct the elements in a canonical form, and finally applying the learned group action to recover the data’s original form [252]. Our method in $SO(3)$ is closely related to this work, with the crucial difference that we use an equivariant decoder and that we learn to reconstruct the Fourier encoding of the data. A more detailed comparison of the two approaches and the benefits of our fully equivariant approach can be found in the Appendix (Sec. A.2) and in Table A.2.1.

3.1 Methods

Network architecture and training. H-(V)AE consists of an encoder that, through learned linear projections and pre-set nonlinear operations, project the data onto a compressed rotationally equivariant latent space. A trained decoder that is similarly constructed then takes this latent projection and reconstructs the input data. The combination of learned linear and pre-set nonlinear operations form equivariant Clebsch-Gordan blocks (CG bl.) both for the encoder and the decoder; see Figure 3.1 and below for details on the structure of a Clebsch-Gordan block.

Using the Clebsch-Gordan blocks, we construct a fully rotationally equivariant architecture for unsupervised learning. Specifically, the encoder takes as input a steerable tensor with maximum degree $\ell_{\max}$ and, via a stack of Clebsch-Gordan blocks, iteratively and equivariantly transfers information from higher degrees to lower ones, down to the final encoder layer with $\ell_{\max} = 1$, resulting in the invariant ($\ell = 0$) and the frame-defining equivariant ($\ell = 1$) embeddings. The frame is constructed by learning two vectors from the $\ell = 1$ embedding in the final layer of the encoder and using Gram-Schmidt to find the corresponding orthonormal basis [205]. The third orthonormal basis vector is then calculated as the cross product of the first two.

The decoder learns to reconstruct the input from the invariant ($\ell = 0$) embedding of the encoder’s final layer $\mathbf{z}$ and the frame, iteratively increasing the maximum degree $\ell_{\max}$ of the intermediate representations by leveraging the CG Tensor Product within the Clebsch-Gordan blocks (Figure 3.1C). To add stochasticity and make the model variational (i.e., constructing H-VAE as opposed to H-AE), we parameterize the *invariant* part of the latent space by an isotropic Gaussian, i.e., we learn two sets of size z invariants, corresponding to means and standard deviations.

We train H-(V)AE to minimize the reconstruction loss $\mathcal{L}_{rec}$ between the original and the reconstructed tensors, and, for H-VAE only, to minimize the Kullback-Leibler divergence D_{KL} of the posterior invariant latent space distribution $q(\mathbf{z}|\mathbf{x})$ from the selected prior $p(\mathbf{z})$ [114]:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \alpha \mathcal{L}_{rec}(\mathbf{x}, \mathbf{x}') + \beta D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \quad (3.1)$$

We use mean square error (MSE) for $\mathcal{L}_{\text{rec}}$, which as we show in Section A.1.4, respects the necessary property of $\text{SO}(3)$ pairwise invariance, ensuring that the model remains rotationally equivariant. We cast an isotropic normal prior to the invariant latent space: $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Hyperparameters α and β control the trade-off between reconstruction accuracy and latent space regularization [90]; see Section A.1.1 for details on tuning of these rates during training.

As a result of this training, H-(V)AE learns a *disentangled* latent space consisting of a maximally informative invariant ($\ell = 0$) component $\mathbf{z}$ of arbitrary size, as well as three orthonormal vectors ($\ell = 1$), which represent the global 3D orientation of the object and reflect the *coordinate frame* of the input tensor. Crucially, the disentangled nature of the latent space is respected at all stages of training, and is guaranteed by the model’s rotational equivariance; see Figure 3.3 for a visual example. We discuss the equivariant properties of H-(V)AE in more detail in Figure A.4.20, and empirically verify the equivariance of our model up to numerical errors in Table A.4.7.

Invariant Conditioning. Since all of a datapoint’s invariant information is compressed in one latent embedding $\mathbf{z}$, it is trivial to condition the latent space with invariant properties of the data. This process can be used to exclude the properties’ information from the latent embedding, thereby allowing the model to focus on more nuanced characteristics, or to allow for conditional generation if training with a variational objective [217]. In our model, we apply conditioning by first linearly projecting the conditioning so that its dimensionality matches that of $\mathbf{z}$, then adding it to $\mathbf{z}$ before and feeding the result into the decoder.

Architecture of a Clebsch-Gordan block. Each Clebsch-Gordan block (CG bl.) consists of a trained linear layer (*Linearity* (Lin)), an efficient tensor product (ETP) to inject nonlinearity in the network, and normalization steps by *Batch Norm* (BN) and *Signal Norm* (SN) (see Section 2.3). We found using batch norm alone often caused activations to explode in the decoder during inference, and introduced signal norm to counteract this effect. While training with signal norm alone is stable, we found keeping both norms to speed up convergence in preliminary experiments (Fig. A.4.1).

3.2 Results

3.2.1 Rotated MNIST on the sphere

We extensively test the performance of H-(V)AE on the MNIST-on-the-sphere dataset [57]. Following ref. [46], we project the MNIST dataset, which includes images of handwritten numbers, onto the lower hemisphere of a discrete unit sphere. We consider two variants of training/test set splits, NR/R and R/R, differing in whether the training/test images have been randomly rotated (R) or not (NR). For each dataset, we map the images to steerable tensors via the Zernike Fourier Transform (ZFT) in Eq. 2.8 and train models with different different sizes of latent spaces ($z = 16$ vs. $z = 120$) and model types (AE vs. VAE). In all cases the model architecture follows from Figure 3.1C.

For variational models, we tune the regularization strength β such as to maximize the expected quality of generated samples. We define samples to be of high quality if they 1) can be correctly classified by a classifier trained on real data, and 2) if they are diverse enough, finding that there is a trade-off between the two. We assess sample quality across β by generating them with conditional models and computing the accuracy of a classifier trained on real data, as well as the variance of generated samples. We settle on β values of 0.6 and 2.8 for the $z = 16$ and $z = 120$ models respectively, but also perform an ablation study and

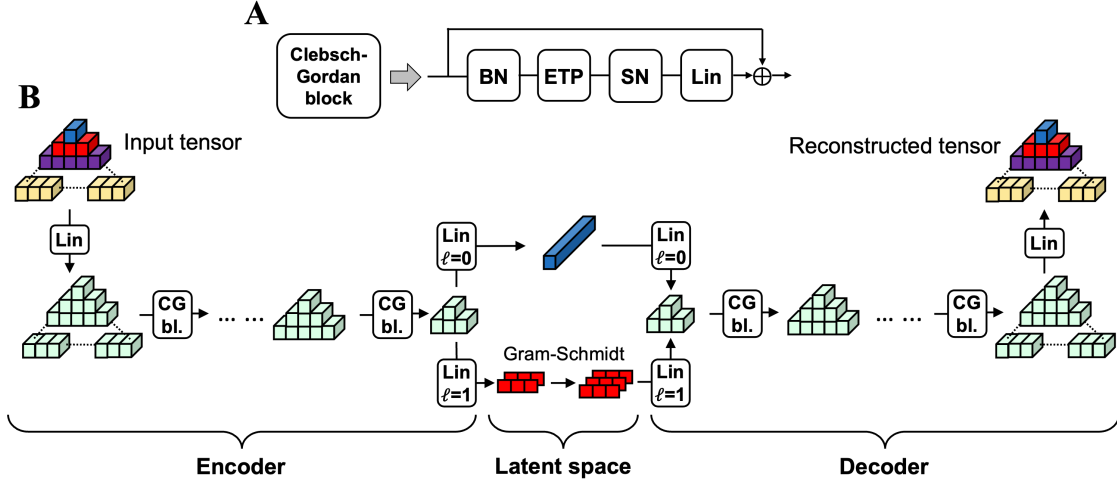


Figure 3.1: **Visualization of H-(V)AE’s architecture.**

A: Schematic of a Clebsch-Gordan Block (CG bl.), with batch norm (BN), efficient tensor product (ETP), and signal norm (SN), and Linear (Lin) operations. **B:** Schematic of the H-AE architecture. We color-code features of different degrees in the input and in the latent space for clarity. The H-VAE schematic differs only in the latent space, where two sets of invariants are learned (means and standard deviations of an isotropic Gaussian distribution).

show that the chosen values of β are near-optimal according to quantitative metrics on the quality of the latent space (Figure A.4.6). See A.3.2 for more details.

We use the metric *Cosine loss* to measure a model’s reconstruction ability. Cosine loss is a normalized dot product generalized to operate on pairs of steerable tensors (akin to cosine similarity), and modified to be interpreted as a loss. Importantly, unlike MSE, Cosine loss is dimensionless, and therefore, comparable across different datasets and encodings of data in tensors of different sizes, though it is agnostic to magnitudes and thus unfit for training; see Section A.1.5 for details.

All trained models achieve very low reconstruction Cosine loss (Table 3.1) with no significant difference between training modes, indicating that the models successfully leverage $SO(3)$ -equivariance to generalize to unseen orientations. Predictably, AE models have lower reconstruction loss than VAE models (since they do not need to find a trade-off between reconstruction error and KL divergence, Eq. 3.1), and so do models with a larger latent space. Nonetheless, H-VAE achieves reliable reconstructions, as shown in Figure 3.2A and Table 3.1.

Interpretability of the latent space is also an important feature of a model. All 8 models produce an invariant latent space that naturally clusters by digit identity. We show this qualitatively for four of the models in Fig. 3.2B and Fig. A.4.2, and quantitatively by clustering the data via K-Means with 10 centroids and computing standard clustering metrics of Purity [5] and V-measure [199] in Table 3.1.

Crucially, the built-in $SO(3)$ -equivariance enables models trained on the non-rotated images to seamlessly generalize to images that have been randomly rotated, as seen by the equivalent performance between models trained and evaluated on the NR/R and R/R splits.

All trained models achieve much better clustering metrics than Rot-Inv AE [137], with the VAE models consistently outperforming the AE models. We also train a linear classifier (LC) to predict digit identity from invariant latent space descriptors, achieving better accuracy to Rot-Inv AE with the same latent space size. We further observe marginal improvements of VAE over AE models in terms of classification accuracy. Using

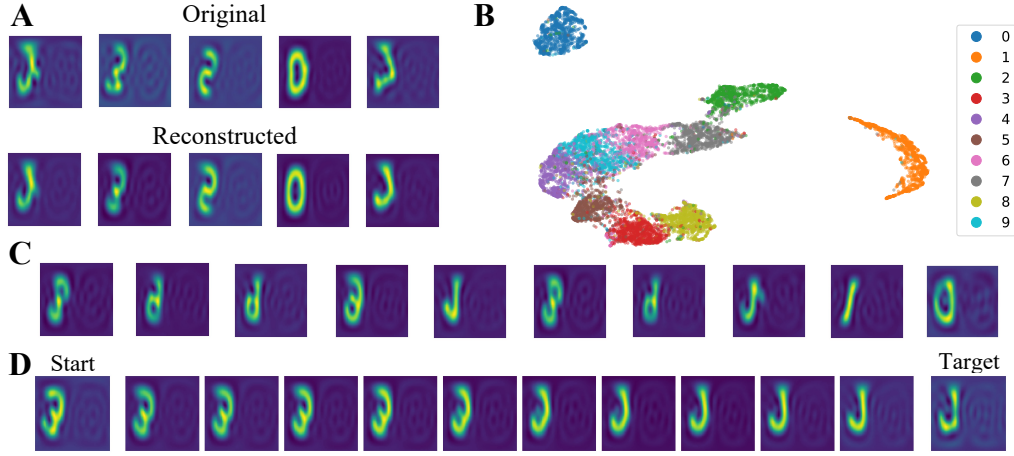


Figure 3.2: **H-VAE on MNIST-on-the-sphere.**

Evaluation on rotated digits for an H-VAE trained on non-rotated digits with $z = 16$. **(A)** Original and reconstructed images in the canonical frame after inverse transform from Fourier space. The images are projected onto a plane. Distortions at the edges and flipping are side-effects of the projection. **(B)** visualization of the latent space via 2D UMAP [146]. Data points are colored by digit identity. **(C)** Random images generated by feeding the decoder invariant embeddings sampled from the prior distribution and the canonical frame. See more examples in Figure A.4.11. **(D)** Example image trajectory by linearly interpolating through the learned invariant latent space. Interpolated invariant embeddings are fed to the decoder alongside the canonical frame. See more examples in Figure A.4.18. MNIST-on-the-sphere dataset is created by projecting data from the planar MNIST on a discrete unit sphere, using the Driscoll-Healey (DH) method with a bandwidth (bw) of 30 [46].

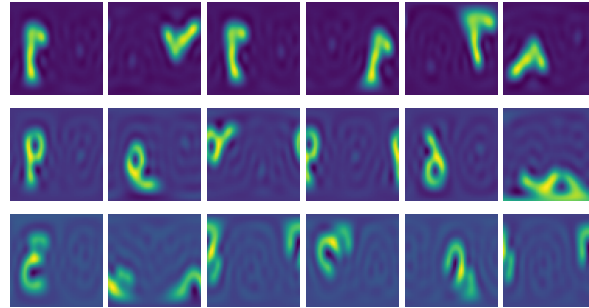


Figure 3.3: **Visual evidence of the disentanglement in the latent space of MNIST-on-the-sphere.**

For each row, the invariant embedding $\mathbf{z}$ is held fixed, and a different frame (i.e., the rotation matrix) is used. Frames are sampled randomly and differ across rows, with the exception of the first column, where the frame is the same. Then, $\mathbf{z}$ and the frame are fed to the decoder and the Inverse Fourier Transform is used to generate the reconstructed spherical image, which is projected onto a plane for the ease of visualization. Modulo the distortions given by projecting the image onto a plane, it is clear that the invariant embedding contains all semantic information, and the frame solely determines the orientation of the image.

Dataset	Type	Method	z	bw	Cosine	Purity	V-meas.	Class. Acc.	P@N	R@N	F1@N	mAP	NDCG
MNIST	Supervised	[44] NR/R	-	30	-	-	-	0.993	-	-	-	-	-
	Unsupervised	[137] NR/R	120	30	-	0.40	0.35	0.894	-	-	-	-	-
		H-AE NR/R (Ours)	120	30	0.017	0.59	0.51	0.920	-	-	-	-	-
		H-AE R/R (Ours)	120	30	0.015	0.65	0.52	0.916	-	-	-	-	-
		H-AE NR/R (Ours)	16	30	0.031	0.66	0.55	0.850	-	-	-	-	-
		H-AE R/R (Ours)	16	30	0.030	0.61	0.52	0.844	-	-	-	-	-
		H-VAE NR/R (Ours)	120	30	0.048	0.74	0.63	0.923	-	-	-	-	-
		H-VAE R/R (Ours)	120	30	0.050	0.73	0.60	0.916	-	-	-	-	-
		H-VAE NR/R (Ours)	16	30	0.068	0.73	0.60	0.878	-	-	-	-	-
		H-VAE R/R (Ours)	16	30	0.067	0.69	0.57	0.855	-	-	-	-	-
Shrec17	Supervised	[67]	-	128	-	-	-	-	0.717	0.737	-	0.685	-
	Unsupervised	[44]	-	128	-	-	-	-	0.719	0.710	0.708	0.679	0.758
		[137]	120	30	-	0.41	0.34	0.578	0.351	0.361	0.335	0.215	0.345
		H-AE (Ours)	40	90	0.130	0.50	0.41	0.654	0.548	0.569	0.545	0.500	0.597
		H-VAE (Ours)	40	90	0.151	0.52	0.42	0.631	0.512	0.537	0.512	0.463	0.568

Table 3.1: **Performance metrics of H-(V)AE on MNIST-on-the-sphere and Shrec17.** Reconstruction Cosine loss, clustering metrics (Purity and V-measure), classification accuracy in the latent space using a linear classifier, and retrieval metrics (Shrec17 only) are shown. For MNIST, we follow ref. [46] to create the MNIST-on-the-sphere dataset by projecting data from the planar MNIST on a discrete unit sphere, using the Driscoll-Healey method with a bandwidth (bw) of 30. We then map the images to steerable tensors via the Zernike Fourier Transform (ZFT) with $L = 10$, and a constant radial function $R_\ell^n = 1$, resulting in tensors with 121 coefficients. We train 8 models with different sizes of latent spaces z (16 vs. 120) and model types (AE vs. VAE). For Shrec17, we follow ref. [46] and project surface information from each model onto an enclosing Driscoll-Healey spherical grid with a bandwidth of 90 via a ray-casting scheme, generating spherical images with 6 channels. We then apply the ZFT with $L = 14$ and a constant radial function $R_\ell^n = 1$ to each channel individually, resulting in a tensor with 1350 coefficients. We only report scores presented in the corresponding papers, and only the best-performing supervised method from the literature.

a K-nearest neighbor (KNN) classifier instead of LC further improves performance on the $z = 16$ models (Table A.4.1).

As H-VAE is a generative model, we generate random spherical images by sampling invariant latent embeddings from the prior distribution, and observing diversity in digit type and style (Fig. 3.2C and Fig.s A.4.11, A.4.12). We further assess the quality of the invariant latent space by generating images via linear interpolation of the invariant embeddings associated with two test images. The interpolated images present spatially consistent transitions (Fig. 3.2D and Fig.s A.4.18, A.4.19, A.4.17), which is a sign of a semantically well-structured latent space.

To understand the meaning of the learned frames, we ask ourselves what the output of the decoder looks like if the frame is held constant; for simplicity, we set it equal to the 3×3 identity matrix. We experimentally find that the reconstructed elements tend to be aligned with each other and hypothesize that the model is implicitly learning to maximize the overlap between training elements, providing empirical evidence in Figure A.4.20. We call this frame the **canonical** frame following the analogy with the canonical elements in ref. [252]. We note that it is possible to rotate original elements to the canonical frame thanks to the equivalence between the frame we learn and the rotation matrix within our implementation; in fact in our experiments, when visualizing reconstructed or sampled MNIST images, we first rotate them to the canonical frame for ease of visualization.

3.2.2 Shrec17

The Shrec17 dataset consists of 51k colorless 3D models belonging to 55 object classes, with a 70/10/20 train/valid/test split [214]. We use the variant of the dataset in which each object is randomly rotated. Converting 3D shapes into spherical images preserves topological surface information, while significantly

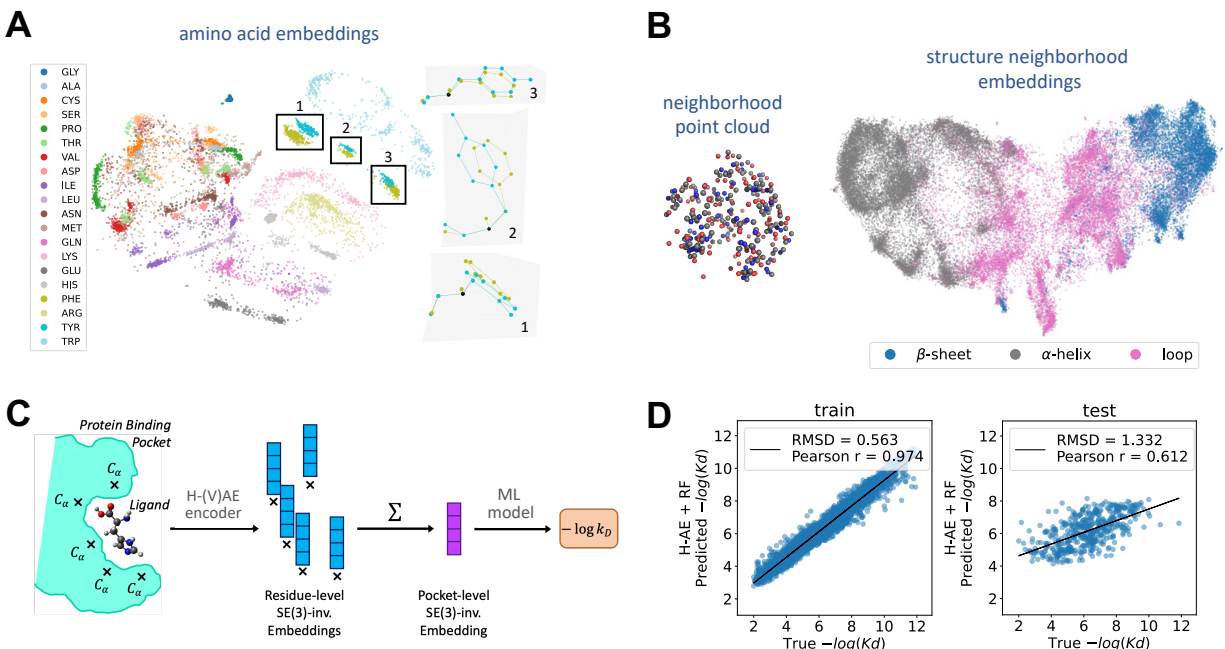


Figure 3.4: Structural embeddings to predict protein-ligand binding affinities with H-(V)AE. (A) H-VAE was trained to reconstruct the Fourier representation of 3D atomic point clouds representing amino acids (colors). The invariant latent space clusters by amino acid conformations. The highlighted clusters for PHE and TYR contain residue pairs with similar conformations; TYR and PHE differ by one oxygen at the end of their benzene rings. We compare conformations by plotting each residue in the standard backbone frame (right); x and y axes are set by the orthonormalized $C\alpha$ -N and $C\alpha$ -C vectors, and z axis is their cross product. For this plot, 1,000 amino acids were used as training data, with network parameters: $\beta = 0.025$ and $z = 2$. (B) Left: An example protein neighborhood (point cloud of atoms) of $10 \text{ \AA}$ around a central residue, used to train the H-AE models, is shown. Right: 2D UMAP visualization of the 128-dimensional invariant latent space learned by H-AE trained on the protein structure neighborhoods with $L = 6$ can separate neighborhoods by the secondary structure of their focal amino acid (colors); a linear classifier trained on 300,000 latent embeddings predicts secondary structure of the focal amino-acid with 90% accuracy. Each point represents a neighborhood. See Section A.3.5 for details on the network architecture and training procedure. (C) We use H-AE to extract the residue-level $SO(3)$ -invariant embeddings in the binding pocket of a protein-ligand structure complex (data from PDBbind [224]). We then sum over these embeddings to form an $SE(3)$ -invariant pocket embedding that is used as an input to a standard Machine Learning model to predict the binding affinity between the protein and the ligand. (D) The predictions on the protein-ligand binding affinities from (C) is shown against the true values for the training (left) and the test (right) sets. We use the data split provided by ATOM3D [231], which devises training and test sets respectively containing 3507, and 490 protein-ligand complexes, with maximum 30% sequence similarity between training and test proteins; see Table 3.2 for a comparison against state-of-the-art methods.

simplifying the representation. We follow ref. [46] and project surface information for each image onto an enclosing spherical grid via a ray-casting scheme and apply ZFT (eq. 2.8) on these transformed images.

We train an AE and a VAE model on ZFT transformed data (Section A.3.3) and, similarly to the MNIST dataset, compute cosine loss, clustering metrics and classification accuracy via a linear classifier. We also compute the standard Shrec17 object retrieval metrics via the latent space linear classifier’s predictions (see [214] for a description of the retrieval metrics). H-AE achieves the best classification and retrieval results for autoencoder-based models, and is competitive with supervised models despite the lower grid bandwidth and the small latent space (Table 3.1). Using KNN classification instead of a linear classifier further improves performance (Table A.4.2). H-VAE achieves slightly worse classification results but better clustering metrics compared to H-AE. While reconstruction loss is low, there is still significant margin of improvement.

3.2.3 Structural embeddings of amino acid neighborhoods to predict function

Here, we provide a strong use case for H-(V)AE in structural biology. Specifically, we learn expressive embeddings for amino acids neighborhoods within protein structures that can be used to learn protein function.

Embeddings of amino acid conformations. As a first, propedeutic step, we train H-(V)AE to reconstruct the 3D structure of individual amino acids, represented as atomic point clouds, extracted from protein structures in the Protein Data Bank (PDB) [18]. Residues of the same type have different conformations and naturally have noisy coordinates, making this problem a natural benchmark for our rotationally equivariant method.

We represent an amino acid by atom-type-specific clouds (C, O, N and S; we exclude H) centered at the residue’s $C\alpha$ and compute the ZFT (eq. 2.8) with $L = 4$ and $N = 20$ within a radius of $10\text{\AA}$ from the residue’s $C\alpha$, and concatenate features of the same degree ℓ , resulting in a tensor with 940 coefficients. We train several H-AE and H-VAE with different architectures, but all with the latent space sizes $z = 2$; see Sec. A.3.4 for details.

We consistently find that the latent space clusters by amino acid conformations (Figure 3.4A), with sharper cluster separations as more training data is added (Figure A.4.22 and A.4.23). We find that test reconstruction loss decreases with more training data but the reconstruction is accurate even with little training data (from 0.153 Cosine loss with 400 training residues to 0.034 with 20,000); Table A.4.3. A similar trend is observed for KNN-based classification accuracy of residues (from 0.842 with 400 training residues to 0.972 with 20,000); (Table A.4.3). Notably, an untrained model, while achieving random reconstruction loss, still produces an informative invariant latent space (0.629 residue type accuracy), suggesting that the forced $SO(3)$ -invariance grants a “warm start” to the encoder. We do not find significant improvements in latent space classification by training with a variational objective, and present ablation results in Table A.4.4.

Embeddings of amino acid structure neighborhoods. The structural neighborhood surrounding an amino acid provides a context for its function (e.g., whether it takes part in interaction with other proteins at a protein-protein interface or not). Indeed, our previous work has shown that supervised learning algorithms can accurately classify focal amino acids based on the composition of their surrounding neighborhoods [184]. Here, we train H-(V)AE to reconstruct residue-specific protein *neighborhoods* - which we define as the point clouds of atoms within $10\text{\AA}$ of a residue’s $C\alpha$ - across the protein universe (Fig. 3.4A). We extract these protein neighborhoods from all proteins in ProteinNet [8]. We then construct each neighborhood’s Fourier representation by computing the ZFT (Eq. 2.8) over the point clouds associated with each atom type within the neighborhood (C, N, O and S) and concatenating atom-specific features of the same degree ℓ .

We train several H-AE models with varying architectures with different maximum spherical degree L and latent space sizes z (see details in Sec. A.3.5); note that we do not experiment with variational models for this task. H-AE shows strong reconstruction ability, but its accuracy worsens with smaller latent space sizes and higher maximum spherical degree L (Table A.4.5). Notably, the learned latent space is smoothly structured according to geometric features of the neighborhoods, such as the presence of secondary structure components (Fig. 3.4B and A.4.25) and number of atoms (Figure A.4.26).

Predicting protein-Ligand Binding Affinity (LBA). The binding interaction between proteins and ligands should be primarily determined by the composition of the protein’s binding pocket in complex with the ligand. Therefore, we hypothesized that the inferred protein structure embeddings (Fig. 3.4B) should contain information about protein-ligand binding interactions, if the neighborhood is defined along the binding pocket and contains atoms from the ligand. To test this hypothesis, we follow the pipeline in Fig. 3.4C. Specifically, given a protein-ligand structure complex, we identify residues in the binding pocket (i.e., residues with $C\alpha$

Model	Ligand Binding Affinity		
	30% Similarity		
	RMSD ↓	Pearson’s r ↑	Spearman’s r ↑
DeepDTA	1.565	0.573	0.574
3DCNN	1.414 ± 0.021	0.550	0.553
GNN	1.570 ± 0.025	0.545	0.533
MaSIF	1.484 ± 0.018	0.467 ± 0.020	0.455 ± 0.014
EGNN	1.492 ± 0.012	0.489 ± 0.017	0.472 ± 0.008
GBPNet	1.405 ± 0.009	0.561	0.557
EGNN + PLM	1.403 ± 0.013	0.565 ± 0.016	0.544 ± 0.005
ProtMD	<u>1.367 ± 0.014</u>	<u>0.601 ± 0.036</u>	<u>0.587 ± 0.042</u>
H-AE + L.R.	1.397 ± 0.019	0.560 ± 0.017	0.568 ± 0.018
H-AE + R.F.	1.332 ± 0.012	0.612 ± 0.009	0.619 ± 0.009

Table 3.2: **Results on the Ligand Binding Affinity task.** Prediction accuracies using H-AE embeddings with Linear Regression (L.R.), and Random Forest (R.F.) Regression are benchmarked against other methods. We choose the H-AE model with best RMSD on validation split, which is the model with $L = 6$ and $z = 128$ for both Linear Regression and Random Forest. For each set of predictions, we use an ensemble of 10 Regressors as we noted a small but consistent improvement in performance. Best scores are in **bold** and second-best scores are underlined. H-AE+R.F. delivers state-of-the-art predictions. Methods are ordered by date of release; see Table A.4.8 for more details.

within 10Å of the ligand) and extract their structure neighborhoods, which include atoms from both the protein and the ligand. We then pass the residue-centered neighborhoods of the binding pocket through a trained H-AE’s encoder to extract their rotationally-invariant embeddings. We highlight that, since the neighborhood centers are well-defined at the residues’ C- α ’s, the embeddings are not only rotationally invariant about their center, but also translationally invariant with the respect to translations of the whole system, i.e., they are effectively SE(3)-invariant.

Since protein-ligand binding affinity is an extensive quantity in the number of interacting residues, we construct a pocket embedding by summing over residue-level embeddings; the resulting pocket embedding is SE(3)-invariant, reflecting the natural symmetry of the LBA task. We use these pocket embeddings as feature vectors to train simple Machine Learning models to predict protein-ligand binding affinities.

To test the performance of our method, we use the LBA dataset in ATOM3D [231] that provides the PDB structure of the protein-ligand complex together with either the measured dissociation constant K_d or the inhibition constant K_i ; see Sec. A.4 for further details. To map between the learned pocket embeddings to the log dissociation (or inhibition) constants we train both a simple Linear and a Random Forest Regressor on a training sets provided by ATOM3D. Fig. 3.4D shows the performance of the model with the Random Forest Regressor and Table 3.2 provide a detailed benchmark of our methods against prior approaches. The Linear model achieves competitive results, whereas the Random Forest Regressor achieves state-of-the-art.

These results demonstrate the utility of unsupervised learning for residue-level protein structure representations in predicting complex protein functions. Most of the competing structure-based methods for LBA (Table 3.2) learn complex graph-based functions on top of simple atomic representation, whereas our method uses simpler machine learning models over rich residue-level representations. Notably, ProtMD (the method competitive to ours) performs a pre-training scheme using expensive molecular dynamics simulations that informs the model about conformational flexibility, information that our method does not have access to. Given the computational cost of training complex atom-level graph-based models, our residue-based approach can offer a more viable alternative for modeling large protein interfaces.

3.3 Summary

In this chapter, we have developed the first end-to-end $SO(3)$ -equivariant unsupervised algorithm, termed Holographic (Variational) Auto-Encoder (H-(V)AE), suitable for data distributed in three dimensions around a given central point. The model learns an invariant embedding describing the data in a “canonical” orientation alongside an equivariant frame describing the data’s original orientation relative to the canonical one.

Prior works have attempted to learn representations that are invariant to certain transformations. For example, in refs. [215, 118] general “shape” embeddings are learned by characterizing a separate “deformation” embedding. However, these networks are not explicitly equivariant to the transformations of interest.

Others proposed to learn an exactly invariant embedding alongside an approximate (but not equivariant) group action to align the input and the reconstructed data. For example, *Mehr et al* [149] learns in quotient space by pooling together the latent encodings of copies of the data that have been transformed with sampled group actions, and backpropagate the minimum reconstruction loss between the decoded element and the transformed copies of the data. This approach is best suited for discrete and finite groups, for which it does not require approximations, and it is computationally expensive as it is akin to data augmentation. *Lohit et al* [137] construct an $SO(3)$ -invariant autoencoder for spherical signals by learning an invariant latent space and minimizing a loss which first finds the rotation that best aligns the true and reconstructed signals. Although this approach is effective for non-discrete data, it still manually imposes rotational invariance, and can only reconstruct signals up to a global rotation. In contrast, H-(V)AE is fully equivariant and only requires simple MSE for reconstruction of data in its original orientation.

A small body of work went beyond invariance to develop equivariant autoencoders. Several methods construct data and group-specific architectures to auto-encode data equivariantly, learning an equivariant representation in the process [92, 119]. Others use supervision to extract class-invariant and class-equivariant representations [69]. A recent theoretical work proposes to train an encoder that encodes elements into an invariant embedding and an equivariant group action, then using a standard decoder that uses the invariants to reconstruct the elements in a canonical form, and finally applying the learned group action to recover the data’s original form [252]. Our method in $SO(3)$ is closely related to this work, with the crucial difference that our network is end-to-end rotationally equivariant, in that we use an equivariant decoder and learn to reconstruct the Fourier encoding of the data. A more detailed comparison of the two approaches and the benefits of our fully equivariant approach can be found in the Appendix (Section A.2) and in Table A.2.1. Recently, generative models of atomic point clouds that are equivariant to Euclidean transformation have been developed for molecular [94] and protein [259, 247] generation in 3D using Normalizing Flows [94] and diffusion processes [203, 259, 247]. Our work is partly related to these, but with some crucial differences. 1) Our method is designed for spherical images and for point clouds with a preferential center, as opposed to point clouds. 2) The use of an autoencoder architecture makes our method suitable for compressing information into a lower-dimensional space, for effective later use in annotation-scarce domains like ligand binding affinity prediction; by contrast the latent space of normalizing flows and diffusion-based models must have the same size as the data, making them unsuitable for this task.

There is also a diverse body of literature on constructing representations of atomic systems that are invariant/equivariant to euclidean symmetries, leveraging Fourier transforms and CG tensor products, with focused applications on materials science [62, 162, 234]

H-(V)AE’s learned embeddings are highly expressive. For example, we used the learned invariants to achieve state-of-the-art unsupervised clustering and classification results on various spherical image datasets. By making our model variational in its invariant latent space, we enhanced the quality of clustering and made

the model generative. Our model is defined fully in spherical Fourier space, and thus, can reach a desired expressiveness without a need for excessive computational resources.

H-(V)AE also produces rich residue-level representations of local neighborhoods in protein structures, which we use as embeddings for downstream structure-based tasks such as Ligand Binding Affinity prediction. Indeed, H-(V)AE representations paired with a simple Random Forest Regressor achieve state-of-the-art results on learning the binding affinity between proteins and small molecule ligands.

More broadly, we expect that H-(V)AE can be used to extract rich, symmetry-aware features from local neighborhoods in spherical images and complex 3D objects, to be used in more complex downstream tasks that benefit from the symmetry constraints. For example, our method can be leveraged for modeling diffusion MRI data, for which rotation-equivariant methods have recently shown to be highly beneficial [160]. In structural biology, we expect our method to be useful for coarse-graining full-atom representations of protein structures to facilitate structure-based predictions of protein function. For example, a large protein graph can be coarse-grained by substituting its full-atom representation with rich embeddings of local structural neighborhoods learned from an unsupervised model. With an added supervised step, these coarse-grained embedding can be leveraged to predict complex protein functions, as we show for predicting ligand binding affinity. This approach is akin to using protein embeddings for sequence data, learned by language models, to inform (few-shot) predictions for protein function [225].

Chapter 4

H-PACKER: HOLOGRAPHIC PACKER

The contents of this chapter are published in Visani et al. [237], under the title *H-Packer: Holographic Rotationally Equivariant Convolutional Neural Network for Protein Side-Chain Packing*.

Determining amino acid side-chain conformations in a protein, known as Protein Side-Chain Packing (or Rotamer Packing), is an essential step in protein folding and the de-novo design of proteins. Computational approaches to protein folding often divide the structure inference problem into two steps: first, they characterize the rigid backbone structure, and then they pack the side-chains associated with the amino acids at each residue. The flexibility of the side-chain makes the search in the space of possible conformations inevitably complex and computationally expensive. The de-novo protein design protocols also rely on similar logical steps: often an amino acid sequence compatible with a desirable backbone structure is to be inferred (designed) [52] and then the associated side-chains should be packed to form the full atomic composition of a protein. Furthermore, it has been suggested that models trained to learn *distributions* of side-chain conformations can be used to predict mutational effects in a zero-shot fashion [144], thereby making rotamer packing a valuable proxy problem for resolving structure-to-function maps.

Many of the conventional methods for side-chain packing rely on physical models through which they find a rotamer that minimizes a physically-reasoned heuristic energy of the protein fold [7, 100, 120, 37, 128]. Popular algorithms include RosettaPacker from the rosetta suite [7], FASPR [100], and SCWRL [120]. However, these computational methods often lack accuracy and speed in their predictions. As deep learning makes strides in protein science, there is a growing effort in developing machine learning methods for rotamer packing. Among these methods is DLPacker [157], which treats the packing problem as an image transformation. This algorithm characterizes the local environment of a given amino acid backbone within a structure as a 3D image, and uses this model to predict the atomic coordinates of the side chain. It then compares the predicted side-chain to a pre-set library of rotamers to select the closest conformation. AttnPacker [148], a more recently developed method, uses a deep graph attention network to model the local geometry of a residue within a structure and is trained to predict the coordinates of the side-chain atoms. Recently, diffusion models over side-chain torsional angles have also been applied, such as DiffPack [262].

The specific conformation - or the ensemble of conformations - adopted by a side-chain is fundamentally determined by the physical forces acting upon it, among which those coming from local sources are the most prominent. Therefore, modeling a side-chain’s conformation as a function of its local atomic environment is a physics-appropriate approximation. Furthermore, the conformation can be described unambiguously by SO(3)-invariant or SO(3)-equivariant variables. Thus, side-chain packing fits nicely within our framework of rotationally equivariant modeling of residue-centered protein neighborhoods.

Here, we tackle the problem of side-chain packing by learning to directly regress over χ (torsional) angles, which are main degrees of freedom determining side-chain conformations. We derive and discuss three possible parameterizations of the χ angles, ultimately settling on regressing over the Sine and Cosine transforms of the angles, which are rotationally invariant. We introduce Holographic-Packer (H-Packer), a deep learning method that packs rotamers by first predicting candidate χ angles from backbone and sequence, and then refines the predictions with a model trained on full-atom structures.

By directly predicting the side-chain χ -angles, H-packer does not rely on comparing its output with a pre-set library of rotamers, making it computationally more efficient than methods like DLPacker. Furthermore,

H-Packer is light-weight (2×3M parameters vs. 208M of AttnPacker) and requires few resources to train (single vs. multiple GPUs for diffusion models like DiffPack). We evaluate the packing performance of H-Packer on standard datasets, and show that it has generally better performance than conventional physics-based methods, and competitive against machine learning solutions. In general, our results suggest that H-Packer has learned complementary features to alternative methods. Code for H-Packer is freely available at <https://github.com/gvisani/hpacker>.

4.1 Methods

4.1.1 Modeling side-chain conformations with χ Angles

While amino acids are composed of a maximum of 10 heavy atoms (in the case of Tryptophan), their 3D conformations can be uniquely described by the value of at most 4 dihedral angles, referred to as the χ angles (Figure 4.1A). This reduction in the number of degrees of freedom is granted due to the physical constraints posed on the remaining internal coordinates (bond angles, bond lengths, and dihedral angles - *redundant internal coordinates*). Specifically, the inter-atomic physical interactions within amino acids often constrain these redundant coordinates to a constant, or a well-defined function of the residue’s χ angles. Therefore, predicting χ angles is the key step in side-chain packing.

H-packer addresses the side-chain packing problem in two steps: (i) it predicts the value of χ angles, and (ii) it reconstructs the atomic coordinates using the predicted χ angles and the constrained values of the redundant internal coordinates. Specifically, we evaluate the redundant internal coordinates from a subset of training data (1,700 structures) by leveraging the `internal_coords` feature of the `biopython` package. We empirically verified that the distributions of values were Gaussian with low variance, and resolved to take their *medians* as the ground truth. Substituting these values for the original ones yields a negligible Null Reconstruction error of approximately 0.127Å (Figure B.2.1). Notably, this error remains unchanged even when using only 100 reference structures instead of 1,700 (Figure B.2.2).

4.1.2 Predicting χ angles using H-Packer

We aim to predict the χ angles associated with a side-chain conformation from the configuration of atoms surrounding a given residue. This atomic neighborhood is associated with the backbone and the side-chain of the neighboring residues in the structure.

During inference only the coordinates of the backbone atoms are known a priori - alongside the identity of the amino acids they belong to. However, physical interactions with the atoms of neighboring side-chains are the true determinants of a residue’s conformation. Therefore, we develop H-Packer into a two-step solution (Figure 4.1C). Specifically, we train two models: one to predict χ angles from the backbone atoms and amino-acid identity alone, the other to predict χ angles from full neighborhoods, i.e., by including the true side-chain atoms of the surrounding residues (minus the residue of interest). At inference time, we use the first model to make an *initial guess* of the side-chain conformations, and then a second model to iteratively *refine* the predictions.

To build the individual models that predict χ angles, we start by considering their symmetries. Notably, χ angles are *invariant* to rigid-body transformations (translations and rotations) of the protein (i.e., they are SE(3) invariant). Translation invariance can be satisfied by choosing a well-defined center for a residue of interest (e.g. the residue’s $C\alpha$), leaving rotational (i.e. SO(3) invariance) about the chosen center to be accounted for. The rotational symmetry and the dependence on local interaction make this problem fall neatly within our framework of SO(3)-equivariant modeling of protein neighborhoods. Thus we build two SO(3)-equivariant models - using the components described in Chapter 2 - to predict a residue’s χ angles,

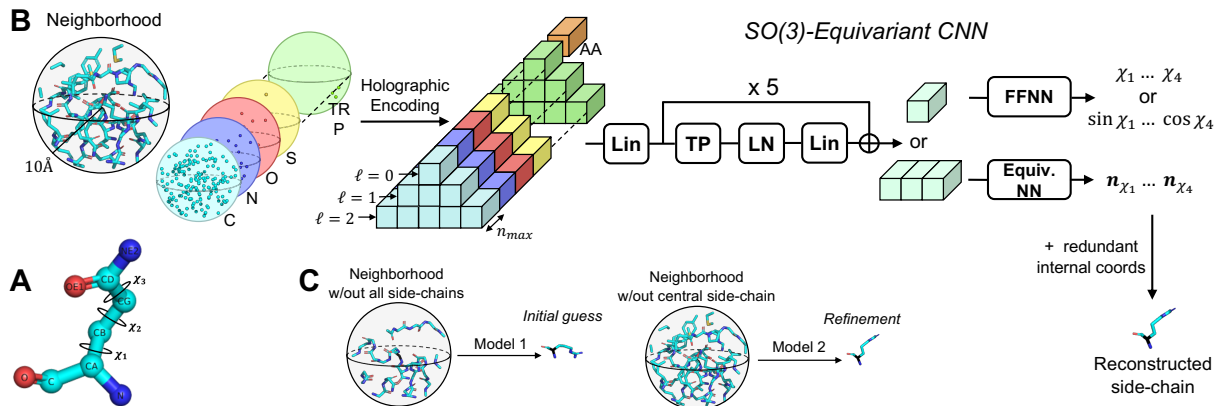


Figure 4.1: Overview of H-Packer.

A: Illustration of Glutamine’s χ angles, of which there are three. **B:** Schematic shows the $SO(3)$ -Equivariant network for side-chain packing by first predicting the missing residue’s χ angles from its surrounding atomic environment, and then using the χ angles to reconstruct the residue’s side-chain. As illustrated in **C**, H-Packer consists of two networks, one trained on backbone atoms only and used to make an initial guess, and one trained on full side-chain neighborhoods and used to refine the predictions.

considering as input the point cloud of atoms within a radius $r = 10$ of the residue’s C- α (with or without the neighboring side-chains).

Holographic Encoding. We encode neighborhoods into holograms as described in Sec. 2.2. We incorporate atom-level input features by dividing the holographic encoding into different *channels* (see Figure 4.1). We consider the following two sets: (i) **Atomic channels:** C, N, O, S, wildcard element excluding hydrogens, partial charge from the Amber99sb force field [180], and (ii) **Amino-Acid channels:** one for each of the 20 canonical amino-acids, plus a wildcard channel. We include the charge value in its dedicated channel as the weights ω_i coupled to the point cloud’s density function. While we train the *initial guess* model using both sets of channels (atomic and amino-acid) as input, we only consider the atomic channels for the *refinement* model. We do this in an effort to make the model’s predictions more grounded in physical interactions. We condition both models with the identity of the residue of interest by concatenating a linear embedding of its one-hot encoding to the input’s invariant ($\ell = 0$) features. This is particularly necessary for the refinement model - which is trained only with atomic channels - since it wouldn’t otherwise know about the identity of the residue of interest.

$SO(3)$ -Equivariant neural network architecture. We use the resulting holograms as inputs to an $SO(3)$ -Equivariant Convolutional Neural Network (Figure 4.1B). Our model is conceptually divided into three parts: **First**, a linear layer that projects data and conditioning to a hidden representation with same number of features per ℓ .

Second, a stack of equivariant blocks connected via additive skip connections, each composed of: (i) feature-wise tensor product nonlinearity, (ii) layer norm with silu nonlinearity, and (iii) a linear layer whose output dimensions are the same as the input’s (see Sec. 2.3). After the final block, we retain only the features of type $\ell = 0$ or $\ell = 1$ depending on the training objective (Section 4.1.3). It should be noted that features of type $\ell = 0$ are rotationally invariant scalars, whereas those associated with $\ell = 1$ are equivariant vectors that transform consistently with the input under rotation. We use $\ell = 1$ features to directly learn the orientation of the intersecting planes that define a side-chain’s dihedral angles χ (see Section 4.1.3).

Third, optionally and only for the models with invariant ($\ell = 0$ output), we apply a standard feed-forward neural network with dropout regularization and SiLU nonlinearity.

4.1.3 Training objectives to infer χ angles

We consider three alternative parameterizations of χ angles, i.e. three possible objective functions:

(i) The angle itself. χ angles are defined between -180° and 180° with a periodicity such that the angles -179° and 179° are to be considered 2° apart, not 358° . Thus, plain MSE loss would pose strong and unnatural constraints on the model. To account for this, we mod the predictions to fall in the valid range, and compute the loss between two angles as the minimum between the computed error and 360° minus the error, resulting in the following loss function:

$$\mathcal{L}_{\text{angles}}(\{\hat{\chi}_i\}_{i=1}^{N_\chi}, \{\chi_i\}_{i=1}^{N_\chi}) = \frac{1}{N_\chi} \sum_{i=1}^{N_\chi} \min(E_{\chi_i}, 2\pi - E_{\chi_i}) \quad \text{where} \quad E_{\chi_i} = (\text{mod}(\hat{\chi}_i, 2\pi) - \chi_i)^2 \quad (4.1)$$

where $\hat{\chi}_i$ and χ_i are the predicted and the true values of the i^{th} χ , respectively, and N_χ is the number of χ angles associated with the residue of interest. In our implementation, the χ angle domain is scaled and shifted to fall in $[0, 2]$ to make the scale of the loss functions comparable between the three representations of the angles.

(ii) Sine and Cosine transforms of the angle. A pair of sine and cosine transformation provides an alternative representation for a χ angle that accounts for its periodicity and is also rotationally invariant; a similar approach is also considered in concurrent work [159]. We directly predict sine and cosine values by feeding 8 outputs from the network to a tanh activation function, which then form the arguments of a MSE loss function:

$$\mathcal{L}_{\text{sin-cos}}(\{\hat{\chi}_i\}_{i=1}^{N_\chi}, \{\chi_i\}_{i=1}^{N_\chi}) = \frac{1}{2N_\chi} \sum_{i=1}^{N_\chi} (\cos \hat{\chi}_i - \cos \chi_i)^2 + (\sin \hat{\chi}_i - \sin \chi_i)^2 \quad (4.2)$$

Notably, this loss function is justified by a nice geometric interpretation, whereby it is equivalent to computing the cosine loss between the 2D vectors that describe the χ angles on the unit circle (proof in Eq. B.1.1).

(iii) Normal vectors to the dihedral plane. χ angles are examples of dihedral angles, meaning that they are defined as the angle between two planes. For χ angles, the two planes are described by subsequent triplets of atoms along the side-chains. Any two subsequent χ angles share one plane. Therefore, any conformation with N_χ angles can be alternatively described by $N_\chi + 1$ planes (or their normal vectors); one of these normal vectors is a redundant internal coordinate (defined by backbone + C β atoms), while others specify the N_χ independent degrees of freedom.

We consider training models to predict the dihedral planes' normal vectors: $\mathbf{n}_{\chi_1} \dots \mathbf{n}_{\chi_4}$. It should be noted that unlike the sine/cosine transformation, the vectors are not invariant to rotations, but *equivariant* of type $\ell = 1$ (geometric vectors) which can be extracted from the H-Packer equivariant network. We use a cosine loss over the true and predicted vectors:

$$\mathcal{L}_{\text{norms}}(\{\hat{\mathbf{n}}_{\chi_i}\}_{i=1}^{N_\chi}, \{\mathbf{n}_{\chi_i}\}_{i=1}^{N_\chi}) = \frac{1}{N_\chi} \sum_{i=1}^{N_\chi} 1 - \langle \hat{\mathbf{n}}_{\chi_i}, \mathbf{n}_{\chi_i} \rangle \quad (4.3)$$

Relevant symmetries in computing loss functions. Some amino acid conformations exhibit a rotation symmetry by π in some of their χ angles. For example, χ_2 of Phenylalanine and Tyrosine indicates the torsion of their benzene rings, thus a rotation by π leaves the conformation physically unchanged. However, as χ angles are formally defined by internal atom names, these equivalent conformations are associated with different χ angle values. We correct for this degeneracy by considering the minimum loss value between considering χ and $\pi - \chi$ as targets during training and evaluation. When computing the error on the atomic coordinates (generally via Root Mean Square Deviation, RMSD) for the full side-chain, we need to consider other such symmetries between non- χ atoms, as listed in Table B.2.1.

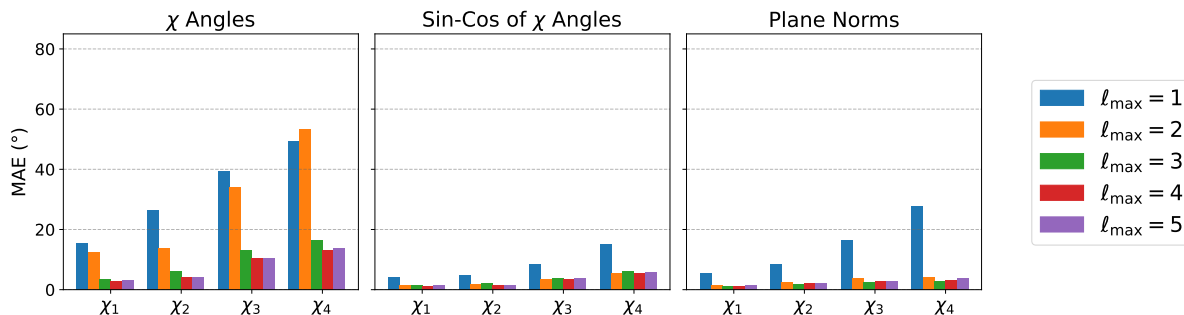


Figure 4.2: **Test MAE for the simple task of predicting χ angles from atomic conformation.** Panels show reconstruction accuracies using three loss functions: the angle χ itself (left), the sin/cos transform of the angle (center), and the normal vectors to the dihedral planes (right), for different maximum angular degrees $\ell_{\max}$ (colors).

4.2 Results

4.2.1 Toy task: inferring χ angles from atomic coordinates

We start by studying the behavior of our model on a simple task: predicting (or rather, calculating) χ angles from the true atomic coordinates of the conformation. We found this to be a useful benchmark to study our model’s behavior.

Setup. We randomly select 160 structures from our real task’s training set (see below) and split them into 100/30/30 for training/validation/testing, respectively. We collect conformations of all residues presenting χ angles, and consider only their heavy atoms (C, N, O, S). We then apply the Zernike encoding varying $\ell_{\max}$ from 1 to 5 and train models with varying $\ell_{\max}$ consistent with that of the input, as well as with different prediction objectives (angles, sin-cos of angles, plane norms). Crucially, we vary the number of hidden channels (decreasing it with higher $\ell_{\max}$) to keep the number of parameters constant around 330k, and thus, removing differences in model capacity as a contributing factor to performance. We do not condition the models with amino-acid identity to make the problem more challenging, and therefore more interesting. We refer to Section B.2 for more details.

Results. Test Mean Absolute Error (MAE) per χ angle for all models is shown in Figure 4.2, and training curves are shown in the Appendix (Figure B.2.3). Notably, the Angle model performs the worst, and is unable to recover the true χ angle with negligible error. The Sin-Cos and Plane Norm models instead recover all χ angles with very low error ($< 5^\circ$) with $\ell_{\max} > 1$. It appears that $\ell_{\max} = 2$ is the minimum sufficient degree needed to solve this task with high accuracy. We note that error is higher for later χ angles. We hypothesise that this is expected for two reasons: (i) later χ angles depend on atoms that are farther way from the center of the neighborhood, thus having lower angular resolution within the Zernike representation, and (ii) there is simply less training data for them. Weighting χ angles in the loss function according to their average frequency partially mitigates the second issue (Figure B.2.4). Notably, the fact that the model performs well without explicit knowledge of amino acid identities implies that it can easily infer the amino acid type from the the number and the relative location of the atoms.

4.2.2 Side-Chain Packing

Dataset. We consider the training and validation datasets used in DLpacker [157], consisting of 19,436 structures with a maximum inter-protein sequence similarity of 50%. Unlike DLpacker, we do not remodel structures with PDB-redo [106] and do not convert selenomethionine residues into methionine. For testing our model, we use the CASP13 and CASP14 targets (82 and 64 structures, respectively). We remove from the

CASP13		Angle MAE ° ↓				Angle Accuracy % ↑			Atom RMSD Å ↓		
Method		χ_1	χ_2	χ_3	χ_4	All	Core	Surface	All	Core	Surface
SCWRL		27.64	28.97	49.75	61.54	56.2	71.3	43.4	0.934	0.495	1.027
FASPR		27.04	28.41	50.30	60.89	56.4	70.3	43.6	0.910	0.502	1.002
RosettaPacker		25.88	28.25	48.13	59.82	58.6	75.3	35.7	0.872	0.422	1.001
DLPacker		22.18	27.00	51.22	70.04	58.8	<u>73.9</u>	45.4	0.772	0.402	0.876
AttnPacker		<u>18.92</u>	<u>23.17</u>	<u>44.89</u>	58.98	<u>62.1</u>	73.7	<u>47.6</u>	<u>0.669</u>	<u>0.366</u>	<u>0.775</u>
DiffPack		15.35	19.19	37.30	50.19	69.5	82.7	57.3	0.579	0.298	0.696
H-Packer ₀ ^{$\ell_{\max}=5$}		26.89	31.95	47.51	52.75	49.3	61.5	40.4	0.961	0.726	1.131
H-Packer ₂ ^{$\ell_{\max}=5$}		23.64	29.47	45.17	53.26	54.4	69.9	43.6	0.863	0.575	1.070
H-Packer ₅ ^{$\ell_{\max}=5$}		23.60	29.40	<u>44.91</u>	<u>52.91</u>	54.7	70.7	43.7	0.858	0.564	1.067
H-Packer _{up} ^{$\ell_{\max}=5$}		<i>20.03</i>	<i>26.88</i>	<i>42.74</i>	<i>52.07</i>	<i>58.4</i>	<i>75.4</i>	<i>46.4</i>	<i>0.765</i>	<i>0.483</i>	<i>0.980</i>

CASP14		Angle MAE ° ↓				Angle Accuracy % ↑			Atom RMSD Å ↓		
Method		χ_1	χ_2	χ_3	χ_4	All	Core	Surface	All	Core	Surface
SCWRL		33.50	33.05	51.61	55.28	45.4	62.5	33.2	1.062	0.567	1.216
FASPR		33.04	32.49	50.15	54.82	46.3	62.4	34.0	1.048	0.594	1.205
RosettaPacker		31.79	28.25	50.54	56.16	47.5	<u>67.2</u>	33.5	1.006	0.501	1.183
DLPacker		29.01	33.00	53.98	72.88	48.0	66.9	33.9	0.929	0.476	1.107
AttnPacker		<u>25.34</u>	<u>28.19</u>	<u>48.77</u>	<u>51.92</u>	<u>50.9</u>	66.2	36.3	<u>0.823</u>	<u>0.438</u>	<u>1.001</u>
DiffPack		21.91	25.54	44.27	55.03	57.5	77.8	43.5	0.770	0.356	0.956
H-Packer ₀ ^{$\ell_{\max}=5$}		32.31	35.90	49.05	50.34	40.8	57.0	31.6	1.087	0.762	1.297
H-Packer ₂ ^{$\ell_{\max}=5$}		29.96	34.32	47.46	50.50	45.0	65.1	34.1	1.011	0.629	1.250
H-Packer ₅ ^{$\ell_{\max}=5$}		29.61	34.03	<u>46.72</u>	50.35	45.2	65.5	34.0	1.002	0.626	1.244
H-Packer _{up} ^{$\ell_{\max}=5$}		<i>26.58</i>	<i>31.54</i>	<i>45.67</i>	<i>49.46</i>	<i>48.1</i>	<i>69.5</i>	<i>36.2</i>	<i>0.915</i>	<i>0.534</i>	<i>1.160</i>

Table 4.1: **Comparative assessment of H-Packer on CASP13 and CASP14.** We present best results in **bold** and second-best underlined. *Italicized* results represent an upper bound to our algorithm’s performance. Performance of models other than H-Packer is taken from [262].

training and validation sets any protein that has sequence similarity above 50% with any of the proteins in the test set.

H-Packer training. We used the Sin/Cos loss function (Eq. 4.2) as it was the best-performing loss in our toy-task; while the Plane Norms loss (Eq. 4.3) also performed well in the toy-task, we found that models trained with the Sin/Cos objective were easier to regularize via dropout in the final invariant feed-forward neural network. The *initial guess* and the *refinement* networks were trained with the same $\ell_{\max}$ of 5 and $n_{\max} = 12$; the latter was chosen such that it included at least one radial function with wavelength lower than the minimum interatomic distance. We also considered models trained with $\ell_{\max} = 4$, tuning the number of hidden features to keep the number of trainable parameters the same as the $\ell_{\max} = 5$ models, and equal to $\sim 3M$. All models were trained for 10 epochs, keeping the model with lowest validation loss at the end of an epoch; see further details in Section B.2. experimented with up-weighting higher-index χ angles in the loss function (Section 4.2.1), but ultimately decided against it in an effort to promote more accurate learning of the lower-index ones χ_1 and χ_2 . Throughout our experiments, we consider the performance of H-Packer models with different number of rounds of refinement. For example, H-Packer₀ denotes the model with *no* refinement. For each model, we also compute an *upper bound* in performance of the refinement process by tasking the *refinement* model to predict χ angles from the ground truth neighboring structures (i.e., the toy task); we denote this by H-Packer_{up}.

Metrics. In line with previous work [148, 262], we evaluate our models on three main metrics. (i)

Method	CASP13				CASP14			
	base	Std.	Sym.	Std.+Sym.	base	Std.	Sym.	Std.+Sym.
H-Packer ₀ ^{$\ell_{\max}=5$}	0.961	0.943	0.923	0.906	1.087	1.070	1.050	1.034
H-Packer ₂ ^{$\ell_{\max}=5$}	0.863	0.842	0.826	0.805	1.011	0.992	0.972	0.953
H-Packer ₅ ^{$\ell_{\max}=5$}	0.858	0.837	0.821	0.800	1.002	0.984	0.964	0.945
H-Packer _{up} ^{$\ell_{\max}=5$}	0.765	0.741	0.730	0.706	0.915	0.895	0.880	0.860

Table 4.2: **Atom RMSD ($\text{\AA}\downarrow$) for H-Packer across all residues with different treatments of the true structure.** **Std:** standardizing the true structure by reconstructing it with our data-derived redundant internal coordinates and the real χ angles. **Sym:** considering the additional non-natural symmetries used by AttnPacker.

	H-Packer ₅	Angle MAE $^{\circ}\downarrow$				Angle Accuracy % $\uparrow$			Atom RMSD $\text{\AA}\downarrow$		
		χ_1	χ_2	χ_3	χ_4	All	Core	Surface	All	Core	Surface
CASP13	$\ell_{\max} = 4$	24.27	29.76	46.32	52.56	53.5	68.8	43.0	0.878	0.594	1.081
	$\ell_{\max} = 5$	23.60	29.40	44.91	52.91	54.7	70.7	43.7	0.858	0.564	1.067
CASP14	$\ell_{\max} = 4$	30.36	34.38	48.76	50.62	43.5	64.1	32.5	1.024	0.648	1.260
	$\ell_{\max} = 5$	29.61	34.03	46.72	50.35	45.2	65.5	34.0	1.002	0.626	1.244

Table 4.3: **Ablation in $\ell_{\max}$ of H-Packer.** Metrics for the other H-packer models can be found in Table B.2.2.

Angle-specific Mean Absolute Error (MAE), (ii) residue-level angle accuracy, defined as the proportion of residues for which the prediction of all χ angles is within 20° of the true value, and (iii) average atomic Root Mean Square Deviation (RMSD) of side-chain atoms across residues. We further distinguish between *Surface* and *Core* residues, as conformations occurring on the surface of proteins are notoriously harder to predict. Surface residues are defined as having at most 15 β -C within $10 \text{ \AA}$ of their β -C, whereas core residues must have at least 20 β -C’s in this range.

Comparative Evaluation on CASP13 and CASP14 targets. Table 4.1 compare H-Packer’s performance in side-chain packing with other computational methods [148, 262, 157, 120, 7, 100]. Despite its simplicity, H-Packer₅ is competitive against the state-of-the-art at predicting χ_3 and χ_4 , but falls behind on χ_1 and χ_2 predictions. This discrepancy indicates that H-Packer has likely learned complementary features to the other models. Moreover, H-Packer mostly outperforms the physics-based computational algorithms [120, 7, 100] and is competitive with DLPacker [157] in all our performance metrics. Interestingly, H-Packer is consistently better than physics-based approaches in terms of overall Atom RMSD, but tends to fall shorter on Angle Accuracy. We present error distributions for H-Packer in Figures B.2.5, B.2.6, B.2.7, and B.2.8.

Interestingly, while H-Packer predictions are improved upon using refinement networks, the performance saturates after 2 steps of refinement; the accuracies after 5 iterations of refinement are comparable to those after only 2 steps (Table 4.1). Therefore, it is unlikely that further refinement could improve H-Packer’s performance to reach its upper bound performance. We hypothesise that training H-Packer to produce confidence scores might help in developing site-specific convergence criteria to help bridge the gap [148, 262].

Our experiments show that H-packer is competitive against physics-based methods and some machine-learning solutions, but its performance still lags behind the state-of-the-art at predicting χ angles closer to the backbone. Overall, the lack of consistent comparative patterns in performance metrics suggests that H-Packer learns features complementary to other approaches.

Ablation in $\ell_{\max}$. Table 4.3 shows how changing $\ell_{\max}$ (from 4 to 5) impacts the performance of H-Packer.

Method	HPacker $_{\ell_{\max}=5}^{\ell_{\max}=5}$	HPacker $_{\ell_{\max}=5}^{\ell_{\max}=5}$	HPacker $_{\ell_{\max}=5}^{\ell_{\max}=5}$	AttnPack	DLPack	RosPack	FASPR	SCWRL4
Rel. Time	1.00	0.51	0.18	0.05	5.70	6.96	0.02	0.67

Table 4.4: **Relative times to undertake full atomic reconstruction.** In our current (unoptimized) implementation, HPacker $_{\ell_{\max}=5}^{\ell_{\max}=5}$ takes 1,482s to reconstruct the 82 CASP13 targets on a single NVIDIA A40 GPU. Times for the other methods were taken from [148].

For the same model capacity, using higher $\ell_{\max}$ consistently yields better performance, indicating that higher angular resolutions of the input can be beneficial for learning this task. This performance improvement comes with a trade-off in training and inference time, which scale superlinearly with $\ell_{\max}$ unless the Tensor Product computation is adequately constrained [44]. For reference, training our models with $\ell_{\max} = 5$ takes $\sim 40\%$ longer than those with $\ell_{\max} = 4$. We leave the hyperparameter optimization of $\ell_{\max}$ to future work.

On computing RMSD fairly. In Table 4.1 we report RMSD computed by measuring the distance between the coordinates of true and predicted atoms, modulo the symmetries we report in B.2.1. However, other algorithms such as AttnPacker [148] consider other symmetries as well, sometimes even between atoms of differing chemical elements. Though these symmetries reflect spatially similar conformations (such as a flip of the Histidine ring), they result in inflated RMSD scores. We show the effect of this inflation on H-Packer predictions in Table 4.2. In the same table, we also show the RMSD computed against true structures that have been "reconstructed" using the true χ angles and the constant values that we use for redundant internal coordinate within H-Packer; we do this in an effort to disentangle the Null Reconstruction Error (Figure B.2.1) from the error given by mistakes in χ angle prediction.

Speed. Table 4.4 shows relative reconstruction speeds for several packing algorithms. Using the current implementation of the reconstruction algorithm, the best-performing H-Packer model is about 7x faster than the popular algorithm RosettaPacker and 6x faster than DLPacker; however, it is considerably slower than AttnPacker. Speed can be considerably cut down by half at the expense of minor performance degradation using two refinement iterations instead of five. However, more considerable speed gains may be achieved by CPU parallelization, when computing holographic encodings of structural neighborhood during initial data processing. Indeed, each initial guessing and refinement step of H-Packer predicts all χ angles at once, but in the current implementation holographic encodings are computed in series, creating a bottleneck that currently accounts for 88% of the inference time (10% is atom placement, and only 2% is making the actual predictions on GPU). We plan on optimizing this aspect in future iterations of the model.

4.3 Summary

In this chapter, we have presented H-Packer, a novel algorithm for predicting side-chain conformations by jointly regressing over the side-chain’s χ angles. H-packer is composed of two simple and fast rotationally equivariant neural networks, the first one is used for making an *initial guess* using the coordinates of backbone atoms alongside residue identity information, while the second one *refines* the predictions by considering the predicted coordinates of the neighboring side-chain atoms. We carefully study three alternative objective functions, eventually deciding on using a geometrically justified loss function over the sine and cosine of χ angles. Our experiments show that H-packer is competitive against physics-based methods and some machine-learning solutions, but its performance still lags behind the state-of-the-art at predicting χ angles closer to the backbone. Overall, the lack of consistent comparative patterns in performance metrics suggests that H-Packer learns features complementary to other approaches. In addition, the formulation of H-packer makes it amenable to easy-to-achieve CPU parallelization to speed up its already fast inference predictions. We further emphasize that H-Packer is remarkably lightweight - $2 \times 3\text{M}$ parameters vs. 208M of AttnPacker - and requires few resources to train - single GPU at < 1 hour per epoch vs. 4 GPUs for 400 epochs for DiffPack (unknown total time). Limitations of the model include: its inability to distinguish between covalent and non-covalent interactions as atomic interactions are not explicitly encoded into the network, and its inherently

lower angular resolution further away from a neighborhood’s center. Future areas of improvement include: enhancing angular resolution by scaling up $\ell_{\max}$ while adjusting the architecture to reduce the resulting computational complexity, and training a confidence model for the predictions and using it to inform the refinement process.

Chapter 5

HERMES

At the time of writing, most of the contents of this chapter are under review, and posted as a preprint Visani et al. [239], under the title *HERMES: Holographic Equivariant neuRal network model for Mutational Effect and Stability prediction*.

This work originated as a re-implementation of the *Holographic-CNN* (H-CNN), developed by my mentor and former labmate Mike Pun [185]. I desired to re-implement the network within the framework I was familiar with, and that had external support, so that I could better use it and better tweak it as I saw fit. Following up on Mike’s initial findings that H-CNN could be used to predict the effect of mutations on diverse protein functions, and considering emergent works that used similar high-level principles as H-CNN to predict mutation effects, I ventured to improve upon H-CNN with the goal of mutation effect prediction in mind. These efforts resulted not only in a new family of models - the HERMES models - but also in a systematic analysis of the strengths, weaknesses and differences among models in the "machine learning for protein mutation effect" literature.

Starting with biological motivations, understanding the effects of amino acid substitutions on protein function is fundamental to biological discovery and engineering, with applications spanning disease variant identification [77, 20], enzyme optimization and engineering [102, 244], viral escape prediction [229, 139, 164], antibody engineering [89], and vaccine antigen stabilization [147, 80, 15, 154, 177].

Mutational effects on thermodynamic stability and binding affinity are particularly well-studied, as stability is typically prerequisite for function [194] and most biological processes are mediated by binding events. While these effects can be measured experimentally via denaturation assays [132], surface plasmon resonance [110], or deep mutational scanning [71, 115, 221], such experiments remain laborious despite recent throughput improvements [232].

Computational approaches offer an alternative. Molecular dynamics simulations can accurately capture short-time (nano seconds) responses, but are limited in predicting substantial conformational changes often induced by mutations [74]. Physics-based energy functions, including FoldX [208] and Rosetta [111], remain widely used to predict mutation-induced stability changes [80], yet are often slow and inaccurate [20]. In recent years, machine learning models have shown considerable progress in this area: models trained to predict amino acid propensities at a given residue from surrounding sequence [190, 151] or structural context [185, 59, 20, 126, 17, 60] can approximate mutational effects across phenotypes.

Recent work improves these pre-trained baselines by modifying the pre-training objective [82, 79], and by fine-tuning to predict phenotype-specific mutational effects [60, 59, 20, 143], with thermodynamic stability as a frequent target for structure-based models [60, 59, 20]. Notable examples include RaSP [20], which fine-tunes a 3D convolutional neural network (CNN) on Rosetta-computed $\Delta\Delta G$ values [39]; Stability-Oracle [59], a graph transformer fine-tuned on experimental stability measurement from the Megascale dataset [232]; and ThermoMPNN [60], which fine-tunes the inverse folding model ProteinMPNN [52] on a different subset of Megascale.

Here, we present HERMES, a family of structure-based models built upon our previous H-CNN architecture [185]. Like H-CNN, HERMES employs a 3D rotationally equivariant, all-atom CNN architecture, but incorporates implementation improvements yielding a $\sim 2.75\times$ speedup and an adaptable architecture enabling fine-tuning for arbitrary downstream tasks. We first pre-train HERMES on an inverse folding

objective (i.e., predicting a residue’s amino acid identity from its surrounding atomic neighborhood within a 10 Å radius), then fine-tune for predicting mutational effects. The HERMES family comprises three model variants that differ in their treatment of structural context: HERMES-*fixed*, which holds the local environment static during prediction; HERMES-*relaxed*, which enables local structure relaxation through explicit side-chain repacking; and HERMES-*amortized*, which implicitly encodes the structural flexibility associated with different substitutions through amortization. We systematically analyze the biases of these models with respect to the physicochemical properties of mutating residues and characterize their utility across different tasks.

On thermodynamic stability benchmarks, we show that fine-tuned HERMES models match or exceed state-of-the-art performance. However, we identify a “wild-type preference bias” introduced by the pre-training objective that is only partially eliminated by fine-tuning. We also demonstrate HERMES’ utility for structure-based vaccine design, where stabilizing viral envelope glycoproteins in their metastable pre-fusion conformation is essential for presenting neutralizing antibody epitopes [147]. Identifying stabilizing mutations traditionally requires domain expertise and costly experimental iteration. While computational approaches using Rosetta [80, 177] or machine learning (e.g., ReCAP [15]) have emerged, HERMES offers significant computational efficiency over Rosetta and, unlike ReCAP, is publicly available. When evaluated on 33 known stabilizing mutations across 5 viral envelopes, HERMES ranks 19 within the top 3 predicted substitutions at each position, with particularly strong performance on mutations that stabilize independently without synergistic interactions. We also illustrate two mutation discovery pipelines using full saturation mutagenesis with HERMES to suggest an initial set of candidates, followed by a refinement of the candidate set with Rosetta. Lastly, we demonstrate that HERMES can be fine-tuned to predict mutational effects on protein-protein binding affinity, benchmarking competitively against existing models.

Code is open source at <https://github.com/StatPhysBio/hermes/tree/main>.

5.1 Methods

HERMES architecture and pre-training on masked amino acid classification.

At its core, HERMES is a 3D, rotationally equivariant convolutional neural network that predicts the propensity of the 20 different canonical amino acids at a masked (removed) focal residue, given its local atomic neighborhood within the protein structure (Fig. 5.1A).

Atomic neighborhoods - centered at the focal residue’s $C\alpha$ and with radius 10 Å - are first featurized by atom type (including added hydrogens), partial charge, and solvent-accessible surface area, and then encoded into *holograms* via the ZFT (Eq. 2.8). Then, a stack of SO(3)-equivariant layers - shown in Fig. 5.1A and detailed in Section 2.3 - map this encoding to a rotation-invariant embedding, which a final MLP converts into amino-acid propensities.

For pre-training, we train HERMES to recover the identity of the masked residues on ProteinNet’s CASP12 set, filtered at 30% sequence identity [8] (Fig. 5.1B). Each model is an ensemble of 10 independently trained networks (3.5M parameters each), with predictions averaged at inference. To improve robustness for zero-shot applications, we additionally train models with 0.50 Å coordinate noise. Pre-training performance is summarized in Table C.1.1. See Extended Methods in Appendix C for further details.

Zero-shot prediction of mutational effects with pre-trained HERMES. The pre-trained HERMES model outputs $P(aa|X_{i,aa})$, the probability of assigning amino acid aa at residue i , given the surrounding atomic environment (neighborhood) $X_{i,aa}$ in the structure. Crucially, $X_{i,aa}$ depends on the aa : the neighborhood geometry has the fingerprint of the masked amino acid during pre-training.

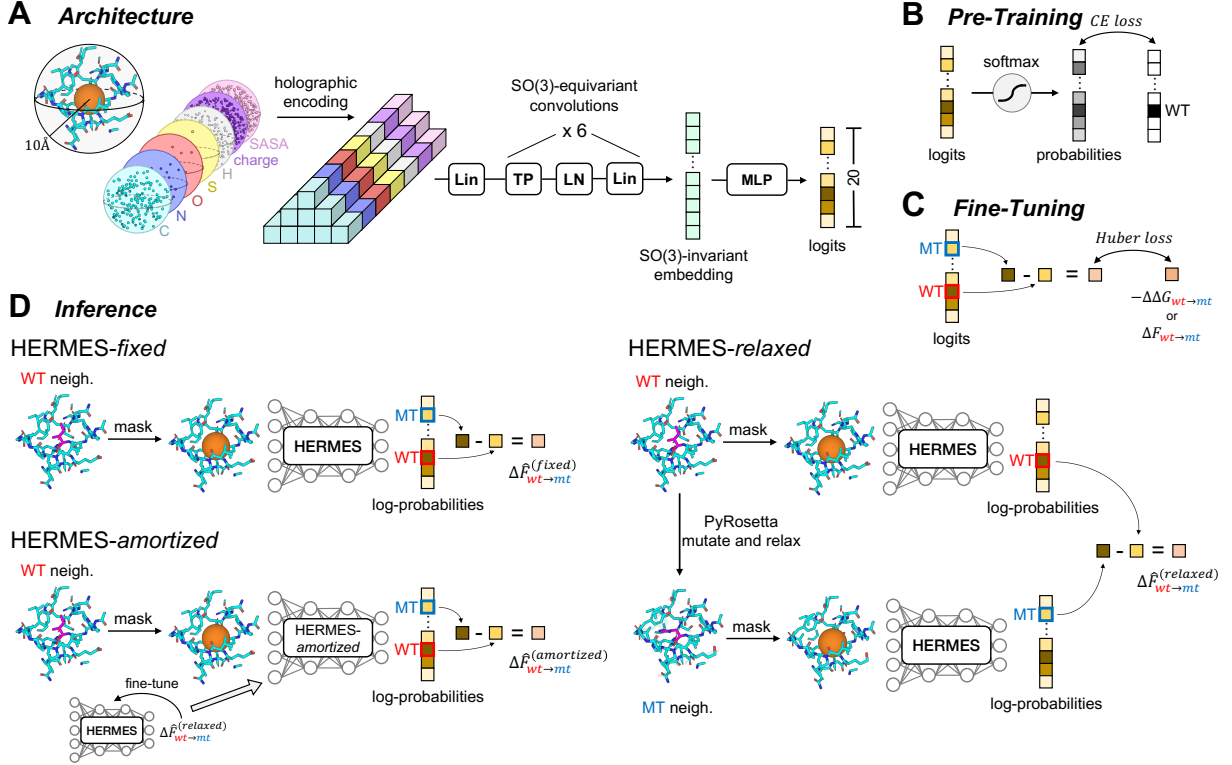


Figure 5.1: Overview of HERMES. (A) Model architecture: HERMES takes as input an all-atom structural neighborhood (10 Å radius) around a masked focal residue. Each atom is represented by its 3D coordinates, element type, partial charge, and solvent-accessible surface area (SASA). The neighborhood is projected onto a Zernike Fourier basis (spherical hologram) and processed by rotation (SO(3)) equivariant layers to produce a rotation-invariant embedding, which is mapped to 20 amino-acid-specific logits (Methods). (B) Pre-training: models are trained to predict the identity of the masked focal residue from its surrounding atomic neighborhood; logits in (A) are converted to amino-acid propensities (probabilities) via a softmax. (C) Fine-tuning for mutational effects: model weights are optimized to regress the logit difference between mutant and wild-type amino acids to the corresponding experimental mutational effect. (D) Inference protocols. **HERMES-fixed** scores a substitution as the difference between the the mutant and wild-type amino-acids logits from a single forward pass, conditioned on the masked wild-type neighborhood X_{wt} . **HERMES-relaxed** conditions the mutant term on an approximate mutant neighborhood $\hat{X}_{wt \rightarrow mt}$ generated in-silico by introducing the mutation on the wild-type structure and locally relaxing the structure with Rosetta [39]. **HERMES-amortized** distills the relaxed protocol by fine-tuning on **HERMES-relaxed** predictions, enabling fast fixed-style inference while retaining relaxation-aware behavior.

Conditional models are widely used for zero-shot prediction of mutational effects on protein function (e.g., [190, 151, 185]). We approximate the effect of wild-type to mutant (wt $\rightarrow$ mt) substitution at residue i by the log-likelihood ratio between the wild-type and the mutant amino acids, conditioned on the respective local atomic neighborhoods $X_{i,wt}$ and $X_{i,mt}$ (omitting the residue index i for clarity):

$$\Delta \hat{F}_{wt \rightarrow mt} \propto \log P(mt|X_{mt}) - \log P(wt|X_{wt}) \quad (5.1)$$

Here, the neighborhood X depends on the identity of the original amino acid (wt or mt), suggesting the potential need for having access to mutant structures for such predictions. The $\hat{\cdot}$ (hat) indicates the model-predicted mutational effects, as opposed to experimental measurements (no hat).

When a mutant structure is unavailable, neighborhoods can be relaxed *in silico* (e.g., by Rosetta [39]), though this procedure is computationally expensive. A common alternative is to score all mutations using a single (typically wild-type) structure [52, 59, 185]. Here, we consider both of these protocols: **HERMES-fixed** evaluates mutational effects using the wild-type structure only (for both mt and wt propensities), while **HERMES-relaxed** evaluates the propensity of the mutant amino acid in the Rosetta-relaxed mt neighborhood

$\hat{X}_{\text{wt} \rightarrow \text{mt}}$ starting from the available wt structure (Fig. 5.1D); see Methods for details. The resulting estimates for mutational effects from these two approaches follow,

$$\begin{aligned}\Delta \hat{F}_{\text{wt} \rightarrow \text{mt}}^{(\text{HERMES-fixed})} &\propto \log P(\text{mt} | X_{\text{wt}}) - \log P(\text{wt} | X_{\text{wt}}) \\ \Delta \hat{F}_{\text{wt} \rightarrow \text{mt}}^{(\text{HERMES-relaxed})} &\propto \log P(\text{mt} | \hat{X}_{\text{wt} \rightarrow \text{mt}}) - \log P(\text{wt} | X_{\text{wt}}).\end{aligned}\quad (5.2)$$

Fine-tuning HERMES to predict protein function. Context-conditioned amino-acid likelihoods provide useful zero-shot proxies for mutational effects across diverse functions [190, 151, 185], but supervised models for specific functions can perform better in practice. Building on prior work [20, 59], we fine-tune HERMES models directly on mutational effect data. Unlike approaches that train a separate regression head [20, 59, 60], we update the model end-to-end so that its *predicted scores* themselves align with measurements. Specifically, as shown in Fig. 5.1C, we make the predicted HERMES-*fixed* $\Delta \hat{F}_{\text{wt} \rightarrow \text{mt}}^{(\text{HERMES-fixed})}$ (eq. 5.2) regress over the experimentally measured mutational effects $\Delta F_{\text{wt} \rightarrow \text{mt}}$ by minimizing a robust Huber loss,

$$\mathcal{L} = \text{HuberLoss}(\Delta \hat{F}_{\text{wt} \rightarrow \text{mt}}^{(\text{HERMES-fixed})}, \Delta F_{\text{wt} \rightarrow \text{mt}}) \quad (5.3)$$

which stabilizes training under outliers while calibrating predictions to the function of interest. See Extended Methods in Appendix C for further details.

Hermes-amortized to encode structural flexibility in HERMES-fixed by amortization. Because the structural-relaxation step in HERMES-*relaxed* is computationally costly (~ 66 times slower than HERMES-*fixed* on a single CPU / A40 GPU, Table C.1.2), we distill HERMES-*relaxed* predictions into the model via a fine-tuning procedure. Concretely, we fine-tune the model so that the mutational-effect predictions produced with the fast HERMES-*fixed* protocol regress to the corresponding HERMES-*relaxed* predictions via Eq. 5.3 (Fig. 5.1D). We perform this fine-tuning on a small subset of neighborhoods extracted from the pre-training proteins ($\sim 15\text{k}$ neighborhoods, 0.5% of the total). The resulting amortized model HERMES-*amortized* runs at HERMES-*fixed* speed (effectively the same protocol, just different model weights) yet closely matches HERMES-*relaxed* performance (Fig. C.1.1).

5.2 Results

5.2.1 Predicting mutational effects on thermodynamic fold-stability

The chief task we evaluated the HERMES models on was to predict mutational effects on thermodynamic folding stability. Thermodynamic folding stability is defined as the change in Gibbs Free ΔG energy upon folding. Thus, the effect of a mutation $\text{wt} \rightarrow \text{mt}$ on folding stability is denoted by $\Delta \Delta G_{\text{wt} \rightarrow \text{mt}} = \Delta G_{\text{mt}} - \Delta G_{\text{wt}}$.

Training and test Data. To enable a direct comparison with recent structure-based stability predictors, we fine-tuned and evaluated HERMES on the same benchmark splits used by RaSP [20], Stability-Oracle [59], and ThermoMPNN [60]. Specifically, we (i) trained on the RaSP Rosetta-derived stability dataset and tested on the experimental benchmark used in RaSP, (ii) used Stability-Oracle’s curated cDNA117k training set and T2837 test set; training on 117k $\Delta \Delta G$ values from ref. [232] filtered by enforcing $\leq 30\%$ sequence identity to the test set, and (iii) trained on ThermoMPNN’s Megascale *train* split and evaluated on its Megascale *test* split, both from ref. [232]; training on 216k and testing on 28k mutation effects with 25% sequence-identity cutoff. Because these splits are defined by the original studies, our results are directly comparable across methods. We additionally note that the Megascale *train* split is not de-duplicated against T2837, and six

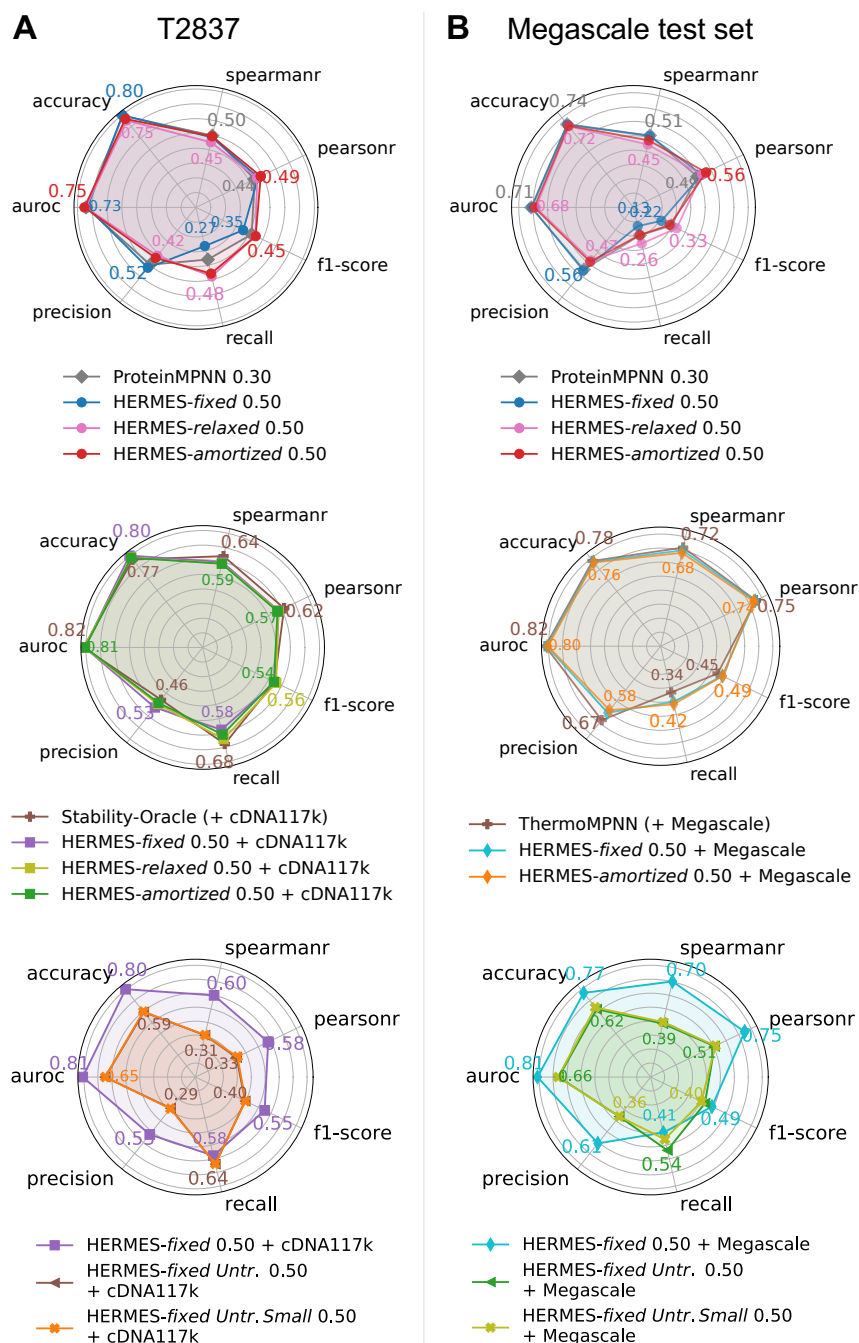


Figure 5.2: Predicting mutational effects on thermodynamic folding stability. Stabilizing-versus-destabilizing classification metrics are computed using $\Delta\Delta G < 0$ (experimental) and $\Delta \log p > 0$ (predicted) as cutoffs for stabilizing mutations. **(A)** T2837 results: zero-shot models (top) and models fine-tuned or only trained on cDNA117k (middle and bottom). **(B)** Megascale test set results: zero-shot models (top) and models fine-tuned or only trained on the Megascale training set (middle and bottom). Model names indicate the architecture, the coordinate-noise amplitude used, and when applicable, the fine-tuning dataset (listed after “+”); *Untr.* is short for *Untrained*, indicating models that had no pre-training and were instead only trained on stability effects. Only models trained with coordinate noise are shown; the noise amplitude is indicated within each model name as standard deviation in Å units. Results for models trained without noise are provided in Fig. C.1.2.

of the T2837 proteins ($\sim 5\%$) have $> 90\%$ sequence-similar homologs in the Megascale training set; see

Methods for more details on these training and test sets.

Noise and side-chain relaxation improve zero-shot model predictions. We first evaluated the *zero-shot* HERMES models (no fine-tuning on experimental data) on the T2837 and Megascale test sets (Fig. 5.2, C.1.2). Pre-training on structures with Gaussian coordinate noise (0.5 Å s.d.) improves performance, consistent with prior work [52, 185]. Relative to HERMES-*fixed*, the packing-aware HERMES-*relaxed* achieves significantly higher recall (0.48 vs. 0.27; p-value < 0.01), with only a slight loss in precision, leading to a higher overall F1-score. HERMES-*amortized* performs similarly to HERMES-*relaxed*: on T2837, and on Megascale, it shows slightly lower recall and F1 while still outperforming HERMES-*fixed*; the p-values for the significance of these performance differences are reported in Fig. C.1.5.

To test whether packing awareness preferentially improves predictions for substitutions that perturb local packing, we stratified mutations by residue size. Wild-type and mutant residues were assigned to small, medium, or large classes using normalized van der Waals volumes [150], and we evaluated stabilizing-versus-destabilizing classification performance within each wt → mt size-transition group (Fig. 5.3).

Switching from HERMES-*fixed* to HERMES-*relaxed* yielded the largest recall gains in identifying stabilizing substitutions between residues with substantially different sizes. This improvement was most pronounced for large→small substitutions: HERMES-*fixed* is constrained by the rigid wild-type cavity, preventing it from recognizing that mutations that create voids could be stabilizing, whereas HERMES-*relaxed* can repack neighbors to fill the space and stabilize the smaller side chain. While precision changes varied across size categories, F1-score improved in all categories, with the largest gains occurring for small→large substitutions, where relaxation can reorganize the local environment to accommodate bulkier side chains that would otherwise clash. Interestingly, HERMES-*amortized*, which learned packing implicitly through fine-tuning, showed comparable improvements for large→small substitutions but much weaker gains for small→large substitutions. This could suggest that stabilizing small→large substitutions may be underrepresented in the fine-tuning training set; the p-values associated with the significance of performance differences between models within each size-transition category, and within models across different size-transition categories are reported in Figs. C.1.4, C.1.5.

For comparison, we evaluated ProteinMPNN [52] on the same tasks. ProteinMPNN outperformed HERMES-*fixed*, and performed on par with HERMES-*relaxed* on both small→large and large→small mutations (Fig. 5.3, C.1.3). We attribute this performance to ProteinMPNN’s input representation: by conditioning only on backbone coordinates and residue identities, ProteinMPNN is less constrained by the explicit wild-type side-chain geometry compared to the all-atom HERMES-*fixed* model. This architectural choice enables implicit reasoning about side-chain flexibility, achieving results comparable to using explicit relaxations in HERMES-*relaxed*. Notably, ProteinMPNN performed better than HERMES-*amortized* on small→large substitutions.

Fine-tuned HERMES models achieve state-of-the-art performance for stability effect prediction. When fine-tuning on the respective stability effect datasets, HERMES outperformed RaSP (Fig. C.1.6), and matched the performance of Stability-Oracle (Fig. 5.2A, C.1.7) and ThermoMPNN (Fig. 5.2B). These results underscore both the effectiveness of the HERMES architecture and the importance of fine-tuning data for test-time accuracy. HERMES was also robust to using ESMFold-resolved structures [130] for either fine-tuning or inference (Fig. C.1.8), supporting practical use cases where only computationally predicted structures are available.

Removing pre-training on wild-type amino-acid classification significantly degraded performance: HERMES models trained *only* for stability prediction performed poorly when trained on either cDNA117k or Megascale,

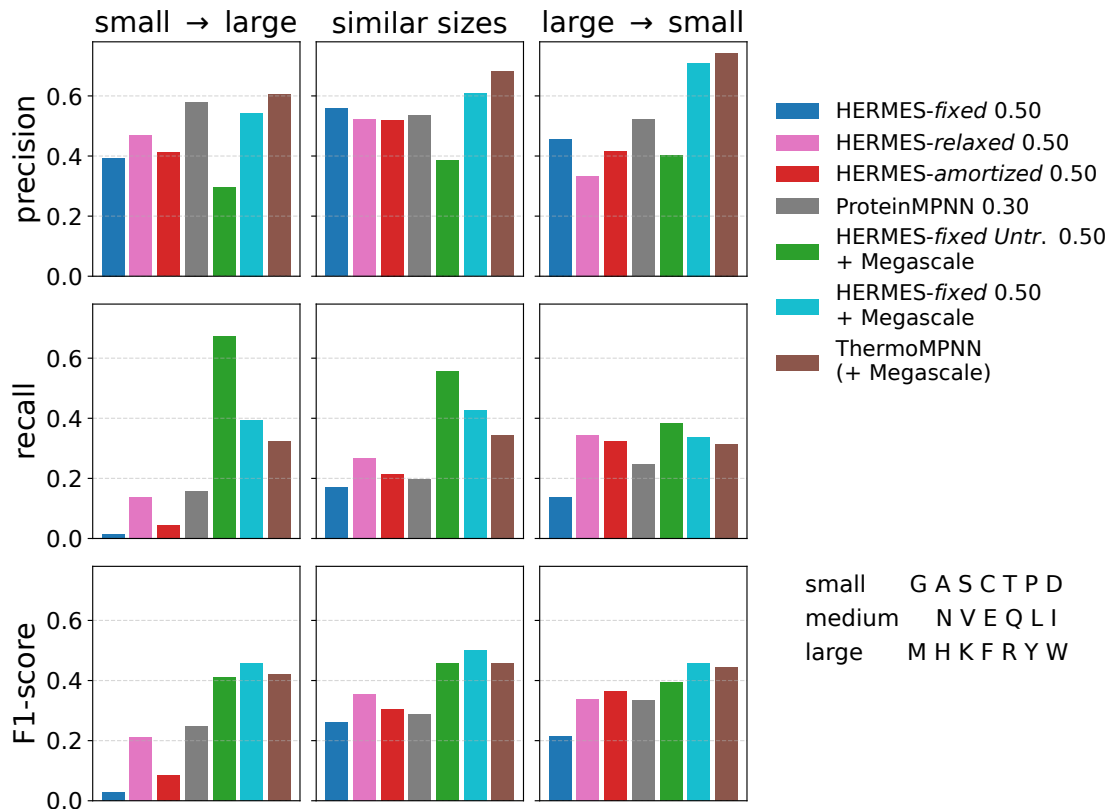


Figure 5.3: Impact of amino acid size changes on predicting mutational effects on protein stability. Amino acids are grouped into three size classes based on van der Waals volume (as listed), and predictions are stratified by wild-type and mutant size classes; “similar sizes” denote substitutions within the same class. Stabilizing-versus-destabilizing classification metrics (rows) are then computed and shown for each stratum (columns), using $\Delta\Delta G < 0$ (experimental) and $\Delta \log p > 0$ (predicted) as cutoffs for stabilizing mutations. P-values for pairwise model comparisons are shown in Fig. C.1.3, and p-values comparing each model’s performance on small→large vs. large→small substitutions are shown in Fig. C.1.4. Model names indicate the architecture, the coordinate-noise amplitude used during pre-training as standard deviation in Å units, and when applicable, the fine-tuning dataset (listed after “+”).

even after reducing capacity from 3.5M parameters to 50k to mitigate overfitting (Fig. C.1.2). A similar failure mode was reported for ThermoMPNN on the Megascade training data [60].

Finally, fine-tuning on stability effects largely eliminated size-dependent bias, yielding comparable recall and F1 scores for small→large and large→small substitutions (Figs. 5.3, C.1.4).

Biophysical interpretation of HERMES predictions for mutational stability effects. We sought to quantify how much models’ mutation preferences could be explained by amino-acid physicochemical properties. Specifically, we asked which properties are shared by amino-acid pairs that the model treats as interchangeable, i.e., substitutions with $\Delta\hat{F} \approx 0$ on average.

To do this, we constructed model-averaged substitution matrices M^{model} whose (α, β) entry represents how neutrally the model treats exchanging amino acids α and β . For each pair (α, β) , we computed the model-predicted mean *absolute* effect, $M_{\alpha, \beta}^{\text{model}} = \langle |\Delta\hat{F}_{\alpha, \beta}^{(\text{model})}| \rangle$, where $\langle \cdot \rangle$ denotes averages over all $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ substitutions in the Megascade test set. This procedure yields a symmetric, non-negative 20×20 matrix for each model, in which values approaching zero indicate greater predicted interchangeability. For comparison, we constructed an analogous matrix using experimentally measured $|\Delta\Delta G|$ values from the

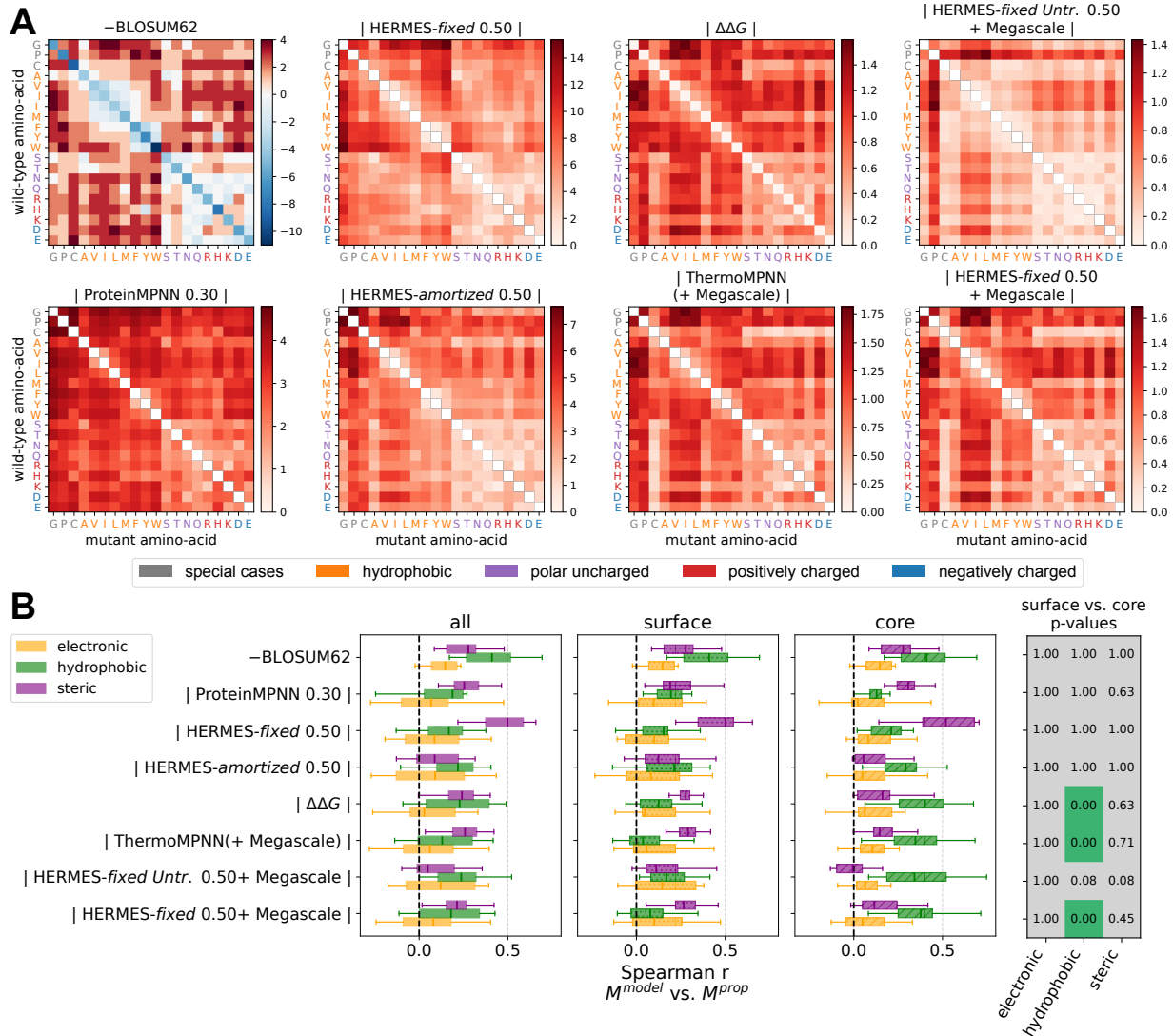


Figure 5.4: Uncovering mutation preferences via model-averaged substitution matrices. (A) Heatmaps of model-averaged substitution matrices M^{model} computed by averaging over the Megascale test set, shown alongside BLOSUM62 and the experimental matrix of mean $|\Delta\Delta G|$ values across mutations. Spearman correlations between matrices are reported in Fig. C.1.10. Core- and surface-restricted matrices (core: SASA $< 1^2$; surface: SASA $> 3^2$) are shown in Fig. C.1.11 and C.1.12. (B) For each model and site subset (all, surface, core), boxplots summarize Spearman correlations between M^{model} and property-difference matrices M^{prop} , grouped by property class (color). P-values from two-tailed t-tests comparing the surface vs. core correlations within each property class are shown on the right. P-values for between-model comparisons of property-class correlations are shown in Fig. C.1.9.

Megascale test set ($M^{|\Delta\Delta G|}$), and include the BLOSUM62 substitution matrix as an additional reference.

Following ref. [186], we also constructed symmetric, nonnegative matrices of *absolute amino-acid property differences*. Specifically, we considered 50 quantitative properties spanning (i) hydrophobic, (ii) electronic, and (iii) steric categories (Table C.1.3), yielding 50 property-specific matrices M^{prop} with the (α, β) entry: $M_{\alpha, \beta}^{\text{prop}} = |\text{prop}_\alpha - \text{prop}_\beta|$. Smaller values indicate greater similarity between the two amino acids with respect to the specified property.

To quantify which biophysical properties each model tends to preserve, we computed the Spearman correlation between the model-averaged substitution matrix M^{model} and each of property-specific amino acid distance matrix M^{prop} (Fig. 5.4B). Consistent with the size-stratified analysis (Fig. 5.3), HERMES-*fixed* predicted amino-acid interchangeability align most strongly with steric properties, particularly average buried-residue volume (Spearman $r = 0.66$) and normalized van der Waals volume (Spearman $r = 0.64$). This correlations was significantly stronger than for the other models (p-values in Fig. C.1.9). In contrast, BLOSUM62, derived from evolutionary substitution frequencies, preferentially preserves hydrophobic properties. When restricting M^{model} to core residues ($\text{SASA} < 1\text{\AA}^2$), the interchangeability matrix associated with stability-fine tuned models (e.g. $M^{\text{HERMES-}fixed\ 0.50 + \text{Megascale}}$) showed stronger alignment with hydrophobic properties, to levels comparable to BLOSUM62 and consistent with the experimental matrix $M^{|\Delta\Delta G|}$. Zero-shot models instead did not show a comparable increase in hydrophobic preservation when restricting to the core (Fig. 5.4B).

Reversibility and path-independence of HERMES predictions with respect to mutations. Mutational effects are, in principle, reversible: the effect of substituting a residue from amino acid α to β should be equal in magnitude and opposite in sign to the reverse change, i.e., $\Delta F_{\alpha \rightarrow \beta} = -\Delta F_{\beta \rightarrow \alpha}$. Moreover, equilibrium quantities such as protein stability free energy are state functions. As such, the net effect of a multi-step substitution depends only on the initial and final amino acids, not on the mutational path. For a path $\alpha \rightarrow \beta \rightarrow \gamma$, this implies: $\Delta F_{\alpha \rightarrow \gamma} = \Delta F_{\alpha \rightarrow \beta} + \Delta F_{\beta \rightarrow \gamma}$.

HERMES predicts mutation effects as differences in amino-acid-specific log-probabilities (logits) $\log p(aa|X_{aa})$ (both zero-shot and after fine-tuning). As log-probability differences under a fixed structural context, these predictions satisfy reversibility by construction:

$$\Delta \hat{F}_{\alpha \rightarrow \beta}^{(\text{model})} = \log P(\beta|X_\beta) - \log P(\alpha|X_\alpha) = -\Delta \hat{F}_{\beta \rightarrow \alpha}^{(\text{model})} \quad (5.4)$$

where we set $X_\alpha = X_\beta$ for HERMES-*fixed* and HERMES-*amortized*, and $X_\beta = \hat{X}_{\alpha \rightarrow \beta}$ for HERMES-*relaxed*. Path-independence follows similarly. In contrast, other structural models such as Stability-Oracle [59] require explicit $19\times$ data augmentation (termed “thermodynamic permutation augmentation” in ref. [59]) to enforce these properties.

Alternatively, reversibility can be assessed in a structure-conditioned manner [20, 60], where the effects of forward $\alpha \rightarrow \beta$ and reverse $\beta \rightarrow \alpha$ substitutions are computed using the outgoing structural contexts, i.e., $\alpha \rightarrow \beta$ is conditioned on X_α and $\beta \rightarrow \alpha$ on X_β (Extended Methods, Appendix C). Under this transformation, we do not automatically expect a “structure-conditioned reversibility”, as forward and reverse transitions are conditioned on different structures. However, a well-balanced model should have near-reversibility in this setting, making this a stringent test of model bias. We measured structure-conditioned reversibility on the Ssym dataset, which contains wild-type and single-mutant structures for 352 mutations across 19 proteins [172]. For each mutation, we computed a “forward” effect (wt $\rightarrow$ mt, conditioned on the wild-type structure) and a “reverse” effect (mt $\rightarrow$ wt, conditioned on the mutant structure).

We found that zero-shot HERMES models and ProteinMPNN predict stability effects more accurately for forward substitutions (wt $\rightarrow$ mt) than for the reverse direction, consistent with previous observations [20, 60]

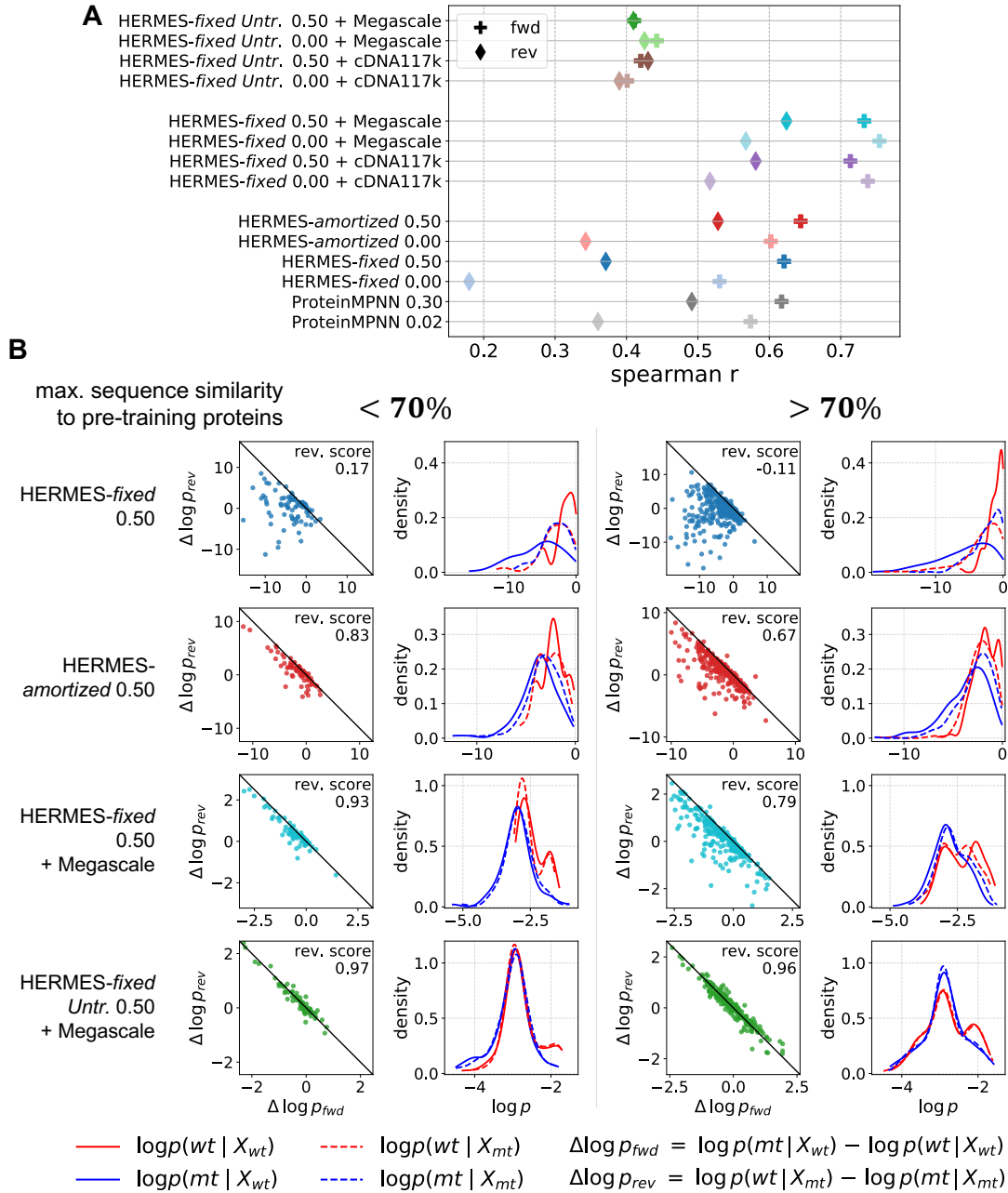


Figure 5.5: Identifying model biases with structure-conditioned reversibility (A) Spearman correlations between experimental stability changes ($\Delta\Delta G$ and model predictions on the Ssym dataset are shown. For each model, we report correlations for forward substitutions ($\Delta \log p_{fwd}$ vs. $\Delta\Delta G$) and the reverse substitutions ($\Delta \log p_{rev}$ vs. $-\Delta\Delta G$). (B) Ssym proteins are stratified by their maximum sequence identity to pre-training proteins ($\geq 70\%$ vs. $< 70\%$). For each split (columns) and four representative HERMES variants (rows), the left panel shows the scatter plots for $\Delta \log p_{fwd}$ vs. $\Delta \log p_{rev}$; an unbiased model should exhibit strong anti-correlation, summarized by the reversibility score (rev.; higher is more reversible; Eq. C.1.3). Reversibility is highest for the model not pre-trained on wild-type amino-acid classification (green; bottom row) and is consistently higher for the low-similarity subset. Fig. C.1.13 shows the reversibility scores across all models in (A). In each column, the right panel shows the distributions of the log-probabilities that make up $\Delta \log p_{fwd} = \log p(mt|X_{wt}) - \log p(wt|X_{wt})$ (solid lines) and $\Delta \log p_{rev} = \log p(wt|X_{mt}) - \log p(mt|X_{mt})$ (dashed lines). All models except for the one that was *not* pre-trained on wild-type amino-acid classification (last row) exhibit elevated $\log p(wt|X_{wt})$.

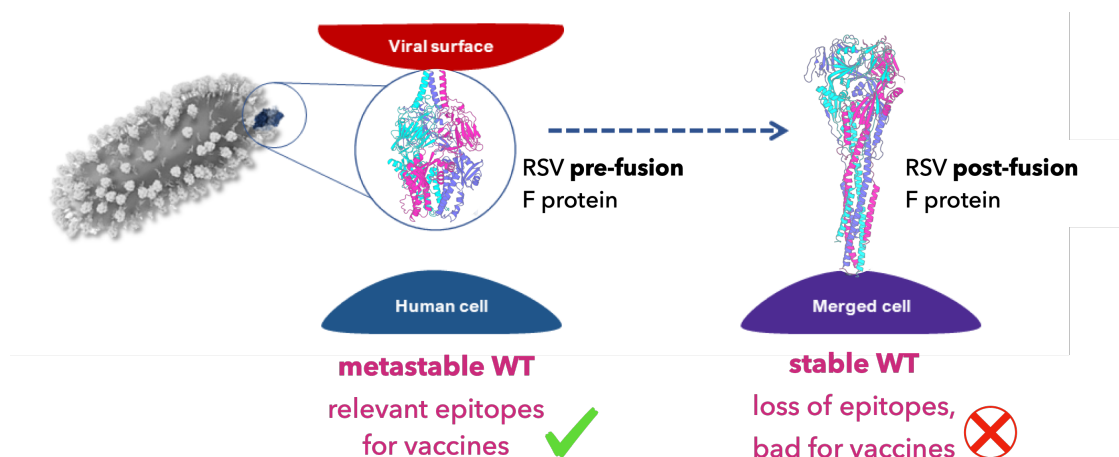


Figure 5.6: Overview of structure-based vaccine design against enveloped viruses. A large proportion of viruses are enveloped by a bi-lipid membrane akin to that of human cells. To infect human cells, they make use of specialized proteins that latch onto the cell's membrane and "fuse" it to the virus' envelope. This fusion process is enabled by a conformational change in the fusion protein: from a "pre-fusion" to a "post-fusion" conformation. Naturally, vaccines should be engineered to elicit immune responses that target epitopes presented *before* the virus has infected the cell, i.e. while the fusion protein is in the pre-fusion conformation. However, the wild-type of fusion viruses is generally more stable in the post-fusion conformation. Therefore, the field of structure-based vaccine design has proposed to use vaccines containing engineered mutants of the fusion proteins that are more stable in pre-fusion relative to post-fusion. Figure adapted from <https://www.niaid.nih.gov/diseases-conditions/safe-and-effective-rsv-protein-vaccines>.

(Fig. 5.5A). Adding coordinate noise during pre-training and fine-tuning on stability effects both reduced this forward-reverse disparity, but did not eliminate it. Models trained *only* on stability effects showed little or no disparity, albeit with substantially lower overall accuracy. Across all pre-trained models, the predicted stability effects of forward mutations from the wild-type tend to have larger magnitudes than those of the corresponding reverse mutations (Fig. 5.5B, scatterplots). This bias stems from the elevated log-probabilities assigned to wild-type amino acids in wild-type structure neighborhoods $\log p(\text{wt}|X_{\text{wt}})$ (Fig. 5.5B, density plots), suggesting a wild-type preference that may reflect pre-training memorization. Consistently, stratifying Ssym proteins by sequence similarity to the pre-training set (Extended Methods, Appendix C) yielded a modest reduction in wild-type preference for lower-similarity proteins (Fig. 5.5B). A more detailed characterization of this bias is left to future work.

5.2.2 Antigen stabilization with HERMES for vaccine design

Structure-based vaccine design seeks to increase vaccine efficacy by engineering viral envelope glycoproteins (hereafter, "antigens") to be more stable in their pre-fusion conformation, relative to their post-fusion conformation (Figure 5.6). Pre-fusion stabilization enriches presentation of neutralization-relevant epitopes and can bias elicited immune responses toward protective specificities. In the literature, it is achieved by applying a minimal set of mutations to the fusion protein that either improve the stability of the pre-fusion conformation, or decrease that of the post-fusion conformation, while preserving the relevant epitopes [147, 80, 15, 154, 177].

Practitioners often classify mutations in different types based on the manner in which they provide relative pre-fusion stabilization (hereafter, "antigen stabilization"). For example, *Cavity-filling mutations* stabilize the pre-fusion conformation by filling cavities located within the hydrophobic core, by inserting a larger hydrophobic side-chain. *Proline mutations* instead are often used to impede post-fusion α -helix formation and stabilize pre-fusion conformations, due to proline's unique lack of amide hydrogens which

prevents it from taking part in stable α -helices. Other mutation types include *electrostatic mutations*, which introduce polar or charged amino acids in environments where they can engage in stabilizing electrostatic interactions, and finally *synergistic mutations*, which involve applying multiple mutations so that they positively interact through a variety of mechanisms. We show examples of mutation types in Fig. C.1.16A.

The use of computational tools to guide antigen stabilization has precedents in the literature, but it is largely limited to the use of scoring functions like Rosetta [80, 177], with few closed-source attempts to use machine learning [15]. Therefore, we sought out to test whether HERMES can be used to aid vaccine design campaigns.

Unfortunately, large scale datasets of experimentally-validated mutations are not yet present. Thus, we started the campaign by manually collecting 33 previously reported antigen-stabilizing mutations drawn from five viral antigens: Influenza HA [154] (3 mutations), RSV-F [147, 123] (7 mutations), hMPV-F [80, 15, 177] (11 mutations), DENV-E [177] (8 mutations), and SARS-CoV-2 spike protein [96] (4 mutations). We then benchmarked model performance on their ability to identify the specific mutants as the most pre-fusion-stabilizing within their respective sites; essentially evaluating per-site recall. We scored mutations using zero-shot and stability-fine-tuned variants of ProteinMPNN and HERMES. Importantly, none of the stabilized variants in this benchmark appeared in the training data of any HERMES model, either during pre-training or during stability fine-tuning.

To quantify performance, we emulated a simple model-guided selection workflow in which a practitioner considers substitutions at a site in descending order of model scores. For each known stabilizing mutation, we record its rank r_{mt} among all 20 amino acids at that position and compare it to the wild-type rank r_{wt} . We posit that a practitioner would select a particular mutant for experimental validation if (1) its rank r_{mt} is better (lower) than that of the wild-type r_{wt} , and (2) its rank is among the top-scoring candidates (lower end). We categorize each predicted-stabilizing mutation (i.e., with $r_{\text{mt}} < r_{\text{wt}}$) as strongly suggested ($r_{\text{mt}} \leq 3$), moderately suggested ($3 < r_{\text{mt}} \leq 6$), or weakly suggested ($r_{\text{mt}} > 6$) (Fig. 5.7 and Table C.1.4). This scheme would group mutations with similar physico-chemical properties into the same rank class. Consequently, models are generally not penalized for ranking a biophysically similar alternative above a known stabilizing mutant, making our performance metrics robust to fine-grained rank differences in different models (see Fig. C.1.14 and Appendix C for a more detail discussion on antigen-stabilizing mutations).

Figs. 5.7, C.1.16 and Table C.1.4 summarize predictions across the 33 stabilizing mutations we studied. Among the machine-learning methods, HERMES-*amortized* performs best: it assigns a better (lower) rank than wild-type to 24 stabilizing mutations, including 19 classified as strongly suggested ($r_{\text{mt}} \leq 3$). Notably, stability fine-tuning does not consistently improve performance over the corresponding zero-shot ProteinMPNN and HERMES-*amortized* on this task (Table C.1.4).

For comparison, we also scored each mutant with Rosetta [39], (Extended Methods, Appendix C). Rosetta recovers stabilizing mutations on par with HERMES-*amortized* (Fig. 5.7 and Table C.1.4); however, 17 variants in this benchmark were originally selected (by the references that discovered the variants) using Rosetta-based screening, which partially biases this baseline in Rosetta’s favor. Most importantly, however, Rosetta inference is substantially more computationally intensive than the machine learning models, limiting its practicality for predicting the stability effects in high-throughput saturation mutagenesis (Fig. 5.7C).

To assess statistical enrichment of model predictions over chance, we compared the predicted number of stabilizing mutations in the strong/moderate categories to a random-ranking baseline using a binomial test (Extended Methods, Appendix C); p-values are reported in Fig. C.1.15. All models except for HERMES-*fixed* and ThermoMPNN significantly outperform random ranking in recovering stabilizing mutations with strong or moderate confidence (Table C.1.4). As an additional baseline, we tested whether the BLOSUM62 (B62) substitution matrix can prioritize stabilizing mutations. BLOSUM62 recovers significantly fewer stabilizing mutations in the strong/moderate categories than HERMES-*amortized* (defined as $r_{\text{mt}}^{\text{B62}} \leq 3$ and $r_{\text{mt}}^{\text{B62}} \leq 6$,

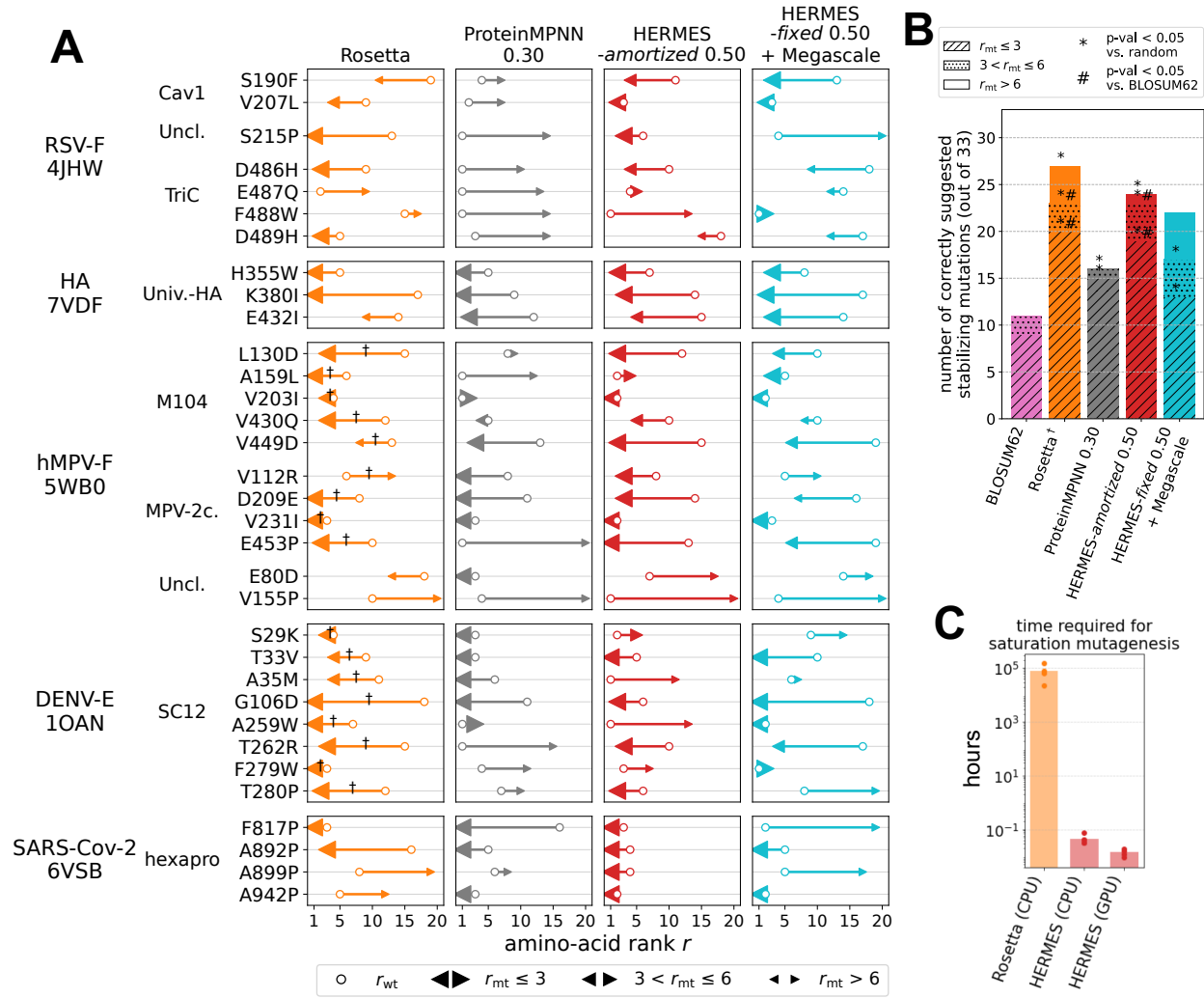


Figure 5.7: Recovering previously-identified antigen-stabilizing mutations with HERMES. (A) Model recall is evaluated on 33 previously reported antigen-stabilizing mutations (rows) across five viral antigens. For each antigen, the PDB structure used for scoring is indicated. Mutations are specified as wild-type→mutant substitutions at the annotated site. Four models (columns) are compared, as labeled above each column; see Table C.1.4 for a more extensive comparison of models on this task. Arrows depict the change in predicted rank from the wild type (open circle) to the stabilizing mutant (arrow tip); arrow size indicates whether the mutant ranks in the top 3 (large), ranks 4–6 (medium), or ranks > 6 (small). The † symbols mark mutations originally proposed as stabilizing by Rosetta. “Univ.-HA” stands for “Universal-HA”; “MPV-2c.” stands for MPV-2cREKR; “Uncl.” stands for “Uncleaved Prefusion-Closed”. (B) Counts of correctly prioritized antigen-stabilizing mutations ($r_{mt} < r_{wt}$), stratified by predicted rank group: strongly suggested ($r_{mt} \leq 3$), moderately suggested ($3 < r_{mt} \leq 6$), or weakly suggested ($6 \leq r_{mt}$) are shown for each model. For comparison, BLOSUM62 is used to rank substitutions into the ($r_{mt} \leq 3$) or ($3 < r_{mt} \leq 6$) groups; under this scheme the wild-type residue is always ranked first. Statistical enrichment is assessed by comparing the number of strong/moderate recalls to a random-ranking baseline and to BLOSUM62 (denoted by (*) and (#), respectively, when binomial tests corrected p-value < 0.05; see Methods). We note that all structures were considered in their native multimeric state, generating symmetric partners with PyMOL when necessary. (C) Estimated times for saturation mutagenesis predictions on the viral antigens considered in this study. Each dot is an antigen. Times shown are for a single CPU and a single NVIDIA A40 GPU. See Table C.1.5 for precise per-antigen times.

noting that $r_{wt}^{B62} = 1$ always; results shown in Fig. 5.7, and binomial test p-values reported in Fig. C.1.15), underscoring the value of incorporating structural context when predicting stabilizing substitutions.

For a more detailed analysis, we manually annotated the 33 antigen-stabilizing mutations based on their mutation types (Table C.1.6) and accordingly stratified model performance (Fig. C.1.16B). Among the models

tested, HERMES-*amortized* most reliably recovers stabilizing proline mutations (7 out of 8; Fig. C.1.16B), highlighting its potential utility for proposing pre-fusion stabilizing prolines. More broadly, all models recover proline and electrostatic mutations at higher rates than expected under BLOSUM62, whereas cavity-filling mutations show weaker gains. A plausible explanation is that cavity-filling changes typically replace a smaller hydrophobic residue with a larger one that preserves core hydrophobicity. Because such substitutions are common in natural sequence variation, they are partly captured by a sequence-derived matrix like BLOSUM62. In contrast, proline and charged substitutions are strongly context-dependent, with effects that hinge on local geometry and environment, and therefore benefit more from structure-aware modeling; see Appendix C for a more detailed discussion.

Lastly, the synergistic RSV-F TriC mutations are difficult to recover for almost all methods (Fig. 5.7). This is expected for the three “acid patch neutralization” substitutions D486H, D489H, and E487Q [147], which were experimentally stabilizing only in combination (i.e., synergy) with F488W [147]. These four residues are tightly clustered in both intra- and inter-chain space, consistent with a synergistic (epistatic) mechanism in which the substitutions act synergistically to stabilize the trimer. A second example of synergy is the DENV-E pair A259W and T262R, which together introduce a favorable cation- π interaction (Fig. C.1.16A.4). Because our evaluation scheme scores mutations in a site-independent manner, the models are not able to capture such multi-residue dependencies. Specifically, HERMES scores mutations at a single site under the assumption that all other amino-acid identities remain fixed, and therefore, cannot recover stabilization mechanisms that arise only from specific combinations of mutations. ProteinMPNN, ThermoMPNN, and Rosetta are likewise evaluated here under the same site-independent protocol for a fair comparison, although they can in principle be applied to score multi-residue variants jointly; see Appendix C for a more detailed discussion.

Beyond the inability to capture epistatic effects, several additional limitations of our approach should be noted. First, to emulate a practitioner’s workflow we evaluate mutations using only their relative ranks; however, the absolute magnitudes of model scores carry additional information and could enable more robust decision-making when prioritizing antigen-stabilizing mutations. Second, because the original studies rarely tested comprehensive alternative substitutions at the same sites, we cannot evaluate the models’ ability to propose different stabilizing mutations that were not assayed experimentally. Third, experimental validation frequently involved multi-mutation constructs, complicating attribution of observed stabilization to any single residue. Finally, the modest number of curated stabilizing mutations limits statistical power and the strength of conclusions in our analyses. Despite these caveats, our results suggest that HERMES can serve as a practical screening tool for rational library design, prioritizing candidate antigen-stabilizing mutations for more efficient experimental validations.

While the above analysis uses experimentally-verified antigen stabilizing mutations as ground truth, it is limited by a low sample size, and the fact that it measures only per-site recall. Therefore, it is not fully indicative of performance during a design campaign, in which precision is also a key metric.

Despite a lack of extensive experimental ground truth, we sought to demonstrate the practical utility of HERMES by designing a workflow for suggesting individual antigen-stabilizing mutations. We considered two pipelines for suggesting mutations with HERMES.

1) *Stabilize pre-fusion core*: the first and simplest pipeline focuses on suggesting mutations that stabilize the pre-fusion conformation, ignoring possible effects to the post-fusion conformation. Specifically, given an pre-fusion structure, we scored all possible mutations all sites in the core (average SASA $< 1\text{\AA}^2$), via $\Delta \log p^{pre-f}$, where higher scores indicate higher predicted stabilized effects. We then sorted the mutations by score, and considered as suggestions the top 80 mutations, with a limit of maximum 3 mutations at the same site to encourage more site variability; crucially, all mutations are predicted to be stabilizing by HERMES ($\Delta \log p^{pre-f} > 0$).

2) *Stabilize pre-fusion and destabilize post-fusion*: for the second pipeline instead we aim to suggest mutations

that both stabilize the pre-fusion conformation, and destabilize the post-fusion conformation. Specifically, we consider as suggestions the top 80 mutations (again, max 3 per site) sorted by $\Delta \log p^{pre-f} - \Delta \log p^{post-f}$, and subject to both $\Delta \log p^{pre-f} > 0$ and $\Delta \log p^{post-f} < 0$. For this approach, we do not pose any constraints on site locations.

Of the HERMES models, we only considered HERMES-*amortized* as it was the best performing model in the ground truth recall analysis. For baselines, we considered mutations sampled with ProteinMPNN, BLOSUM62, and at random. Specifically, for ProteinMPNN we selected 80 mutations per pipeline in the same exact manner as with HERMES. For BLOSUM62 we randomly selected 80 sites per pipeline (either in pre-fusion core only or unrestrictedly), and took the top two mutations by BLOSUM62 score. Finally, we sampled 160 mutations at random (max 3 per site, either in pre-fusion core only or unrestrictedly). For all models and pipelines, we excluded mutations to and from cysteine residues, both to avoid breaking disulfide bonds, and in consideration of the fact that stabilizing cysteine mutations usually come in pairs precisely to form disulfide bonds.

Due to the lack of experimental ground truth, we resort to using Rosetta as an oracle. Rosetta scores are computed analogously to the previous analysis (Figure 5.7), but with higher sample size. Specifically, for each mutation, the full structure with both wild-type and mutant amino-acid is relaxed and scored 50 times, yielding a distribution of 50 scores in Rosetta Energy Units (REU). These are then summarized into a single composite score (S_{ros}) by averaging the best (lowest) 20% of the values. Then, a suggested mutation is considered a ‘‘Rosetta hit’’ if either $\Delta S_{ros}^{pre-f} > 0$, or both $\Delta S_{ros}^{pre-f} > 0$ and $\Delta S_{ros}^{post-f} < 0$, respectively following the analogous pipeline used for suggesting mutations.

Using Rosetta as oracle implies a model’s performance should be interpreted as how much it agrees with Rosetta, not explicitly with experiments. However, we still consider this a useful exercise. Indeed, Rosetta’s ability to identify stabilizing mutations is validated by our recall analysis 5.7, and by previous studies with experimental validation [80, 177]. Despite this, Rosetta is very slow (Figure 5.7C), prohibitively so if CPU resources are limited. Thus, first using a fast model - like HERMES - to rank all possible mutations such that top mutations are agreed upon by Rosetta can accelerate the finding of truly antigen-stabilizing mutations. Furthermore, using the consensus of multiple models to generate candidates with the highest likelihood of experimental success is a ubiquitous strategy in protein design [31, 1], and adopted in this thesis also in Chapter 6.

We ran both design pipelines on the RSV-F and DENV-E proteins, reporting respective results in Figures 5.8 and C.1.17. HERMES-*amortized* consistently outperforms - in terms of Rosetta hits - all baselines, with the exception of ProteinMPNN in the ‘‘stabilize pre-fusion and destabilize post-fusion’’ pipeline. The greatest alignment with Rosetta is observed in the ‘‘stabilize pre-fusion core’’ pipeline, where 7 of the top 10 mutations are validated by Rosetta, compared to 4 by ProteinMPNN and 2 if picked at random (Figure 5.8). Plotting the amino-acid pairs involved in successfully suggested mutations, reveals that HERMES can identify cavity-filling mutations in the core (small hydrophobic $\rightarrow$ large hydrophobic and polar $\rightarrow$ hydrophobic) as well as electrostatic mutations (polar $\rightarrow$ positively or negatively charged) (Figure 5.8A, middle).

Manually inspecting mutations that were suggested by HERMES and validated by Rosetta, we identified A298I and N175P promising candidates (Figure 5.8). In the pre-fusion conformation, alanine 298 is in an under-packed region of the core, surrounded by hydrophobic residues (with the exception of a single threonine); mutating it to an isoleucine provides better packing without requiring any relaxation of the neighborhood. Furthermore, even though this mutation was identified without concern for the post-fusion conformation, site 298 is on the surface of the post-fusion conformation thus a mutation to Isoleucine would not likely not further stabilize it (Figure 5.8A, right) ¹. N175P instead was identified with the pipeline

¹After having already written this, my collaborator Zac Jones discovered that the A298I mutation was declared in a patent, which

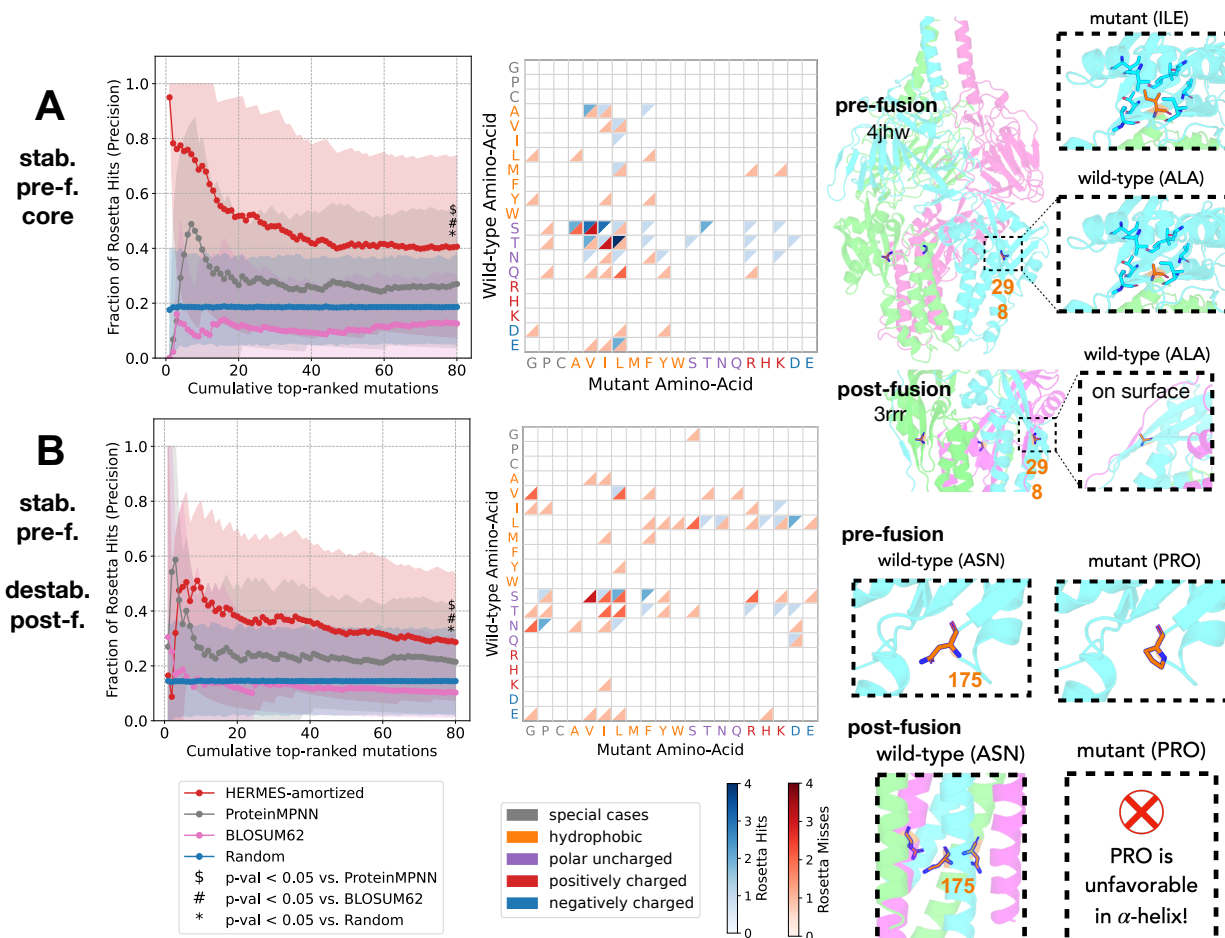


Figure 5.8: Identifying new antigen-stabilizing mutations with HERMES for RSV-F. Panel **A** shows results using the “stabilize pre-fusion core” pipeline, whereas panel **B** shows results using the “stabilize pre-fusion and destabilize post-fusion” pipeline. Each panel is split in three sub-panels. *Left*: Rosetta hit rate as a function of number of top- k suggested mutations (for all methods except for Random, for which we show the average hit rate of 20 random samples of k mutations from the pool of 160 randomly-selected mutations). Rosetta hits were computed multiple times via 200 bootstrap samples with replacement of the 50 REU scores. Solid lines and markers denote the mean across bootstrap samples, whereas shaded regions denote the 5-95 percentile range. We note that all BLOSUM62 suggestions had a score ≥ 1 and are ranked by score just like HERMES and ProteinMPNN. P-values were computed via Wilcoxon signed-rank test and corrected for multiple testing via Bonferroni within each panel. *Center*: heatmap showing number of Rosetta hits (blue) and misses (red) for each amino-acid mutation pair suggested by HERMES-amortized. Here, Rosetta hits were computed using all 50 REU scores instead of bootstrap samples. *Right*: examples of promising antigen-stabilizing mutation identified by each pipeline. Mutant structures were generated using PyMOL’s protein mutagenesis wizard.

that includes the predicted effect on the post-fusion conformation, and is a canonical example of a proline mutation: in pre-fusion the site is on a loop, specifically on a kink, which allows for prolines; in post-fusion instead it is on a α -helix, which generally cannot accommodate prolines (Figure 5.8B, right).

Overall, our analyses suggest that HERMES-amortized can speed-up antigen stabilization campaigns, by quickly evaluating all possible mutations and suggesting a pool of highly-likely antigen-stabilizing mutations for experimental validation.

is validating for the pipeline! <https://patents.google.com/patent/US20170298101A1/en>

Method	Per-Struct. Pearson	Per-Struct. Spearman	Overall Pearson	Overall Spearman
Rosetta* [173]	0.328	0.299	0.311	0.347
FoldX* [56]	0.391	0.364	0.356	0.351
DDGPred* [211]	0.371	0.343	0.652	0.439
ESM-IF* [98]	0.231	0.209	0.296	0.287
MIF-Net.* [143]	0.395	0.348	0.667	0.480
RDE-Net.* [143]	0.469	0.433	0.642	0.527
Vanilla Pythia PPI [†] [228]	0.478	0.449	0.709	0.537
Pythia-PPI [†] [228]	0.565	0.527	0.785	0.637
ProteinMPNN 0.02	0.281	0.282	0.331	0.315
ProteinMPNN 0.30	0.270	0.255	0.334	0.289
HERMES-fixed 0.00	0.306	0.287	0.285	0.272
HERMES-fixed 0.50	0.317	0.308	0.291	0.286
HERMES-amortized 0.00	0.231	0.241	0.290	0.242
HERMES-amortized 0.50	0.222	0.239	0.276	0.222
HERMES-fixed 0.00 + cDNA117k	0.347	0.331	0.380	0.342
HERMES-fixed 0.50 + cDNA117k	0.305	0.294	0.344	0.288
HERMES-fixed 0.00 + SKEMPI Easy	0.471	0.433	0.578	0.476
HERMES-fixed 0.50 + SKEMPI Easy	0.430	0.389	0.512	0.420
HERMES-fixed 0.00 + SKEMPI Medium	0.472	0.430	0.576	0.466
HERMES-fixed 0.50 + SKEMPI Medium	0.407	0.368	0.497	0.403
HERMES-fixed 0.00 + SKEMPI Difficult	0.435	0.398	0.395	0.380
HERMES-fixed 0.50 + SKEMPI Difficult	0.399	0.359	0.328	0.322

Table 5.1: Benchmark on predicting mutational effects on protein-protein binding in SKEMPI. The Spearman/Pearson correlations between model-predicted effects of single point mutations and experimental values from the SKEMPI v2.0 dataset [103] are reported both across mutations within each structure individually (“per-structure” correlations), and over mutations pooled across all complexes (“overall” correlations). Entries marked with * are taken from ref. [143] and, for machine learning-based methods (all except Rosetta and FoldX), were obtained using 3-fold cross-validation over SKEMPI complexes. Entries marked with † are taken from ref. [228], and were obtained via 5-fold cross validation on the SKEMPI complex structures. We evaluate HERMES using three train/test splits defined from SKEMPI metadata, with increasing difficulty: (i) *Easy*, a random split; (ii) *Medium*, which groups sites with similar binding sites (hold-out proteins) into the same split; (iii) *Difficult*, which groups sites from the same held-out protein types (functional classes) into the same split (see Extended Methods, Appendix C for details). The HERMES models with most comparable training procedure to the other machine learning models are those fine-tuned on the *Easy* split, for which we used 3-fold cross-validation with splits defined by PDB complex. Note that, similar to HERMES, the machine learning models in ref. [143] were fine-tuned on SKEMPI $\Delta\Delta G_{\text{binding}}$ labels only, whereas Pythia-PPI [228] was trained on a mixture of SKEMPI binding labels and FireProtDB stability labels [223] and further refined via self-distillation.

5.2.3 Predicting binding effect of mutations

Predicting the effects of mutations on binding affinity is a central step in target-specific protein design. In prior work, we demonstrated the utility of HERMES for predicting how mutations in short peptide antigens alter binding to T-cell receptors in the context of MHC complexes [241]. We further leveraged this capability for de novo design of peptide antigens intended to elicit specific T-cell responses [241]. Here, we evaluate HERMES on the single-point mutation effects from the SKEMPI v2.0 dataset [103], which spans a substantially broader range of protein-protein interactions and binding-affinity perturbations.

In a zero-shot setting, HERMES exhibits measurable predictive signal (HERMES-*fixed* 0.50; Spearman’s $\rho = 0.286$). ProteinMPNN achieves comparable overall performance, whereas physics-based approaches (Rosetta [39] and FoldX [208]) perform modestly better (Spearman’s $\rho \approx 0.35$). In contrast to our results for stability prediction, HERMES-*fixed* models correlate more strongly with binding effects than HERMES-*amortized* models. Interestingly, fine-tuning for stability (HERMES-*fixed* 0.00+ cDNA117k) yields a slight improvement in binding-effect prediction. A detailed comparison across models is provided in Table 5.1.

Next, we fine-tuned HERMES directly on SKEMPI using 3-fold cross-validation. To control for information leakage under structural similarity, we introduced three homology-aware splitting strategies of increasing difficulty; the most stringent split prevents complexes from the same interaction “class” (e.g., antibody-antigen or TCR-pMHC) from appearing in different folds; see Extended Methods, Appendix C for details. Fine-tuned HERMES models achieve substantially stronger correlations, including under the *Difficult* split (Table 5.1).

Finally, we compared HERMES to recent state-of-the-art methods, RDE-Network and MIF-Network [143], DDGPred [211], ESM-IF [98] and Pythia-PPI [228]. These baselines were also trained on SKEMPI using 3- or 5-fold cross-validation under protocols analogous, but not identical, to our *easy* split. HERMES fine-tuned on SKEMPI-Easy performs competitively with RDE-Network and MIF-Network, but trails Pythia-PPI (Table 5.1). Notably, RDE-Network and MIF-Network are fine-tuned on SKEMPI $\Delta\Delta G$ labels in a manner similar to our approach, whereas Pythia-PPI incorporates additional training procedures that further boost performance (Extended Methods, Appendix C). These procedures are, in principle, applicable to HERMES as well, but we leave a systematic evaluation of such extensions, and of HERMES as a general framework for binding-affinity prediction, to future work.

5.3 Summary

In this chapter, we introduced HERMES, a family of fast, structure-based machine learning models for predicting the effects of mutations on protein function. HERMES leverages SO(3)-equivariant neural networks operating on local atomic neighborhoods in a protein structure to predict amino acid propensities, whose differences can be used to predict mutational effects. This formulation yields a computationally efficient predictor that is naturally suited to high-throughput settings. HERMES captures mutational impacts on diverse phenotypes, including thermodynamic stability and protein-protein binding affinity, and it provides a practical tool for antigen stabilization in vaccine design.

A central finding is that encoding packing flexibility is essential for accurate predictions across mutations involving amino acids of different sizes. HERMES-*fixed*, a lightweight protocol that evaluates mutations in the rigid wild-type structure, exhibits strong bias toward size-conserving substitutions. Explicitly modeling relaxation via Rosetta [39], (HERMES-*relaxed* protocol) resolves this bias but incurs substantial computational cost. Our amortized approach offers an effective compromise: by fine-tuning on relaxed predictions for just 0.5% of pre-training data, HERMES-*amortized* learns implicit packing flexibility that can be leveraged for predictions at HERMES-*fixed* speed. This capability proves particularly valuable for antigen stabilization, where HERMES-*amortized* recovers over half of the verified stabilizing mutations in our benchmark and shows strong performance on proline substitutions (87.5% recovery), which stabilize proteins by restricting backbone conformational freedom.

Beyond zero-shot use, HERMES can be fine-tuned directly on experimental labels using a simple end-to-end procedure. Notably, both the native and the fine-tuned models use the difference of amino acid scores for predicting mutational effects, which yields an inherent reversibility with respect to mutation order—a thermodynamic consistency property that is often enforced via explicit $19\times$ data augmentation [59].

With fine-tuning, HERMES achieves competitive accuracy on thermodynamic stability benchmarks when trained on the same data as prior methods [20, 59, 60]. Interestingly, stability fine-tuning does not consistently

transfer to antigen stabilization and can even degrade performance. One plausible explanation is dataset mismatch: the high-throughput stability datasets used for fine-tuning [232] are enriched for relatively small, compact domains, whereas vaccine antigens are often larger, multi-domain, and conformationally heterogeneous, with stabilizing mutations frequently targeting quaternary contacts, glycoprotein-specific features, or prefusion-state constraints. Resolving this discrepancy will likely require broader supervision that better reflects antigen structure and design objectives, or task-aligned fine-tuning data.

We also demonstrate that pre-training on wild-type amino acid classification remains necessary for strong performance; models trained only on currently-available experimental stability data fail. Notably, pre-trained models exhibit “wild-type preference,” assigning elevated probabilities to wild-type residues in wild-type structural contexts. This bias correlates with sequence similarity to pre-training proteins, suggesting partial memorization rather than purely generalizable learning. Fine-tuning partially ameliorates but does not eliminate this effect, representing a fundamental trade-off in current training paradigms. More robust pre-training objectives, stronger regularization, or explicit debiasing strategies may be required to fully address this issue.

A key application we highlight is structure-based vaccine design, where practitioners seek to stabilize substitutions that preserve a desired prefusion conformation of a vaccine antigen to elicit immune responses targeting relevant epitopes. For this task, a useful model should propose stabilizing mutation candidates across different sites of an antigen. On the limited available data comprising 33 verified antigen-stabilizing mutations across five viruses, HERMES performs well in recovering these mutations. We also develop two pipelines to suggest mutations with HERMES across the full antigen, showing close agreement with predictions made with the physics-based model Rosetta, which we suggest to use to further filter the candidate set of mutations. Moreover, the strict locality of the model makes it straightforward and fast to apply to large antigens and assemblies that can pose challenges for architectures modeling global interactions in proteins. Our results show that local all-atom environments modeled by HERMES can encode multiple stabilization mechanisms, including cavity filling, post-fusion disruptions and pre-fusion backbone rigidification via prolines, and favorable electrostatic remodeling, while also underscoring limitations for mechanisms that depend on long-range coupling, or multi-site epistasis.

We further evaluated HERMES on predicting changes in protein–protein binding affinity using the SKEMPI dataset [103]. In the zero-shot setting, HERMES shows measurable predictive signal, and supervised fine-tuning on SKEMPI substantially improves performance. Recent state-of-the-art approaches for binding prediction like Pythia-PPI [228] indicate that design choices such as self-distillation can substantially improve performance [228]; we expect HERMES would similarly benefit from such approaches, representing a clear next step.

Several directions could extend HERMES’ capabilities. Developing larger, more diverse training datasets, particularly for antigen stabilization and binding, would likely improve performance, as our results indicate that training data dominates architectural differences in determining benchmark performance. The locality of HERMES appears to be a double-edged sword: local neighborhoods reduce input complexity, remove constraints on protein size, and improve scalability, but necessarily limit the model’s ability to capture long-range epistasis and allosteric coupling. We view HERMES as a hypothesis-generation tool whose predictions are most reliable when mutational effects are driven primarily by short-range interactions and local packing. Productive directions for future work include better characterizing when locality suffices, further reducing pre-training-induced biases, and developing multi-scale approaches that efficiently combine local all-atom predictors with complementary global or multi-site models to capture mechanisms beyond the local neighborhood.

Chapter 6

MAPPING THE T-CELL RECEPTOR SPECIFICITY LANDSCAPE WITH HERMES

The contents of this chapter are published in Visani et al. [241], under the title *T cell receptor specificity landscape revealed through de novo peptide design*. This work was started by my mentor and former labmate Mike Pun. His first-hand contributions can be found in his thesis [183].

T-cells play a key role in the adaptive immune system, contributing to immune surveillance and response to pathogens. T-cells recognize pathogen-derived epitopes in the form of short protein fragments (peptides) displayed on specific molecules known as Major Histocompatibility Complexes (MHCs). The binding between a T-cell receptor (TCR) and a peptide-MHC (pMHC) is essential to mediate a protective T-cell response to an infection. To defend the host against a multitude of pathogens, a highly diverse TCR repertoire is generated through a somatic VDJ recombination process, and selected to target infected cells with remarkable sensitivity and specificity.

Predicting the TCR-pMHC binding specificity is an important step in characterizing the efficacy of the immune system in countering different pathogens. Such understanding would aid with disease diagnoses and development of diagnostic tests for early detection of autoimmune diseases [12]. A predictive model for TCR-pMHC specificity can be used to design engineered TCRs that recognize cancer antigens [135], enhancing the effectiveness of adoptive cell transfer and Chimeric Antigen Receptor (CAR) T-cell therapies [124, 156], as well as the design of soluble TCRs and bispecific T-cell engager therapeutics [97]. Moreover, it can enable the design of optimized antigens for vaccines to elicit robust T-cell responses against specific pathogens or tumors [202, 22, 195].

One bottleneck in deciphering the TCR-pMHC code is the lack of well-curated datasets. High-throughput immune repertoire sequencing, combined with comparative analyses of hosts facing similar immune challenges, and the development of functional assays (e.g., pMHC-tetramer staining [261] or MIRA [166]), have enabled the identification of diverse TCR pools with potential reactivity to specific antigens [50], collected in repositories like Immune Epitope Database (IEDB) [242] and VDJdb [216]. However, the current databases are unbalanced, with only a handful of dominant peptides (e.g. derived from SARS-CoV-2, CMV, EBV, Influenza, etc.) and MHC alleles present, each linked to hundreds of TCRs.

As a result, statistical inference and machine learning methods trained on such data to predict TCR-pMHC reactivities lack generalizability beyond their training data [101, 250, 152]. On the structural side, only a few hundred TCR-pMHC crystal structures have been experimentally generated to date. This is in part because TCR-pMHC interactions are typically of low affinity and are difficult to stabilize to the level needed for CryoEM or X-ray crystallography [179]. Importantly, this scarcity hinders computational prediction methods like AlphaFold in modeling reliable TCR-pMHC structures [26, 260]. Machine learning models, such as large language models, pre-trained on vast unannotated protein datasets can generate meaningful data representations [192, 131]. Fine-tuning these models on smaller, labeled datasets for specific tasks (e.g., predicting mutation impacts on stability or function) has proven highly effective [145, 206, 245, 88]. Pre-trained sequence-based protein language models have been employed to predict reactivities of antibodies to antigens [245, 88], or TCRs to pMHC complexes [28, 152]. However, lack of generalizability due to sparse and skewed training data has remained a challenge in these models.

Protein-protein interactions in general, and immune-pathogen interaction in particular, rely on the complementarity in the 3D structures and amino acid compositions of the interacting protein pairs. Even though

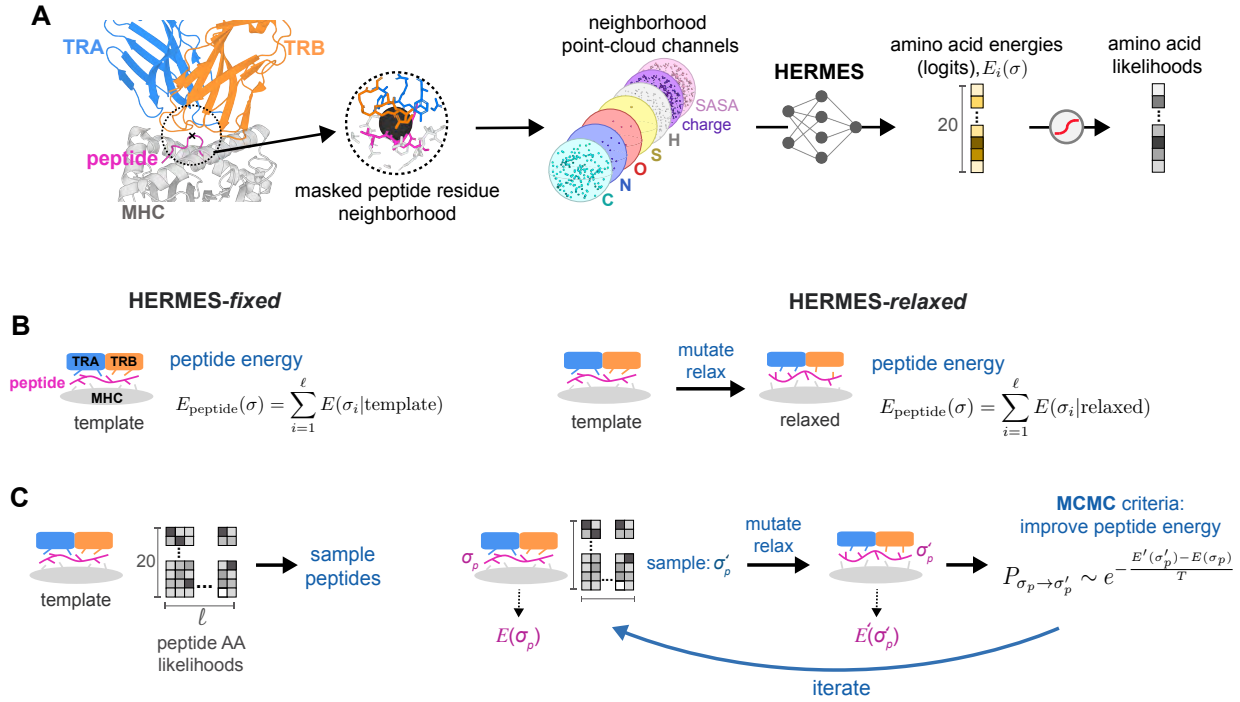


Figure 6.1: HERMES model for predicting and designing TCR-pMHC interactions. (A) HERMES is a structure-based, rotationally equivariant neural network pre-trained on the protein universe to predict amino acid preferences at a residue from its 3D structural environment [238, 182]. It takes as input all atoms within 10Å of a focal residue (with that residue’s atoms masked), featurizing them by atom type, charge, and SASA. HERMES then outputs energies $E_i(\sigma)$ for the 20 possible amino acids σ at residue i , which serve as logits in a softmax to estimate amino acid likelihoods. (B) To predict TCR-pMHC affinity, we use two protocols: HERMES-fixed (left) and HERMES-relaxed (right). In HERMES-fixed we use a fixed structural template closest to the TCR-pMHC of interest as input to HERMES, and evaluate the peptide energy as the sum of HERMES-computed per-residue energies for the peptide. In HERMES-relaxed mutations are introduced to the template’s peptide to match the peptide of interest, followed by local repacking in PyRosetta [38]. Peptide energy is then evaluated similar to HERMES-fixed, but using the relaxed structure. (C) We also use two protocols for peptide design. Similar to (B) we fix the closest structural template in HERMES-fixed, and use HERMES’ likelihood matrix for possible amino acids at each peptide position, which we sample to generate new peptide sequences. In HERMES-relaxed (right), we begin with a random peptide σ , and locally pack it with PyRosetta, and also generate an amino acid likelihood matrix (similar to HERMES-fixed) for this packed structure. We sample a new sequence σ' from this likelihood matrix, and pack it with PyRosetta. We evaluate peptide energies $E_{\text{peptide}}(\sigma)$, and $E_{\text{peptide}}(\sigma')$ for both peptides, using their respective PyRosetta-relaxed structures, and we use these energies as the criterion to sample improved sequences with MCMC (SI). This procedure is then iterated until convergence. Peptides designed by both approaches are then filtered by TCRdock [26] or Alpha-Fold 3 [2] to assure the stability of the complex.

the interacting amino acids can be far apart in a protein sequence, they are close to each other in structure, resulting in local interactions that can determine immune-pathogen recognition. Machine learning models trained on protein structures instead of sequences can learn local structural representations for amino acid statistics, which tend to be more generalizable across different protein families and beyond their training sets [51, 21, 153, 238, 182, 58].

Here, we present a structure-based approach to predict interaction affinities between TCRs and peptides presented on MHC class I, and to design reactive peptides for specific TCR-MHC-I complexes (TCR-MHC for short). Due to the limited availability of structural data for TCRs, we use HERMES— a physics-guided, 3D-equivariant machine learning model trained on the entire protein universe to predict amino acid preferences based on their local structural environments [182, 238].

We demonstrate that HERMES accurately predicts both binding affinity of TCR-pMHC Class I complexes and

the functional responses and T-cell activities induced by a range of peptides presented on MHC molecules. Affinity of TCR-pMHC complexes, measured as the dissociation constant K_d by surface plasmon resonance, is labor intensive to obtain. Functional readout used to measure T-cell activities, including measurements of T-cell proliferation, labeled cellular activation markers, and cytokine (e.g., IFN- γ) release, are easier to obtain than affinity measurements. Yet, functional readouts are shaped not only by the TCR-pMHC affinity, but also by other cellular factors such as the T-cell's differentiation state (e.g., naive or memory), self-reactivity of the TCR [9, 3], and the avidity effects driven by TCR/co-receptor clustering and pMHC density on antigen-presenting cells [222]. Although HERMES is more suited to predict molecular affinities, we show that its scores nonetheless can well track T-cell activities, making it a reliable tool for predicting T-cell response to antigens.

Lastly, we show that HERMES can be used to design novel peptides binding specific TCR-MHC complexes, achieving up to 50% experimental validation accuracy across diverse TCR-MHC systems. cancer vaccines. By leveraging our design algorithm, we characterize the specificity of a diverse range of TCRs, providing a quantitative measure for the diversity of the peptide-MHC antigens that a TCR can recognize in humans and mice.

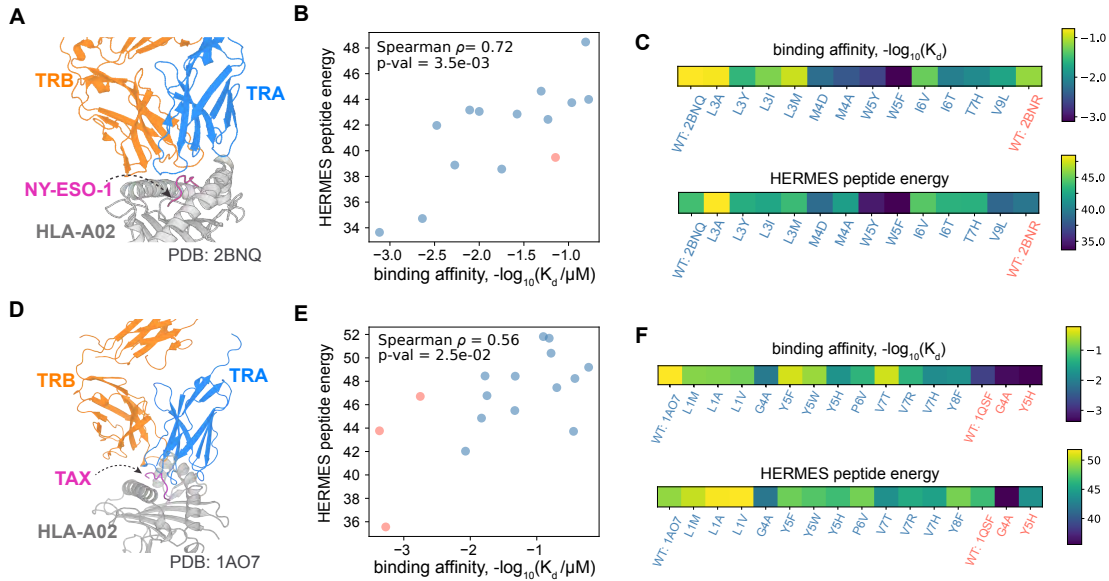


Figure 6.2: Predicting binding affinity of TCR-pMHC complexes. (A) The structure of the NY-ESO-1 peptide (purple), in complex with HLA-A*02:01 and a purified specific 1G4 TCR is shown (PDB ID: 2BNQ [41]). This structure together with that of a peptide variant with C9V mutation (PDB ID: 2BNR [41]) are used as templates to predict the binding affinity of the TCR-MHC complex to other peptide variants. (B) The binding affinities ($-\log_{10}(K_d/\mu\text{M})$) of TCR-MHC complexes with 14 distinct peptides, measured using surface plasmon resonance, are shown against the predicted HERMES-relaxed peptide energies, using the closest structural template for each variant; 2BNQ is used for blue points and 2BNR for the orange point. The Spearman ρ between predictions and measurements is reported in the panel. (C) The heatmaps show the experimental affinity values (top) and predicted peptide energies (bottom) for all the peptide variants, with the identity of the mutations color coded based on the closest structural template. (D-F) Similar to (A-C) but for the Tax viral peptide in complex with HLA-A*02:01 and a specific human TCR. Two structural templates are used [75] for evaluating peptide energies with PDB IDs: 1AO7 (shown in D, and used for blue points), and 1QSF (used for orange points).

6.1 Model

T-cell response is mediated by the interactions between TCRs and pMHC complexes. To model the TCR-pMHC interaction, we use HERMES [238], a 3D rotationally equivariant structure-based neural network

model, trained on the protein universe; see Fig. 6.1A. HERMES predicts the amino acid propensity from its surrounding 3D protein environment, a measure that we have previously shown to serve as a reliable proxy for assessing the effect of mutations on protein stability or binding [182, 238].

For TCR-pMHC complexes, we seek to determine how changes in a peptide sequence impact the binding affinity of a TCR-pMHC complex, and ultimately, the T-cell response to the antigen. To do so, we characterize a score, which we term peptide energy, for a peptide σ , given its surrounding TCR and MHC structural environment,

$$E_{\text{peptide}}(\sigma; \text{TCR}, \text{MHC}) = \sum_{i=1}^{\ell} E(\sigma_i; \text{TCR}, \text{MHC}, \sigma_{/i}) \quad (6.1)$$

We assume that each peptide residue contributes linearly to the total energy by the amount $E(\sigma_i; \text{TCR}, \text{MHC}, \sigma_{/i})$, where σ_i is the amino acid at position i of the peptide. A residue’s energy contribution E is evaluated by HERMES (logits of the network), taking in as input the atomic composition of the surrounding TCR, MHC, and the rest of the peptide $\sigma_{/i}$ (excluding the i^{th} residue) within a 10 Å of σ_i ’s α -Carbon; see Fig. 6.1 and SI for details.

Model type	Model	TCR-pMHC affinity		T-cell activity							
		1G4 TCR	A6 TCR	H2-scDb	TCR1	TCR2	TCR3	TCR4	TCR5	TCR6	TCR7
Substitution Matrix	BLOSUM62 [87]	0.83	0.56*	-0.05	-0.07	0.14	0.38	0.33	0.42*	0.55	-0.01
Structure Prediction	TCRdock [26]	0.62*	0.88	N/A	0.02	0.13	0.21	0.18*	-0.02	0.19	0.06
	TCRdock-no template [26]	0.25	-0.14	N/A	-0.03	0.44	0.11	-0.22	0.28*	0.14	0.06
Sequence based ML	TAPIR [68]	0.29	0.35	N/A	0.36	0.36	-0.00	0.12	0.39*	0.08	-0.01
Structure based ML	ESM-IF1 [99]	0.38*	0.17	0.12	0.43*	0.41	0.19	-0.15	0.28*	0.25	0.08
	ProteinMPNN [51]	-0.01	0.49	0.55	0.46*	0.52	0.49*	-0.28	0.45	0.41*	0.31
Structure based ML	HERMES-fixed	0.29	0.63*	0.46*	0.56	0.71	0.59	-0.06	0.37*	0.42*	-0.01
	HERMES-relaxed	0.72*	0.56*	0.29	0.31	0.57	0.55*	-0.20	0.37*	0.42*	0.14*

Table 6.1: Benchmarking of models for predicting TCR-pMHC binding affinities and T-cell activities.

The table lists the Spearman ρ of models’ predictions against experimentally measured binding affinities (Fig. 6.2) and T-cell activities (Fig. 6.3) in response to peptide variants. The HERMES-fixed and HERMES-relaxed performance are from models with no noise (0.00) in training data, and the ProteinMPNN performance is from the model with a small noise amplitude of 0.02 Å; see SI for details. We show in **bold** the best model for each dataset, and any model whose performance is *not* significantly different from the best model (p -value > 0.05 associated with the difference in the Fisher’s z-transformed Spearman correlations) is indicated with an asterisk (*). See Tables D.7.2 and D.7.3 in the SI for p -values and additional models.

To compute peptide energy, we input to HERMES an experimentally or computationally determined structure of a specified TCR-MHC complex bound to at least one peptide variant. Experimental data on the impact of peptide substitutions on the TCR-pMHC structure is limited, and computational models often fail to capture subtle conformational changes in the structure due to amino acid substitutions in a peptide. Given these constraints, we adopt two approaches to estimate peptide energies across diverse peptide variants for a given TCR-MHC complex:

- **HERMES-fixed:** In the simplest approach, we choose a TCR-pMHC structure with the same TCR and MHC as our query and a peptide sequence closest to the peptide of interest. This structure is then used as our template in HERMES to compute the peptide energy as described in eq. D.1.1 (Fig. 6.1B). This method does not alter the underlying structure and assumes that peptide amino acid substitutions do not significantly change the conformation of the TCR-pMHC complex.
- **HERMES-relaxed:** Since amino acid substitutions can locally reshape a protein complex, we introduce a more involved protocol to account for these structural changes. We begin with the closest available structure for the TCR-pMHC complex. We then mutate the original peptide to the desired state and

apply a relaxation procedure using PyRosetta to pack the substituted amino acid [38] (SI). During relaxation, side-chain and backbone atoms of the peptide are allowed to move, while only the side-chain atoms of the MHC and TCR chains within a 10 Å radius of the peptide are flexible. We then calculate the peptide energy with HERMES (eq.D.1.1), using the relaxed structure as input (Fig. 6.1B). Since PyRosetta’s relaxation procedure is stochastic, we average the peptide energy across 100 realizations of these relaxations. For ablation purposes, we also include results obtained by selecting the relaxed conformation with the lowest (most favorable) Rosetta energy, and using its HERMES peptide energy.

6.2 Results

6.2.1 Predicting binding affinity of TCR-pMHC complexes

Binding between TCRs and pMHC complexes is necessary to mediate a T-cell response. The binding affinity of a natural TCR-pMHC complex is relatively low, with a dissociation constant K_d in the range of 1 – 100 μ M [54, 179, 264], in contrast to nanomolar affinities for antibody-antigen complexes. Engineered TCRs can achieve much higher affinities in a nanomolar [220] to picomolar [127, 63] range.

Surface plasmon resonance (SPR) spectroscopy provides accurate measurements of the dissociation constant (K_d) for specific TCR-pMHC systems. In these experiments, one protein—either the TCR or the MHC—is immobilized on a conducting plate, and the other is introduced in solution to bind to it. This binding alters the local refractive index near the plate, affecting the resonance signal. We used a published dataset of SPR-measured affinities for two TCR-MHC complexes binding to an ensemble of peptides, for which crystal structures of the complexes with at least one peptide are available [6, 176]; see Table D.7.1 for details.

In our first example, we examined TCR 1G4 binding to NY-ESO-1 peptide variants, which is a cancer-testis antigen commonly expressed in many cancers and is targeted by immunotherapies [66]. We used two structural templates, one with the wild-type peptide SLLMWITQC (NY-ESO-1), and the other with a more immunogenic peptide, in which the Cysteine at position 9 is substituted by Valine (C9V) [40, 41] (Fig. 6.2A). For variants differing by a single amino acid, our predictions with HERMES-*relaxed* correlated well (Spearman $\rho = 0.72$) with experimentally measured affinities from refs. [6, 176]; see Fig. 6.2B, C.

Next, we tested the A6 TCR binding to the HTLV-1 Tax peptide LLFGYPVYV. We employed two structural templates: one with the wild-type, the other with the mutant peptide with substitution Y8A [75] (Fig. 6.2D and Table D.7.1). Again, the HERMES-*relaxed* peptide energies correlated well (Spearman $\rho = 0.56$) with the affinities, measured in ref. [176] (Fig. 6.2E, F).

Overall, HERMES predictions aligned closely with these limited experimental affinity measurements, with HERMES-*relaxed* showing a better agreement with affinity measurements than HERMES-*fixed*. We compared our approach for scoring peptides to alternative methods, including the baseline substitution matrices (BLOSUM62 [87] and the TCR-specific substitution matrices from ref. [140]), the score from the TCR-pMHC structure prediction algorithm TCRdock [26], the sequence-based machine learning algorithms for TCRs (TAPIR [68], NetTCR [104], and TULIP [152]), and the structure-based inverse folding models (ESM-IF1 [99] and ProteinMPNN [51]); see Table 6.1 for key benchmarking results and Table D.7.2 and Fig. D.7.1 for a more extensive comparison of different methods, including the different HERMES models, and different scoring schemes by ProteinMPNN.

For the 1G4 TCR, BLOSUM62 [87] achieves the highest correlation with experimental affinity data, whereas template-guided TCRdock [26] leads for the A6 system; in contrast, TCRdock without using a homologous structure as a template underperforms. In both systems, HERMES achieves performance that is statistically indistinguishable from the top performers (Fisher’s z-test p-value > 0.05). Overall, BLOSUM62, template-guided TCRdock, and HERMES rank strongly as predictors of TCR-pMHC affinities across the tested datasets.

In addition, for the 1G4 system, ESM-IF1 [99] and the TCR-specific substitution matrices from ref. [140] also perform competitively. However, due to the limited number of affinity measurements, we lack statistical power to resolve finer differences among the models.

It should be noted that although BLOSUM62 is one of the best contenders for affinity prediction, its sequence-averaged construction ignores local structural context and therefore assigns identical substitution scores regardless of a residue’s environment (Fig. D.7.2); similar limitations apply to the TCR-specific substitution matrices from ref. [140]. HERMES, by contrast, is structure-aware and can attribute distinct scores to the same mutation when it occurs in different conformational settings—an ability that is critical for nuanced mutational forecasts (Fig. D.7.2). Determining the biophysical circumstances under which such context-agnostic scoring schemes suffice, and when structure-aware approaches become necessary, will require systematic evaluation on broader, more diverse datasets.

Lastly, we probed HERMES’s robustness to potential inaccuracies in structures generated by the widely used AlphaFold3 (AF3) algorithm [2], replacing the crystal structure inputs with AF3 models; see SI for details. Both the template-guided and the de novo AF3 structures remained within 1.1 Å RMSD of the respective crystal structures (Table D.7.4). For both HERMES and ProteinMPNN, replacing experimental structures with AF3 predictions altered predictions for the 1G4 and A6 TCRs binding to pMHC variants, but the changes were not statistically significant (Fisher’s z-test p-value > 0.05); see Fig. D.7.4. It is nonetheless expected that the predictions should suffer in the absence of reliable structural templates, as seen also in other methods such as TCRdock [26] (Tables 6.1, D.7.2).

6.2.2 Predicting T-cell response to peptide antigens

T-cell activation is shaped by factors beyond TCR-pMHC affinities, including the T-cell’s differentiation state, its cross-reactivity to self-antigens [9, 3], and avidity effects from TCR/co-receptor cross-linking [222]. Nonetheless, because readouts of T-cell activity (e.g., T-cell proliferation, and production of activation markers) are easier to obtain than SPR-based affinity measurements, we next asked whether HERMES scores, which are more attuned for predicting TCR-pMHC binding affinity, can serve as predictors of T-cell activities.

As the first system, we examined T-cell responses to all single-point mutants of a native peptide, as measured in ref. [140] for different TCR-pMHC systems. Specifically, we make predictions for (i) three TCRs (TCRs 1-3) recognizing variants of a highly immunogenic HLA-A*02:01-restricted human cytomegalovirus (CMV) epitope NLVPMVATV (NLV), (ii) three TCRs (TCRs 4-6) recognizing variants of the weaker HLA-A02:01-restricted melanoma gp100 self-antigen IMDQVPFSV, and (iii) a TCR (TCR 7) recognizing variants of weakly immunogenic HLA-B*27:05-restricted pancreatic cancer neo-peptide GRLKALCQR, believed to have elicited an effective immune response in long-term cancer survivors [140]. T-cell activity was quantified by determining the half-maximal effective concentration (EC_{50}) from the dose-response curves of 4-1BB⁺ CD8⁺ T-cells across varying peptide concentrations; see Methods on details of inferring EC_{50} .

With the exception of TCR1 (where a TCR-pMHC template with HLA-A*02:01 and one peptide variant was already available), we relied on AlphaFold3 (AF3) [2] to model the remaining TCR-pMHC complexes; see Table D.7.1 and for template information and AF3 sequence inputs for TCRs 2-7. As shown in Figs. 6.3A, and D.7.3, HERMES-*fixed* peptide energy predictions (eq. D.1.1) correlate well with experimental EC_{50} measurements for the CMV peptide variants, reaching up to Spearman correlation $\rho = 0.71$ for TCR2, and outperforming the alternative models. For TCR1 and TCR3, the performance of ProteinMPNN [51] is worse but statistically indistinguishable from HERMES (Tables 6.1, D.7.3, and Fig. D.7.1). Moreover, replacing the crystal structure for TCR1 with an AF3 model yields comparable performance, underscoring HERMES’s robustness to structural inputs so long as the structural models are of high quality (Figs. D.7.5, D.7.4). Correlations are lower for TCRs 4-6 targeting variants of the melanoma gp100 self-antigen, with HERMES

achieving Spearman correlation $\rho = 0.42$ for TCR6 as its best prediction in this class, with an accuracy comparable to ProteinMPNN [51] (Fig. 6.3B and Tables 6.1, D.7.3). HERMES underperforms for TCR7 responding to the pancreatic cancer neo-peptide GRLKALCQR (Fig. 6.3C), for which only the ProteinMPNN's predictions show significant albeit weak correlation with data (Table 6.1). The reduced accuracy of HERMES for cancer epitopes compared to the variants of the viral CMV epitope could reflect the impact of antagonism from self-antigens similar to tumor epitopes, hindering T-cell responses [9, 3]—though additional experiments are needed to confirm this.

Sequence-based models (TAPIR [68], NetTCR [104], and TULIP [152]) are outperformed by structure-based models tested here in all cases except for TCR4, which has the lowest AF3 confidence at the TCR-pMHC interface (i.e., a lower confidence in its fold); see Table D.7.4 and Fig. D.7.4 for the AF3 confidence metrics for all systems. Here, even the basic substitution models (BLOSUM62 [87] and the TCR-specific matrices from ref. [140]) outperform the structure-based predictors (Tables 6.1, D.7.3). This suggests that structural approaches hinge on a reliable structure as input, though more data is needed to draw definitive conclusions (Fig. D.7.4).

Next, we examined the therapeutic T-cell responses to all single-point mutants of the HLA-A*02:01-restricted p53-derived neoepitope p53^{R175H}, HMTEVVRHC, measured in ref. [97]. This T-cell therapy approach used a designed bi-specific Fab antibody that binds both to the TCR-CD3 complex on T-cells and to the variants of the p53^{R175H}-MHC complex on tumor cells (Table D.7.1). A strong binding to an epitope activates the engaged T-cell, and the extent of such activity was measured by the production of IFN- γ . The resulting HERMES-*fixed* predicted peptide energies (eq. D.1.1) show a Spearman $\rho = 0.46$ with the experimentally measured IFN- γ production (Figs. 6.3D, D.7.3). ProteinMPNN achieves a slightly higher correlation (Spearman $\rho = 0.55$), but this improvement over HERMES is not statistically significant (Fisher's z-test p-value > 0.05); see Table 6.1. Lastly, we found that replacing the crystal structure with the AF3 model substantially degrades HERMES and ProteinMPNN performances (Fig. D.7.4, Tables 6.1, D.7.3), as the AF3 model consistently mis-docks the bispecific antibody on the pMHC complex (Table D.7.4 and Fig. D.7.5).

Across the limited number of systems examined, our analyses indicate that—when a suitable structural template is available—HERMES can estimate epitope mutational effects at the pMHC-TCR interface and, in the H2-scDb case, at a Fab-pMHC interface. In this benchmark, ProteinMPNN performs comparably to HERMES, indicating that inverse-folding algorithms not explicitly trained for this task may nonetheless capture the impact of peptide mutations on T-cell responses. However, definitive conclusions about the generalizability of these methods to neoepitopes will require larger datasets with broader mutational coverage across peptide variants. Importantly, predictive performance for both approaches depends on access to a high-quality structure (experimental or computational), which limits applicability in systems without suitable structural models (Fig. D.7.4); see SI and Tables 6.1 and D.7.3 for a detailed benchmark of different approaches, including different HERMES models.

energy effectively predicts T-cell responses to neoepitopes presented by different MHC-I alleles. Moreover, HERMES predictions are not limited to TCR-pMHC interactions but also extend to antibody-pMHC interactions, commonly leveraged in T-cell therapies for their higher achievable affinities;

6.2.3 Design of novel peptides reactive to a TCR-MHC complex

TCR-engineered T-cell therapies are designed to target defined neoantigens, but specificity remains a central challenge. Engineered high-affinity TCRs can cross-react with untested self-peptides, causing severe off-target toxicity; in a MAGE-A3 directed trial, recognition of a Titin-derived peptide led to fatal cardiac events [133]. Therefore, developing approaches to predict the distribution of peptides recognized by engineered TCRs could enable more stringent preclinical screening and help mitigate off-target risk in patients.

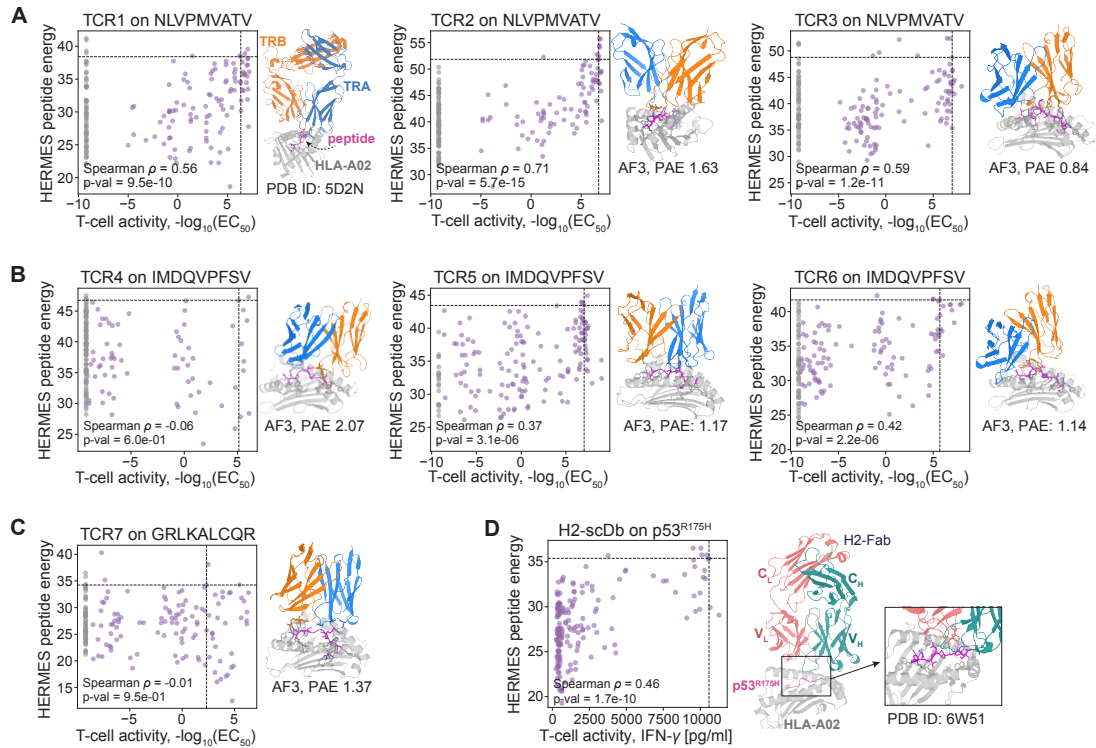


Figure 6.3: Predicting T-cell activity in response to peptide variants. (A–C) The predicted HERMES-*fixed* peptide energies versus experimentally measured T-cell activity are shown for (A) HLA-A*02:01-restricted CMV peptide variants interacting with TCR1–TCR3 (panels), (B) HLA-A*02:01-restricted melanoma gp100 peptide variants interacting with TCR4–TCR6, and (C) HLA-B*27:05-restricted pancreatic cancer peptide variants interacting with TCR7. In each case, T-cell activity in response to all single amino acid mutations from the wild-type is measured. T-cell activity is based on fitted EC_{50} values from dose-response curves of 4-1BB⁺ CD8⁺ T-cells (SI; data from [140]). Gray points indicate peptides with T-cell responses below detection limit (no EC_{50} obtained) and are excluded from the reported Spearman correlations. Except for TCR1 (resolved structure, PDB ID: 5D2N [258]), the other TCR-pMHC complexes were modeled with AF3, and the model’s peptide PAE for each wild-type peptide is denoted. These structures are used as templates to estimate peptide energies with HERMES-*fixed*; see Table D.7.1 for details on the templates. In all panels, TCR- α (TRA) is shown in blue, TCR- β (TRB) in orange, the peptide in magenta, and HLA in gray. (D) The predicted HERMES-*fixed* peptide energies are shown against experimentally measured T-cell activity for single point mutants of the HLA-A*02:01-restricted p53^{R175H} neoepitope from ref. [97]. T-cell response (IFN- γ production) is mediated through a bispecific antibody binding to the pMHC complex. The co-crystallized structure of the bispecific antibody with the p53^{R175H}-HLA-A*02:01 complex (PDB ID: 6W51 [97]) is used as the template for HERMES-*fixed* predictions. For all systems (A–D), the heatmaps for the predicted and the experimentally measured mutational effects are shown in Fig. D.7.3.

To map the peptide-recognition landscape of TCRs, we present a structure-guided pipeline for designing immunogenic peptides predicted to elicit strong TCR recognition when presented by MHC class I. Leveraging HERMES’s ability to predict T-cell responses to peptides presented by MHC-I, our approach begins with an existing template structure—experimentally or computationally resolved—of a TCR-MHC in complex with at least one peptide variant, which we refer to as the wild-type peptide. We then generate candidate peptides using two strategies, with different degrees of complexity; see Fig. 6.1C for a schematic description of the two pipelines.

In the basic pipeline (HERMES-*fixed*), we sequentially mask the atoms of each amino acid along the peptide, one at a time. Using the HERMES model, we predict the probability of each amino acid type for a given residue, based on the local structural environment within 10 Å of the masked residue. Repeating this procedure for every amino acid along the peptide yields a position weight matrix (PWM) that represents the

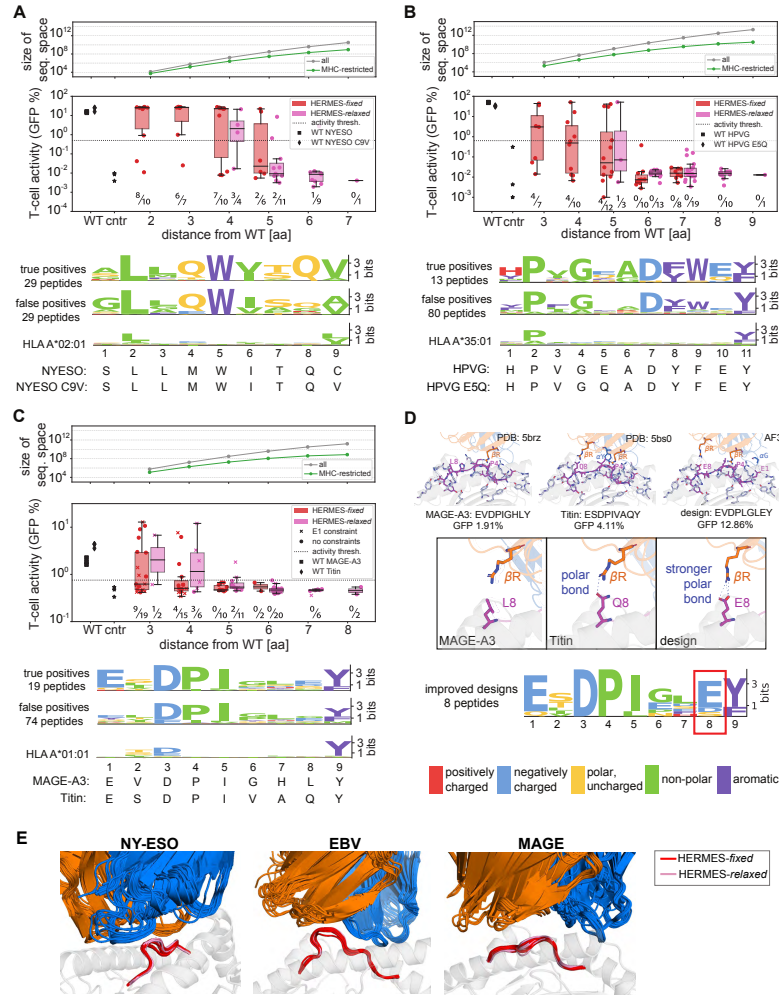


Figure 6.4: De novo peptide design to elicit a T-cell response. (A-C) Validation results are shown for designed peptides built upon the wild-type templates from (A) NY-ESO peptide against TCR 1G4 and HLA-A*02:01 (PDB IDs: 2BNR, 2BNQ [41]), (B) EBV epitopes against the TK3 TCR and the HLA-B*35:01 (PDB IDs: 3MV7 [84], 4PRP [136]), and (C) MAGE-A3 and Titin epitopes against an engineered TCR and HLA-A*01:01 (PDB IDs: 5BRZ, 5BS0 [187]). Designs were validated using Jurkat cells with endogenous NFAT-eGFP reporter expressing the paired TCR, interacting with peptide presented by the MHC in each system. The reported T-cell activities (percentage GFP-positive) are averaged over three replicates. The box plots show T-cell activity for designed peptides at different Hamming distances from the wild-type; the black line is the median, the box spans the 25-75% quantiles, and dots indicate individual data points. Red and pink boxes/markers correspond to designs with *HERMES-fixed* and *HERMES-relaxed* models, respectively; square markers denote wild-type peptides, and stars are negative controls (no peptide). The dashed line marks the activation threshold, and numbers below each bar show the relative number of successful designs out of total designs. The top plot in each panel shows the size of the peptide sequence space at different distances from the wild-types, with the gray line indicating the complete sequence space, and the green indicating the typical number of sequences likely to be presented by the MHC of each system; see SI for details. The bottom shows sequence motifs as PWMs for (i) true-positive designs that induced T-cell activation, (ii) false positives that did not, and (iii) the MHC presentation motif from the MHC Motif Atlas [226]. (D) 3D structures of TCR-pMHC complexes focused on the peptide are shown for MAGE-A3 (PDB ID: 5BRZ), Titin (PDB ID: 5BS0), and the highest-activity design in (C) modeled by AF3 [2]. The enhanced activity for Titin versus MAGE-A3, and for the designed peptide versus both wild-types follows from the formation of polar bonds between position 8 of the peptide (magenta) and the CDR3 loop of TCR β (orange). The PWM highlights the prominence of E₈ among the eight designs with improved T-cell activity in (C). (E) Structures predicted with AF3 for the activity-inducing designed peptides (above the dashed lines in A-C) are displayed in complex with their cognate TCR-MHC's. For each system, designs generated by *HERMES-fixed* (red) and *HERMES-relaxed* (pink) are overlaid after aligning the MHC molecule (in gray). Slight shifts are seen in the backbone conformations of the peptides and the TRA (blue) and TRB (orange) loops.

probabilities of different amino acids at each peptide position. Peptides are then sampled by drawing from this PWM. It should be noted that HERMES-*fixed* retains the structural fingerprints of the original peptide, as it does not account for local structural changes induced by amino acid substitutions; see Fig. 6.1C and D.7.6A, and SI for more details.

To incorporate structural flexibility and relaxation following amino acid substitutions, we introduce the design pipeline of HERMES-*relaxed*, using simulated annealing and Markov chain Monte Carlo (MCMC) sampling. We begin with the template TCR-MHC structure and a completely random peptide sequence, which is packed *in silico* within the structure using PyRosetta's packing functionality, followed by the Relax protocol [38]. During relaxation, both side-chain and backbone atoms of the peptide are allowed to move, while only the side-chain atoms of the MHC and TCR chains within 10 Å of the peptide are flexible. With this relaxed structure, we apply the HERMES-*fixed* method to generate a PWM and sample a new peptide, which is then packed and relaxed to form a new structure. We use MCMC to determine whether to accept the newly sampled peptide by comparing its HERMES peptide energy (favorability), evaluated in its relaxed conformation, to that of the original peptide (eq. D.1.1). We then iteratively perform this procedure, incrementally reducing the MCMC temperature at each step, until the results converge; see Figs 6.1C, D.7.6B-C, and SI for more details.

The peptides designed by both approaches are then filtered by TCRdock [26] or AlphaFold3 [2], using the Predicted Alignment Error (PAE) between TCR and pMHC interfaces, to assure folding and binding of the resulting TCR-pMHC structures. For each system, we define a specific PAE threshold close to the TCRdock PAE of the system's TCR-MHC in complex with the native (wild-type) peptide, which is known a priori to activate T-cells. As a result, the PAE thresholds vary slightly across different systems; see SI for details.

We tested the accuracy of our peptide design pipeline in three systems, with the structural templates taken from (i) the cancer-testis antigen NY-ESO-1 in complex with the TCR 1G4 and HLA-A*02:01, (ii) a peptide derived from the Epstein-Barr virus (EBV) in complex with the TK3 TCR and the HLA-B*35:01, and (iii) the immuno-therapeutic target MAGE-A3 in complex with an engineered TCR and HLA-A*01:01. For brevity, we refer to these three systems as NY-ESO, EBV and MAGE, respectively; See Table D.7.5 for information on the structure templates used in each case. These systems were selected because each one has high-resolution crystal structures of the same TCR-MHC scaffold bound to at least two distinct peptide variants, and that they span a therapeutically relevant spectrum that includes both engineered (for MAGE and NY-ESO) and natural (for EBV) TCRs directed against viral as well as tumor antigens. It should be noted that as we progressed from NY-ESO to EBV and MAGE, we chose to explore the sequence space farther from the wild-type peptides, leading to more challenging designs.

We validated our designs using Jurkat cells expressing a specific TCR and an endogenous NFAT-eGFP reporter to indicate T-cell activation in response to different peptides. We tested the TCR-MHC specificity in all systems—NY-ESO, MAGE, EBV—under 96 different conditions, including the de novo designed peptides, positive controls (wild-type peptides from the template structures), and an unstimulated “no peptide” control. Peptides were presented on artificial APCs (aAPCs), expressing the specified HLA in each system. For NY-ESO designs, we determined the the percentage of GFP-positive cells through both flow cytometry and fluorescence microscopy. Given the consistency between the two experimental approaches (Fig. D.7.7) we relied on fluorescence microscopy only to measure GFP levels induced by different peptides for MAGE and EBV designs; see Appendix for experimental details.

We used two structural templates for the NY-ESO system: one containing the original 9-amino acid peptide SLLMWITQC, and another with the more immunogenic C9V substitution [40, 41]. Our de novo designs differed from the closest wild-type peptide by 2 to 7 amino acids, with a TCRdock PAE ≤ 5.5 . For this system only, we selected 35 negative designs (designed by HERMES, but with PAE > 5.5) for experimental testing: only one of these was a false negative, which interestingly, had a desirable AF3 PAE score (as low as the wild-type peptides); see Figs. D.7.8, D.7.9. For the 58 positive designed peptides (PAE ≤ 5.5), we

achieved an overall design accuracy of 50%, with GFP levels significantly higher than the negative control; see SI for details. The accuracy of predictions decreases as the sequence divergence of the de novo designs from the wild-type peptides increases (Fig. 6.4A, D.7.9). HERMES-*fixed* achieved higher design accuracy of 70% within its smaller designed sequence subspace of up to 5 substitutions from the wild-types, whereas HERMES-*relaxed* explored sequences further from the wild-type peptides (4-7 substitutions) but with a reduced accuracy of 24% (Fig. 6.4A). The main difference between the true and false positive sequences was at position 8—from strongly preferring glutamine within the true positives, to having similar preferences among glutamine, glycine, arginine, or methionine within the false positives (Fig. 6.4A).

For the EBV system, we used two structural templates: one with the 11-amino acid peptide HPVGEADYFEY—commonly referred to as HPV—derived from the viral latent nuclear protein EBNA-1, and the other with an epitope from a wide-spread viral variant, with a glutamine at position 5 [136]. Unlike NY-ESO, the HPV peptide has a more flexible conformation with a helical turn within the MHC binding pocket. Our de novo designs differed from the closest wild-type peptide by 3 to 9 amino acids, with a TCRDock PAE ≤ 5.1 . Across the 93 de novo designs, we achieved an overall accuracy of 14%, associated with the peptides that induced significant GFP levels in T-cells. The accuracy of predictions decreases with increasing the sequence divergence from the wild-types (Fig. 6.4B, D.7.10). Notably, we achieved 40% accuracy among the designs within 3-5 amino acid distance of one of the wild-type templates, while none of the designs with larger than 5 amino acid distance were successful. Lastly, we observe a strong preference for glutamic acid at position 10 of the successful designs relative to the false positives (Fig. 6.4B).

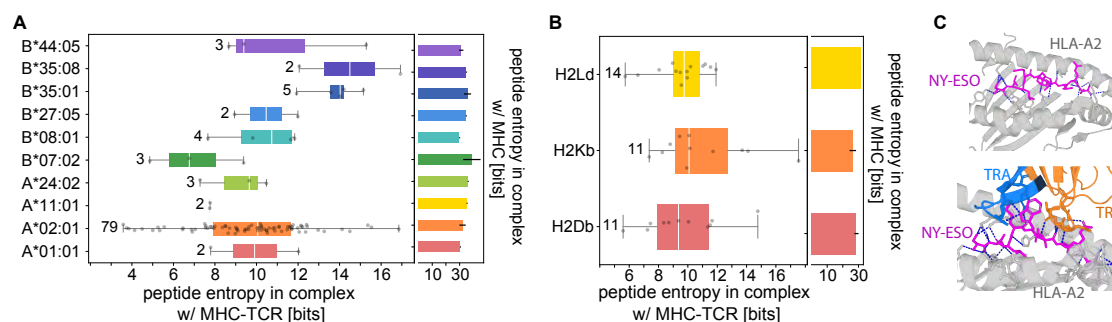


Figure 6.5: Evaluating T-cell specificity from de novo designed peptides. (A) T-cell specificity is measured by peptide entropy, which is computed from the HERMES-*fixed* predicted amino acid likelihood matrix, starting from a TCR-pMHC structural template. Each point on the plot corresponds to the entropy estimate from a distinct TCR-pMHC template, grouped by their MHC allele. The box plots display the distribution of peptide entropies for TCR-pMHC templates sharing the same MHC allele, with a white line marking the median and the box spanning the 25% to 75% quantiles. The number on the side of each box indicates the number of distinct structural templates we used for that MHC class; see Table D.7.6 for details on these structures. The side plot shows peptide degeneracy for MHC presentation alone, evaluated as the entropy of the peptide position weight matrices gathered from the MHC Motif Atlas [226]. The error bars indicate the standard deviation of peptide degeneracies across motifs of different lengths, associated with a given MHC allele. (B) Similar to (A) but for degeneracy of peptides interacting with TCRs and MHCs in mice. (C) Polar bonds (blue dashed lines) between the NY-ESO peptide (magenta) and HLA-A*02:01 (gray) is shown at the top (PDB ID: 1S9W [249]). The additional polar bonds formed between the peptide and TRA (blue) and TRB (orange) of an interacting TCR are shown at the bottom (PDB ID: 2BNR [41]). These additional polar bonds limit the degeneracy of the peptides that can interact with a TCR-MHC complex compared to those presented by the same MHC.

For the MAGE system, we used two structural templates: one with the 9-amino acid MAGE-A3 peptide EVDPIGHLY and another with the Titin-derived self-peptide ESDPIVAQY, which is expressed in cardiomyocytes [187]. off-target response of the affinity-enhanced MAGE-A3-specific engineered TCR that resulted in severe cardiac toxicity and deaths of two patients [133]. Our de novo designs differed from the closest wild-type peptide by 3 to 8 amino acids, with a TCRDock PAE ≤ 5.2 (Fig. 6.4C). From the resulting 93 de novo designs, 20% activated T-cells with significant levels of GFP expression. Similar to the other systems, the accuracy of predictions decreases with increasing the sequence divergence from the wild-types (Fig. 6.4C).

We achieved 30% accuracy among the designs with 3-5 to amino acid distance, while none of the designs with larger than 5 amino acid distance from the wild-types were successful.

Fig. 6.4C shows differences in amino acid composition between successful and failed designs. Alanine and glycine scanning experiments have previously suggested that MAGE-A3 peptide variants with an E-DPI--Y motif activate T-cells in this system [36]. Consistent with these findings, the fraction of T-cell activating variants in our designs increases as their sequences more closely resemble this motif (Fig. D.7.11).

In a subset of designs using the MAGE-A3 template, we fixed the glutamic acid at position 1 (E_1), which is part of the E-DPI—Y pattern. In both MAGE-A3 and Titin E_1 is within 4 Å of the MHC and forms strong polar interactions with it (Fig. 4D). Fixing E_1 improved the design accuracy to 1/3, compared to 1/9 when this amino acid was not constrained. Even without this constraint in the design protocol, E_1 appeared in 2/3 of successful designs, and in none of the unsuccessful ones (Fig. D.7.11). Notably, among the more distant designs made with HERMES-*relaxed*, only those containing a fixed E_1 were able to activate T-cells (Fig. 6.4C). This suggests that constraining essential amino acids can enable broader exploration of sequence space at other positions, possibly due to epistatic interactions. However, further investigation is needed to confirm this.

Several of our designs outperformed the wild-type peptides in activating T-cells: eight surpassed Titin's GFP levels, and eleven exceeded MAGE-A3's (Fig. 6.4C). These improved designs strongly favor glutamic acid (E) at position 8 of the peptide, where MAGE-A3 has leucine (L), and Titin has glutamine (Q) (Fig. 6.4D). In Titin, the glutamine forms a polar bond with an arginine (R) in the interacting TCR β , and it is likely responsible for increasing Titin's activity relative to MAGE-A3's. The glutamic acid substitution in our designs further strengthens this polar bond (Fig. 6.4D), resulting in a stronger T-cell response. For NY-ESO and EBV, the wild-type peptides were strongly immunogenic and induced high levels of GFP, and many of our designs induced comparable T-cell responses, but did not improve upon those (Fig. 6.4).

To gauge the impact of using AF3 structures as design templates, we reran the HERMES-*fixed* protocol with AF3 predicted structures (with and without template guidance), and obtained peptide PWMs that were almost identical to those derived from using crystal structures (SI and Fig. D.7.12). An interesting exception is the MAGE system, where using the AF3 template introduced a strong preference for glutamic acid at position 1 of the peptide (Fig. D.7.12C), mirroring the pattern observed among the designed peptides that induced strong T-cell response (Fig. 6.4D).

Although no direct experimental comparisons were performed, computational evaluations using TCRdock PAE and presentation score by MHC-I alleles indicate that HERMES designs outperform or match those generated by ProteinMPNN [51], the corresponding MHC position weight matrix [142, 141, 11, 226], or the BLOSUM substitution matrix [87] (SI and Figs. D.7.13, D.7.14, D.7.15). A notable exception is in the MAGE system, where ProteinMPNN designs generally have more favorable PAE scores, albeit with a lower proportion of designs passing the antigen presentation filter by the MHC molecule (Fig. D.7.15).

In all the three systems we used the PAE values from TCRdock and AF3 as filtering criteria to select designs for experimental validation (see SI). The PAE values reported by the structure prediction algorithms have been shown to be noisy indicators of protein folding fidelity and design quality [196]. While larger experimental libraries are needed to draw definitive conclusions, it appears that PAEs from TCRdock and AF3 have some complementary predictive strengths (Figs. D.7.8, D.7.16), and therefore, combining their information could lead to a more robust design decisions (SI). Both of these scores correlate only weakly with HERMES (Figs. D.7.8, D.7.13, D.7.14, D.7.15), so they can provide complementary design filters. Furthermore, designed peptides with higher (more favorable) HERMES energies more often activated T-cells (Fig. D.7.16), suggesting that a more restrictive sampling (e.g., with a lower MCMC temperature for HERMES-*relaxed* protocol) could yield a higher success rate, albeit at the expense of peptide diversity.

Overall, HERMES demonstrates a potential for designing highly immunogenic peptides up to five amino acid substitutions away from the wild-type—a task that would otherwise require exploring a vast sequence space of roughly 10^8 to 10^{10} possibilities for a 9-residue to an 11-residue peptide (Fig. 6.4A-C). This capability raises an important question: to what extent do these substantial changes in sequence alter the binding geometry of the peptide inside the MHC-TCR cleft? To this end, we used AF3 to model the TCR-pMHC interfaces for the experimentally validated immunogenic designs from the three systems. Although we see slight variation in both peptide and TCR backbone conformations, overall, we found that our designed immunogenic peptides adopt conformations very close to their wild-type templates, even when those designs were generated with *HERMES-relaxed* (Fig. 6.4E). We attribute this structural convergence to two potential factors: (i) residual limitations in HERMES peptide-backbone sampling strategy, and (ii) an inherent AF3 bias toward conformations it encountered during training. Given that TCRs can cross-react with peptides of markedly different conformational poses [191], developing methods that can better explore peptide conformational space within the TCR-MHC groove remains a compelling direction for future research.

6.2.4 Structural basis for TCR specificity

TCRs exhibit substantial degeneracy in their recognition, with some autoimmune T-cells experimentally shown to recognize over one million distinct peptides [253]. However, the full extent of TCR degeneracy for typical receptors remains unclear, largely due to the limitations in high-throughput experimental assays for TCR recognition of many peptides presented on different MHCs.

Our computational peptide design framework helps address this limitation, at least for the TCR-MHC complexes with known structural templates. Specifically, for a given TCR-MHC pair, we can use *HERMES-fixed* protocol to generate a peptide position weight matrix (PWM), representing the ensemble of peptides presented by the MHC and recognized by the TCR. We use the entropy of this PWM as a proxy for TCR degeneracy.

Fig. 6.5A shows the entropy of these peptide distributions for 105 TCR-MHC pairs across 10 human MHC-I alleles; see Table D.7.6 for details on these structures. The median peptide entropy across these TCR-MHC pairs is approximately 10 bits, indicating that a typical TCR-MHC pair can recognize on the order of $\approx 10^3$ peptides. In contrast, examining the degeneracy of MHC-I presentation alone in humans—using the peptide PWMs gathered from the MHC Motif Atlas [226]—yields a median entropy of 31 bits, implying that MHC-I molecules can typically recognize $\approx 2 \times 10^9$ distinct peptides (Fig. 6.5A). A similar analysis on 36 TCR-MHC pairs spanning three mouse MHC-I alleles shows comparable trends, with a MHC-I degeneracy on the order of $\approx 5 \times 10^8$ peptides (29 bits), and the TCR-MHC degeneracy of $\approx 10^3$ peptides (10 bits); see Fig. 6.5B, and Table D.7.6 for details on these structures. This pronounced reduction in entropy—observed by comparing the distribution of peptides recognized by MHC-I molecules alone to that recognized by TCR-MHC-I complexes—quantifies how TCRs constrain the accessible antigenic shape space, as illustrated by the polar interactions that they form with peptides in Fig. 6.5C.

As an alternative to MHC Motif Atlas [226], we explored the ability of *HERMES-fixed* to model MHC degeneracy. When structural coverage is extensive (roughly > 30 structures of distinct peptides for a given MHC allele), the motifs produced by HERMES align closely with those in the MHC Motif Atlas (SI, Fig. D.7.17, and Table D.7.7). With sparser structural coverage, HERMES underestimates the peptide diversity, likely because backbone conformations are insufficiently sampled. Given the imbalanced distribution of structures across MHC alleles, we therefore continue to use MHC Motif Atlas [226] as our reference for computing MHC motif specificity.

Because our analyses focus on interactions involving typical TCR and MHC molecules, our entropy-based degeneracy estimates are smaller than those from previous theoretical studies, which often relied on

experimental observations from autoimmune T-cells or limited synthetic peptide libraries [210]. Moreover, the HERMES-*fixed* model generates PWMs based on peptides whose backbone conformations match those of the wild-type template, likely underestimating the true degeneracy of TCR-MHC complexes (Fig. 6.5). Although HERMES-*relaxed* can partially address this issue, its designs are only reliable within roughly five amino acids of the native peptides with similar backbone conformations (Fig. 6.4), preventing a full exploration of the possible peptide poses for a given TCR-MHC complex. Nonetheless, our estimates can serve as informative lower bounds on typical MHC-I and TCR degeneracies.

6.3 Summary

In this chapter, we introduced a structure-based machine learning model to predict TCR-pMHC interactions and to design novel reactive peptides for TCR-MHC complexes. Our approach builds on HERMES, a physically motivated equivariant neural network model that we previously developed [238, 182]. Trained on the protein universe, HERMES predicts amino acid propensities based on their atomic environments in protein structures, and has been shown to implicitly learn biophysical interatomic interactions in proteins [182], enabling it to make highly generalizable predictions. This generalizability is particularly important for TCR-pMHC complexes, where only a modest number of experimentally resolved structures are available, and computational algorithms often struggle to accurately predict new structures [26, 260].

We found that HERMES-predicted peptide energies closely tracked both TCR-pMHC binding affinities and peptide-induced T-cell activities across diverse systems, including variants from a CMV-derived viral epitope [140], the NY-ESO cancer-testis antigen [6, 176], and a melanoma self-antigen [140]. Notably, HERMES also accurately predicted the activity of p53-derived neoepitope variants in the context of a bispecific antibody-mediated T-cell therapy [97]. This result suggests the algorithm’s potential applicability: given a reliable structural template close to the system of interest, HERMES can predict peptide-induced T-cell responses, whether through direct TCR interactions or antibody-mediated mechanisms. However, drawing more definitive conclusions about the generalizability of HERMES would require examining a larger datasets of TCRs interacting with pMHC variants.

Based on the HERMES predictions for TCR-pMHC reactivities, we presented a computational protocol for de novo design of immunogenic peptides that can elicit robust T-cell responses. Starting from the structural template of a TCR-MHC with a native (wild-type) peptide, we developed an MCMC search guided by HERMES peptide energies to explore the vast sequence space of roughly 10^8 (10^{10}) possibilities for 9-residue (11-residue) peptides, to identify immunogenic candidates. Experimental validation across three systems—with native peptides from NY-ESO and MAGE cancer epitopes, plus a viral epitope from EBV—revealed that our designs reliably activated T-cells, even with up to five substitutions from the native peptide used in the structural template. Notably, several MAGE-derived variants outperformed or matched the native peptide in T-cell activation, demonstrating the method’s potential to enhance peptide potency.

This capability can be relevant to personalized peptide-based cancer vaccines, which aim to induce durable T-cell memory against a patient’s tumor antigens to reduce the risk of rebound [202, 243, 195]. Despite their promise, the success of cancer vaccines has been limited by low peptide immunogenicity and poor uptake by antigen-presenting cells [219]. Recent approaches have employed deep mutational scanning experiments to identify more immunogenic cancer neoantigens against specific TCR candidates [140]. However, these experiments are often limited to scanning single amino acid mutations from the wild-type peptide due to the combinatorial explosion when exploring multiple mutations. Moreover, the precise TCRs responding to peptide vaccines in a patient are often unknown, and many immune responses are induced de novo [243]. With the caveat of access to suitable structural templates, HERMES could inform T-cell vaccine design by computationally generating diverse peptide libraries predicted to engage a broad set of candidate TCRs

(identified independently), nominating vaccine epitopes for subsequent safety and efficacy testing.

These results show the potential of HERMES for designing peptide-based cancer vaccines with neoantigens, providing an avenue for personalized immunotherapy strategies.

Accurately modeling T-cell specificity remains challenging, in part due to the limited functional data on TCR-epitope pairs, and the highly cross-reactive nature of TCR-epitope interactions. A design protocol like the one presented here, coupled with experimental validation, offers a promising path forward for future studies. By iteratively generating candidate peptide libraries, testing them experimentally, and refining the model with each iteration, it should be possible to more efficiently explore the high dimensional antigenic sequence space. This active learning strategy has a potential for revealing the “shape space” of TCR-epitope interactions, with broad applicabilities for early disease diagnosis, and for design of targeted immunotherapies. Moreover, this approach can help mitigate immunotherapy-related toxicity by identifying and excluding potential self-reactivities—beyond single-point mutations in the target epitope—such as the off-target Titin reactivity observed in an engineered T-cell therapy against MAGE-A3 that led to a severe cardiac toxicity [133].

Although our findings highlight the promise of structure-based models for immune recognition, a key limitation remains the scarcity of high-quality TCR-pMHC structural templates, despite advances from tools like AlphaFold3 [2] and TCRDock [26]. On the other hand, while TCR sequences are more widely available, sequence-based models often lack generalizability because existing paired TCR-pMHC data are skewed toward just a few peptides [216, 242]. A promising direction for future work lies in developing multi-modal flexible methods that integrate structural and sequence-based models, leveraging both the depth and generalizability offered by structural information and the scalability of sequence data.

TCRs can cross-react with diverse peptides in multiple orientations and conformation [191]. To reflect this flexibility, we sampled peptide conformational space (i) implicitly, by applying noise to the coordinates during training, and (ii) explicitly, by locally relaxing the atomic environment after each amino acid substitution. Even so, AlphaFold3 predicts that our experimentally validated immunogenic designed peptides retain similar structural poses as their wild-type templates. This structural convergence may partly stem from AlphaFold3’s bias toward structures represented in its training set, but it also exposes limitations in our current sampling strategy. Future work should therefore aim to co-design peptide sequence, peptide backbone geometry, together with the CDR loop conformations of the interacting TCR, enabling a more comprehensive exploration of the peptide-TCR conformational landscape.

Although this work primarily examines MHC class I-restricted peptides, the broad applicability of structure-based modeling suggests that similar methods could be adapted for the more complex TCR-peptide-MHCII interactions, thereby offering new insights into CD4⁺ T-cell responses. Moreover, incorporating additional immunological factors—such as TCR clustering, co-receptor interactions, cross-reactivity with self-antigens, and functional immune profiling—can further enhance the accuracy of T-cell response predictions and improve the resulting de novo peptide design pipeline.

Chapter 7

CONCLUSION AND OUTLOOK

Creating accurate models describing how a protein carries its function holds promise to both enable and speed up scientific discovery and translational research. Since almost all protein functions derive from the inter-atomic interactions among the proteins' constituent amino-acids, first-principles methods rooted in physics are established approaches for building models that generalize across diverse protein functions. Despite their promise of generalizability, physics-based methods are still greatly limited in usability by slow speeds and high need for computational resources.

In recent years, machine learning (ML) - driven by highly expressive neural networks - has risen to fill this gap by substituting physics-based first-principles methods with effective models learned from data, each specific to a single or a circumscribed set of tasks and to a particular representation choice. The main appeal of ML models is that they can be both fast and accurate, thus enabling high-throughput applications and analyses that are often not feasible with physics-based methods. However, balancing the trade-off between speed and accuracy rests on carefully curating model expressivity, training data, and choice of representation, such that the learned "shortcuts" that render the effective model fast are also predictive of real-world phenomena.

In this thesis, I attempted to achieve this balance by applying the very first-principles thinking that ML has been trying to render obsolete. Indeed, the principle that a protein successfully carries its function owing to a circumscribed set of local inter-atomic effects can help machine learning research just as much as molecular dynamics. Therefore, I developed data-driven models of *local atomic environments*, reasoning that the resulting effective models would capture the very local inter-atomic effects that we know drive protein function, thus rendering the models predictive across diverse functional settings.

Just as inter-atomic interactions are a fundamental principle underlying protein function, one of their core elements is their symmetry with respect to rotations and translations in 3D: the global orientation of the protein or protein-protein complex has no bearing on the occurring interactions. The field of Geometric Deep Learning offers a principled framework to constrain neural networks to make predictions that respect desired symmetries without the use of additional training data. This ensures that the learned effective models possess these symmetries, deleting at the root the risk that they have learned symmetry-breaking shortcuts.

In this thesis, I combined these first-principle considerations into a unified neural network framework for building *rotationally-equivariant models of residue-centered atomic neighborhoods*, which is described in detail in Chapter 2. I then applied this framework to develop several models spanning diverse aspects of protein structure and function, which all achieve predictive performances closely matching or exceeding the state-of-the-art, evidencing the efficacy of the proposed framework.

In Chapter 3, I applied the $SO(3)$ -equivariant framework to the domain of unsupervised learning, developing the Holographic (Variational) AutoEncoder (H-(V)AE). H-(V)AE's key idea is to disentangle the latent space into a maximally informative invariant embedding - which learns a compact representation of the input in a "canonical" orientation - and a single rotation matrix that learns the orientation of the input relative to the "canonical" one. Through simple experiments on spherical images, we provide evidence that the "canonical" orientation is one where all the training samples are maximally overlapped on top of each other. Finally, when applied to protein neighborhoods H-(V)AE was able to learn compact embeddings that efficiently encode key biophysical and structural properties of the neighborhoods, and that can be successfully used as features for downstream tasks; indeed, pairing them with a Random Forest Regressor [29] resulted in

state-of-the-art performance on predicting protein-ligand binding affinity on the ATOM3D Ligand Binding Affinity dataset [231]. Furthermore, H-(V)AE neighborhood embeddings are currently being used by a colleague to easily identify structural features that are indicative of a protein’s functional family, following similar principles as [201, 48].

Then, in Chapter 4, I used the SO(3)-equivariant framework to tackle the task known as "rotamer packing", which concerns finding physically correct side-chain conformations that are consistent with a backbone structure and an amino-acid sequence that is predicted to fold into the backbone. This task is common in protein design pipelines, as it complements models that design backbone structures [248, 125] and amino-acid sequences [52] without modeling exact atomic placement of the side-chains. The method I proposed first predicts each side-chain conformation using a local model of backbone-atom and spatially localized amino-acid labels, and then iteratively refines the predictions using a model of the previously-predicted all-atom conformations. The resulting algorithm - H-Packer - is not state-of-the-art on common benchmarks, but nonetheless its sufficiently high performance, speed and ease-of-use have enabled it to be widely used by the community, especially through the popular BioEmu codebase [125].

In Chapter 5, I revisited the work by my former colleague and mentor, Michael Pun, on H-CNN, a model trained to predict the likelihood of finding each amino-acid in a local atomic context. In his work, Mike provided proof-of-principle results that the learned likelihoods could be used to zero-shot the effects of mutations on a protein’s stability and on protein-protein binding affinity. In this Chapter, I contributed several methodological improvements, culminating in the HERMES family of models. These improvements include: a $\times 2.75$ faster and easier to optimize neural network architecture; improved modeling of local relaxation as a result of mutations at no inference-time cost, with the model known as HERMES-*amortized*; an efficient pipeline for fine-tuning HERMES on desired mutation effects, used to create HERMES-*amortized* as well as state-of-the-art models of stability effects when fine-tuning on experimental stability measurements; a clean code repository, easy to install and use even for practitioners. I also probed HERMES and competing models for their strengths, weaknesses, and biases, tying them to methodological design choices. These analyses, other than serving as blueprints for the community, reveal that HERMES’ full-atom input endows it with a strong preference for steric-preserving substitutions, and also that pre-training on wild-type amino-acid classification introduces a wild-type preference bias that is stronger for proteins with closer homologs in the pre-training data. Finally, I provided evidence that HERMES-*amortized* can identify mutations known to stabilize viral antigens in a conformation amenable for vaccine development, and presented an engineering pipeline that leverages HERMES-*amortized* to speed-up vaccine development.

Finally, in Chapter 6, I showed further evidence for the impact that HERMES can have on biological discovery. Specifically, I studied the interaction between a given T-Cell Receptor (TCR) and a peptide antigen presented on an MHC molecule. These interactions are key events that regulate adaptive immunity, and their modeling has been largely elusive, owing to a lack of exhaustive data, coupled with the high level of degeneracy characteristic of the system (a single TCR can interact with many peptides, and vice-versa). In this chapter, I used HERMES to model the distribution of peptides that are presented by a specific MHC, and recognized by a specific TCR, a problem known among immunologists as that of TCR specificity. Our results show that HERMES, without almost any specific knowledge¹ of TCR-pMHC interactions, but access to a reliable wild-type TCR-pMHC structure, can be used to design a broad range of peptides that a target TCR interacts with. HERMES not only successfully identifies peptide residues that do not contribute to TCR binding and thus can be ignored, but is also able to suggest peptides that engage in stronger interactions with the TCR than the wild-type. This chapter is not only a testament to HERMES’ ability to model amino-acid substitutions in diverse structural contexts, it is also a testament to the importance of paradigm shifts. Indeed, prior to this work, most research on TCR specificity prediction and design was performed using sequence representations

¹barring what was in the pre-training data, which deduplication at 30% similarity greatly reduced, see Chapter 5

and relied on experimental training-data distributions heavily skewed towards high TCR diversity and low peptide diversity. Instead, using a structural representation relieves us from relying on experimental data for training, and enables the use of a zero-shot model like HERMES. This of course comes at the expense of having a reliable TCR-pMHC structure in the first place, which is not always available and ignores any possible alternative conformation.

The successes of the methods presented in this thesis are a testament to the effectiveness of the design choices that underlie the structure-based, residue-centered framework used to build all of them. This effectiveness likely hinges on two main factors.

First, the neural networks built with this framework are particularly efficient at modeling the atomic environments to the degree that is required by the tasks considered in this thesis. Indeed, the $SO(3)$ -equivariant neural networks built in this thesis are strictly less expressive than several $SE(3)$ -equivariant alternatives commonly referenced in the atomic modeling literature [16, 129, 230]. Their expressiveness - and associated computational complexity - is however unnecessary for the tasks considered here. In fact, those neural networks are often used to fit force measurements *on each atom*, and thus require representing the neighborhood and all the interactions with high atom-level precision. By contrast, the neural networks in this thesis encode atomic environments into a lossy representation, and processes it with way fewer expensive tensor product operations. However, they are also only required to make predictions *for the focal residue*, a generally more “coarse-grained” setting that likely requires less precise details. Therefore, the success of the models in this thesis is indicative that less expressive models are sufficient for the tasks considered here. Finding the right level of expressiveness for a task is important: not only because lower expressiveness reduces the risk for overfitting [19], but also because it often reduces computational complexity, and the risk that something might go wrong during model development. Indeed, strictly equivariant and expressive models [16, 129, 230] have been known to be both very computationally expensive and difficult to optimize, to the point that recent works have been trying to relax the constraint of equivariance to improve optimization properties [175]. Instead, I believe the framework presented in this thesis will stand the test of time owing to its balanced simplicity, which keeps it fast and easy to optimize without compromising on equivariance.

Second, the focus on modeling local atomic environments directly results in the learning of effective models over the majority of inter-atomic and inter-residue interactions that render single protein folds and protein-protein interactions stable, which are hallmarks of protein function. This is in contrast for example with models trained to model amino-acid distributions in functional protein sequences without worrying about structure, which have been shown to capture meaningful correlates between amino-acid variability and functional variability only as deeply as evolution has sampled them [263].

The design choices of locality and focusing on static atomic environments also pose obvious limits. For example, as discussed in Chapters 5 and 6, these choices limit the accuracy of HERMES to mutation effects that are driven by local interactions surrounding the mutated site, and in particular with minimal necessary side-chain relaxations upon mutations (a limit only partially mitigated by HERMES-*amortized*). However, what is most dangerous about these limits - and others that may be unveiled - is the ignorance of them. Indeed, these limits simply define the domain in which the learned effective model is applicable. Therefore, characterizing them provides a “how to” manual for when the model is applicable and what it can accomplish: not much different from a function contract in software engineering. Knowledge of these limits also turns the model into a hypothesis generation machine. For example, the success of HERMES-*fixed* on scoring and design of TCR-pMHC interactions could be seen as evidence that TCR-pMHC interactions are largely driven by local effects, and that neural substitutability at TCR-pMHC interfaces is largely driven by preservation of steric properties - a hypothesis that is actually supported by a complementary line of evidence [186]. Last, being able to make useful predictions with low computational requirements and high speed unlocks use cases of the model that are not feasible with slower albeit potentially more accurate approaches.

Nevertheless, just like HERMES-*amortized* improved the model's performance without changing its speed, other improvements are possible within the framework. A relatively small improvement would be distilling the predictions of each 10-model HERMES ensemble into a single model [91], to further increase inference speed and make the models even more CPU-friendly. I would also want to estimate prediction uncertainty for both HERMES and H-Packer. This would have the primary purpose of guiding users' expectations with respect to model predictions, but also the predicted uncertainty itself may be predictive of relevant physical quantities, for example side-chain conformational entropy in the case of H-Packer. A natural place to start would be reporting and analyzing the 10-model ensemble variance of HERMES.

Thinking instead about large-scale extensions of the framework, insightful future work would be training HERMES without the full side-chains, but still with local neighborhoods, and using an input parameterization similar to that of the *initial guess* H-Packer model. This would disentangle locality and all-atom design choices, resulting in a model with inductive biases "in-between" those of HERMES and of ProteinMPNN. Finally, I envision the efforts in this thesis to potentially be unified in a single model that performs auto-regressive prediction of both amino-acid labels and side-chain configurations, with the additional inclusion in the neighborhood featurization of atomic types beyond those found in proteins - like phosphorus and common metallic ions. Such a model would still only use local atomic neighborhoods as input, but it would include "mask" and amino-acid-specific tokens to handle the cases in which amino-acid labels and side-chain conformations are respectively unknown; in many ways it would be a direct competitor of LigandMPNN [53], but developed with strict locality and our rotationally equivariant framework. This model would be a direct substitute of both HERMES and H-Packer, and in addition be able to (1) design structure-conditioned sequences without assuming side-specific independence, and (2) better design protein interfaces against non-protein molecules like DNA, ions, and small-molecule ligands.

REFERENCES

- [1] Josh Abramson et al. “Accurate structure prediction of biomolecular interactions with AlphaFold 3”. en. In: *Nature* 630.8016 (June 2024). Publisher: Nature Publishing Group, pp. 493–500. ISSN: 1476-4687. DOI: 10.1038/s41586-024-07487-w. URL: <https://www.nature.com/articles/s41586-024-07487-w> (visited on 05/01/2026).
- [2] Josh Abramson et al. “Accurate structure prediction of biomolecular interactions with AlphaFold 3”. en. In: *Nature* 630.8016 (June 2024), pp. 493–500. ISSN: 0028-0836,1476-4687. DOI: 10.1038/s41586-024-07487-w.
- [3] Sooraj R Achar et al. “Universal antigen encoding of T cell activation from high-dimensional cytokine dynamics”. en. In: *Science* 376.6595 (May 2022), pp. 880–884. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.abl5311.
- [4] Bruce Alberts et al. *Molecular Biology of the Cell*. 4th. Garland Science, 2002. ISBN: 978-0-8153-3218-3 978-0-8153-4072-0.
- [5] Mark Aldenderfer and Roger Blashfield. *Cluster Analysis*. 2455 Teller Road, Thousand Oaks California 91320 United States of America: SAGE Publications, Inc., 1984. ISBN: 978-0-8039-2376-8 978-1-4129-8364-8. DOI: 10.4135/9781412983648. URL: <https://methods.sagepub.com/book/cluster-analysis> (visited on 09/21/2022).
- [6] Milos Aleksic et al. “Dependence of T cell antigen recognition on T cell receptor-peptide MHC confinement time”. en. In: *Immunity* 32.2 (Feb. 2010), pp. 163–174. ISSN: 1074-7613,1097-4180. DOI: 10.1016/j.immuni.2009.11.013.
- [7] Rebecca F. Alford et al. “The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design”. eng. In: *Journal of Chemical Theory and Computation* 13.6 (June 2017), pp. 3031–3048. ISSN: 1549-9626. DOI: 10.1021/acs.jctc.7b00125.
- [8] Mohammed AlQuraishi. “ProteinNet: a standardized data set for machine learning of protein structure”. In: *BMC Bioinformatics* 20.1 (June 2019), p. 311. ISSN: 1471-2105. DOI: 10.1186/s12859-019-2932-0. URL: <https://doi.org/10.1186/s12859-019-2932-0> (visited on 09/23/2022).
- [9] Grégoire Altan-Bonnet and Ronald N Germain. “Modeling T cell antigen discrimination based on feedback control of digital ERK responses”. en. In: *PLoS Biol.* 3.11 (Nov. 2005), e356. ISSN: 1544-9173, 1545-7885. DOI: 10.1371/journal.pbio.0030356.
- [10] Brandon Anderson, Truong-Son Hy, and Risi Kondor. *Cormorant: Covariant Molecular Neural Networks*. Nov. 25, 2019. DOI: 10.48550/arXiv.1906.04015. arXiv: 1906.04015[physics, stat]. URL: <http://arxiv.org/abs/1906.04015> (visited on 01/24/2023).
- [11] Massimo Andreatta and Morten Nielsen. “Gapped sequence alignment using artificial neural networks: application to the MHC class I system”. en. In: *Bioinformatics* 32.4 (Feb. 2016), pp. 511–517. ISSN: 1367-4811,1367-4803. DOI: 10.1093/bioinformatics/btv639.
- [12] Meriem Attaf, Eric Huseby, and Andrew K Sewell. “ $\alpha\beta$ T cell receptors as predictors of health and disease”. en. In: *Cell. Mol. Immunol.* 12.4 (July 2015), pp. 391–399. ISSN: 1672-7681,2042-0226. DOI: 10.1038/cmi.2014.134.

- [13] Sarp Aykent and Tian Xia. *GBPNet | Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Aug. 2022. URL: <https://dl.acm.org/doi/abs/10.1145/3534678.3539441> (visited on 01/24/2023).
- [14] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. arXiv:1607.06450 [cs, stat]. July 2016. DOI: 10.48550/arXiv.1607.06450. URL: <http://arxiv.org/abs/1607.06450> (visited on 11/17/2022).
- [15] Mark J. G. Bakkers et al. “Efficacious human metapneumovirus vaccine based on AI-guided engineering of a closed prefusion trimer”. en. In: *Nature Communications* 15.1 (July 2024). Publisher: Nature Publishing Group, p. 6270. ISSN: 2041-1723. DOI: 10.1038/s41467-024-50659-5. URL: <https://www.nature.com/articles/s41467-024-50659-5> (visited on 09/19/2025).
- [16] Simon Batzner et al. “E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials”. en. In: *Nature Communications* 13.1 (May 2022). Publisher: Nature Publishing Group, p. 2453. ISSN: 2041-1723. DOI: 10.1038/s41467-022-29939-5. URL: <https://www.nature.com/articles/s41467-022-29939-5> (visited on 04/22/2026).
- [17] S. Benevenuta et al. “An antisymmetric neural network to predict free energy changes in protein variants”. en. In: *Journal of Physics D: Applied Physics* 54.24 (Mar. 2021). Publisher: IOP Publishing, p. 245403. ISSN: 0022-3727. DOI: 10.1088/1361-6463/abedfb. URL: <https://dx.doi.org/10.1088/1361-6463/abedfb> (visited on 07/05/2024).
- [18] H. M. Berman et al. “The Protein Data Bank”. eng. In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 235–242. ISSN: 0305-1048. DOI: 10.1093/nar/28.1.235.
- [19] Christopher Bishop. *Pattern Recognition and Machine Learning*. en. 2006. URL: <https://link.springer.com/book/9780387310732> (visited on 04/30/2026).
- [20] Lasse M Blaabjerg et al. “Rapid protein stability prediction using deep learning representations”. In: *eLife* 12 (May 2023). Ed. by José D Faraldo-Gómez et al. Publisher: eLife Sciences Publications, Ltd, e82593. ISSN: 2050-084X. DOI: 10.7554/eLife.82593. URL: <https://doi.org/10.7554/eLife.82593> (visited on 06/28/2024).
- [21] Lasse M Blaabjerg et al. “Rapid protein stability prediction using deep learning representations”. en. In: *Elife* 12 (May 2023). ISSN: 2050-084X. DOI: 10.7554/eLife.82593.
- [22] Eryn Blass and Patrick A Ott. “Advances in the development of personalized neoantigen-based therapeutic cancer vaccines”. en. In: *Nat. Rev. Clin. Oncol.* 18.4 (Jan. 2021), pp. 215–229. ISSN: 1759-4774. DOI: 10.1038/s41571-020-00460-2.
- [23] Kevin Borisiak, Gian Marco Visani, and Armita Nourmohammad. *Loop-Diffusion: an equivariant diffusion model for designing and scoring protein loops*. arXiv:2409.18201 [physics]. Sept. 2024. DOI: 10.48550/arXiv.2409.18201. URL: <http://arxiv.org/abs/2409.18201> (visited on 04/30/2026).
- [24] Samuel R. Bowman et al. *Generating Sentences from a Continuous Space*. May 12, 2016. DOI: 10.48550/arXiv.1511.06349. arXiv: 1511.06349 [cs]. URL: <http://arxiv.org/abs/1511.06349> (visited on 09/22/2022).
- [25] John P. Boyd and Fu Yu. “Comparing seven spectral methods for interpolation and for solving the Poisson equation in a disk: Zernike polynomials, Logan–Shepp ridge polynomials, Chebyshev–Fourier Series, cylindrical Robert functions, Bessel–Fourier expansions, square-to-disk conformal mapping and radial basis functions”. In: *Journal of Computational Physics* 230.4 (Feb. 20, 2011), pp. 1408–

1438. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2010.11.011. URL: <https://www.sciencedirect.com/science/article/pii/S0021999110006133> (visited on 09/22/2022).
- [26] Philip Bradley. “Structure-based prediction of T cell receptor:peptide-MHC interactions”. In: *Elife* 12 (Jan. 2023), e82813. ISSN: 2050-084X. DOI: 10.7554/eLife.82813.
- [27] Johannes Brandstetter et al. *Geometric and Physical Quantities Improve E(3) Equivariant Message Passing*. Mar. 26, 2022. DOI: 10.48550/arXiv.2110.02905. arXiv: 2110.02905[cs,stat]. URL: <http://arxiv.org/abs/2110.02905> (visited on 07/17/2022).
- [28] Barbara Bravi et al. “A transfer-learning approach to predict antigen immunogenicity and T-cell receptor specificity”. en. In: *Elife* 12 (Sept. 2023), e85126. ISSN: 2050-084X. DOI: 10.7554/eLife.85126.
- [29] Leo Breiman. “Random forests”. en. In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32.
- [30] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. arXiv:2104.13478 [cs, stat]. May 2021. DOI: 10.48550/arXiv.2104.13478. URL: <http://arxiv.org/abs/2104.13478> (visited on 09/22/2022).
- [31] Jasper Butcher et al. *De novo Design of All-atom Biomolecular Interactions with RFDiffusion3*. en. ISSN: 2692-8205 Pages: 2025.09.18.676967 Section: New Results. Nov. 2025. DOI: 10.1101/2025.09.18.676967. URL: <https://www.biorxiv.org/content/10.1101/2025.09.18.676967v2> (visited on 05/01/2026).
- [32] Patrick O Byrne and Jason S McLellan. “Principles and practical applications of structure-based vaccine design”. In: *Current Opinion in Immunology* 77 (Aug. 2022), p. 102209. ISSN: 0952-7915. DOI: 10.1016/j.coi.2022.102209. URL: <https://www.sciencedirect.com/science/article/pii/S0952791522000565> (visited on 10/06/2025).
- [33] Octav Caldararu, Tom L. Blundell, and Kasper P. Kepp. “A base measure of precision for protein stability predictors: structural sensitivity”. en. In: *BMC Bioinformatics* 22.1 (Feb. 2021), p. 88. ISSN: 1471-2105. DOI: 10.1186/s12859-021-04030-w. URL: <https://doi.org/10.1186/s12859-021-04030-w> (visited on 09/17/2025).
- [34] Herbert B. Callen. *Thermodynamics and an Introduction to Thermostatistics*. English. New York, NY: Wiley, 1985. ISBN: 978-0-471-86256-7.
- [35] Christiam Camacho et al. “BLAST+: architecture and applications”. eng. In: *BMC bioinformatics* 10 (Dec. 2009), p. 421. ISSN: 1471-2105. DOI: 10.1186/1471-2105-10-421.
- [36] Brian J Cameron et al. “Identification of a Titin-derived HLA-A1-presented peptide as a cross-reactive target for engineered MAGE A3-directed T cells”. en. In: *Sci. Transl. Med.* 5.197 (Aug. 2013), 197ra103. ISSN: 1946-6234,1946-6242. DOI: 10.1126/scitranslmed.3006034.
- [37] Yang Cao et al. “Improved side-chain modeling by coupling clash-detection guided iterative search with rotamer relaxation”. In: *Bioinformatics* 27.6 (Mar. 15, 2011), pp. 785–790. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btr009. URL: <https://doi.org/10.1093/bioinformatics/btr009> (visited on 10/05/2023).
- [38] Sidhartha Chaudhury, Sergey Lyskov, and Jeffrey J Gray. “PyRosetta: a script-based interface for implementing molecular modeling algorithms using Rosetta”. en. In: *Bioinformatics* 26.5 (Mar. 2010), pp. 689–691. ISSN: 1367-4803, 1367-4811. DOI: 10.1093/bioinformatics/btq007.

- [39] Sidhartha Chaudhury, Sergey Lyskov, and Jeffrey J. Gray. “PyRosetta: a script-based interface for implementing molecular modeling algorithms using Rosetta”. eng. In: *Bioinformatics (Oxford, England)* 26.5 (Mar. 2010), pp. 689–691. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btq007.
- [40] J L Chen et al. “Identification of NY-ESO-1 peptide analogues capable of improved stimulation of tumor-reactive CTL”. en. In: *J. Immunol.* 165.2 (July 2000), pp. 948–955. ISSN: 0022-1767,1550-6606. DOI: 10.4049/jimmunol.165.2.948.
- [41] Ji-Li Chen et al. “Structural and kinetic basis for heightened immunogenicity of T cell vaccines”. en. In: *J. Exp. Med.* 201.8 (Apr. 2005), pp. 1243–1255. ISSN: 0022-1007,1540-9538. DOI: 10.1084/jem.20042323.
- [42] Ji-Li Chen et al. “Structural and kinetic basis for heightened immunogenicity of T cell vaccines”. In: *Journal of Experimental Medicine* 201.8 (Apr. 2005), pp. 1243–1255. ISSN: 0022-1007. DOI: 10.1084/jem.20042323. URL: <https://doi.org/10.1084/jem.20042323> (visited on 02/12/2025).
- [43] Ratul Chowdhury et al. “Single-sequence protein structure prediction using a language model and deep learning”. en. In: *Nature Biotechnology* 40.11 (Nov. 2022). Publisher: Nature Publishing Group, pp. 1617–1623. ISSN: 1546-1696. DOI: 10.1038/s41587-022-01432-w. URL: <https://www.nature.com/articles/s41587-022-01432-w> (visited on 04/30/2026).
- [44] Oliver J. Cobb et al. *Efficient Generalized Spherical CNNs*. Mar. 8, 2021. DOI: 10.48550/arXiv.2010.11661. arXiv: 2010.11661[astro-ph]. URL: <http://arxiv.org/abs/2010.11661> (visited on 07/16/2022).
- [45] Peter J. A. Cock et al. “Biopython: freely available Python tools for computational molecular biology and bioinformatics”. In: *Bioinformatics* 25.11 (June 2009), pp. 1422–1423. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp163. URL: <https://doi.org/10.1093/bioinformatics/btp163> (visited on 06/28/2024).
- [46] Taco S. Cohen et al. *Spherical CNNs*. Feb. 25, 2018. DOI: 10.48550/arXiv.1801.10130. arXiv: 1801.10130[cs,stat]. URL: <http://arxiv.org/abs/1801.10130> (visited on 07/16/2022).
- [47] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. en. In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274. URL: <https://doi.org/10.1007/BF02551274> (visited on 04/30/2026).
- [48] Sebastian Daberdaku and Carlo Ferrari. “Exploring the potential of 3D Zernike descriptors and SVM for protein–protein interface prediction”. en. In: *BMC Bioinformatics* 19.1 (Feb. 2018), p. 35. ISSN: 1471-2105. DOI: 10.1186/s12859-018-2043-3. URL: <https://doi.org/10.1186/s12859-018-2043-3> (visited on 05/05/2026).
- [49] Tom Darden, Darrin York, and Lee Pedersen. “Particle mesh Ewald: An N·log(N) method for Ewald sums in large systems”. In: *The Journal of Chemical Physics* 98.12 (June 1993), pp. 10089–10092. ISSN: 0021-9606. DOI: 10.1063/1.464397. URL: <https://doi.org/10.1063/1.464397> (visited on 04/30/2026).
- [50] Pradyot Dash et al. “Quantifiable predictive features define epitope-specific T cell receptor repertoires”. en. In: *Nature* 547.7661 (July 2017), pp. 89–93. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature22383.

- [51] J Dauparas et al. “Robust deep learning-based protein sequence design using ProteinMPNN”. en. In: *Science* 378.6651 (Sept. 2022), pp. 49–56. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.add2187.
- [52] J. Dauparas et al. “Robust deep learning-based protein sequence design using ProteinMPNN”. In: *Science* 378.6615 (Oct. 2022). Publisher: American Association for the Advancement of Science, pp. 49–56. DOI: 10.1126/science.add2187. URL: <https://www.science.org/doi/10.1126/science.add2187> (visited on 10/04/2023).
- [53] Justas Dauparas et al. “Atomic context-conditioned protein sequence design using LigandMPNN”. In: *Nature Methods* 22.4 (Mar. 2025), pp. 717–723. DOI: 10.1038/s41592-025-02626-1.
- [54] M M Davis et al. “Ligand recognition by alpha beta T cell receptors”. en. In: *Annu. Rev. Immunol.* 16.1 (1998), pp. 523–544. ISSN: 0732-0582,1545-3278. DOI: 10.1146/annurev.immunol.16.1.523.
- [55] Antonio Deiana et al. “Intrinsically disordered proteins and structured proteins with intrinsically disordered regions have different functional roles in the cell”. en. In: *PLOS ONE* 14.8 (Aug. 2019). Publisher: Public Library of Science, e0217889. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0217889. URL: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0217889> (visited on 04/30/2026).
- [56] Javier Delgado et al. “FoldX 5.0: working with RNA, small molecules and a new graphical interface”. In: *Bioinformatics* 35.20 (Oct. 2019), pp. 4168–4169. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz184. URL: <https://doi.org/10.1093/bioinformatics/btz184> (visited on 09/11/2024).
- [57] Li Deng. “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]”. In: *IEEE Signal Processing Magazine* 29.6 (Nov. 2012). Conference Name: IEEE Signal Processing Magazine, pp. 141–142. ISSN: 1558-0792. DOI: 10.1109/MSP.2012.2211477.
- [58] Daniel J Diaz et al. “Stability Oracle: a structure-based graph-transformer framework for identifying stabilizing mutations”. en. In: *Nat. Commun.* 15.1 (July 2024), p. 6170. ISSN: 2041-1723,2041-1723. DOI: 10.1038/s41467-024-49780-2.
- [59] Daniel J. Diaz et al. “Stability Oracle: a structure-based graph-transformer framework for identifying stabilizing mutations”. en. In: *Nature Communications* 15.1 (July 2024). Publisher: Nature Publishing Group, p. 6170. ISSN: 2041-1723. DOI: 10.1038/s41467-024-49780-2. URL: <https://www.nature.com/articles/s41467-024-49780-2> (visited on 09/10/2024).
- [60] Henry Dieckhaus et al. “Transfer learning to leverage larger datasets for improved prediction of protein stability changes”. In: *Proceedings of the National Academy of Sciences* 121.6 (Feb. 2024). Publisher: Proceedings of the National Academy of Sciences, e2314853121. DOI: 10.1073/pnas.2314853121. URL: <https://www.pnas.org/doi/10.1073/pnas.2314853121> (visited on 09/17/2025).
- [61] Yuan-Hua Ding et al. “Four A6-TCR/Peptide/HLA-A2 Structures that Generate Very Different T Cell Signals Are Nearly Identical”. In: *Immunity* 11.1 (July 1999), pp. 45–56. ISSN: 1074-7613. DOI: 10.1016/S1074-7613(00)80080-1. URL: <https://www.sciencedirect.com/science/article/pii/S1074761300800801> (visited on 02/12/2025).
- [62] Ralf Drautz. “Atomic cluster expansion for accurate and transferable interatomic potentials”. In: *Physical Review B* 99.1 (Jan. 8, 2019). Publisher: American Physical Society, p. 014104. DOI: 10.1103/PhysRevB.99.014104. URL: <https://link.aps.org/doi/10.1103/PhysRevB.99.014104> (visited on 09/22/2022).

- [63] Steven M Dunn et al. “Directed evolution of human T cell receptor CDR2 residues by phage display dramatically enhances affinity for cognate peptide-MHC without increasing apparent cross-reactivity”. en. In: *Protein Sci.* 15.4 (Apr. 2006), pp. 710–721. ISSN: 0961-8368,1469-896X. DOI: 10.1110/ps.051936406.
- [64] Peter Eastman et al. “OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation”. In: *Journal of Chemical Theory and Computation* 9.1 (Jan. 2013). Publisher: American Chemical Society, pp. 461–469. ISSN: 1549-9618. DOI: 10.1021/ct300857j. URL: <https://doi.org/10.1021/ct300857j> (visited on 06/28/2024).
- [65] Ahmed Elnaggar et al. “ProfTrans: Toward Understanding the Language of Life Through Self-Supervised Learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 44.10 (Oct. 2022), pp. 7112–7127. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3095381.
- [66] Ali Esfandiary and Soudeh Ghafouri-Fard. “New York esophageal squamous cell carcinoma-1 and cancer immunotherapy”. en. In: *Immunotherapy* 7.4 (2015), pp. 411–439. ISSN: 1750-743X,1750-7448. DOI: 10.2217/imt.15.3.
- [67] Carlos Esteves, Ameesh Makadia, and Kostas Daniilidis. *Spin-Weighted Spherical CNNs*. Oct. 26, 2020. DOI: 10.48550/arXiv.2006.10731. arXiv: 2006.10731[cs]. URL: <http://arxiv.org/abs/2006.10731> (visited on 09/22/2022).
- [68] Ethan Fast, Manjima Dhar, and Binbin Chen. “TAPIR: a T-cell receptor language model for predicting rare and novel targets”. en. In: *bioRxiv* (Sept. 2023), p. 2023.09.12.557285. DOI: 10.1101/2023.09.12.557285.
- [69] Ilya Feige. “Invariant-equivariant representation learning for multi-class data”. In: (Feb. 10, 2022). URL: <https://openreview.net/forum?id=B1e4wo09K7> (visited on 09/22/2022).
- [70] Állan J. R. Ferrari et al. *Large-scale discovery, analysis, and design of protein energy landscapes*. en. Pages: 2025.03.20.644235 Section: New Results. Mar. 2025. DOI: 10.1101/2025.03.20.644235. URL: <https://www.biorxiv.org/content/10.1101/2025.03.20.644235v1> (visited on 04/30/2026).
- [71] Douglas M. Fowler and Stanley Fields. “Deep mutational scanning: a new style of protein science”. en. In: *Nature Methods* 11.8 (Aug. 2014). Publisher: Nature Publishing Group, pp. 801–807. ISSN: 1548-7105. DOI: 10.1038/nmeth.3027. URL: <https://www.nature.com/articles/nmeth.3027> (visited on 07/05/2024).
- [72] Fabian B. Fuchs et al. *SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks*. Nov. 24, 2020. DOI: 10.48550/arXiv.2006.10503. arXiv: 2006.10503[cs, stat]. URL: <http://arxiv.org/abs/2006.10503> (visited on 07/16/2022).
- [73] P. Gainza et al. “Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning”. In: *Nature Methods* 17.2 (Feb. 2020). Number: 2 Publisher: Nature Publishing Group, pp. 184–192. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0666-6. URL: <https://www.nature.com/articles/s41592-019-0666-6> (visited on 01/24/2023).
- [74] Vytautas Gapsys et al. “Accurate and Rigorous Prediction of the Changes in Protein Free Energies in a Large-Scale Mutation Scan”. en. In: *Angewandte Chemie International Edition* 55.26 (2016). _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.201510054>, pp. 7364–7368. ISSN: 1521-3773. DOI: 10.1002/anie.201510054. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.201510054> (visited on 07/05/2024).

- [75] D N Garboczi et al. “Structure of the complex between human T-cell receptor, viral peptide and HLA-A2”. en. In: *Nature* 384.6605 (Nov. 1996), pp. 134–141. issn: 0028-0836,1476-4687. doi: 10.1038/384134a0.
- [76] Mario Geiger and Tess Smidt. *e3nn: Euclidean Neural Networks*. arXiv:2207.09453 [cs]. July 2022. doi: 10.48550/arXiv.2207.09453. URL: <http://arxiv.org/abs/2207.09453> (visited on 09/22/2022).
- [77] Lukas Gerasimavicius, Xin Liu, and Joseph A. Marsh. “Identification of pathogenic missense mutations using protein stability predictors”. en. In: *Scientific Reports* 10.1 (Sept. 2020). Publisher: Nature Publishing Group, p. 15387. issn: 2045-2322. doi: 10.1038/s41598-020-72404-w. URL: <https://www.nature.com/articles/s41598-020-72404-w> (visited on 07/05/2024).
- [78] U. Göbel et al. “Correlated mutations and residue contacts in proteins”. eng. In: *Proteins* 18.4 (Apr. 1994), pp. 309–317. issn: 0887-3585. doi: 10.1002/prot.340180402.
- [79] Chengyue Gong et al. “Evolution-Inspired Loss Functions for Protein Representation Learning”. en. In: June 2024. URL: [https://openreview.net/forum?id=y5L8W0KRUX&referrer=%5Bthe%20profile%20of%20Chengyue%20Gong%5D\(%2Fprofile%3Fid%3D~Chengyue_Gong1\)](https://openreview.net/forum?id=y5L8W0KRUX&referrer=%5Bthe%20profile%20of%20Chengyue%20Gong%5D(%2Fprofile%3Fid%3D~Chengyue_Gong1)) (visited on 04/14/2025).
- [80] Karen J. Gonzalez et al. “A general computational design strategy for stabilizing viral class I fusion proteins”. en. In: *Nature Communications* 15.1 (Feb. 2024). Publisher: Nature Publishing Group, p. 1335. issn: 2041-1723. doi: 10.1038/s41467-024-45480-z. URL: <https://www.nature.com/articles/s41467-024-45480-z> (visited on 09/19/2025).
- [81] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: <http://www.deeplearningbook.org>.
- [82] Cade W. Gordon, Amy X. Lu, and Pieter Abbeel. “Protein Language Model Fitness is a Matter of Preference”. en. In: Oct. 2024. URL: <https://openreview.net/forum?id=UvPdpa4LuV> (visited on 04/14/2025).
- [83] Ragul Gowthaman and Brian G Pierce. “TCR3d: The T cell receptor structural repertoire database”. In: *Bioinformatics* 35.24 (Dec. 2019), pp. 5323–5325. issn: 1367-4803. doi: 10.1093/bioinformatics/btz517. URL: <https://doi.org/10.1093/bioinformatics/btz517> (visited on 05/19/2025).
- [84] Stephanie Gras et al. “Allelic polymorphism in the T cell receptor and its impact on immune responses”. en. In: *J. Exp. Med.* 207.7 (July 2010), pp. 1555–1567. issn: 0022-1007,1540-9538. doi: 10.1084/jem.20100603.
- [85] Stephanie Gras et al. “Allelic polymorphism in the T cell receptor and its impact on immune responses”. In: *Journal of Experimental Medicine* 207.7 (June 2010), pp. 1555–1567. issn: 0022-1007. doi: 10.1084/jem.20100603. URL: <https://doi.org/10.1084/jem.20100603> (visited on 02/12/2025).
- [86] James M Heather et al. “Stitchr: stitching coding TCR nucleotide sequences from V/J/CDR3 information”. en. In: *Nucleic Acids Res.* 50.12 (July 2022), e68. issn: 0305-1048,1362-4962. doi: 10.1093/nar/gkac190.
- [87] S Henikoff and J G Henikoff. “Amino acid substitution matrices from protein blocks”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 89.22 (Nov. 1992), pp. 10915–10919. issn: 0027-8424,1091-6490. doi: 10.1073/pnas.89.22.10915.

- [88] Brian L Hie et al. “Efficient evolution of human antibodies from general protein language models”. en. In: *Nat. Biotechnol.* 42.2 (Feb. 2024), pp. 275–283. ISSN: 1087-0156,1546-1696. DOI: 10.1038/s41587-023-01763-2.
- [89] Brian L. Hie et al. “Efficient evolution of human antibodies from general protein language models”. en. In: *Nature Biotechnology* 42.2 (Feb. 2024). Publisher: Nature Publishing Group, pp. 275–283. ISSN: 1546-1696. DOI: 10.1038/s41587-023-01763-2. URL: <https://www.nature.com/articles/s41587-023-01763-2> (visited on 07/05/2024).
- [90] Irina Higgins et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: International Conference on Learning Representations. July 21, 2022. URL: <https://openreview.net/forum?id=Sy2fzU9gl> (visited on 09/22/2022).
- [91] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. eprint: arXiv:1503.02531.
- [92] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. “Transforming Auto-Encoders”. In: *Artificial Neural Networks and Machine Learning – ICANN 2011*. Ed. by Timo Honkela et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 44–51. ISBN: 978-3-642-21735-7. DOI: 10.1007/978-3-642-21735-7_6.
- [93] Scott A. Hollingsworth and Ron O. Dror. “Molecular Dynamics Simulation for All”. In: *Neuron* 99.6 (Sept. 2018), pp. 1129–1143. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2018.08.011. URL: <https://www.sciencedirect.com/science/article/pii/S0896627318306846> (visited on 04/30/2026).
- [94] Emiel Hoogeboom et al. *Equivariant Diffusion for Molecule Generation in 3D*. June 16, 2022. DOI: 10.48550/arXiv.2203.17003. arXiv: 2203.17003[cs, q-bio, stat]. URL: <http://arxiv.org/abs/2203.17003> (visited on 07/16/2022).
- [95] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8. URL: <https://www.sciencedirect.com/science/article/pii/0893608089900208> (visited on 04/30/2026).
- [96] Ching-Lin Hsieh et al. “Structure-based design of prefusion-stabilized SARS-CoV-2 spikes”. In: *Science* 369.6510 (Sept. 2020). Publisher: American Association for the Advancement of Science, pp. 1501–1505. DOI: 10.1126/science.abd0826. URL: <https://www.science.org/doi/10.1126/science.abd0826> (visited on 10/06/2025).
- [97] Emily Han-Chung Hsiue et al. “Targeting a neoantigen derived from a common TP53 mutation”. en. In: *Science* 371.6533 (Mar. 2021), eabc8697. ISSN: 0036-8075,1095-9203. DOI: 10.1126/science.abc8697.
- [98] Chloe Hsu et al. “Learning inverse folding from millions of predicted structures”. en. In: *Proceedings of the 39th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, June 2022, pp. 8946–8970. URL: <https://proceedings.mlr.press/v162/hsu22a.html> (visited on 09/11/2024).
- [99] Chloe Hsu et al. “Learning inverse folding from millions of predicted structures”. In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 8946–8970.

- [100] Xiaoqiang Huang, Robin Pearce, and Yang Zhang. “FASPR: an open-source tool for fast and accurate protein side-chain packing”. In: *Bioinformatics (Oxford, England)* 36.12 (June 1, 2020), pp. 3758–3765. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btaa234.
- [101] Giulio Isacchini et al. “Deep generative selection models of T and B cell receptor repertoires with soNNia”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 118.14 (Apr. 2021). ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2023141118.
- [102] Toyokazu Ishida. “Effects of Point Mutation on Enzymatic Activity: Correlation between Protein Electronic Structure and Motion in Chorismate Mutase Reaction”. In: *Journal of the American Chemical Society* 132.20 (May 2010). Publisher: American Chemical Society, pp. 7104–7118. ISSN: 0002-7863. DOI: 10.1021/ja100744h. URL: <https://doi.org/10.1021/ja100744h> (visited on 07/05/2024).
- [103] Justina Jankauskaitė et al. “SKEMPI 2.0: an updated benchmark of changes in protein–protein binding energy, kinetics and thermodynamics upon mutation”. In: *Bioinformatics* 35.3 (Feb. 2019), pp. 462–469. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty635. URL: <https://doi.org/10.1093/bioinformatics/bty635> (visited on 07/02/2024).
- [104] Mathias Fynbo Jensen and Morten Nielsen. “NetTCR 2.2 - Improved TCR specificity predictions by combining pan- and peptide-specific training strategies, loss-scaling and integration of sequence similarity”. en. In: *eLife* 12 (Feb. 2024), RP93934. DOI: 10.7554/elife.93934.2.
- [105] Bowen Jing et al. *Learning from Protein Structure with Geometric Vector Perceptrons*. arXiv:2009.01411 [cs, q-bio, stat]. May 2021. DOI: 10.48550/arXiv.2009.01411. URL: <http://arxiv.org/abs/2009.01411> (visited on 07/17/2022).
- [106] R. P. Joosten et al. “PDB_REDO: automated re-refinement of X-ray structure models in the PDB”. In: *Journal of Applied Crystallography* 42.3 (June 1, 2009). Publisher: International Union of Crystallography, pp. 376–384. ISSN: 0021-8898. DOI: 10.1107/S0021889809008784. URL: <http://scripts.iucr.org/cgi-bin/paper?he5437> (visited on 10/06/2023).
- [107] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. en. In: *Nature* 596.7873 (Aug. 2021). Publisher: Nature Publishing Group, pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. URL: <https://www.nature.com/articles/s41586-021-03819-2> (visited on 07/03/2024).
- [108] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. en. In: *Nature* 596.7873 (Aug. 2021), pp. 583–589. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-021-03819-2.
- [109] Mostafa Karimi et al. “DeepAffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks”. In: *Bioinformatics* 35.18 (Sept. 15, 2019), pp. 3329–3338. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz111. URL: <https://doi.org/10.1093/bioinformatics/btz111> (visited on 01/24/2023).
- [110] Robert Karlsson. “SPR for molecular interaction analysis: a review of emerging application areas”. en. In: *Journal of Molecular Recognition* 17.3 (2004). eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/jmr.660>, pp. 151–161. ISSN: 1099-1352. DOI: 10.1002/jmr.660. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jmr.660> (visited on 07/05/2024).

- [111] Elizabeth H. Kellogg, Andrew Leaver-Fay, and David Baker. “Role of conformational sampling in computing mutation-induced changes in protein structure and stability”. en. In: *Proteins: Structure, Function, and Bioinformatics* 79.3 (2011). _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.22921>, pp. 830–838. ISSN: 1097-0134. DOI: 10.1002/prot.22921. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.22921> (visited on 07/05/2024).
- [112] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: <http://arxiv.org/abs/1412.6980> (visited on 03/10/2024).
- [113] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980[cs]. URL: <http://arxiv.org/abs/1412.6980> (visited on 09/23/2022).
- [114] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: (Dec. 20, 2013). DOI: 10.48550/arXiv.1312.6114. URL: <https://arxiv.org/abs/1312.6114v10> (visited on 09/23/2022).
- [115] Justin B Kinney and David M McCandlish. “Massively Parallel Assays and Quantitative Sequence–Function Relationships”. In: *Annu. Rev. Genomics Hum. Genet.* 20. Volume 20, 2019 (Aug. 2019), pp. 99–127. ISSN: 1527-8204, 1545-293X. DOI: 10.1146/annurev-genom-083118-014845.
- [116] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. *Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network*. Nov. 10, 2018. DOI: 10.48550/arXiv.1806.09231. arXiv: 1806.09231[cs, stat]. URL: <http://arxiv.org/abs/1806.09231> (visited on 07/16/2022).
- [117] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. “Clebsch–Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: https://papers.nips.cc/paper_files/paper/2018/hash/a3fc981af450752046be179185ebc8b5-Abstract.html (visited on 03/09/2024).
- [118] Kaushik Koneripalli et al. “Rate-Invariant Autoencoding of Time-Series”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). ISSN: 2379-190X. May 2020, pp. 3732–3736. DOI: 10.1109/ICASSP40776.2020.9053983.
- [119] Adam R. Kosiorek et al. *Stacked Capsule Autoencoders*. Dec. 2, 2019. DOI: 10.48550/arXiv.1906.06818. arXiv: 1906.06818[cs, stat]. URL: <http://arxiv.org/abs/1906.06818> (visited on 09/22/2022).
- [120] Georgii G. Krivov, Maxim V. Shapovalov, and Roland L. Dunbrack Jr. “Improved prediction of protein side-chain conformations with SCWRL4”. In: *Proteins: Structure, Function, and Bioinformatics* 77.4 (2009). _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.22488>, pp. 778–795. ISSN: 1097-0134. DOI: 10.1002/prot.22488. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.22488> (visited on 10/05/2023).
- [121] Stephan T. Kudlacek et al. “Designed, highly expressing, thermostable dengue virus 2 envelope protein dimers elicit quaternary epitope antibodies”. In: *Science Advances* 7.42 (Oct. 2021). Publisher: American Association for the Advancement of Science, eabg4084. DOI: 10.1126/sciadv.abg4084. URL: <https://www.science.org/doi/10.1126/sciadv.abg4084> (visited on 01/06/2026).

- [122] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. “Zero-data learning of new tasks”. In: *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2. AAAI’08*. Chicago, Illinois: AAAI Press, July 2008, pp. 646–651. ISBN: 978-1-57735-368-3. (Visited on 04/30/2026).
- [123] Yi-Zong Lee et al. “Rational design of uncleaved prefusion-closed trimer vaccines for human respiratory syncytial virus and metapneumovirus”. en. In: *Nature Communications* 15.1 (Nov. 2024). Publisher: Nature Publishing Group, p. 9939. ISSN: 2041-1723. DOI: 10.1038/s41467-024-54287-x. URL: <https://www.nature.com/articles/s41467-024-54287-x> (visited on 10/06/2025).
- [124] Rom Leidner et al. “Neoantigen T-cell receptor gene therapy in pancreatic cancer”. en. In: *N. Engl. J. Med.* 386.22 (June 2022), pp. 2112–2119. ISSN: 0028-4793,1533-4406. DOI: 10.1056/NEJMoa2119662.
- [125] Sarah Lewis et al. “Scalable emulation of protein equilibrium ensembles with generative deep learning”. In: *Science* 389.6761 (July 2025). Publisher: American Association for the Advancement of Science, eadv9817. DOI: 10.1126/science.adv9817. URL: <https://www.science.org/doi/10.1126/science.adv9817> (visited on 05/05/2026).
- [126] Bian Li et al. “Predicting changes in protein thermodynamic stability upon point mutation with deep 3D convolutional neural networks”. en. In: *PLOS Computational Biology* 16.11 (Nov. 2020). Publisher: Public Library of Science, e1008291. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1008291. URL: <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008291> (visited on 07/05/2024).
- [127] Yi Li et al. “Directed evolution of human T-cell receptors with picomolar affinities by phage display”. en. In: *Nat. Biotechnol.* 23.3 (Mar. 2005), pp. 349–354. ISSN: 1087-0156,1546-1696. DOI: 10.1038/nbt1070.
- [128] Shide Liang et al. “Fast and accurate prediction of protein side-chain conformations”. In: *Bioinformatics* 27.20 (Oct. 15, 2011), pp. 2913–2914. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btr482. URL: <https://doi.org/10.1093/bioinformatics/btr482> (visited on 10/05/2023).
- [129] Yi-Lun Liao and Tess Smidt. *Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs*. June 23, 2022. DOI: 10.48550/arXiv.2206.11990. arXiv: 2206.11990[physics]. URL: <http://arxiv.org/abs/2206.11990> (visited on 07/16/2022).
- [130] Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. In: *Science* 379.6637 (Mar. 2023). Publisher: American Association for the Advancement of Science, pp. 1123–1130. DOI: 10.1126/science.ade2574. URL: <https://www.science.org/doi/10.1126/science.ade2574> (visited on 09/07/2024).
- [131] Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. en. In: *Science* 379.6637 (Mar. 2023), pp. 1123–1130. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.ade2574.
- [132] Kresten Lindorff-Larsen and Kaare Teilum. “Linking thermodynamics and measurements of protein stability”. In: *Protein Engineering, Design and Selection* 34 (Feb. 2021), gzab002. ISSN: 1741-0126. DOI: 10.1093/protein/gzab002. URL: <https://doi.org/10.1093/protein/gzab002> (visited on 07/05/2024).
- [133] Gerald P Linette et al. “Cardiovascular toxicity and titin cross-reactivity of affinity-enhanced T cells in myeloma and melanoma”. en. In: *Blood* 122.6 (Aug. 2013), pp. 863–871. ISSN: 0006-4971,1528-0020. DOI: 10.1182/blood-2013-03-490565.

- [134] Bingxu Liu et al. “Design of high specificity binders for peptide-MHC-I complexes”. In: *bioRxiv* 2024.11.28.625793 (Nov. 2024). DOI: 10.1101/2024.11.28.625793.
- [135] Bingxu Liu et al. “Design of high-specificity binders for peptide-MHC-I complexes”. en. In: *Science* 389.6758 (July 2025), pp. 386–391. ISSN: 0036-8075,1095-9203. DOI: 10.1126/science.adv0185.
- [136] Yu Chih Liu et al. “A molecular basis for the interplay between T cells, viral mutants, and human leukocyte antigen micropolymorphism”. en. In: *J. Biol. Chem.* 289.24 (June 2014), pp. 16688–16698. ISSN: 0021-9258,1083-351X. DOI: 10.1074/jbc.M114.563502.
- [137] Suhas Lohit and Shubhendu Trivedi. *Rotation-Invariant Autoencoders for Signals on Spheres*. Dec. 8, 2020. DOI: 10.48550/arXiv.2012.04474. arXiv: 2012.04474[cs]. URL: <http://arxiv.org/abs/2012.04474> (visited on 08/16/2022).
- [138] Pedro E. M. Lopes, Olgun Guvench, and Alexander D. MacKerell. “Current Status of Protein Force Fields for Molecular Dynamics Simulations”. en. In: *Molecular Modeling of Proteins*. Ed. by Andreas Kukol. New York, NY: Springer, 2015, pp. 47–71. ISBN: 978-1-4939-1465-4. DOI: 10.1007/978-1-4939-1465-4_3. URL: https://doi.org/10.1007/978-1-4939-1465-4_3 (visited on 04/30/2026).
- [139] Marta Łuksza and Michael Lässig. “A predictive fitness model for influenza”. en. In: *Nature* 507.7490 (Feb. 2014), pp. 57–61. ISSN: 0028-0836. DOI: 10.1038/nature13087.
- [140] Marta Łuksza et al. “Neoantigen quality predicts immunoediting in survivors of pancreatic cancer”. en. In: *Nature* 606.7913 (May 2022), pp. 389–395. ISSN: 0028-0836. DOI: 10.1038/s41586-022-04735-9.
- [141] Claus Lundegaard, Ole Lund, and Morten Nielsen. “Accurate approximation method for prediction of class I MHC affinities for peptides of length 8, 10 and 11 using prediction tools trained on 9mers”. en. In: *Bioinformatics* 24.11 (June 2008), pp. 1397–1398. ISSN: 1367-4803,1367-4811. DOI: 10.1093/bioinformatics/btn128.
- [142] Claus Lundegaard et al. “NetMHC-3.0: accurate web accessible predictions of human, mouse and monkey MHC class I affinities for peptides of length 8-11”. en. In: *Nucleic Acids Res.* 36.Web Server issue (July 2008), W509–12. ISSN: 0305-1048,1362-4962. DOI: 10.1093/nar/gkn202.
- [143] Shitong Luo et al. “Rotamer Density Estimator is an Unsupervised Learner of the Effect of Mutations on Protein-Protein Interaction”. en. In: Sept. 2022. URL: https://openreview.net/forum?id=_X9Yl1K2mD (visited on 09/11/2024).
- [144] Shitong Luo et al. *Rotamer Density Estimator is an Unsupervised Learner of the Effect of Mutations on Protein-Protein Interaction*. Pages: 2023.02.28.530137 Section: New Results. Mar. 1, 2023. DOI: 10.1101/2023.02.28.530137. URL: <https://www.biorxiv.org/content/10.1101/2023.02.28.530137v1> (visited on 09/21/2023).
- [145] Ali Madani et al. “Large language models generate functional protein sequences across diverse families”. en. In: *Nat. Biotechnol.* (Jan. 2023). ISSN: 1087-0156, 1546-1696. DOI: 10.1038/s41587-022-01618-2.
- [146] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. Sept. 17, 2020. DOI: 10.48550/arXiv.1802.03426. arXiv: 1802.03426[cs, stat]. URL: <http://arxiv.org/abs/1802.03426> (visited on 09/28/2022).

- [147] Jason S. McLellan et al. “Structure-Based Design of a Fusion Glycoprotein Vaccine for Respiratory Syncytial Virus”. In: *Science* 342.6158 (Nov. 2013). Publisher: American Association for the Advancement of Science, pp. 592–598. DOI: 10.1126/science.1243283. URL: <https://www.science.org/doi/10.1126/science.1243283> (visited on 09/19/2025).
- [148] Matthew McPartlon and Jinbo Xu. “An end-to-end deep learning method for protein side-chain packing and inverse folding”. In: *Proceedings of the National Academy of Sciences* 120.23 (June 6, 2023). Publisher: Proceedings of the National Academy of Sciences, e2216438120. DOI: 10.1073/pnas.2216438120. URL: <https://www.pnas.org/doi/10.1073/pnas.2216438120> (visited on 09/21/2023).
- [149] Éloi Mehr et al. “Manifold Learning in Quotient Spaces”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. ISSN: 2575-7075. June 2018, pp. 9165–9174. DOI: 10.1109/CVPR.2018.00955.
- [150] Hu Mei et al. “A new set of amino acid descriptors and its application in peptide QSARs”. en. In: *Peptide Science* 80.6 (2005). _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/bip.20296>, pp. 775–786. ISSN: 1097-0282. DOI: 10.1002/bip.20296. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/bip.20296> (visited on 09/05/2025).
- [151] Joshua Meier et al. “Language models enable zero-shot prediction of the effects of mutations on protein function”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 29287–29303. URL: <https://proceedings.neurips.cc/paper/2021/hash/f51338d736f95dd42427296047067694-Abstract.html> (visited on 06/28/2024).
- [152] Barthélemy Meynard-Piganeau et al. “TULIP: A transformer-based unsupervised language model for interacting peptides and T cell receptors that generalizes to unseen epitopes”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 121.24 (June 2024), e2316401121. ISSN: 0027-8424,1091-6490. DOI: 10.1073/pnas.2316401121.
- [153] Kevin Michalewicz, Mauricio Barahona, and Barbara Bravi. “ANTIPASTI: Interpretable prediction of antibody binding affinity exploiting normal modes and deep learning”. en. In: *Structure* 32.12 (Dec. 2024), 2422–2434.e5. ISSN: 0969-2126,1878-4186. DOI: 10.1016/j.str.2024.10.001.
- [154] Fin J. Milder et al. “Universal stabilization of the influenza hemagglutinin by structure-based redesign of the pH switch regions”. In: *Proceedings of the National Academy of Sciences* 119.6 (Feb. 2022). Publisher: Proceedings of the National Academy of Sciences, e2115379119. DOI: 10.1073/pnas.2115379119. URL: <https://www.pnas.org/doi/full/10.1073/pnas.2115379119> (visited on 09/19/2025).
- [155] Federico Miotello et al. “An Analysis of Musical Instruments Directivity Based on Spatial Complexity”. In: *2025 33rd European Signal Processing Conference (EUSIPCO)*. Sept. 2025, pp. 421–425. DOI: 10.23919/EUSIPCO63237.2025.11226645. URL: <https://ieeexplore.ieee.org/abstract/document/11226645> (visited on 04/22/2026).
- [156] Justin M Mirazee et al. “Enhancing CAR T cell therapy through in silico modeling of hinge sequence length based on epitope location”. en. In: *Biophys. J.* 123.3 (Feb. 2024), 552a. ISSN: 0006-3495,1542-0086. DOI: 10.1016/j.bpj.2023.11.3338.
- [157] Mikita Misiura et al. “DLPacker: Deep learning for prediction of amino acid side chain conformations in proteins”. In: *Proteins: Structure, Function, and Bioinformatics* 90.6 (2022). _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.26311>, pp. 1278–1290. ISSN: 1097-0134. DOI: 10.1002/prot.26311. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.26311> (visited on 10/04/2023).

- [158] Yorgo Modis et al. “A ligand-binding pocket in the dengue virus envelope glycoprotein”. In: *Proceedings of the National Academy of Sciences* 100.12 (June 2003). Publisher: Proceedings of the National Academy of Sciences, pp. 6986–6991. DOI: 10.1073/pnas.0832193100. URL: <https://www.pnas.org/doi/full/10.1073/pnas.0832193100> (visited on 01/07/2026).
- [159] Abhishek Mukhopadhyay et al. *ZymePackNet: rotamer-sampling free graph neural network method for protein sidechain prediction*. Pages: 2023.05.05.539648 Section: New Results. May 6, 2023. DOI: 10.1101/2023.05.05.539648. URL: <https://www.biorxiv.org/content/10.1101/2023.05.05.539648v1> (visited on 09/14/2023).
- [160] Philip Müller et al. *Rotation-Equivariant Deep Learning for Diffusion MRI*. Feb. 13, 2021. DOI: 10.48550/arXiv.2102.06942. arXiv: 2102.06942[cs]. URL: <http://arxiv.org/abs/2102.06942> (visited on 06/01/2023).
- [161] Albert Musaelian et al. *Learning Local Equivariant Representations for Large-Scale Atomistic Dynamics*. Apr. 11, 2022. DOI: 10.48550/arXiv.2204.05249. arXiv: 2204.05249[cond-mat,physics:physics]. URL: <http://arxiv.org/abs/2204.05249> (visited on 09/22/2022).
- [162] Felix Musil et al. “Physics-Inspired Structural Representations for Molecules and Materials”. In: *Chemical Reviews* 121.16 (Aug. 25, 2021). Publisher: American Chemical Society, pp. 9759–9815. ISSN: 0009-2665. DOI: 10.1021/acs.chemrev.1c00021. URL: <https://doi.org/10.1021/acs.chemrev.1c00021> (visited on 09/22/2022).
- [163] Athi N. Naganathan and Victor Muñoz. “Scaling of Folding Times with Protein Size”. In: *Journal of the American Chemical Society* 127.2 (Jan. 2005). Publisher: American Chemical Society, pp. 480–481. ISSN: 0002-7863. DOI: 10.1021/ja044449u. URL: <https://doi.org/10.1021/ja044449u> (visited on 04/30/2026).
- [164] Richard A Neher, Colin A Russell, and Boris I Shraiman. “Predicting evolution from the shape of genealogical trees”. en. In: *Elife* 3 (Nov. 2014). ISSN: 2050-084X. DOI: 10.7554/eLife.03568.
- [165] Thin Nguyen et al. “GraphDTA: predicting drug–target binding affinity with graph neural networks”. In: *Bioinformatics* 37.8 (Apr. 15, 2021), pp. 1140–1147. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btaa921. URL: <https://doi.org/10.1093/bioinformatics/btaa921> (visited on 01/24/2023).
- [166] Sean Nolan et al. “A large-scale database of T-cell receptor beta (TCR β) sequences and binding associations from natural and synthetic exposure to SARS-CoV-2”. en. In: *Front. Immunol.* 16 (Jan. 2025), p. 1488851. ISSN: 1664-3224. DOI: 10.3389/fimmu.2025.1488851.
- [167] Armita Nourmohammad, Michael Pun, and Gian Marco Visani. “Machine-Learning Model Reveals Protein-Folding Physics”. en. In: *Physics* 15 (Nov. 2022). Publisher: American Physical Society, p. 183. DOI: 10.1103/PhysRevLett.129.238101. URL: <https://physics.aps.org/articles/v15/183?s=09> (visited on 04/30/2026).
- [168] J. N. Onuchic, Z. Luthey-Schulten, and P. G. Wolynes. “Theory of protein folding: the energy landscape perspective”. eng. In: *Annual Review of Physical Chemistry* 48 (1997), pp. 545–600. ISSN: 0066-426X. DOI: 10.1146/annurev.physchem.48.1.545.
- [169] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. “DeepDTA: deep drug–target binding affinity prediction”. In: *Bioinformatics* 34.17 (Sept. 1, 2018), pp. i821–i829. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty593. URL: <https://doi.org/10.1093/bioinformatics/bty593> (visited on 01/24/2023).

- [170] Mark Palatucci et al. “Zero-shot Learning with Semantic Output Codes”. In: *Advances in Neural Information Processing Systems*. Vol. 22. Curran Associates, Inc., 2009. URL: https://papers.nips.cc/paper_files/paper/2009/hash/1543843a4723ed2ab08e18053ae6dc5b-Abstract.html (visited on 04/30/2026).
- [171] Albert C. Pan et al. “Quantitative Characterization of the Binding and Unbinding of Millimolar Drug Fragments with Molecular Dynamics Simulations”. In: *Journal of Chemical Theory and Computation* 13.7 (July 2017). Publisher: American Chemical Society, pp. 3372–3377. ISSN: 1549-9618. DOI: 10.1021/acs.jctc.7b00172. URL: <https://doi.org/10.1021/acs.jctc.7b00172> (visited on 04/30/2026).
- [172] Corrado Pancotti et al. “Predicting protein stability changes upon single-point mutation: a thorough comparison of the available tools on a new dataset”. In: *Briefings in Bioinformatics* 23.2 (Mar. 2022), bbab555. ISSN: 1477-4054. DOI: 10.1093/bib/bbab555. URL: <https://doi.org/10.1093/bib/bbab555> (visited on 06/28/2024).
- [173] Hahnbeom Park et al. “Simultaneous Optimization of Biomolecular Energy Functions on Features from Small Molecules and Macromolecules”. In: *Journal of Chemical Theory and Computation* 12.12 (Dec. 2016). Publisher: American Chemical Society, pp. 6201–6212. ISSN: 1549-9618. DOI: 10.1021/acs.jctc.6b00819. URL: <https://doi.org/10.1021/acs.jctc.6b00819> (visited on 09/11/2024).
- [174] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. ISSN: 1533-7928. URL: <http://jmlr.org/papers/v12/pedregosa11a.html> (visited on 09/23/2022).
- [175] Stefanos Pertigkiozoglou et al. *Recurrent Equivariant Constraint Modulation: Learning Per-Layer Symmetry Relaxation from Data*. arXiv:2602.02853 [cs]. Feb. 2026. DOI: 10.48550/arXiv.2602.02853. URL: <http://arxiv.org/abs/2602.02853> (visited on 05/06/2026).
- [176] Johannes Pettmann et al. “The discriminatory power of the T cell receptor”. en. In: *Elife* 10 (May 2021). ISSN: 2050-084X. DOI: 10.7554/eLife.67092.
- [177] Thanh T. N. Phan et al. “A conserved set of mutations for stabilizing soluble envelope protein dimers from dengue and Zika viruses to advance the development of subunit vaccines”. In: *Journal of Biological Chemistry* 298.7 (July 2022), p. 102079. ISSN: 0021-9258. DOI: 10.1016/j.jbc.2022.102079. URL: <https://www.sciencedirect.com/science/article/pii/S0021925822005191> (visited on 09/19/2025).
- [178] Rob Phillips et al. *Physical Biology of the Cell*. English. London : New York, NY: Garland Science, 2013. ISBN: 978-0-8153-4450-6.
- [179] Kurt H Piepenbrink et al. “Methods for quantifying T cell receptor binding affinities and thermodynamics”. en. In: *Methods Enzymol.* 466 (Nov. 2009), pp. 359–381. ISSN: 0076-6879,1557-7988. DOI: 10.1016/S0076-6879(09)66015-8.
- [180] Jay W. Ponder and David A. Case. “Force Fields for Protein Simulations”. In: *Advances in Protein Chemistry*. Vol. 66. Protein Simulations. Academic Press, Jan. 2003, pp. 27–85. DOI: 10.1016/S0065-3233(03)66002-X. URL: <https://www.sciencedirect.com/science/article/pii/S006532330366002X> (visited on 06/28/2024).

- [181] Fabrizio Pucci et al. “Quantification of biases in predictions of protein stability changes upon mutations”. In: *Bioinformatics* 34.21 (Nov. 2018), pp. 3659–3665. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty348. URL: <https://doi.org/10.1093/bioinformatics/bty348> (visited on 09/17/2025).
- [182] Michael N Pun et al. “Learning the shape of protein microenvironments with a holographic convolutional neural network”. In: *Proceedings of the National Academy of Sciences* 121.6 (2024), e2300838121. DOI: 10.1073/pnas.2300838121. eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.2300838121>.
- [183] Michael N. Pun. “Rotationally Equivariant Learning of Generalizable Protein Structure-To-Function Maps - ProQuest”. en. PhD thesis. Department of Physics, University of Washington, 2023. URL: <https://digital.lib.washington.edu/bitstreams/8997edca-80f6-4649-9ee4-64a04b132409/download> (visited on 04/24/2026).
- [184] Michael N. Pun et al. *Learning the shape of protein micro-environments with a holographic convolutional neural network*. Pages: 2022.10.31.514614 Section: New Results. Nov. 1, 2022. DOI: 10.1101/2022.10.31.514614. URL: <https://www.biorxiv.org/content/10.1101/2022.10.31.514614v1> (visited on 06/01/2023).
- [185] Michael N. Pun et al. “Learning the shape of protein microenvironments with a holographic convolutional neural network”. In: *Proceedings of the National Academy of Sciences* 121.6 (Feb. 2024). Publisher: Proceedings of the National Academy of Sciences. DOI: 10.1073/pnas.2300838121. URL: <https://www.pnas.org/doi/10.1073/pnas.2300838121> (visited on 09/16/2024).
- [186] Andrew G. T. Pyo et al. “Data-Driven Discovery of Biophysical T Cell Receptor Cospecificity Rules”. In: *PRX Life* 3.3 (July 2025). Publisher: American Physical Society, p. 033005. DOI: 10.1103/14j1-wrh5. URL: <https://link.aps.org/doi/10.1103/14j1-wrh5> (visited on 05/06/2026).
- [187] Marine C C Raman et al. “Direct molecular mimicry enables off-target cardiovascular toxicity by an enhanced affinity TCR designed for cancer immunotherapy”. en. In: *Sci. Rep.* 6.1 (Jan. 2016), p. 18851. ISSN: 2045-2322,2045-2322. DOI: 10.1038/srep18851.
- [188] Birkir Reynisson et al. “NetMHCpan-4.1 and NetMHCIIpan-4.0: improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data”. In: *Nucleic Acids Research* 48.W1 (July 2020), W449–W454. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa379. URL: <https://doi.org/10.1093/nar/gkaa379> (visited on 01/28/2025).
- [189] Eve Richardson et al. “IMMREP25: Unseen Peptides”. In: *bioRxiv* (2026). Publisher: Cold Spring Harbor Laboratory, pp. 2026–03. URL: <https://www.biorxiv.org/content/10.64898/2026.03.30.715276.abstract> (visited on 04/30/2026).
- [190] Adam J. Riesselman, John B. Ingraham, and Debora S. Marks. “Deep generative models of genetic variation capture the effects of mutations”. en. In: *Nature Methods* 15.10 (Oct. 2018). Publisher: Nature Publishing Group, pp. 816–822. ISSN: 1548-7105. DOI: 10.1038/s41592-018-0138-4. URL: <https://www.nature.com/articles/s41592-018-0138-4> (visited on 07/02/2024).
- [191] Timothy P Riley et al. “T cell receptor cross-reactivity expanded by dramatic peptide-MHC adaptability”. en. In: *Nat. Chem. Biol.* 14.10 (Oct. 2018), pp. 934–942. ISSN: 1552-4450, 1552-4469. DOI: 10.1038/s41589-018-0130-4.

- [192] Alexander Rives et al. “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 118.15 (Apr. 2021). issn: 0027-8424, 1091-6490. doi: 10.1073/pnas.2016239118.
- [193] James Robinson et al. “IMGT/HLA and IMGT/MHC: sequence databases for the study of the major histocompatibility complex”. en. In: *Nucleic Acids Res.* 31.1 (Jan. 2003), pp. 311–314. issn: 1362-4962,0305-1048. doi: 10.1093/nar/gkg070.
- [194] Gabriel J. Rocklin et al. “Global analysis of protein folding using massively parallel design, synthesis, and testing”. In: *Science* 357.6347 (July 2017). Publisher: American Association for the Advancement of Science, pp. 168–175. doi: 10.1126/science.aan0693. URL: <https://www.science.org/doi/10.1126/science.aan0693> (visited on 07/05/2024).
- [195] Luis A Rojas et al. “Personalized RNA neoantigen vaccines stimulate T cells in pancreatic cancer”. en. In: *Nature* (May 2023), pp. 1–7. issn: 0028-0836, 1476-4687. doi: 10.1038/s41586-023-06063-y.
- [196] James P Roney and Sergey Ovchinnikov. “State-of-the-Art Estimation of Protein Model Accuracy Using AlphaFold”. en. In: *Phys. Rev. Lett.* 129.23 (Dec. 2022), p. 238101. issn: 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.129.238101.
- [197] James P. Roney, Chenxi Ou, and Sergey Ovchinnikov. *Protein Diffusion Models as Statistical Potentials*. en. ISSN: 2692-8205 Pages: 2025.12.09.693073 Section: New Results. Mar. 2026. doi: 10.64898/2025.12.09.693073. URL: <https://www.biorxiv.org/content/10.64898/2025.12.09.693073v3> (visited on 04/30/2026).
- [198] James P. Roney and Sergey Ovchinnikov. “State-of-the-Art Estimation of Protein Model Accuracy Using AlphaFold”. In: *Physical Review Letters* 129.23 (Nov. 2022). Publisher: American Physical Society, p. 238101. doi: 10.1103/PhysRevLett.129.238101. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.129.238101> (visited on 04/30/2026).
- [199] Andrew Rosenberg and Julia Hirschberg. “V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure”. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. CoNLL-EMNLP 2007. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 410–420. URL: <https://aclanthology.org/D07-1043> (visited on 09/21/2022).
- [200] Bernhard Rupp. *Biomolecular Crystallography: Principles, Practice, and Application to Structural Biology*. English. New York: Garland Science, 2010. isbn: 978-0-8153-4081-2.
- [201] Lee Sael and Daisuke Kihara. “Improved protein surface comparison and application to low-resolution protein structure data”. In: *BMC Bioinformatics* 11.Suppl 11 (Dec. 2010), S2. issn: 1471-2105. doi: 10.1186/1471-2105-11-S11-S2. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC3024873/> (visited on 05/05/2026).
- [202] Ugur Sahin and Özlem Türeci. “Personalized vaccines for cancer immunotherapy”. en. In: *Science* 359.6382 (Mar. 2018), pp. 1355–1360. issn: 0036-8075, 1095-9203. doi: 10.1126/science.aar7112.
- [203] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. *E(n) Equivariant Graph Neural Networks*. Feb. 16, 2022. doi: 10.48550/arXiv.2102.09844. arXiv: 2102.09844[cs, stat]. URL: <http://arxiv.org/abs/2102.09844> (visited on 07/17/2022).

- [204] T. Savioli et al. “Engine Mass Flow Estimation through Neural Network Modeling in Semi-Transient Conditions: A New Calibration Approach”. en. In: *Fluids* 9.10 (Oct. 2024). Publisher: Multidisciplinary Digital Publishing Institute, p. 239. ISSN: 2311-5521. DOI: 10.3390/fluids9100239. URL: <https://www.mdpi.com/2311-5521/9/10/239> (visited on 04/30/2026).
- [205] Erhard Schmidt. “Zur Theorie der linearen und nichtlinearen Integralgleichungen”. In: *Mathematische Annalen* 63.4 (Dec. 1, 1907), pp. 433–476. ISSN: 1432-1807. DOI: 10.1007/BF01449770. URL: <https://doi.org/10.1007/BF01449770> (visited on 09/22/2022).
- [206] Robert Schmirler, Michael Heinzinger, and Burkhard Rost. “Fine-tuning protein language models boosts predictions across diverse tasks”. en. In: *Nat. Commun.* 15.1 (Aug. 2024), p. 7407. ISSN: 2041-1723,2041-1723. DOI: 10.1038/s41467-024-51844-2.
- [207] Kristof T. Schütt et al. *SchNet: A continuous-filter convolutional neural network for modeling quantum interactions*. arXiv:1706.08566 [stat]. Dec. 2017. DOI: 10.48550/arXiv.1706.08566. URL: <http://arxiv.org/abs/1706.08566> (visited on 04/22/2026).
- [208] Joost Schymkowitz et al. “The FoldX web server: an online force field”. In: *Nucleic Acids Research* 33.suppl_2 (July 2005), W382–W388. ISSN: 0305-1048. DOI: 10.1093/nar/gki387. URL: <https://doi.org/10.1093/nar/gki387> (visited on 07/05/2024).
- [209] *SE(3)-Transformers for PyTorch | NVIDIA NGC*. NVIDIA NGC Catalog. URL: https://catalog.ngc.nvidia.com/orgs/nvidia/resources/se3transformer_for_pytorch (visited on 10/04/2023).
- [210] Andrew K Sewell. “Why must T cells be cross-reactive?” en. In: *Nat. Rev. Immunol.* 12.9 (Sept. 2012), pp. 669–677. ISSN: 1474-1733,1474-1741. DOI: 10.1038/nri3279.
- [211] Sisi Shan et al. “Deep learning guided optimization of human antibody against SARS-CoV-2 variants with broad neutralization”. In: *Proceedings of the National Academy of Sciences* 119.11 (Mar. 2022). Publisher: Proceedings of the National Academy of Sciences, e2122954119. DOI: 10.1073/pnas.2122954119. URL: <https://www.pnas.org/doi/full/10.1073/pnas.2122954119> (visited on 09/11/2024).
- [212] M. Shanker, M. Y. Hu, and M. S. Hung. “Effect of data standardization on neural network training”. In: *Omega* 24.4 (Aug. 1, 1996), pp. 385–397. ISSN: 0305-0483. DOI: 10.1016/0305-0483(96)00010-2. URL: <https://www.sciencedirect.com/science/article/pii/0305048396000102> (visited on 09/23/2022).
- [213] David E. Shaw et al. “Anton 3: twenty microseconds of molecular dynamics simulation before lunch”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC ’21. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 1–11. ISBN: 978-1-4503-8442-1. DOI: 10.1145/3458817.3487397. URL: <https://dl.acm.org/doi/10.1145/3458817.3487397> (visited on 04/30/2026).
- [214] *SHREC 2017: Large-scale 3D Shape Retrieval from ShapeNet Core55*. URL: <https://shapenet.cs.stanford.edu/shrec17/> (visited on 09/23/2022).
- [215] Zhixin Shu et al. *Deforming Autoencoders: Unsupervised Disentangling of Shape and Appearance*. June 18, 2018. DOI: 10.48550/arXiv.1806.06503. arXiv: 1806.06503[cs]. URL: <http://arxiv.org/abs/1806.06503> (visited on 09/22/2022).
- [216] Mikhail Shugay et al. “VDJdb: a curated database of T-cell receptor sequences with known antigen specificity”. en. In: *Nucleic Acids Res.* 46.D1 (Jan. 2018), pp. D419–D427. ISSN: 1362-4962,0305-1048. DOI: 10.1093/nar/gkx760.

- [217] Kihyuk Sohn, Honglak Lee, and Xinchun Yan. “Learning Structured Output Representation using Deep Conditional Generative Models”. In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. URL: https://papers.nips.cc/paper_files/paper/2015/hash/8d55a249e6baa5c06772297520da2051-Abstract.html (visited on 10/19/2023).
- [218] Vignesh Ram Somnath, Charlotte Bunne, and Andreas Krause. *Multi-Scale Representation Learning on Proteins*. Apr. 4, 2022. DOI: 10.48550/arXiv.2204.02337. arXiv: 2204.02337[cs, q-bio]. URL: <http://arxiv.org/abs/2204.02337> (visited on 01/24/2023).
- [219] Kefan Song and Suzie H Pun. “Design and evaluation of synthetic delivery formulations for peptide-based cancer vaccines”. en. In: *BME Front.* 5 (Mar. 2024), p. 0038. ISSN: 2765-8031. DOI: 10.34133/bmef.0038.
- [220] Carolina M Soto et al. “MHC-class I-restricted CD4 T cells: a nanomolar affinity TCR has improved anti-tumor efficacy in vivo compared to the micromolar wild-type TCR”. en. In: *Cancer Immunol. Immunother.* 62.2 (Feb. 2013), pp. 359–369. ISSN: 0340-7004,1432-0851. DOI: 10.1007/s00262-012-1336-z.
- [221] Tyler N. Starr et al. “Deep Mutational Scanning of SARS-CoV-2 Receptor Binding Domain Reveals Constraints on Folding and ACE2 Binding”. In: *Cell* 182.5 (Sept. 2020), 1295–1310.e20. ISSN: 0092-8674. DOI: 10.1016/j.cell.2020.08.012. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7418704/> (visited on 07/05/2024).
- [222] Jennifer D Stone, Adam S Chervin, and David M Kranz. “T-cell receptor binding affinities and kinetics: impact on T-cell activity and specificity”. en. In: *Immunology* 126.2 (Feb. 2009), pp. 165–176. ISSN: 0019-2805,1365-2567. DOI: 10.1111/j.1365-2567.2008.03015.x.
- [223] Jan Stourac et al. “FireProtDB: database of manually curated protein stability data”. In: *Nucleic Acids Research* 49.D1 (Jan. 2021), pp. D319–D324. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa981. URL: <https://doi.org/10.1093/nar/gkaa981> (visited on 10/13/2025).
- [224] Minyi Su et al. “Comparative Assessment of Scoring Functions: The CASF-2016 Update”. In: *Journal of Chemical Information and Modeling* 59.2 (Feb. 25, 2019). Publisher: American Chemical Society, pp. 895–913. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.8b00545. URL: <https://doi.org/10.1021/acs.jcim.8b00545> (visited on 01/24/2023).
- [225] Kyle Swanson, Howard Chang, and James Zou. “Predicting Immune Escape with Pretrained Protein Language Model Embeddings”. In: *Proceedings of the 17th Machine Learning in Computational Biology meeting*. Machine Learning in Computational Biology. ISSN: 2640-3498. PMLR, Dec. 19, 2022, pp. 110–130. URL: <https://proceedings.mlr.press/v200/swanson22a.html> (visited on 01/24/2023).
- [226] Daniel M Tadros et al. “The MHC Motif Atlas: a database of MHC binding specificities and ligands”. en. In: *Nucleic Acids Res.* 51.D1 (Jan. 2023), pp. D428–D437. ISSN: 0305-1048,1362-4962. DOI: 10.1093/nar/gkac965.
- [227] Timothy J. C. Tan et al. “High-throughput identification of prefusion-stabilizing mutations in SARS-CoV-2 spike”. en. In: *Nature Communications* 14.1 (Apr. 2023). Publisher: Nature Publishing Group, p. 2003. ISSN: 2041-1723. DOI: 10.1038/s41467-023-37786-1. URL: <https://www.nature.com/articles/s41467-023-37786-1> (visited on 04/30/2026).
- [228] Fangting Tao et al. “Reliable prediction of protein–protein binding affinity changes upon mutations with Pythia-PPI”. In: *National Science Review* 12.6 (June 2025), nwaf231. ISSN: 2095-5138. DOI: 10.1093/nsr/nwaf231. URL: <https://doi.org/10.1093/nsr/nwaf231> (visited on 10/13/2025).

- [229] Nicole N. Thadani et al. “Learning from prepandemic data to forecast viral escape”. en. In: *Nature* 622.7984 (Oct. 2023). Publisher: Nature Publishing Group, pp. 818–825. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06617-0. URL: <https://www.nature.com/articles/s41586-023-06617-0> (visited on 07/05/2024).
- [230] Nathaniel Thomas et al. *Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds*. May 18, 2018. DOI: 10.48550/arXiv.1802.08219. arXiv: 1802.08219[cs]. URL: <http://arxiv.org/abs/1802.08219> (visited on 07/16/2022).
- [231] Raphael Townshend et al. “ATOM3D: Tasks on Molecules in Three Dimensions”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1* (Dec. 6, 2021). URL: <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c45147dee729311ef5b5c3003946c48f-Abstract-round1.html> (visited on 01/24/2023).
- [232] Kotaro Tsuboyama et al. “Mega-scale experimental analysis of protein folding stability in biology and design”. en. In: *Nature* 620.7973 (Aug. 2023). Publisher: Nature Publishing Group, pp. 434–444. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06328-6. URL: <https://www.nature.com/articles/s41586-023-06328-6> (visited on 04/14/2025).
- [233] Wu-Ki Tung. *Group Theory in Physics*. 1985. ISBN: 978-9971-966-57-7.
- [234] Martin Uhrin. “Through the eyes of a descriptor: Constructing complete, invertible descriptions of atomic environments”. In: *Physical Review B* 104.14 (Oct. 26, 2021), p. 144110. ISSN: 2469-9950, 2469-9969. DOI: 10.1103/PhysRevB.104.144110. arXiv: 2104.09319[cond-mat]. URL: <http://arxiv.org/abs/2104.09319> (visited on 07/17/2022).
- [235] Pascal Vincent et al. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. ICML ’08. New York, NY, USA: Association for Computing Machinery, July 5, 2008, pp. 1096–1103. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294. URL: <https://doi.org/10.1145/1390156.1390294> (visited on 11/18/2022).
- [236] Gian Marco Visani, Aayush Verma, and William S. DeWitt. *Additive baselines furnish no evidence for epistasis learning by MULTI-evolve*. en. ISSN: 2692-8205 Pages: 2026.04.23.719915 Section: Contradictory Results. Apr. 2026. DOI: 10.64898/2026.04.23.719915. URL: <https://www.biorxiv.org/content/10.64898/2026.04.23.719915v1> (visited on 04/30/2026).
- [237] Gian Marco Visani et al. “H-Packer: Holographic Rotationally Equivariant Convolutional Neural Network for Protein Side-Chain Packing”. en. In: *Proceedings of the 18th Machine Learning in Computational Biology meeting*. ISSN: 2640-3498. PMLR, Mar. 2024, pp. 230–249. URL: <https://proceedings.mlr.press/v240/visani24a.html> (visited on 06/28/2024).
- [238] Gian Marco Visani et al. “HERMES: Holographic Equivariant neuRal network model for Mutational Effect and Stability prediction”. en. In: *bioRxiv* 2024.07.09.602403 (Oct. 2024). DOI: 10.1101/2024.07.09.602403.
- [239] Gian Marco Visani et al. *HERMES: Holographic Equivariant neuRal network model for Mutational Effect and Stability prediction*. en. ISSN: 2692-8205 Pages: 2024.07.09.602403 Section: New Results. Jan. 2026. DOI: 10.1101/2024.07.09.602403. URL: <https://www.biorxiv.org/content/10.1101/2024.07.09.602403v4> (visited on 04/24/2026).

- [240] Gian Marco Visani et al. “Holographic-(V)AE: An end-to-end $SO(3)$ -equivariant (variational) autoencoder in Fourier space”. In: *Physical Review Research* 6.2 (Apr. 2024). Publisher: American Physical Society, p. 023006. DOI: 10.1103/PhysRevResearch.6.023006. URL: <https://link.aps.org/doi/10.1103/PhysRevResearch.6.023006> (visited on 06/28/2024).
- [241] Gian Marco Visani et al. “T cell receptor specificity landscape revealed through de novo peptide design”. In: *Proceedings of the National Academy of Sciences* 122.42 (Oct. 2025). Publisher: Proceedings of the National Academy of Sciences, e2504783122. DOI: 10.1073/pnas.2504783122. URL: <https://www.pnas.org/doi/10.1073/pnas.2504783122> (visited on 12/16/2025).
- [242] Randi Vita et al. “The Immune Epitope Database (IEDB): 2018 update”. en. In: *Nucleic Acids Res.* 47.D1 (Jan. 2019), pp. D339–D343. ISSN: 1362-4962,0305-1048. DOI: 10.1093/nar/gky1006.
- [243] Mathias Vormehr, Türeci, Ö, and Ugur Sahin. “Harnessing tumor mutations for truly individualized cancer vaccines”. en. In: *Annu. Rev. Med.* 70.1 (Jan. 2019), pp. 395–407. ISSN: 1545-326X,0066-4219. DOI: 10.1146/annurev-med-042617-101816.
- [244] Xiaoyu Wang et al. “D3DistalMutation: a Database to Explore the Effect of Distal Mutations on Enzyme Activity”. In: *Journal of Chemical Information and Modeling* 61.5 (May 2021). Publisher: American Chemical Society, pp. 2499–2508. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.1c00318. URL: <https://doi.org/10.1021/acs.jcim.1c00318> (visited on 07/05/2024).
- [245] Yiquan Wang et al. “An explainable language model for antibody specificity prediction using curated influenza hemagglutinin antibodies”. en. In: *Immunity* 57.10 (Oct. 2024), 2453–2465.e7. ISSN: 1074-7613,1097-4180. DOI: 10.1016/j.immuni.2024.07.022.
- [246] J. J. Ward et al. “Prediction and functional analysis of native disorder in proteins from the three kingdoms of life”. eng. In: *Journal of Molecular Biology* 337.3 (Mar. 2004), pp. 635–645. ISSN: 0022-2836. DOI: 10.1016/j.jmb.2004.02.002.
- [247] Joseph L. Watson et al. *Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models*. Pages: 2022.12.09.519842 Section: New Results. Dec. 10, 2022. DOI: 10.1101/2022.12.09.519842. URL: <https://www.biorxiv.org/content/10.1101/2022.12.09.519842v1> (visited on 10/23/2023).
- [248] Joseph L. Watson et al. “De novo design of protein structure and function with RFDiffusion”. en. In: *Nature* 620.7976 (Aug. 2023). Publisher: Nature Publishing Group, pp. 1089–1100. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06415-8. URL: <https://www.nature.com/articles/s41586-023-06415-8> (visited on 05/05/2026).
- [249] Andrew I Webb et al. “Functional and structural characteristics of NY-ESO-1-related HLA A2-restricted epitopes and the design of a novel immunogenic analogue”. en. In: *J. Biol. Chem.* 279.22 (May 2004), pp. 23438–23446. ISSN: 0021-9258,1083-351X. DOI: 10.1074/jbc.M314066200.
- [250] Anna Weber, Jannis Born, and María Rodríguez Martínez. “TITAN: T-cell receptor specificity prediction with bimodal attention networks”. en. In: *Bioinformatics* 37.Suppl_1 (July 2021), pp. i237–i244. ISSN: 1367-4803,1367-4811. DOI: 10.1093/bioinformatics/btab294.
- [251] Maurice Weiler et al. “3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/488e4104520c6aab692863cc1dba45af-Paper.pdf.

- [252] Robin Winter et al. *Unsupervised Learning of Group Invariant and Equivariant Representations*. Feb. 15, 2022. DOI: 10.48550/arXiv.2202.07559. arXiv: 2202.07559[cs]. URL: <http://arxiv.org/abs/2202.07559> (visited on 07/17/2022).
- [253] Linda Wooldridge et al. “A Single Autoimmune T Cell Receptor Recognizes More Than a Million Different Peptides *”. en. In: *J. Biol. Chem.* 287.2 (Jan. 2012), pp. 1168–1177. ISSN: 0021-9258, 1083-351X. DOI: 10.1074/jbc.M111.289488.
- [254] J. Michael Word et al. “Asparagine and glutamine: using hydrogen atom contacts in the choice of side-chain amide orientation1”. In: *Journal of Molecular Biology* 285.4 (Jan. 1999), pp. 1735–1747. ISSN: 0022-2836. DOI: 10.1006/jmbi.1998.2401. URL: <https://www.sciencedirect.com/science/article/pii/S0022283698924019> (visited on 06/28/2024).
- [255] Daniel E. Worrall et al. *Harmonic Networks: Deep Translation and Rotation Equivariance*. Apr. 11, 2017. DOI: 10.48550/arXiv.1612.04642. arXiv: 1612.04642[cs,stat]. URL: <http://arxiv.org/abs/1612.04642> (visited on 09/22/2022).
- [256] Fang Wu et al. “Pre-training of Equivariant Graph Matching Networks with Conformation Flexibility for Drug Binding”. In: *Advanced Science* 9.33 (Nov. 2022), p. 2203796. ISSN: 2198-3844, 2198-3844. DOI: 10.1002/advs.202203796. arXiv: 2204.08663[cs]. URL: <http://arxiv.org/abs/2204.08663> (visited on 01/24/2023).
- [257] Fang Wu et al. *When Geometric Deep Learning Meets Pretrained Protein Language Models*. Dec. 6, 2022. DOI: 10.48550/arXiv.2212.03447. arXiv: 2212.03447[cs,q-bio]. URL: <http://arxiv.org/abs/2212.03447> (visited on 01/24/2023).
- [258] Xinbo Yang et al. “Structural basis for clonal diversity of the public T cell response to a dominant human Cytomegalovirus Epitope”. en. In: *J. Biol. Chem.* 290.48 (Nov. 2015), pp. 29106–29119. ISSN: 0021-9258,1083-351X. DOI: 10.1074/jbc.M115.691311.
- [259] Jason Yim et al. *SE(3) diffusion model with application to protein backbone generation*. May 22, 2023. DOI: 10.48550/arXiv.2302.02277. arXiv: 2302.02277[cs,q-bio,stat]. URL: <http://arxiv.org/abs/2302.02277> (visited on 10/23/2023).
- [260] Rui Yin et al. “TCRmodel2: high-resolution modeling of T cell receptor recognition using deep learning”. en. In: *Nucleic Acids Res.* (May 2023). ISSN: 0305-1048, 1362-4962. DOI: 10.1093/nar/gkad356.
- [261] Shu-Qi Zhang et al. “High-throughput determination of the antigen specificities of T cell receptors in single cells”. en. In: *Nat. Biotechnol.* 36.12 (Nov. 2018), pp. 1156–1159. ISSN: 1087-0156,1546-1696. DOI: 10.1038/nbt.4282.
- [262] Yangtian Zhang et al. *DiffPack: A Torsional Diffusion Model for Autoregressive Protein Side-Chain Packing*. June 1, 2023. DOI: 10.48550/arXiv.2306.01794. arXiv: 2306.01794[cs,q-bio]. URL: <http://arxiv.org/abs/2306.01794> (visited on 09/21/2023).
- [263] Zhidian Zhang et al. “Protein language models learn evolutionary statistics of interacting sequence motifs”. In: *Proceedings of the National Academy of Sciences* 121.45 (Nov. 2024). Publisher: Proceedings of the National Academy of Sciences, e2406285121. DOI: 10.1073/pnas.2406285121. URL: <https://www.pnas.org/doi/10.1073/pnas.2406285121> (visited on 05/06/2026).
- [264] Shi Zhong et al. “T-cell receptor affinity and avidity defines antitumor response and autoimmunity in T-cell immunotherapy”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 110.17 (Apr. 2013), pp. 6973–6978. ISSN: 0027-8424,1091-6490. DOI: 10.1073/pnas.1221609110.

Appendix A

SUPPLEMENT FOR CHAPTER 3

A.1 Modeling details

A.1.1 Tuning the hyperparameters of the training objective

We find it practical to scale the reconstruction loss by a dataset-specific scalar α since the MSE loss varies in average magnitude across datasets. When training H-VAE, we find it beneficial to keep $\beta = 0$ for a few epochs (E_{rec}) so that the model can learn to perform meaningful reconstructions, and then linearly increasing it to the desired value for E_{warmup} epochs to add structure to the latent space, an approach first used by Bowman et al. [24].

A.1.2 Data normalization

As per standard machine learning practice [212], we normalize the data. We do this by dividing each tensor by the average square-root total norm of the training tensors, analogously to the Signal Norm. This strategy puts the target values on a similar scale as the normalized activations learned by the network, which we speculate to favor gradient flow.

A.1.3 Implementation details

Without loss of generality, we use real spherical harmonics for implementation of H-(V)AE. We leverage e3nn [76], using their computation of the real spherical harmonics and their Clebsch-Gordan coefficients.

In our code, we offer the option to use the Full Tensor Product instead of the ETP. Specifically, at each block we allow the users to specify whether to compute the Tensor Product element-wise or fully-connected within the multiplicity, and whether to compute using efficient or fully connected degree mixing.

A.1.4 Pairwise invariant reconstruction loss

To reconstruct a signal within an equivariant model it is desirable to have a *pairwise invariant* reconstruction loss, i.e., a loss $\mathcal{L}_{\text{rec}}$ such that $\mathcal{L}_{\text{rec}}(\mathbf{x}, \mathbf{y}) = \mathcal{L}_{\text{rec}}(\mathbf{D}(\mathbf{g})\mathbf{x}, \mathbf{D}(\mathbf{g})\mathbf{y})$ where $\mathbf{D}$ is the representation of the group element $\mathbf{g}$ acting on the space that $\mathbf{x}$ and $\mathbf{y}$ inhabit (e.g. a rotation matrix if $\mathbf{x}$ and $\mathbf{y}$ are vectors in Euclidean 3D space, or a degree- ℓ wigner-D matrix if $\mathbf{x}$ and $\mathbf{y}$ are degree- ℓ vectors). This property is necessary for the model to remain equivariant, i.e., given that the network is agnostic to the transformation of the input under group operation $\mathbf{x} \rightarrow \mathbf{D}(\mathbf{g})\mathbf{x}$ by producing a similarly transformed output $\mathbf{y} \rightarrow \mathbf{D}(\mathbf{g})\mathbf{y}$, we want the reconstruction loss to be agnostic to the same kind of transformation as well

The MSE loss is pairwise invariant for any degree- ℓ feature on which $\text{SO}(3)$ acts via the ℓ 's Wigner-D matrix. Consider two degree- ℓ features $\mathbf{x}_\ell$ and $\mathbf{y}_\ell$ acted upon by a Wigner-D matrix $D_\ell(\mathbf{g})$ parameterized

by rotation $\mathbf{g}$ (we drop the $\mathbf{g}$ and ℓ indexing for clarity):

$$\begin{aligned}
 \text{MSE}(\mathbf{D}\mathbf{x}, \mathbf{D}\mathbf{y}) &= (\mathbf{D}\mathbf{x} - \mathbf{D}\mathbf{y})^T (\mathbf{D}\mathbf{x} - \mathbf{D}\mathbf{y}) \\
 &= (\mathbf{D}(\mathbf{x} - \mathbf{y}))^T (\mathbf{D}(\mathbf{x} - \mathbf{y})) \\
 &= (\mathbf{x} - \mathbf{y})^T \mathbf{D}^T \mathbf{D} (\mathbf{x} - \mathbf{y}) \\
 &= (\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}) \quad \text{since Wigner-D matrices are unitary} \\
 &= \text{MSE}(\mathbf{x}, \mathbf{y})
 \end{aligned} \tag{A.1.1}$$

Since the MSE loss is pairwise invariant for every pair of degree- ℓ features, it is thus pairwise invariant for pairs of steerable tensors composed via direct products of steerable features.

A.1.5 Cosine loss

We use the metric *Cosine loss* to measure a model's reconstruction ability. Cosine loss is a normalized dot product generalized to operate on pairs of steerable tensors (akin to cosine similarity), and modified to be interpreted as a loss (see Sec. A.1.5 for details),

$$\begin{aligned}
 \text{Cosine}(\mathbf{x}, \mathbf{y}) &= 1 - \frac{\mathbf{x} \odot \mathbf{y}}{\sqrt{(\mathbf{x} \odot \mathbf{x})(\mathbf{y} \odot \mathbf{y})}}, \\
 \text{with } \mathbf{x} \odot \mathbf{y} &= \sum_{\ell'} (\mathbf{x}_{\ell'} \otimes_{cg} \mathbf{y}_{\ell'})_{\ell=0}
 \end{aligned} \tag{A.1.2}$$

Importantly, unlike MSE, which depends on the characteristics of the data (e.g., the size of the data tensors), Cosine loss is dimensionless and therefore, interpretable and comparable across datasets. A measure with these characteristics is practically useful for training of a network because it provides an estimate for how much better the reconstructions can get if the network's hyperparameters were to be further optimized. For example, looking at the Cosine loss in Table A.4.5, we see that our model trained on Shrec17 (best Cosine = 0.130) is not as well optimized as our model trained on MNIST (best Cosine = 0.017). Using MSE, the trend is reversed (1.8×10^{-3} vs. 6.7×10^{-3}), since the scale of MSE depends on the size of the irreps of the data (Fig. A.4.4). Nonetheless, as a measure of a model's reconstruction ability, Cosine loss correlates almost perfectly with MSE for a given dataset and network, especially in the mid-to-low reconstruction quality regime (SpearmanR = 0.99, Fig. A.4.4 and Table A.4.5). It should also be noted that since Cosine loss ignores the relative norms of data features, it is unable to reconstruct norms and thus, not suitable as a training objective.

Proof: Invariance of Cosine loss. The generalized dot product $\odot$ from Eq. A.1.2 is pairwise invariant in the same way that the dot product between two 3D vectors depends only on their relative orientations but not the global orientation of the whole two-vector system. Therefore, the whole Cosine loss expression is pairwise invariant, since all of its components are pairwise invariant.

A.2 Using a non-equivariant decoder

Winter *et al* [252] propose to construct group-equivariant autoencoders by using an equivariant encoder that learns an invariant embedding and a group element, and an unconstrained decoder which uses the invariants alone to reconstruct each datapoint in the "canonical" form, before applying the learned group action in the output space. By contrast, for SO(3) we propose to use an equivariant decoder, whereby the learned group

element is fed as input to the decoder. Such “unconstrained decoder” procedure can in principle be merged with our equivariant encoder and Fourier-space approach in two ways. For each, we argue in favor of using our equivariant decoder.

1) Reconstructing the Fourier coefficients of the data. To apply the learned group element on the decoder’s output, the Wigner-D matrices for the data’s irreps need to be computed from the group element. Then, the Wigner-D matrices can be used to “rotate” the tensor. This has to be done on-the-fly, and it can be done quickly using functions provided in the e3nn package [76] and by smartly vectorizing operations. We implemented this procedure by using a simple Multi-Layer Perceptron with SiLU non-linearities as a decoder. By using e3nn to compute Wigner-D matrices in batches, and by clever construction of tensor multiplications such that runtime scales linearly with ℓ_{max} and is constant with regards to multiplicity and batch size, we achieve models that run with comparable speed to those using our equivariant decoder, and have comparable performance on MNIST (Table A.2.1). Given the empirical similarities we observe, though on a limited use case, we favor the simplicity and elegance of our equivariant decoder. “Simplicity” because we construct the decoder to be symmetric to the encoder, thus endowing it automatically with similar representational power and without the need to tune an architecture made with different base components. Furthermore, we highlight that our method generates intermediate equivariant representations in the decoder, rather than intermediate invariant representations. These intermediate equivariant representations may be of interest to study in and of themselves.

2) Reconstructing the data in real space. In this case, we do not have to compute Wigner-D matrices on-the-fly, since the learned frame can be used directly in the output space as a rotation matrix. However, since the encoder only sees a truncated Fourier representation of the data, which is by construction lossy, while the loss is computed over fine-grained real-space, this model might be too difficult to train. We suspect this would make the model akin to a denoising autoencoder [235] and it might be interesting to analyze, but that would be beyond the scope of this paper. To avoid the denoising effect, we could learn to reconstruct data in real space after an Inverse Fourier Transform (IFT). However, computing the IFT on-the-fly is very expensive and requires a discretization of the input space, to the point of being prohibitive for point clouds. This is not a bottleneck for the Forward Fourier Transform if the cloud is parameterized by Dirac-Delta distributions, i.e., for point clouds, as the integral can be computed exactly (Eq. 2.8).

Method	z	Speed	MSE	Cosine	Purity	V-meas.	LC Class. Acc.	KNN Class. Acc.
H-AE NR/R	16	1.0	1.2×10^{-3}	0.031	0.66	0.55	0.850	0.902
H-AE unconst. decoder NR/R	16	1.3	1.1×10^{-3}	0.028	0.62	0.53	0.838	0.907
H-VAE NR/R	16	1.0	2.8×10^{-3}	0.068	0.73	0.60	0.878	0.897
H-VAE unconst. decoder NR/R	16	1.3	2.9×10^{-3}	0.073	0.68	0.56	0.850	0.872

Table A.2.1: **Performance comparison between our H-(V)AE and a H-(V)AE with ref. [252]’s non-equivariant decoder formulation, on the MNIST-on-the-sphere dataset.**

The non-equivariant decoders are constructed as simple MLPs with SiLU non-linearities, with the following hidden layer sizes: [32,64,128,160,256]. We keep the number of parameters approximately the same to make model comparison fair, but we do not tune the architecture of the invariant decoders. All other training details are kept the same (Sec. A.3).

A.3 Experimental details

A.3.1 Architecture specification

We describe model architectures as follows. We specify the number of blocks B , which is the same for the encoder and the decoder. We specify two lists, (i) DegreesList which contains the maximum degree $\ell_{max,b}$ of

the output of each block b , and (ii) MultList, containing the multiplicity sizes C_b , of each block b . These lists are in the order as they appear in the encoder, and are reversed for the decoder. When it applies, we specify the output multiplicity of the initial linear projection C_{init} . As noted in the main text, we use a fixed formula to determine $\ell_{\text{max},b}$, but we specify it for clarity.

A.3.2 MNIST on the sphere

Model architectures. For models with invariant latent space size $z = 16$, we use 6 blocks, DegreesList = [10, 10, 8, 4, 2, 1] and MultList = [16, 16, 16, 16, 16, 16], with a total of 227k parameters.

For models with invariant latent space size $z = 120$, we use 6 blocks, DegreesList = [10, 10, 8, 4, 2, 1] and MultList = [16, 16, 16, 32, 64, 120], with a total of 453k parameters.

Training details. We keep the learning schedule as similar as possible for all models. We use $\alpha = 50$. We train VAE models for 80 epochs using the Adam optimizer [113] with default parameters, a batch size of 100, and an initial learning rate of 0.001, which we decrease exponentially by one order of magnitude over 25 epochs. We tune β according to the procedure in outlined below, $E_{\text{rec}} = 25$ and $E_{\text{warmup}} = 35$. We instead train AE models for 50 epochs using the "reduce learning rate on plateau" scheme, reducing the learning rate by a factor of 5 if the validation loss has not improved for over one epoch. We utilize the model with the lowest loss on validation data, only after the end of the warmup epochs for VAE models. Training took ~ 3.3 minutes per epoch on a single NVIDIA A40 GPU for each model.

Tuning the regularization strength β with the help of conditioning. As outlined in Section 3.2.1, we select optimal values of β by optimizing the expected sample quality of the generated samples. We define samples to be of high quality if they 1) can be correctly classified by a classifier trained on real data, and 2) if they are diverse enough. We leverage *conditional* models to generate samples of known digit identity. Thus, we train an SO(3)-equivariant classifier - with the same architecture as H-(V)AE's encoder - to classify the digit identity of real data at 97% accuracy, as well as *conditional* H-VAE models with varying β . We then sample the conditional models to construct synthetic datasets of 10,000 spherical images, balanced across digit identity. Finally, we compute 1) the accuracy of the classifier on the synthetic datasets, and 2) the empirical variance of generated samples. We find that there is a trade-off between the two metrics: if β is too low, the generate samples are very diverse but do not resemble real digits, whereas if β is too high, samples of the same digit start looking more and more alike (Figure A.4.16). We empirically select $\beta = 0.6$ and $\beta = 2.8$ for the $z = 16$ and $z = 120$ models, respectively, based on the trade-off between the above two metrics. We note that this procedure relies on the assumption that the same regularization strength β has the same effect on conditional vs. un-conditional models on sample quality. We believe this to be likely for the following three reasons: 1) the reconstruction loss is similar for same values of β (Figure A.4.7); 2) the optimal values of β found using the conditional models approximately optimize the latent space metrics of un-conditional models (Figure A.4.6); 3) visual inspection of the conditional vs. unconditional samples across same values of β (Figures A.4.14 and A.4.11, A.4.15 and A.4.12, A.4.16 and A.4.13). We perform this procedure only on the NR/R dataset and later use the same selected β to train models on the R/R dataset.

A.3.3 Shrec17

Model architectures. Both AE and VAE models have $z = 40$, 7 blocks, DegreesList = [14, 14, 14, 8, 4, 2, 1], MultList = [12, 12, 12, 20, 24, 32, 40], $C_{\text{init}} = 12$, with a total of 518k parameters.

Training details. We keep the learning schedule as similar as possible for all models. We use $\alpha = 1000$. We train all models for 120 epochs using the Adam optimizer with default parameters, a batch size of 100, and an initial learning rate of 0.0025, which we decrease exponentially by two orders of magnitude over the entire 120 epochs. For VAE models, we use $\beta = 0.2$, $E_{\text{rec}} = 25$ and $E_{\text{warmup}} = 10$. We utilize the model with the lowest loss on validation data, only after the end of the warmup epochs for VAE models. Training took ~ 11 hours on a single NVIDIA A40 GPU for each model.

A.3.4 Embedding of amino acid conformations

Pre-processing of protein structures. We sample residues from the set of training structures pre-processed as described in Sec. A.3.5.

Fourier projection. We set the maximum radial frequency to $N = 20$ as it corresponds to a radial resolution matching the minimum inter-atomic distances after rescaling the atomic neighborhoods of radius 10.0 to fit within a sphere of radius 1.0, necessary for Zernike transform.

The multiplicity composition of the data tensors can be described in a notation - analogous to that used by e3nn but without parity specifications - which specifies the multiplicity C_ℓ for each feature of degree ℓ in single units $C_\ell \times \ell$: $44 \times 0 + 40 \times 1 + 40 \times 2 + 36 \times 3 + 36 \times 4$.

Model architectures. All models have $z = 2$, 6 blocks, DegreesList = [4, 4, 4, 4, 2, 1], MultList = [60, 40, 24, 16, 16, 8], $C_{\text{init}} = 48$, with a total of 495k parameters. We note that the initial projection is necessary since the multiplicity differs across feature degrees in the data tensors.

Training details. We keep the learning schedule as similar as possible for all models. We use $\alpha = 400$. We train all models for 80 epochs using the Adam optimizer with default parameters and an initial learning rate of 0.005, which we decrease exponentially by one order of magnitude over 25 epochs. For VAE models, we use $E_{\text{rec}} = 25$ and $E_{\text{warmup}} = 10$. We utilize the model with the lowest loss on validation data, only after the end of the warmup epochs for VAE models.

We vary the batch size according to the size of the training and the validation datasets. We use the following (dataset_size-batch_size) pairs: (400-4), (1,000-10), (2,000-20), (5,000-50), (20,000-20). Training took ~ 45 minutes on a single NVIDIA A40 GPU for each model.

Evaluation. We perform our data ablations by considering training and validation datasets of the following sizes: 400, 1,000, 2,000, 5,000 and 20,000. We keep relative proportions of residue types even in all datasets. We perform the data ablation with H-AE as well as H-VAE models with $\beta = 0.025$ and 0.1.

We further perform a β ablation using the full (20,000) dataset, over the following choices of β : [0(AE), 0.025, 0.05, 0.1, 0.25, 0.5].

For robust results, we train 3 versions of each model and compute averages of quantitative metrics of reconstruction loss and classification accuracy.

For a fair comparison across models with varying amounts of training and validation data, we perform a 5-fold cross-validation-like procedure over the 10k test residues, where the classifier is trained over 4 folds of the test data and evaluated on the fifth one. If validation data is needed for model selection (e.g. for LC), we use 10% of the training data.

A.3.5 Embedding of protein structure neighborhoods

Pre-processing of protein structures. We model protein neighborhoods extracted from tertiary protein structures from the Protein Data Bank (PDB) [18]. We use ProteinNet’s splittings for training and validation sets to avoid redundancy, e.g. due to similarities in homologous protein domains [8]. Since PDB ids were only provided for the training and validation sets, we used ProteinNet’s training set as both our training and validation set and ProteinNet’s validation set as our testing set. Specifically, we make a [80%, 20%] split in the ProteinNet’s training data to define our training and validation sets. This splitting resulted in 10,957 training structures, 2,730 validation structures, and 212 testing structures.

Projection details. We set the maximum radial frequency to $N = 20$ as it corresponds to a radial resolution matching the minimum inter-atomic distances after rescaling the atomic neighborhoods of radius 10.0 Å to fit within a sphere of radius 1.0, necessary for Zernike transform. We vary L and construct models tailored to each one (Table A.3.1).

Model architectures. See Table A.3.1. We note that the initial projection (C_{init}) is necessary since the multiplicity differs across feature degrees in the data tensors, and the ETP necessitates equal multiplicity for all degrees.

Training details. We keep the learning schedule the same for all models. We use $\alpha = 1000$. We train all models for 8 epochs using the Adam optimizer with default parameters, a batch size of 512, and a constant learning rate of 0.001. We utilize the model with the lowest loss on validation data.

z	L	Tensor size	C_{init}	MultList	DegreesList	# Params	Training Speed (hours)
64	3	616	44	[50,50,50,64,64,64]	[3,3,3,3,2,1]	631k	3.0
64	4	940	44	[50,50,50,64,64,64]	[4,4,4,4,2,1]	950k	3.8
64	6	1708	44	[50,50,50,64,64,64]	[6,6,6,4,2,1]	1.6M	5.9
64	8	2604	44	[50,50,50,64,64,64]	[8,8,8,4,2,1]	2.4M	9.3
128	3	616	44	[60,60,60,90,128,128]	[3,3,3,3,2,1]	1.2M	3.8
128	4	940	44	[60,60,60,90,128,128]	[4,4,4,4,2,1]	1.7M	4.4
128	6	1708	44	[60,60,60,90,128,128]	[6,6,6,4,2,1]	2.6M	6.0
128	8	2604	44	[60,60,60,90,128,128]	[8,8,8,4,2,1]	3.8M	10.1
256	3	616	44	[90,90,90,120,200,256]	[3,3,3,3,2,1]	2.8M	4.9
256	4	940	44	[90,90,90,120,200,256]	[4,4,4,4,2,1]	3.8M	5.6
256	6	1708	44	[90,90,90,120,200,256]	[6,6,6,4,2,1]	4.4M	6.0
256	8	2604	44	[90,90,90,120,200,256]	[8,8,8,4,2,1]	7.9M	14.5

Table A.3.1: Model architecture hyperparameters and training time for each H-AE trained on protein neighborhoods.

A.3.6 Latent space classification

Linear classifier. We implement the linear classifier as a one-layer fully connected neural network with input size equal to the invariant embedding of size z , and output size equal to the number of desired classes. We use cross entropy loss with logits as training objective, which we minimize for 250 epochs using the Adam optimizer with batch size 100, and initial learning rate of 0.01. We reduce the learning rate by one order of magnitude every time the loss on validation data stops improving for 10 epochs (if validation data is not

provided, the training data is used). At evaluation time, we select the class with the highest probability value. We use PyTorch for our implementation.

KNN Classifier We use the `sklearn` [174] implementation with default parameters. At evaluation time, we select the class with the highest probability value.

A.3.7 Clustering metrics

Purity [5]. We first assign a class to each cluster based on the most prevalent class in it. Purity is computed as the sum of correctly classified items divided by the total number of items. Purity measures classification accuracy, and ranges between 0 (worst) and 1 (best).

V-measure [199]. This common clustering metric strikes a balance between favoring homogeneous (high homogeneity score) and complete (high completeness score) clusters. Clusters are defined as homogeneous when all elements in the same cluster belong to the same class (akin to a precision). Clusters are defined as complete when all elements belonging to the same class are put in the same cluster (akin to a recall). The V-measure is computed as the harmonic mean of homogeneity and completeness in a given clustering.

A.3.8 On the complementary nature of classification accuracy and clustering metrics

The clustering metrics “purity” and “V-measure” and the supervised metric “classification accuracy” characterize different qualities of the latent space, and, while partly correlated, they are complementary to each other.

Both classes of metrics are computed by comparing the ground truth labels to the predicted labels, and they mainly differ by how the predicted labels are assigned; the clustering metrics use an unsupervised clustering algorithm, while the classification metric uses a supervised classification algorithm to do so. As a result, these metrics focus on different features of the latent space. For example, the clustering metrics are largest when the test data naturally forms clusters with all data points of the same label. While this case can result in a high supervised classification accuracy, clustering is not a necessary condition for high classification accuracy. Indeed, the supervised signal could make the predicted labels depend more heavily on a subset of the latent space features, instead of relying on all of them equally, which is what the clustering algorithm naturally does. Therefore, it is reasonable to conclude that having higher clustering metrics and a lower classification accuracy is a sign that class-related information is more evenly distributed across the latent space dimensions. Overall, the complementary aspect of these metrics makes it necessary to use all of them when comparing the performance of different models in each task.

A.4 Protein-Ligand Binding Affinity Prediction

A.4.1 Data Preprocessing Details

We leverage the H-AE models trained on Protein Neighborhoods (Sec. A.3.5) to predict the binding affinity between a protein and a ligand, given their structure complex— an important task in structural biology. For this task, we use the data from the “refined-set” of PDBBind [224], containing $\sim 5,000$ structures. We use the dataset splits provided by ATOM3D [231] to benchmark our predictions. ATOM3D provides two splits based on the maximum sequence similarity between proteins in the training set and the validation / test sets. We use the most challenging split for our benchmarking in which the sequences of the two sets have at most 30% similarity.

For each complex, we first identify residues in the binding pocket, which we define as residues for which the $C\alpha$ is within $10\text{\AA}$ of any of the ligand’s atoms. We then extract the $10\text{\AA}$ neighborhood for each of the pocket

residues that can contain atoms from both the protein and the ligand. We compute the ZFT (Eq. 2.8) for each neighborhood with maximum degree L matching the degree used by the H-AE of interest. We then compute residue-level $SO(3)$ -invariant embeddings by running the pre-trained H-AE encoder on the neighborhoods surrounding each of the residues in the pocket. We sum over the residue-level embeddings to compute a single pocket-level embedding, which is $SE(3)$ -invariant by construction. Each pocket embedding is used as a feature vector within a standard Machine Learning Regression model to predict protein-ligand binding affinity, provided by PDBBind either in the form of a *dissociation constant* K_d , or an *inhibition constant* K_i . Due to data limitations, and consistent with other studies on LBA, we do not distinguish between K_d and K_i and regress over the negative log of either of the constants that is provided by PDBBind; the transformed quantity is closely related to the binding free energy of protein-ligand interactions.

The protein-ligand binding free energy is an extensive quantity, meaning that its magnitude depends on the number of residues in the binding pocket. In other words, a larger protein-ligand complex can establish a stronger binding. The pocket embedding, which we define as the sum of the residue-level embeddings, is a simple quantity that is extensive and $SE(3)$ -invariant, and therefore, suitable for regressing over protein-ligand binding free energy. Nonetheless, this map is not unique and other extensive transformations to pool together residue-level embeddings into a pocket-level embedding can be used for this purpose.

It should be noted that in the protein-ligand structural neighborhoods we only include the ligand atoms that are found in proteins and were used in the training of the H-AE models (i.e., C, N, O, and S). About 31% of the ligands contain other kinds of atoms, and since our model does not “see” these atoms, we hypothesize that our predictions are worse in these cases. In fact, for H-AE+R.F. (Table 3.2), the Spearman’s correlation over the ligands containing only protein atoms is higher than for the ligands containing other kinds of atoms by about 0.03 points. Therefore, we expect training H-(V)AE to recognize more atom types - or at least making it aware that some other unspecified non-protein atom is present - would yield even better results; as protein structures are often found in complex with other non-protein entities, e.g. ligands and ions, there is training data available for constructing such models.

A.4.2 Machine Learning models used for prediction

Linear Regression. We implement a linear regression model as a one-layer fully connected neural network with one output layer. We use MSE loss as training objective, which we minimize for 250 epochs using the Adam optimizer with batch size 32, and initial learning rate of 0.01. We reduce the learning rate by one order of magnitude every time the loss on validation data stops improving for 10 epochs. We use PyTorch for our implementation.

Random Forest Regression. We use the `sklearn` [174] implementation. We tune hyperparameters via grid search, choosing the combination minimizing RMSD on validation data. We consider the following grid of hyperparameter values:

- `max_features`: [1.0, 0.333, sqrt]
- `min_samples_leaf`: [2, 5, 10, 15]
- `n_estimators`: [32, 64, 100]

A.4.3 Extended Discussion of Results

Table A.4.8 shows benchmarking results on the split of ATOM3D with 30% sequence similarity, extending Table 3.2 in the main text with more baselines. Notably, we show the prediction results for the models that

only use the $\text{SO}(3)$ -invariant ($\ell = 0$) component of each neighborhood’s Zernike transform (our Zernike inv. baseline), instead of the complete embeddings learned by H-AE. Using H-AE consistently outperforms this baseline, implying that invariant information encoded in higher spherical degrees, and extracted by H-AE, can lead to more expressive models for downstream regression tasks.

Type	Method	z	bw	LC Acc.	KNN Acc.
Unsupervised	H-AE NR/R	120	30	0.920	0.920
	H-AE R/R	120	30	0.916	0.923
	H-AE NR/R	16	30	0.850	0.902
	H-AE R/R	16	30	0.844	0.901
	H-VAE NR/R	120	30	0.923	0.907
	H-VAE R/R	120	30	0.916	0.900
	H-VAE NR/R	16	30	0.878	0.897
	H-VAE R/R	16	30	0.855	0.880

Table A.4.1: **Evaluation of network performances for MNIST-on-the-sphere using a K-Nearest Neighbors (KNN) classifier instead of linear classifier in the latent space.** Results are significantly better than when using a linear classifier for models with smaller ($z = 16$) latent space, comparable or worse for the other models. Interestingly, VAE models have worse KNN classification performance than AE models, reversing the trend seen with of Linear Classification.

Type	Method	z	bw	Class. Acc.	P@N	R@N	F1@N	mAP	NDCG
Unsupervised + LC	H-AE	40	90	0.654	0.548	0.569	0.545	0.500	0.597
	H-VAE	40	90	0.631	0.512	0.537	0.512	0.463	0.568
Unsupervised + KNN	H-AE	40	90	0.672	0.560	0.572	0.555	0.501	0.599
	H-VAE	40	90	0.658	0.541	0.558	0.539	0.487	0.591

Table A.4.2: **Evaluation of network performances for Shrec17 using a KNN classifier instead of linear classifier in the latent space.** Results are better than when using a linear classifier.

# train	H-AE				H-VAE ($\beta = 0.025$)				H-VAE ($\beta = 0.1$)			
	MSE	Cosine loss	LC Acc.	KNN Acc.	MSE	Cosine loss	LC Acc.	KNN Acc.	MSE	Cosine loss	LC Acc.	KNN Acc.
0	1.3×10^{-2}	1.015	0.409	0.629	1.5×10^{-2}	0.981	0.424	0.656	1.5×10^{-2}	0.981	0.424	0.656
400	9.4×10^{-4}	0.153	0.586	0.842	9.8×10^{-4}	0.160	0.616	0.848	1.0×10^{-3}	0.163	0.558	0.780
1,000	5.9×10^{-4}	0.099	0.583	0.856	6.3×10^{-4}	0.101	0.569	0.854	6.9×10^{-4}	0.113	0.564	0.844
2,000	4.5×10^{-4}	0.073	0.560	0.900	4.9×10^{-4}	0.081	0.554	0.905	5.5×10^{-4}	0.092	0.593	0.890
5,000	3.3×10^{-4}	0.053	0.629	0.940	3.3×10^{-4}	0.053	0.638	0.961	4.3×10^{-4}	0.072	0.588	0.921
20,000	2.2×10^{-4}	0.034	0.578	0.972	2.4×10^{-4}	0.037	0.667	0.971	2.9×10^{-4}	0.047	0.662	0.966

Table A.4.3: **Quantitative data ablation results on the Toy amino acids dataset.** A random-guessing classifier has an expected accuracy of 0.050.

β	MSE	Cosine	LC Acc.	KNN Acc.
0 (AE)	2.2×10^{-4}	0.034	0.580	0.972
0.025	2.4×10^{-4}	0.037	0.666	0.971
0.05	2.5×10^{-4}	0.039	0.669	0.968
0.1	2.9×10^{-4}	0.047	0.661	0.966
0.25	6.8×10^{-4}	0.132	0.597	0.854
0.5	1.1×10^{-3}	0.203	0.467	0.722

Table A.4.4: **Quantitative AE vs. VAE ablation results on the Toy amino acids dataset.** Models were trained on the full dataset (# train = 20,000).

Dataset	Method	L	z	bw	# Params	MSE	Cosine
MNIST	H-AE NR/R	10	120	30	453k	6.5×10^{-4}	0.017
	H-AE R/R	10	120	30	453k	5.8×10^{-4}	0.015
	H-AE NR/R	10	16	30	227k	1.2×10^{-3}	0.031
	H-AE R/R	10	16	30	227k	1.2×10^{-3}	0.030
	H-VAE NR/R	10	120	30	453k	1.9×10^{-3}	0.048
	H-VAE R/R	10	120	30	453k	2.0×10^{-3}	0.050
	H-VAE NR/R	10	16	30	227k	2.8×10^{-3}	0.068
	H-VAE R/R	10	16	30	227k	2.7×10^{-3}	0.067
Shrec17	H-AE	14	40	90	518k	1.8×10^{-4}	0.130
	H-VAE	14	40	90	518k	2.2×10^{-4}	0.151
Protein NBs	H-AE	3	64	-	631k	3.3×10^{-4}	0.084
	H-AE	4	64	-	950k	3.9×10^{-4}	0.125
	H-AE	6	64	-	1.6M	5.4×10^{-4}	0.219
	H-AE	8	64	-	2.4M	6.3×10^{-4}	0.287
	H-AE	3	128	-	1.2M	2.1×10^{-4}	0.053
	H-AE	4	128	-	1.7M	2.8×10^{-4}	0.087
	H-AE	6	128	-	2.6M	4.0×10^{-4}	0.156
	H-AE	8	128	-	3.8M	4.9×10^{-4}	0.213
	H-AE	3	256	-	2.8M	1.0×10^{-4}	0.025
	H-AE	4	256	-	3.8M	1.9×10^{-4}	0.060
	H-AE	6	256	-	4.4M	2.7×10^{-4}	0.102
	H-AE	8	256	-	7.9M	3.5×10^{-4}	0.148

Table A.4.5: **Test MSE and Cosine loss for H-(V)AE models trained on MNIST, Shrec17 and Protein Neighborhoods.** MSE and Cosine values are strongly correlated within datasets but not across datasets.

Dataset	Method	L	z	TP-type	C_{init}	ChannelsList	DegreesList	# Params	Speed	MSE	Cosine
MNIST	H-AE	10	16	ETP	None	[16,16,16,16,16,16]	[10,10,8,4,2,1]	227k	1.0	1.2×10^{-3}	0.031
	H-AE	10	16	Full-TP	None	[7,6,6,6,16]	[10,8,4,2,1]	229k	1.1	9.8×10^{-4}	0.025
	H-AE	10	16	Full-TP	None	[5,5,5,5,7,16]	[10,10,8,4,2,1]	227k	1.7	8.9×10^{-4}	0.024
Shrec17	H-AE	14	40	ETP	12	[12,12,12,20,24,32,40]	[14,14,14,8,4,2,1]	518k	1.0	1.8×10^{-4}	0.130
	H-AE	14	40	Full-TP	None	[5,5,5,8,40]	[14,8,4,2,1]	518k	0.9	1.9×10^{-4}	0.137
	H-AE	14	40	Full-TP	None	[4,3,3,6,6,6,40]	[14,14,14,8,4,2,1]	513k	1.9	2.0×10^{-4}	0.142
Protein NBs	H-AE	6	64	ETP	44	[50,50,50,64,64,64]	[6,6,6,4,2,1]	1.6M	1.0	5.4×10^{-4}	0.219
	H-AE	6	64	Full-TP	None	[8,8,12,36]	[6,4,2,1]	1.6M	2.1	6.7×10^{-4}	0.286
	H-AE	8	64	ETP	44	[50,50,50,64,64,64]	[6,6,6,4,2,1]	2.4M	1.0	6.3×10^{-4}	0.287
	H-AE	8	64	Full-TP	None	[8,8,12,36]	[6,4,2,1]	2.6M	3.3	7.6×10^{-4}	0.371

Table A.4.6: **Training speed and reconstruction ablations of H-(V)AE models with different Tensor Product rules.** To make comparison fair, models were trained using the same training hyperparameters as described in A.3, and all models were constructed to have comparable number of parameters. Speed was computed as training time and divided by the time of the model using ETP within each dataset. Models with the ETP usually generate better reconstructions and are usually the fastest. The speed and performance gains of the ETP are most apparent on the Protein Neighborhoods task, where we also note that, as the angular resolution of the data (L) is increased from 6 to 8, the relative speed gain of the ETP over the Full-TP is significantly accentuated (from 2.1x to 3.3x).

Dataset	Method	L	z	# train	Equiv. Error	Abs. Value
MNIST	H-AE NR/R	10	16	-	$(2.3 \pm 2.7) \times 10^{-4}$	$(1.0 \pm 0.9) \times 10^{-1}$
	H-VAE NR/R	10	16	-	$(2.5 \pm 1.9) \times 10^{-4}$	$(1.3 \pm 1.8) \times 10^{-1}$
Shrec17	H-AE	14	40	-	$(1.4 \pm 2.4) \times 10^{-5}$	$(0.4 \pm 1.1) \times 10^{-2}$
	H-VAE	14	40	-	$(1.4 \pm 3.5) \times 10^{-5}$	$(0.4 \pm 2.2) \times 10^{-2}$
Toy Aminoacids	H-AE	4	2	1,000	$(4.7 \pm 3.7) \times 10^{-5}$	$(2.6 \pm 4.1) \times 10^{-2}$
	H-VAE $\beta = 0.025$	4	2	1,000	$(5.4 \pm 3.9) \times 10^{-5}$	$(2.8 \pm 4.0) \times 10^{-2}$
	H-AE	4	2	20,000	$(9.3 \pm 6.4) \times 10^{-5}$	$(3.5 \pm 4.7) \times 10^{-2}$
	H-VAE $\beta = 0.025$	4	2	20,000	$(9.0 \pm 4.7) \times 10^{-5}$	$(3.0 \pm 4.3) \times 10^{-2}$
Protein NBs	H-AE	3	64	-	$(3.3 \pm 4.2) \times 10^{-5}$	$(1.6 \pm 2.1) \times 10^{-2}$
	H-AE	4	64	-	$(0.6 \pm 0.6) \times 10^{-5}$	$(0.6 \pm 2.1) \times 10^{-2}$
	H-AE	6	64	-	$(0.9 \pm 1.0) \times 10^{-5}$	$(0.4 \pm 1.3) \times 10^{-2}$
	H-AE	8	64	-	$(0.7 \pm 1.0) \times 10^{-5}$	$(0.2 \pm 0.9) \times 10^{-2}$
	H-AE	3	128	-	$(1.1 \pm 1.3) \times 10^{-5}$	$(1.0 \pm 3.0) \times 10^{-2}$
	H-AE	4	128	-	$(0.6 \pm 0.7) \times 10^{-5}$	$(0.6 \pm 2.5) \times 10^{-2}$
	H-AE	6	128	-	$(0.1 \pm 0.1) \times 10^{-5}$	$(0.2 \pm 1.7) \times 10^{-2}$
	H-AE	8	128	-	$(0.1 \pm 0.1) \times 10^{-5}$	$(0.1 \pm 1.0) \times 10^{-2}$
	H-AE	3	256	-	$(0.6 \pm 0.4) \times 10^{-5}$	$(0.6 \pm 2.0) \times 10^{-2}$
	H-AE	4	256	-	$(0.4 \pm 0.6) \times 10^{-5}$	$(0.6 \pm 2.2) \times 10^{-2}$
	H-AE	6	256	-	$(0.1 \pm 0.1) \times 10^{-5}$	$(0.2 \pm 1.6) \times 10^{-2}$
	H-AE	8	256	-	$(0.09 \pm 0.07) \times 10^{-5}$	$(0.1 \pm 1.3) \times 10^{-2}$

Table A.4.7: **Mean equivariance error for some of our trained H-(V)AE models.** Errors were computed over 2,000 randomly sampled spherical tensors, each with a randomly sampled rotation. Standard deviation is shown alongside the mean. We also show the mean and standard deviation of the absolute value of the output coefficients, to enable contextualization of the measured equivariance error. The equivariance error due to numerical error (absolute difference in coefficients by rotating input vs. output tensor) is consistently three orders of magnitude lower than the typical absolute value of the coefficients, indicating that equivariance is preserved. The same trend occurs for *untrained* models (not shown here for simplicity).

Method	Ligand Binding Affinity 30% Similarity			
	RMSD ↓	Pearson's r ↑	Spearman's r ↑	Kendall's τ ↑
DeepDTA [169]	1.565	0.573	0.574	-
DeepAffinity [109]	1.893 ± 0.650	0.415	0.426	-
Cormorant [10]	1.568 ± 0.012	0.389	0.408	-
ProtTrans [65]	1.544 ± 0.015	0.438 ± 0.053	0.434 ± 0.058	-
3DCNN [231]	1.414 ± 0.021	0.550	0.553	-
GNN [231]	1.570 ± 0.025	0.545	0.533	-
MaSIF [73]	1.484 ± 0.018	0.467 ± 0.020	0.455 ± 0.014	-
DGAT [165]	1.719 ± 0.047	0.464	0.472	-
DGIN [165]	1.765 ± 0.076	0.426	0.432	-
DGAT-GCN [165]	1.550 ± 0.017	0.498	0.496	-
GVP-GNN [105]	1.648 ± 0.014	0.213 ± 0.013	0.164 ± 0.009	0.110 ± 0.012
EGNN [203]	1.492 ± 0.012	0.489 ± 0.017	0.472 ± 0.008	0.329 ± 0.014
HoloProt [218]	1.464 ± 0.006	0.509 ± 0.002	0.500 ± 0.005	-
GBPNNet [13]	1.405 ± 0.009	0.561	0.557	-
EGNN + PLM [257]	1.403 ± 0.013	0.565 ± 0.016	0.544 ± 0.005	0.379 ± 0.007
ProtMD [256]	1.367 ± 0.014	<u>0.601 ± 0.036</u>	<u>0.587 ± 0.042</u>	-
Zernike Inv. + Linear Regression	1.455 ± 0.005	0.513 ± 0.005	0.516 ± 0.006	0.357 ± 0.005
Zernike Inv. + Random Forest	<u>1.361 ± 0.011</u>	0.587 ± 0.009	0.584 ± 0.010	<u>0.408 ± 0.008</u>
H-AE + Linear Regression	1.397 ± 0.019	0.560 ± 0.017	0.568 ± 0.018	0.397 ± 0.016
H-AE + Random Forest	1.332 ± 0.012	0.612 ± 0.009	0.619 ± 0.009	0.436 ± 0.006

Table A.4.8: **Comprehensive benchmarking results on the Ligand Binding Affinity task with Atom3D's 30% similarity split.** Models are sorted by date of release. In addition to the H-AE informed models, we also report the performance of baseline models that only use the SO(3)-invariant ($\ell = 0$) component of each neighborhood's Zernike transform (Zernike Inv.). H-AE consistently outperforms this baseline, indicating that the SO(3)-invariant information from the higher-degree features extracted by H-AE are informative for this regression task. Best scores are in **bold** and second-best scores are underlined. Errors for our models are computed as the standard deviation in prediction by 10 Machine Learning models trained with bootstrapped data.

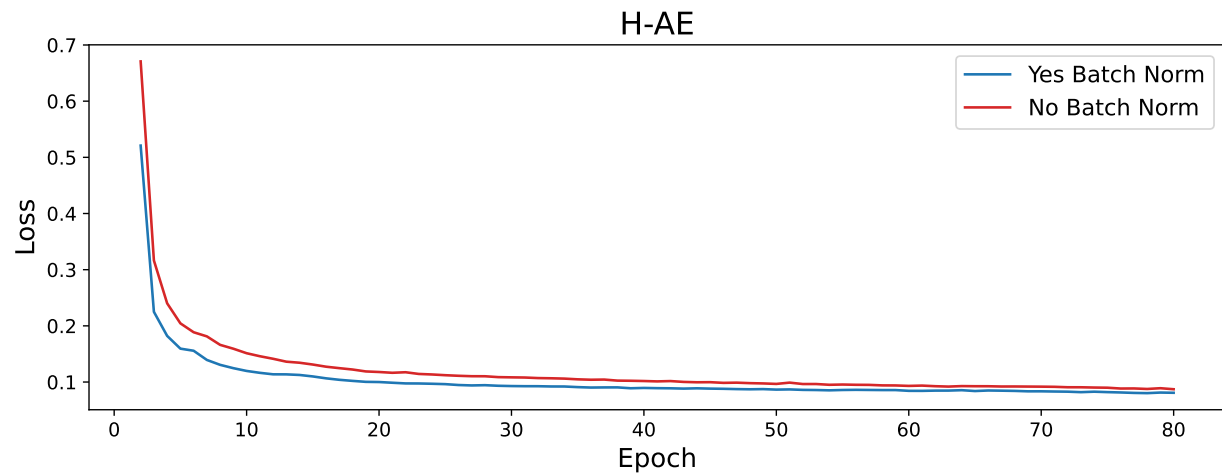


Figure A.4.1: **Training loss trace of H-AE, with and without Batch Norm, on MNIST-on-the-sphere.** Models were trained with the (NR/R; $z = 16$; AE) specification. The loss on validation data follows the same trend, but it is not shown for simplicity.

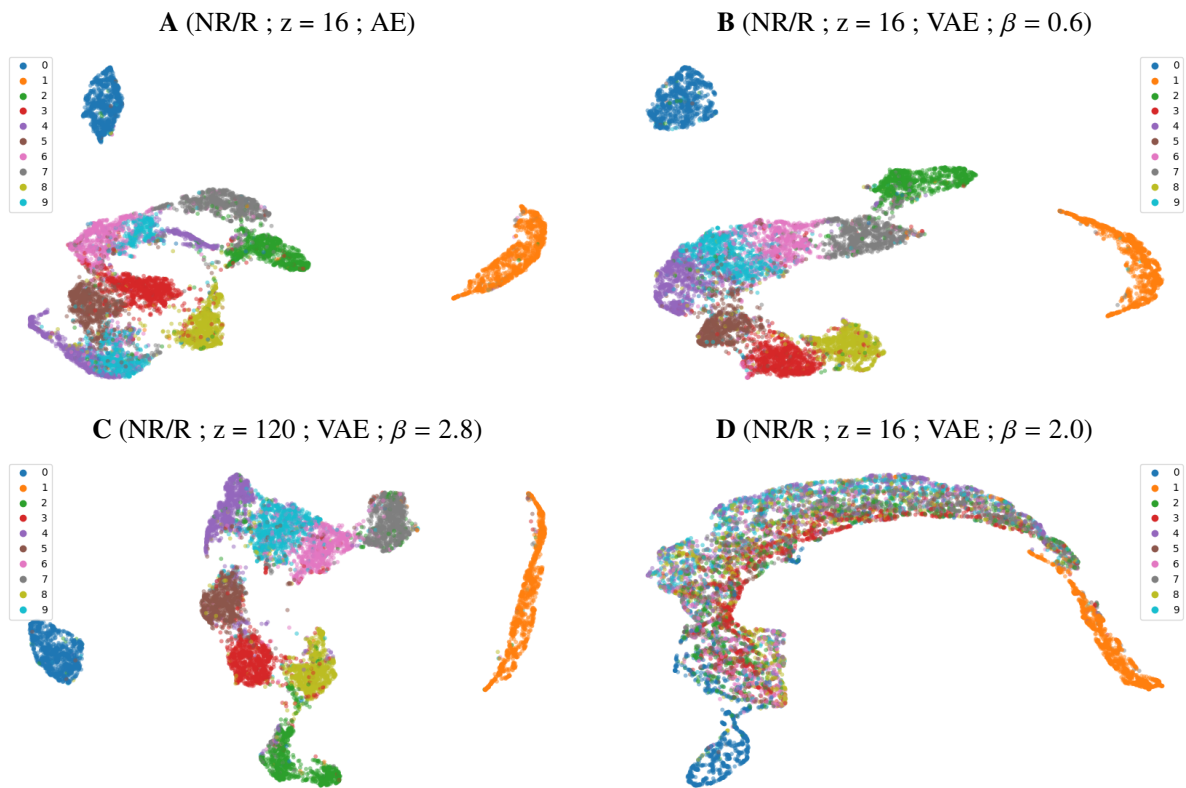


Figure A.4.2: **UMAP visualization of the invariant latent space learned by selected H-(V)AE models trained on MNIST-on-the-sphere.** Notably, variational models with appropriately-tuned strength of the prior (β) present the most cohesive clusters based on digit identity (panels B and C). If the prior is too strong (panel D), the latent space loses most of the digit-based clustering.

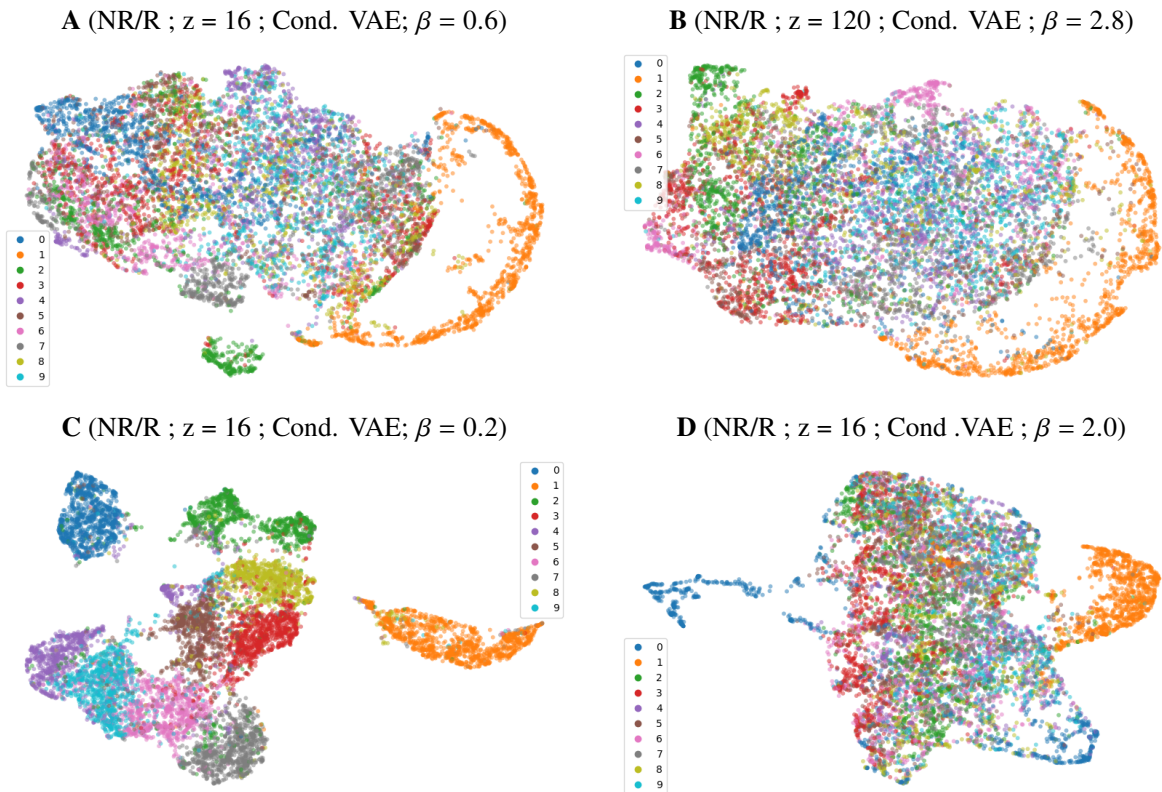


Figure A.4.3: **UMAP visualization of the invariant latent space learned by selected *conditional* H-VAE models trained on MNIST-on-the-sphere.** Contrary to using standard models, with a conditional VAE we expect the latent space to not contain information about the conditioning (in this case, the digit identity). We find this is true generally for high-enough strength of the prior β (panels A, B and D) but not true if β is too low (panel C).

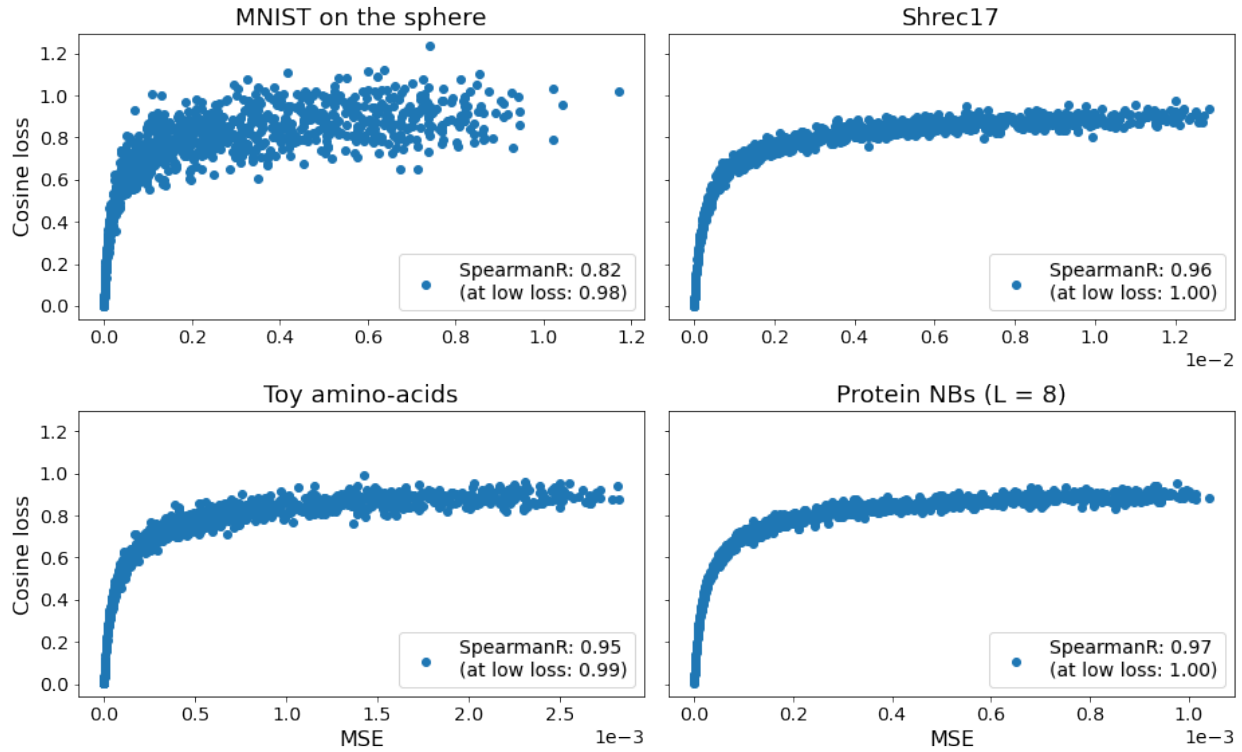


Figure A.4.4: Correlation between Cosine loss and MSE values between pairs of random tensors. For each dataset, we sample a batch of $N = 1000$ tensors with dataset-specific feature degrees and channel sizes, where each coefficient is sampled from a normal distribution. We mimic the normalization step performed in the real experiment and normalize each tensor by the average total norm of the batch. We then generate a “noisy” version of each tensor by adding (normalized) Gaussian noise to each coefficient with standard deviation sampled from a uniform distribution between 0 and some maximum noise level (10 in these plots). This procedure results in N pairs of tensors with varying degrees of similarity between them. We compute the MSE and Cosine loss for all N pairs of tensors and visualize them. The two loss values are well correlated in rank as measured by Spearman Correlation. The correlation is significantly stronger in the regime of reconstruction loss below a Cosine loss of 0.5 (SpearmanR ~ 0.99), a value well above the maximum Cosine loss achieved by H-(V)AE in all our experiments. All the p-values for the Spearman Correlations shown in the plot are significant (< 0.05).

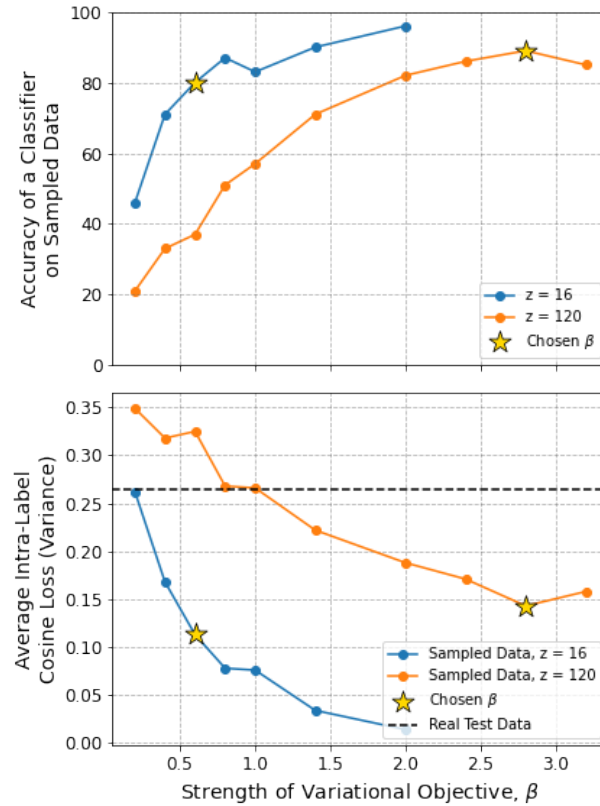


Figure A.4.5: Trade-off across regularization strength β between quality (top) and diversity (bottom) of MNIST-on-the-sphere samples generated by H-VAE with digit conditioning. We train conditional H-VAE models with varying values of regularization strength β , as well as an $SO(3)$ -equivariant classifier with high accuracy (97%) on real data. We generate 10,000 conditional samples evenly distributed across labels. We approximate the samples' quality by the classifier's accuracy (top) and their variance by the intra-label cosine loss across 100,000 samples pairs; in the bottom plot we also show the intra-label cosine loss on real data. The plots indicate a clear trade-off between sample quality and diversity. In particular, as β increases above a certain threshold (higher for a larger latent space), the samples start looking all the same (see Figure A.4.16 for examples). We chose values of β upon visual inspection and indicate them with a star. We further note that all our models tend to generate samples with lower diversity than real data, as indicated by the cosine loss being below the black dotted line in the bottom plot; this is to be expected since VAEs notoriously tend to "smooth-out" fine details, but the effect is less pronounced with a larger latent space.

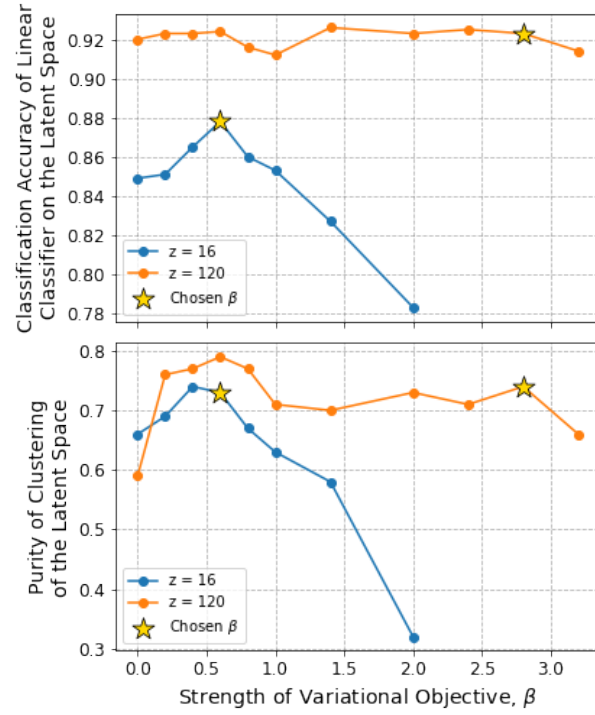


Figure A.4.6: **Ablation in regularization strength β on MNIST-on-the-sphere test set.** We indicate the chosen values β with stars. We emphasize that the values of β were chosen according to visual inspection in Figure A.4.5 using conditional models.

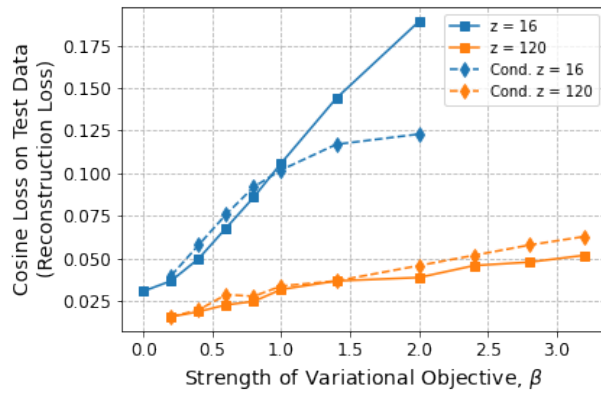


Figure A.4.7: **Reconstruction loss across β for both unconditional and conditional H-VAE models on MNIST-on-the-sphere.**

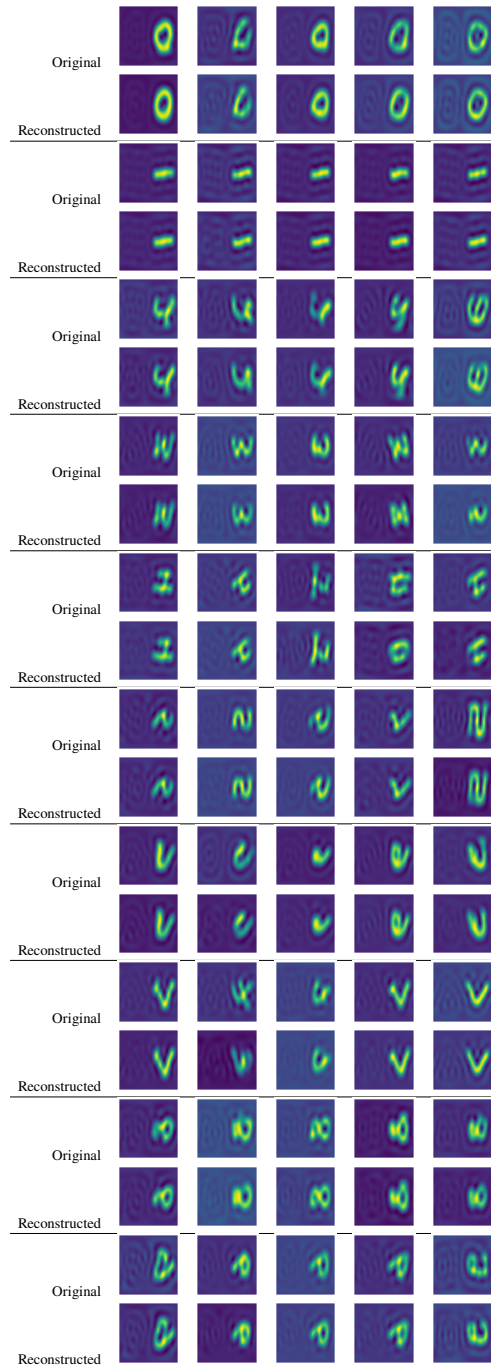


Figure A.4.8: **Reconstructions of Random Test Images by the (NR/R; $z = 16$; AE) MNIST-on-the-sphere model.**

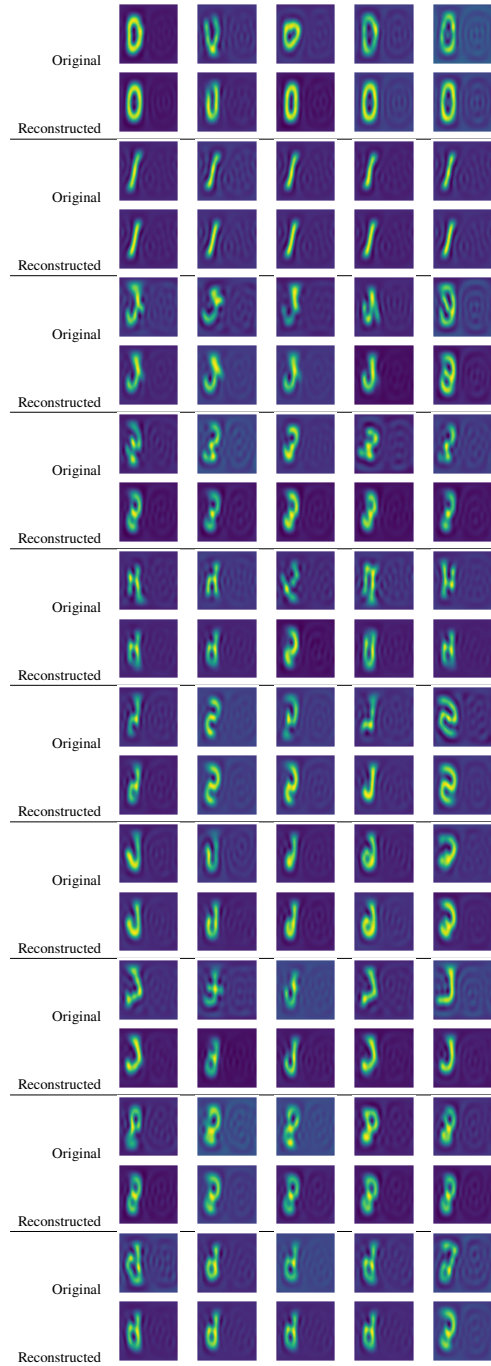


Figure A.4.9: **Reconstructions of Random Test Images by the (NR/R; $z = 16$; VAE; $\beta = 0.6$) MNIST-on-the-sphere model.**

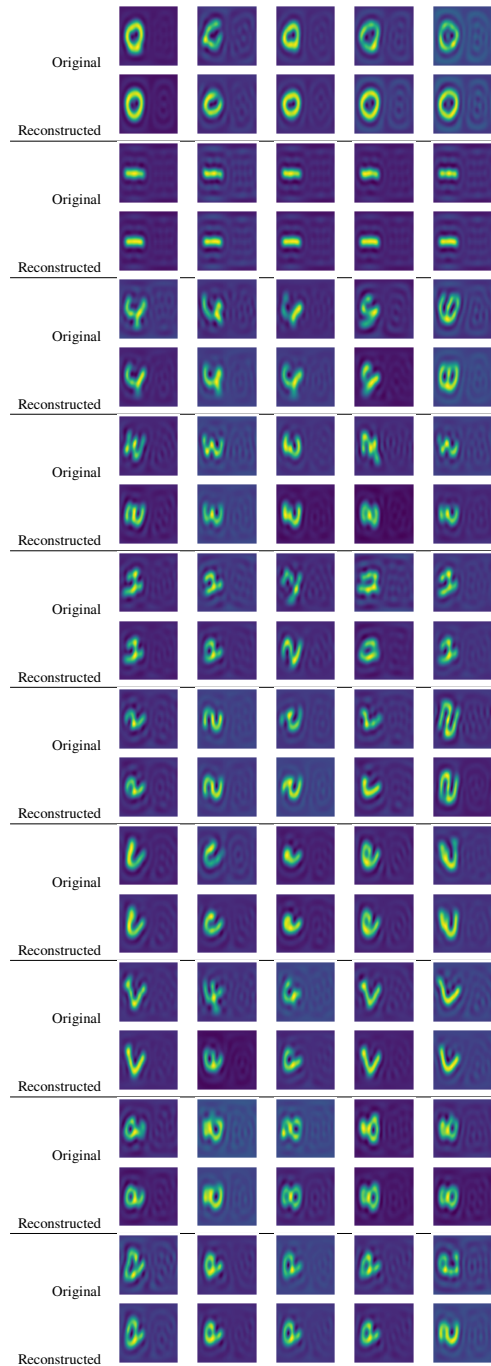


Figure A.4.10: **Reconstructions of Random Test Images by the (NR/R; $z = 120$; VAE; $\beta = 2.8$) MNIST-on-the-sphere model.**

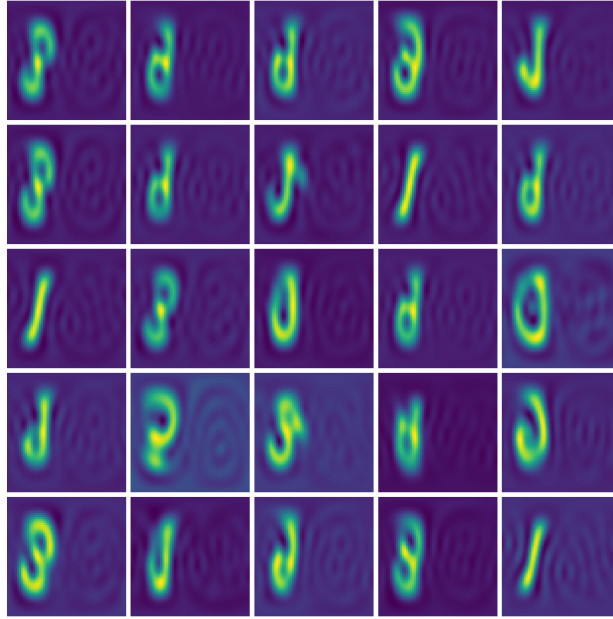


Figure A.4.11: **Random samples generated by the (NR/R; $z = 16$; VAE; $\beta = 0.6$) MNIST-on-the-sphere model.** We sample invariant latent embeddings from the prior distribution (isotropic normal) and feed them to the decoder alongside the canonical frame to generate tensors. We then compute the inverse SFT to map the generated tensor to images in real space. The samples show a wide range of diversity in digit and style.

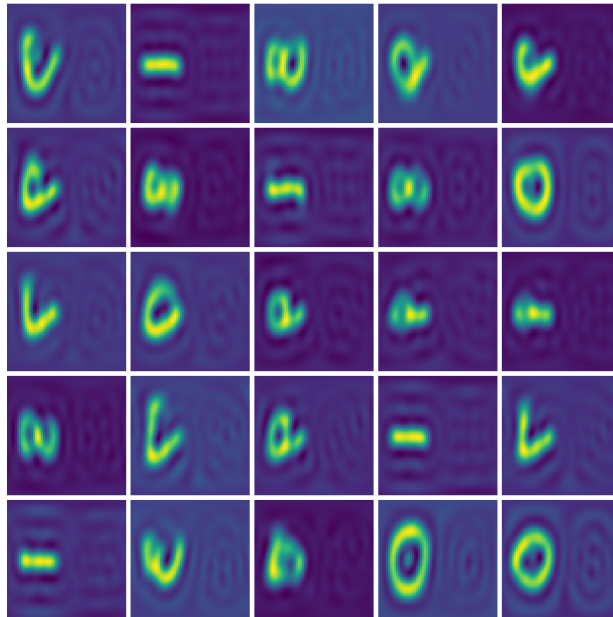


Figure A.4.12: **Random samples generated by the (NR/R; $z = 120$; VAE; $\beta = 2.8$) MNIST-on-the-sphere model.** We sample invariant latent embeddings from the prior distribution (isotropic normal) and feed them to the decoder alongside the canonical frame to generate tensors. We then compute the inverse SFT to map the generated tensor to images in real space. The samples show a wide range of diversity in digit and style.

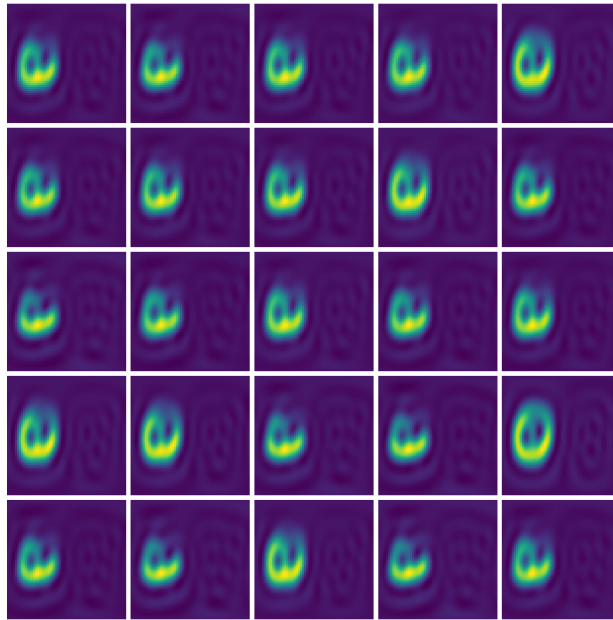


Figure A.4.13: **Random samples generated by the (NR/R; $z = 16$; VAE; $\beta = 2.0$) MNIST-on-the-sphere model.** We sample invariant latent embeddings from the prior distribution (isotropic normal) and feed them to the decoder alongside the canonical frame to generate tensors. We then compute the inverse SFT to map the generated tensor to images in real space. Samples look approximately all the same and like a "mush" of all digits, a clear sign that the strength of the prior β is too high.

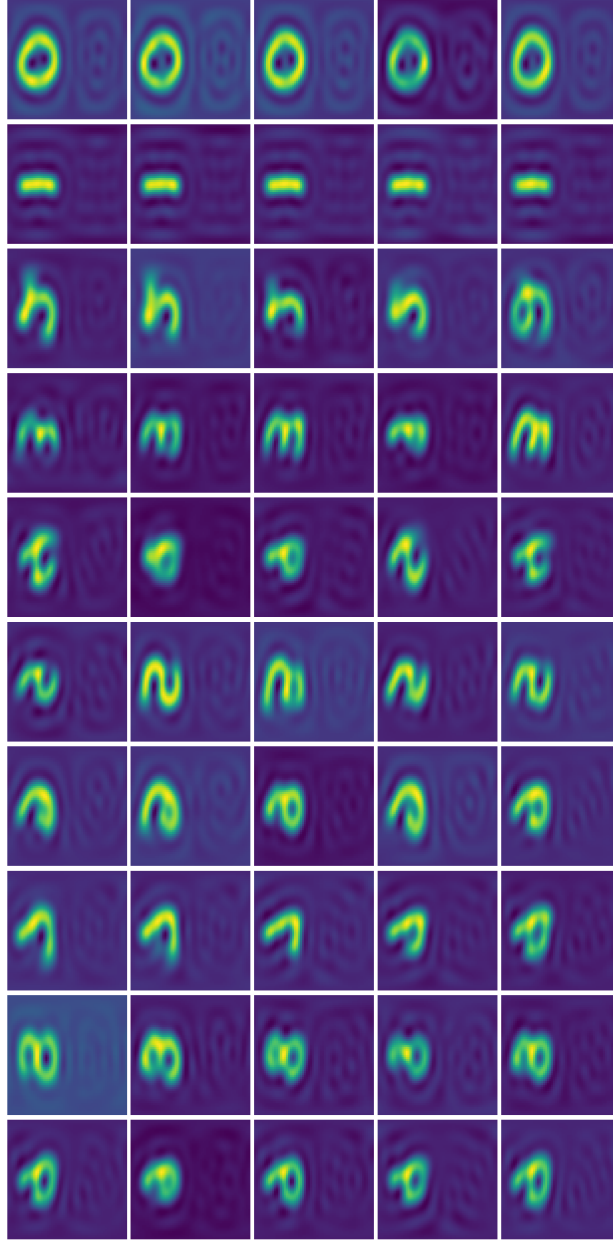


Figure A.4.14: **Random conditional samples generated by the (NR/R; $z = 16$; VAE; $\beta = 0.6$; conditional) MNIST-on-the-sphere model.** We sample invariant latent embeddings from the prior distribution (isotropic normal), add digit conditioning, and feed them to the decoder alongside the canonical frame to generate tensors. We then compute the inverse SFT to map the generated tensor to images in real space. Samples reflect their conditioning digit and show a wide variety of styles.

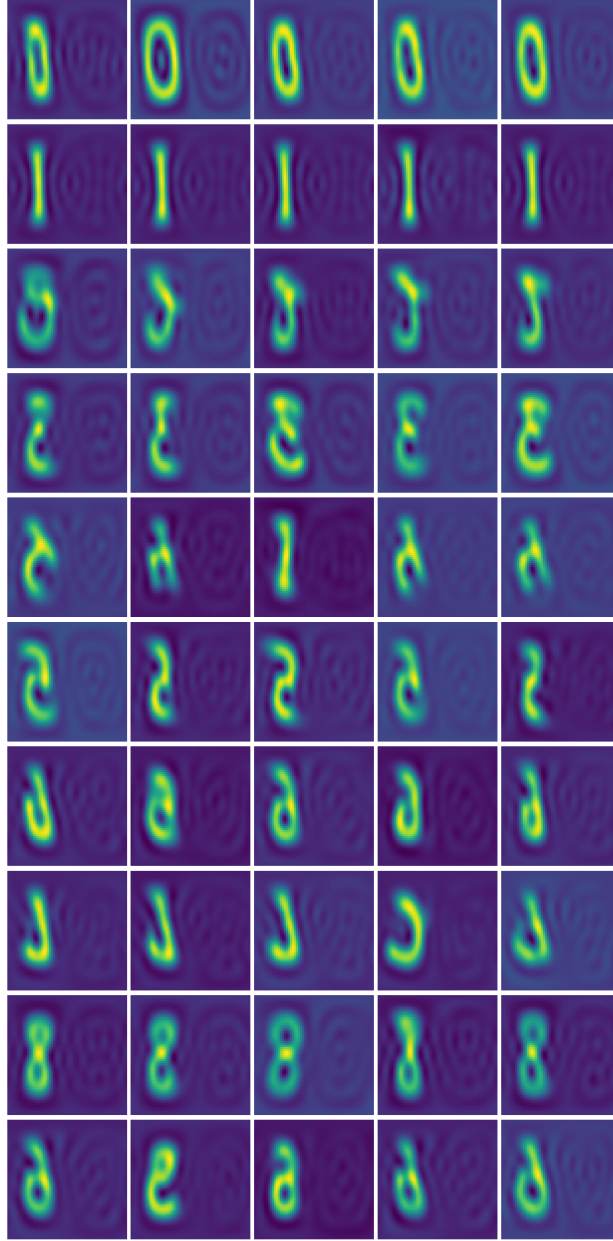


Figure A.4.15: **Random conditional samples generated by the (NR/R; $z = 120$; VAE; $\beta = 2.8$; conditional) MNIST-on-the-sphere model.** We sample invariant latent embeddings from the prior distribution (isotropic normal), add digit conditioning, and feed them to the decoder alongside the canonical frame to generate tensors. We then compute the inverse SFT to map the generated tensor to images in real space. Samples reflect their conditioning digit and show a wide variety of styles.

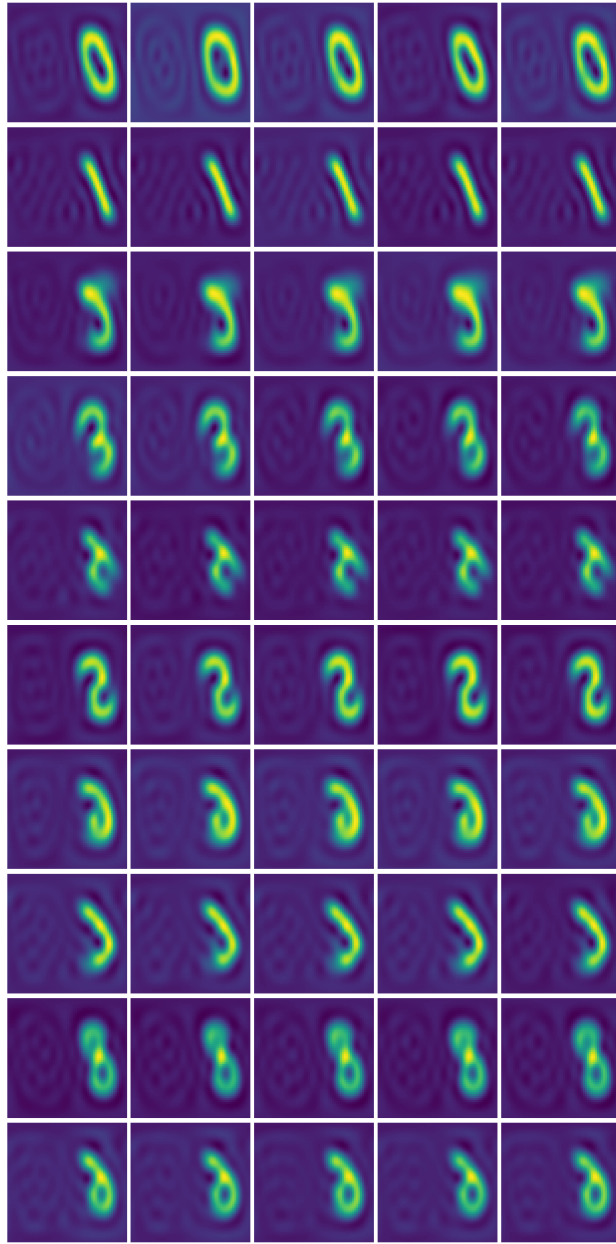


Figure A.4.16: **Random conditional samples generated by the (NR/R; $z = 16$; VAE; $\beta = 2.0$; conditional) MNIST-on-the-sphere model.** We sample invariant latent embeddings from the prior distribution (isotropic normal), add digit conditioning, and feed them to the decoder alongside the canonical frame to generate tensors. We then compute the inverse SFT to map the generated tensor to images in real space. Samples look the same within each digit, a clear sign that the strength of the prior β is too high.

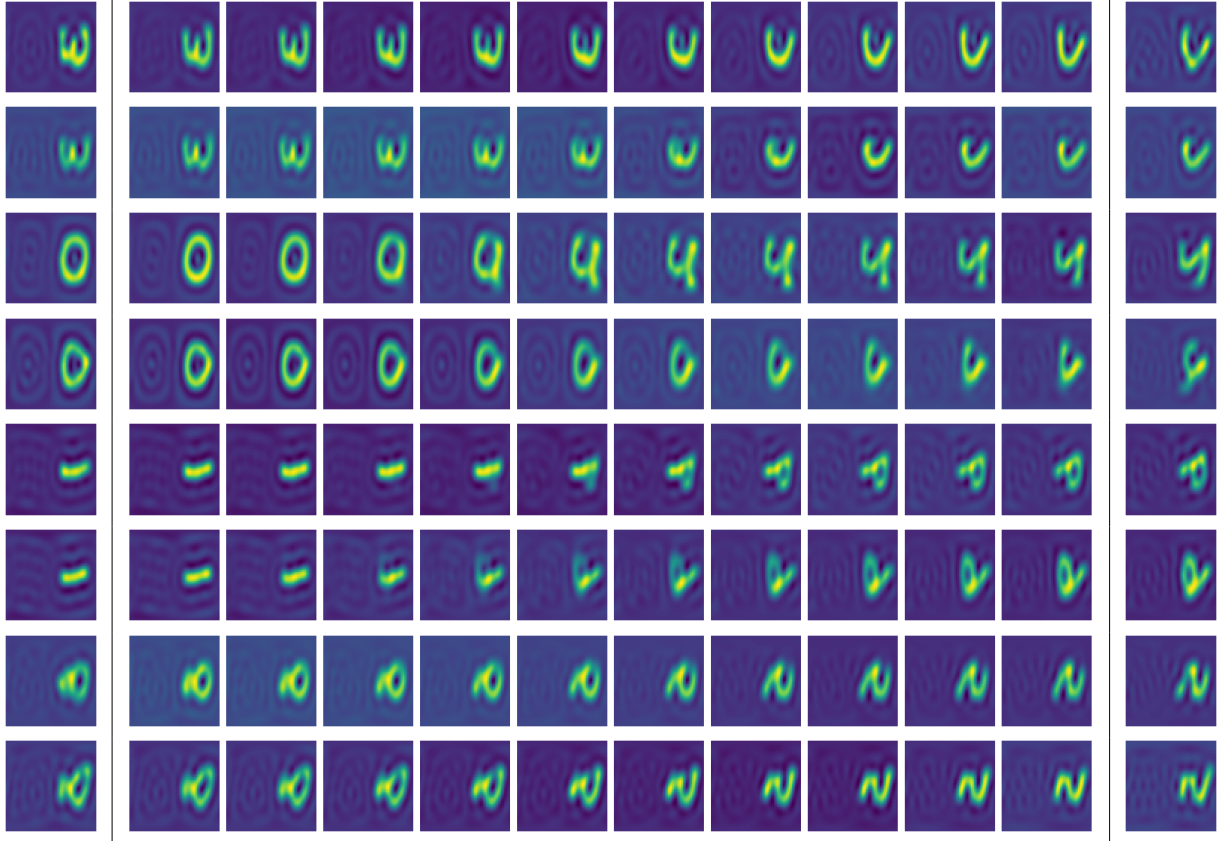


Figure A.4.17: **Trajectories across the latent space for the (NR/R; $z = 16$; AE) MNIST-on-the-sphere model.** We compute pairs of invariant latent embeddings using the model’s encoder, and linearly interpolate between them through the latent space. We then feed the interpolated embeddings into the decoder, together with the canonical frame, and compute the inverse SFT to get the image in real space. The left and right columns show the original images (after forward and inverse SFT) rotated to be placed in the learned canonical frame, whereas the center rows show the interpolated images. We can see that all trajectories are smooth, respecting spatial consistency, sign of a well structured latent space.

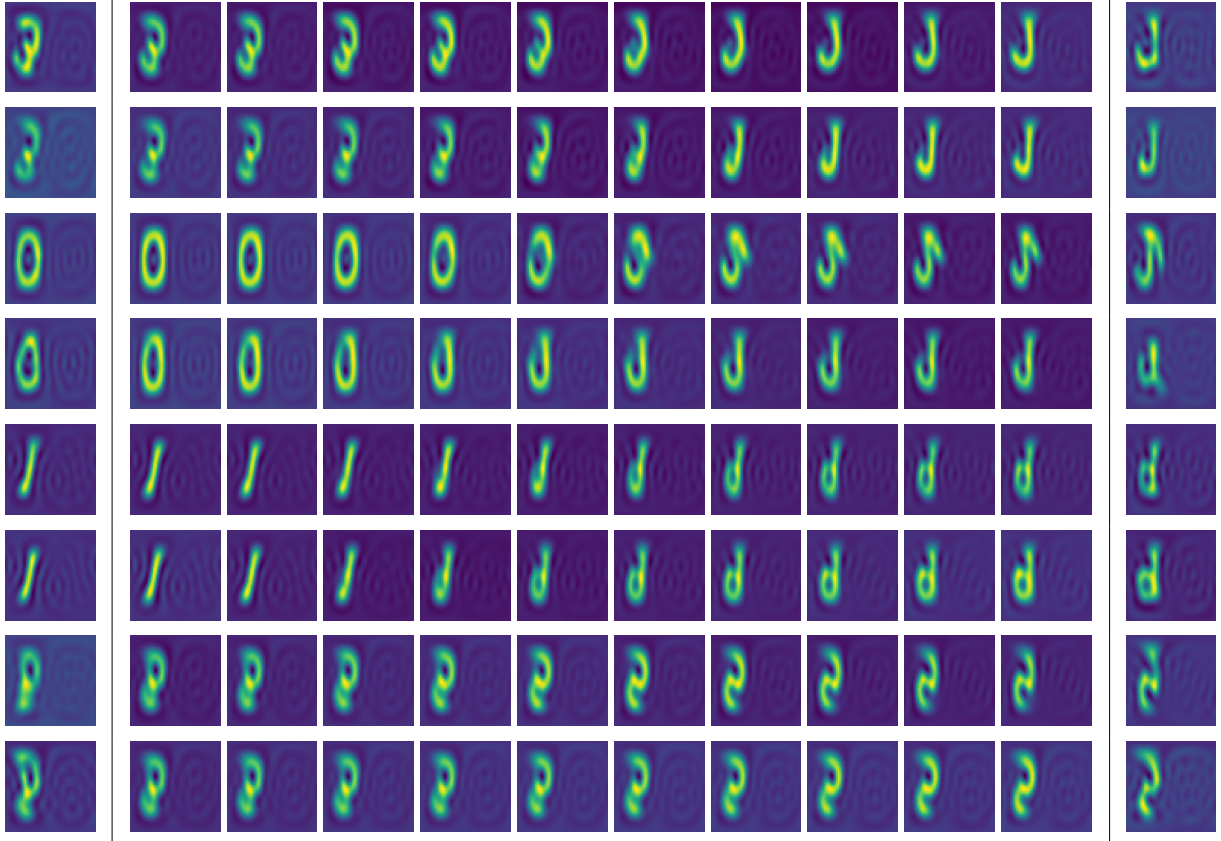


Figure A.4.18: **Trajectories across the latent space for the (NR/R; $z = 16$; VAE; $\beta = 0.6$) MNIST-on-the-sphere model.** We compute pairs of invariant latent embeddings using the model's encoder, and linearly interpolate between them through the latent space. We then feed the interpolated embeddings into the decoder, together with the canonical frame, and compute the inverse SFT to get the image in real space. The left and right columns show the original images (after forward and inverse SFT) rotated to be placed in the learned canonical frame, whereas the center rows show the interpolated images. We can see that all trajectories are smooth, respecting spatial consistency, sign of a well structured latent space.

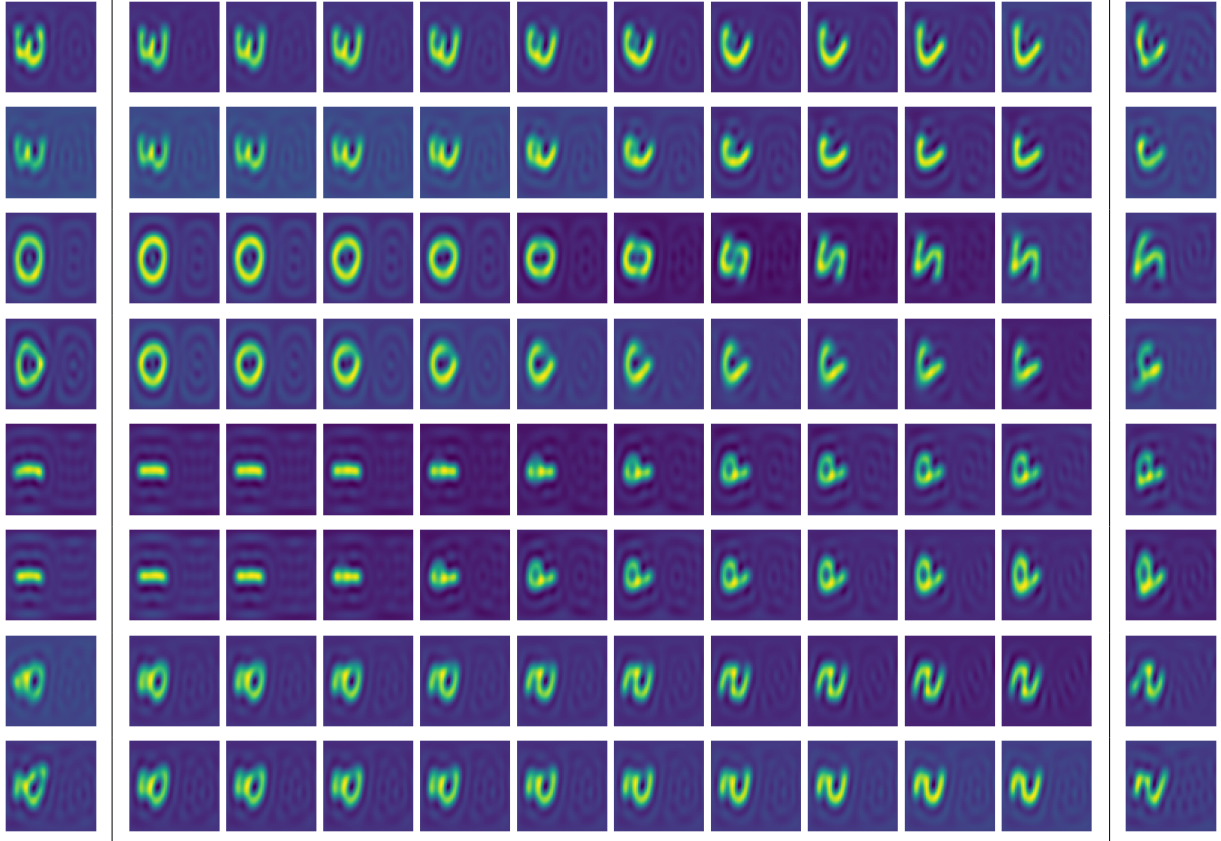


Figure A.4.19: **Trajectories across the latent space for the (NR/R; $z = 120$; VAE; $\beta = 2.8$) MNIST-on-the-sphere model.** We compute pairs of invariant latent embeddings using the model's encoder, and linearly interpolate between them through the latent space. We then feed the interpolated embeddings into the decoder, together with the canonical frame, and compute the inverse SFT to get the image in real space. The left and right columns show the original images (after forward and inverse SFT) rotated to be placed in the learned canonical frame, whereas the center rows show the interpolated images. We can see that all trajectories are smooth, respecting spatial consistency, sign of a well structured latent space.

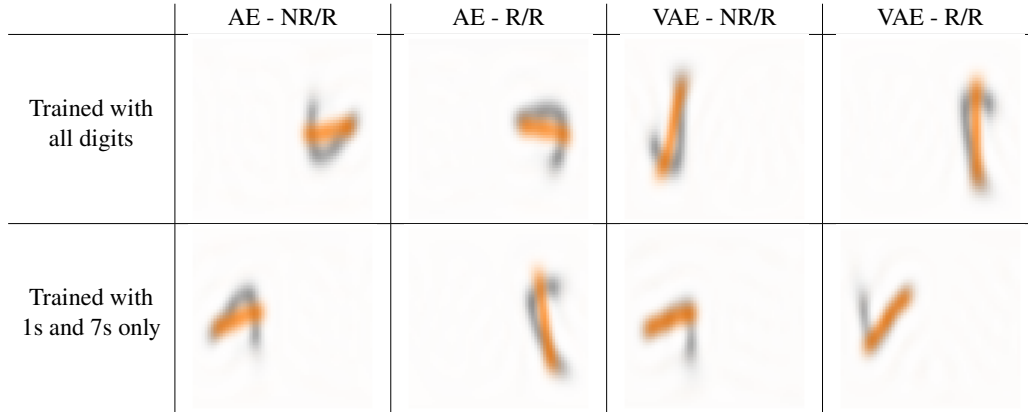


Figure A.4.20: **H-(V)AE implicitly learns to maximally overlap training images on MNIST-on-the-sphere.**

For each of the 4 models with $z = 16$, we train a version using only images containing 1s and 7s. For each of the resulting 8 models, we visualize the sum of training images of digits 1 and 7, when rotated to the canonical frame. We compute the sums of images with the same digit, and overlay them with different colors for ease of visualization. We test the hypothesis as whether H-(V)AE learns frames that align the training images such that they maximally overlap; we do so in two ways.

First: if the hypothesis were true, all canonical images of the same digit should maximally or near-maximally overlap - since they have very similar shape - and thus, their overlays would look like a “smooth” version of that digit. Indeed, we find this statement to be true for all models irrespective of their training strategy.

Second: we consider the alignment of images of different digits. We take 1s and 7s as examples given their similarity in shape. If the hypothesis were true, models trained with only 1s and 7s should align canonical 1s along the long side of canonical 7s; indeed we find this to be the case for two out of four cases, interestingly corresponding to the variational models. The same alignment between 1s and 7s, however, does not necessarily hold for models trained with all digits. This is because maximizing overlap across a set of diverse shapes does not necessarily maximize the overlap within any independent pair of such shapes. Indeed, we find that canonical 1s and canonical 7s do not overlap optimally with each other for models trained with all digits.

We note that these tests do not provide a formal proof, but rather empirical evidence of the characteristics of frames learned by H-(V)AE on the MNIST-on-the-sphere task.

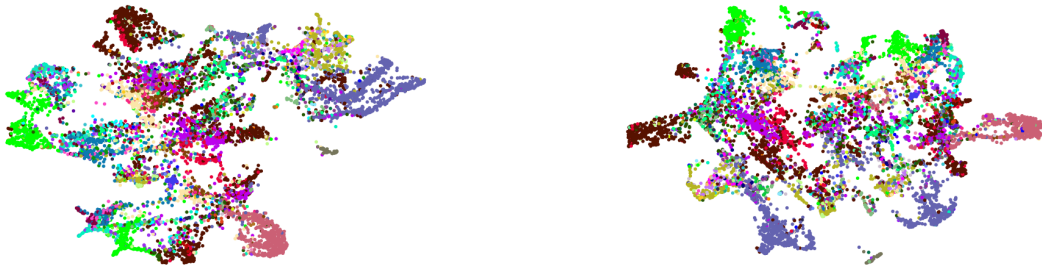


Figure A.4.21: **2D visualization via UMAP of the invariant latent embeddings of Shrec17 test data learned by H-(V)AE.** Left: H-AE, Right: H-VAE. Points are colored by class (55 classes).

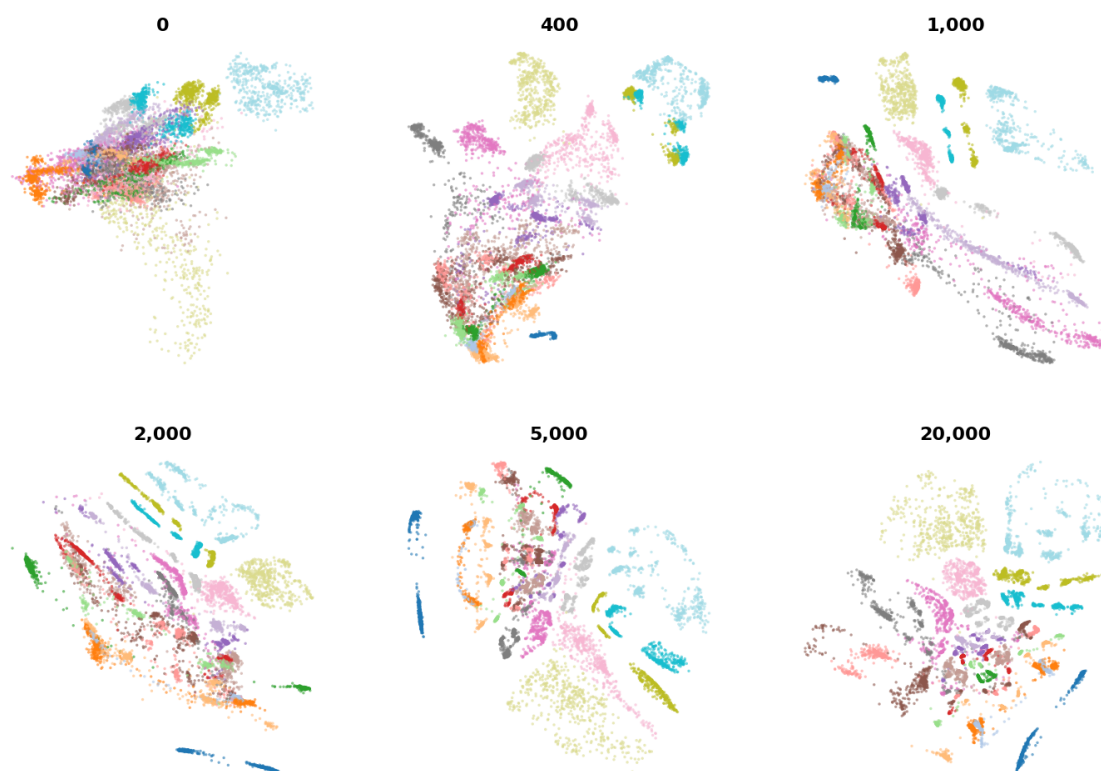


Figure A.4.22: **Amino acid latent space learned by H-AE.** Visualization of the test data's invariant latent space learned by H-AE trained with varying amounts of the training data. As more training data is added, the separation of clusters containing residues with most similar conformations becomes more distinct. Notably, even with no training data, conformation clusters can be identified.

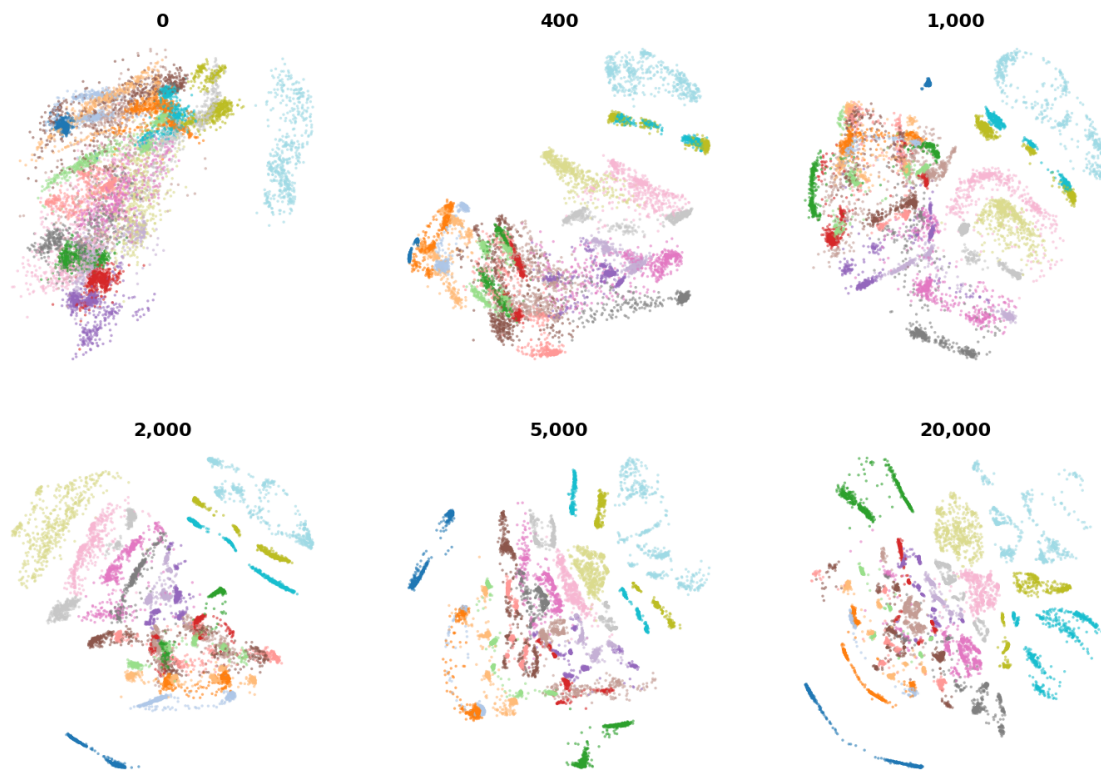


Figure A.4.23: **Amino acid latent space learned by H-VAE.** Visualization of the test data's invariant latent space learned by H-VAE ($\beta = 0.025$) trained with varying amounts of training data.

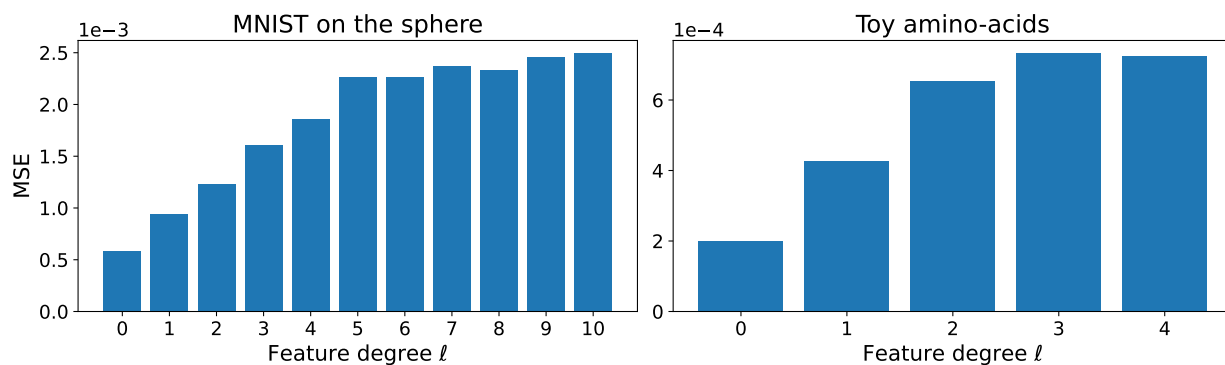


Figure A.4.24: **Reconstruction loss as a function of feature degree ℓ .** Test reconstruction loss (MSE) of H-VAE split by feature degree ℓ , for the MNIST-on-the-sphere (left) and Toy amino acids dataset (right). In both cases, features of larger degrees are harder to reconstruct accurately. The increase in loss is more steep for smaller degrees.

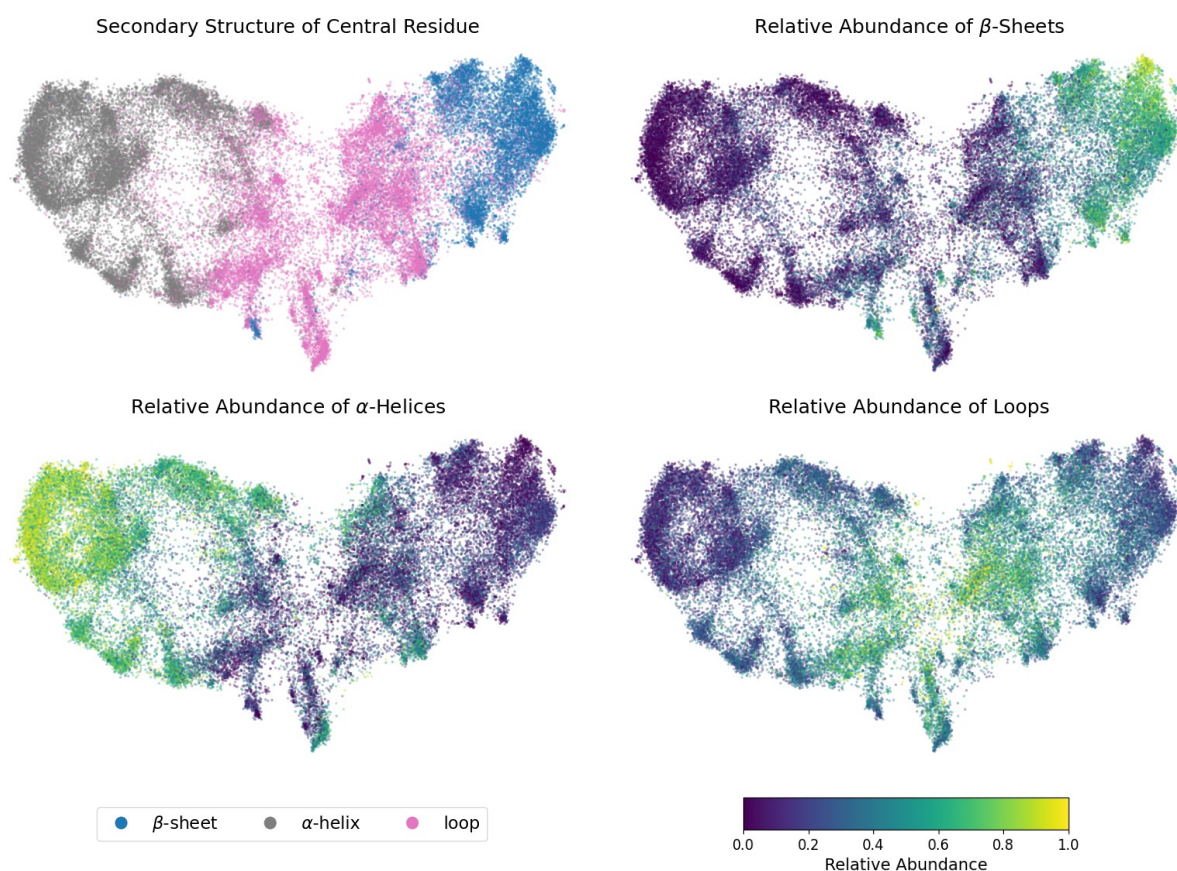


Figure A.4.25: **2D UMAP visualization of the 128-dimensional invariant latent space learned by H-AE trained on the protein structure neighborhoods with $L = 6$, annotated by secondary structure.** Within each figure, each point represents a neighborhood. Top left panel: points are colored by secondary structure of the neighborhood's central residue; a simple logistic regression model trained on the embeddings of 300,000 training neighborhoods is able to recover the central residue's secondary structure on the test set with 90% accuracy. Other panels: points are colored by the relative abundance of a single secondary structure component, computed as the proportion of atoms within the neighborhood that are assigned to that secondary structure.



Figure A.4.26: **2D UMAP visualization of the 128-dimensional invariant latent space learned by H-AE trained on the protein structure neighborhoods with $L = 6$, annotated by number of atoms** Within each figure, each point represents a neighborhood. Points are colored by the *total power* of the neighborhood's zernike projection, which is a proxy for the neighborhood's **number of atoms**.

Appendix B

SUPPLEMENT FOR CHAPTER 4

B.1 Proof of equivalence between MSE and cosine loss in predicting Sine and Cosine of χ angles

Let χ_i and $\hat{\chi}_i$ be the true and the predicted angle values for a residue's i^{th} χ angle, respectively. Let $\mathbf{v}_i \equiv [\sin \chi_i, \cos \chi_i]$ and $\hat{\mathbf{v}}_i \equiv [\sin \hat{\chi}_i, \cos \hat{\chi}_i]$ denote their respective 2D vectors on the unit circle, whose two components are the sine and cosine of the angles themselves. Then we have:

$$\begin{aligned}
 \text{MSE}_{\sin-\cos}(\hat{\chi}_i, \chi_i) &= \frac{1}{2}(\cos \hat{\chi}_i - \cos \chi_i)^2 + \frac{1}{2}(\sin \hat{\chi}_i - \sin \chi_i)^2 \\
 &= 1 - \cos \hat{\chi}_i \cos \chi_i - \sin \hat{\chi}_i \sin \chi_i \\
 &= 1 - \langle \hat{\mathbf{v}}_i, \mathbf{v}_i \rangle \\
 &= \text{CosineLoss}(\hat{\mathbf{v}}_i, \mathbf{v}_i)
 \end{aligned} \tag{B.1.1}$$

where the jump from step 1 to step 2 is granted by the trigonometric identity $\sin^2 \theta + \cos^2 \theta = 1$.

B.2 Training Details

B.2.1 Toy Task

All models were constructed with 5 equivariant blocks and no invariant feed-forward neural network (FFNN) for the two models predicting invariant quantities. All models were trained using Adam [113] for 30 epochs with learning rate 0.001 and a batch size of 32. The model exhibiting lowest validation loss was used.

B.2.2 Side-chain Packing

Models with $\ell_{\max} = 4$ have five equivariant blocks with a per- ℓ hidden feature size of 128. For models with $\ell_{\max} = 5$ we use 5 blocks with per- ℓ size 96. In doing so, the two models have comparable number of parameters (3M). All models have a 3-layer FFNN with silu nonlinearity and dropout normalization rate of 0.1. We found it useful to tune the dropout rate to prevent overfitting. We train all models for 10 epochs, keeping the model with lowest validation loss at the end of an epoch (convergence usually happened by epoch ~ 8); models with $\ell_{\max} = 5$ took roughly 50 minutes per epoch to train on a single NVIDIA A40 GPU, while models with $\ell_{\max} = 4$ took 35 minutes.

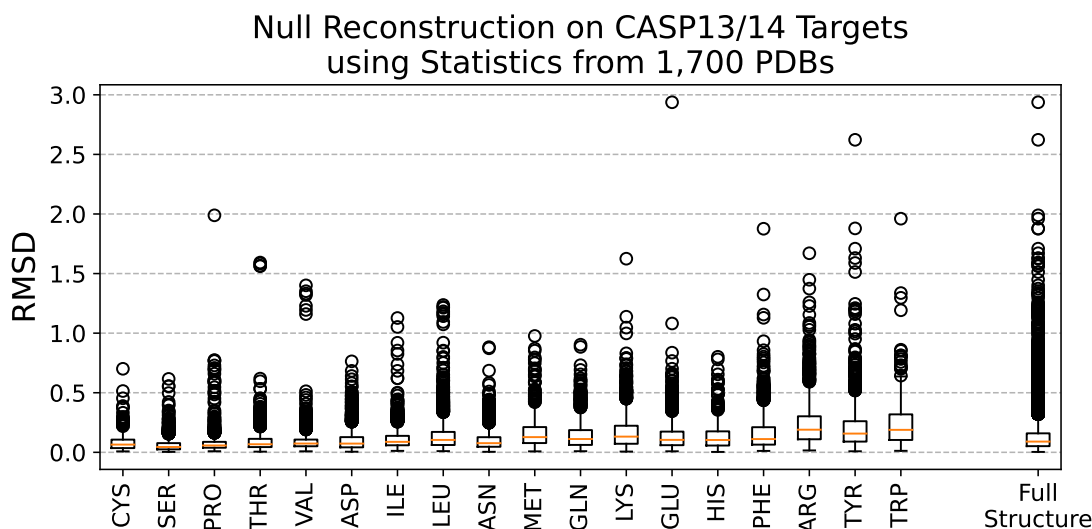


Figure B.2.1: Null Reconstruction Error on Test data by using data-derived values as redundant internal coordinates. Effectively, each redundant internal coordinate (i.e. excluding chi angles) gets substituted from a single value computed as the median of the corresponding value in a reference dataset. Across all structures, the average Null Reconstruction RMSD is 0.127 Å. We notice a small number of outliers (single residues with abnormally high RMSD), but we do not investigate the causes.

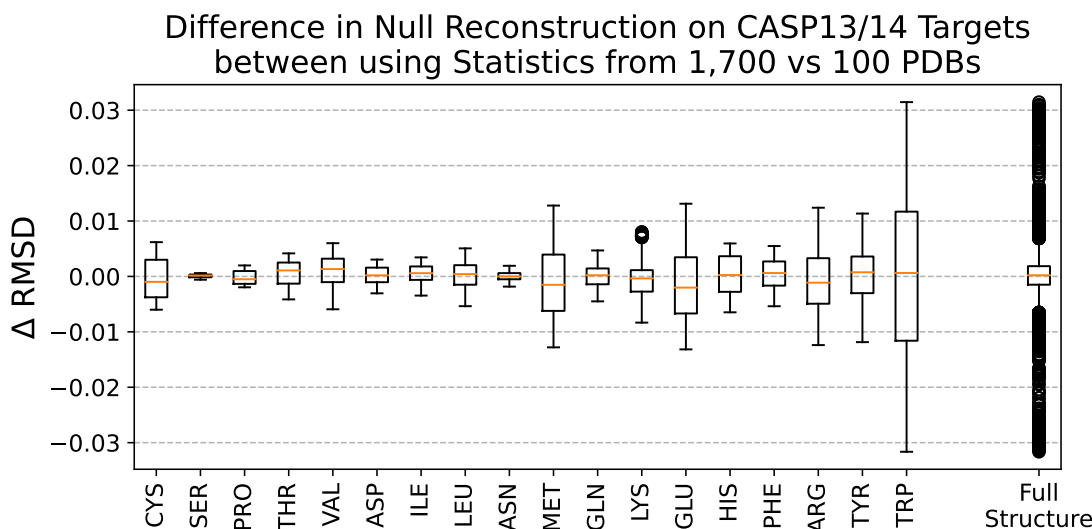


Figure B.2.2: Difference in Null Reconstruction Error on Test data by using a smaller set of structures for reference.

Amino-Acid	π -Symmetric χ	Pairs of Equivalent Atom names
ARG	-	NH1-NH2
TYR	χ_2	CD1-CE1, CD2-CE2
PHE	χ_2	CD1-CE1, CD2-CE2
ASP	χ_2	OD1-OD2
GLU	χ_3	OE1-OE2

Table B.2.1: List of conformational symmetries that we take into consideration.

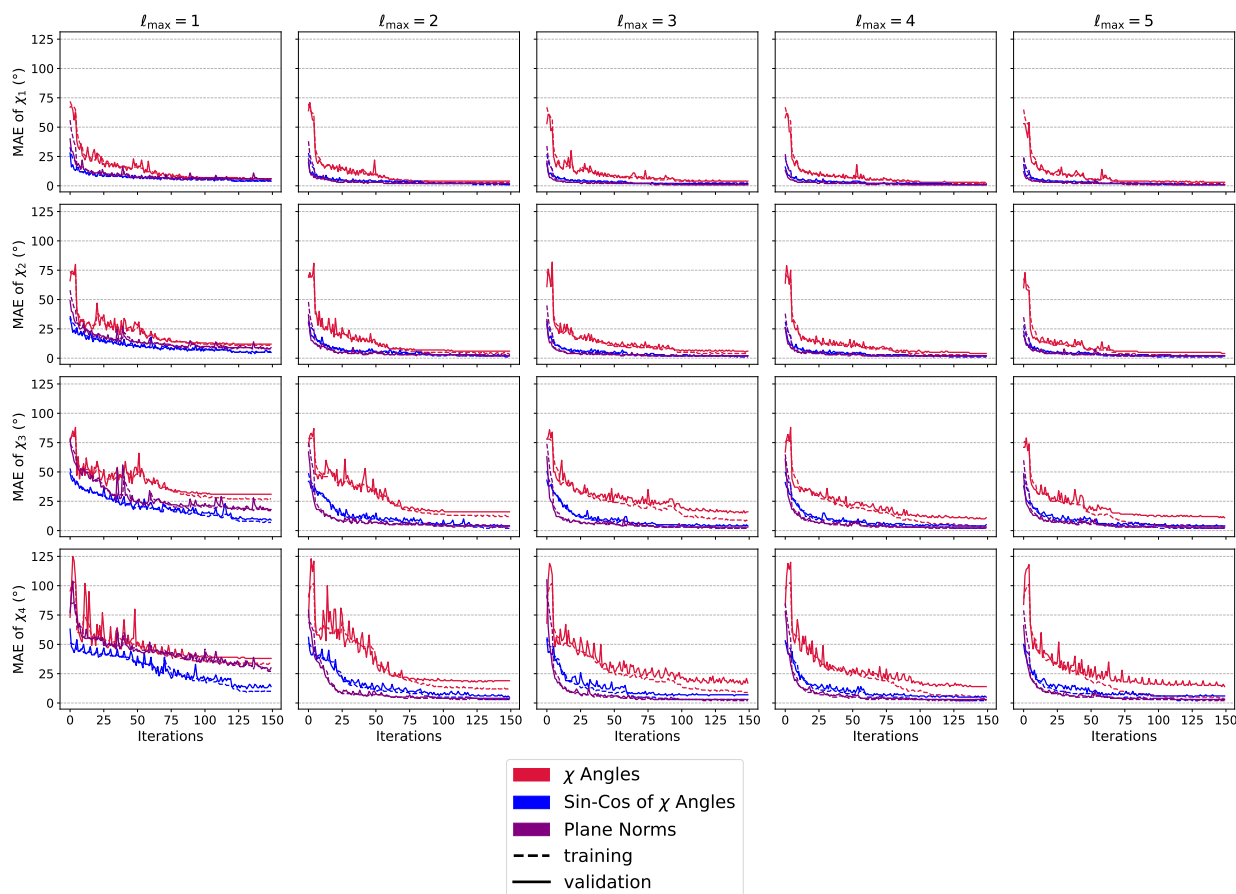


Figure B.2.3: χ angles error trace during training for the simple task of predicting χ angles from true amino-acid conformations. The Sin-Cos and Plane Norms models show comparable convergence curves, except for $\ell_{\max} = 1$ where the Plane Norms model struggles with χ_3 and χ_4 . The Angles model has the worst convergence trace, even overfitting for high $\ell_{\max}$.

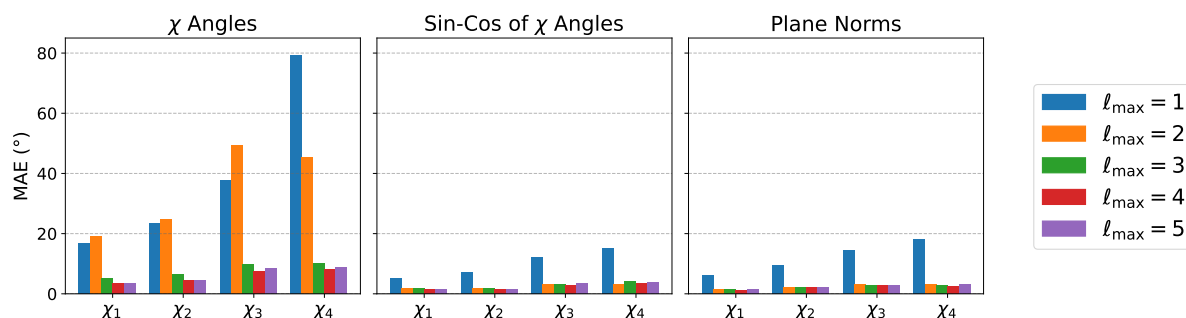


Figure B.2.4: Test MAE for the simple task of predicting χ angles from atomic conformation, with loss weighted by frequency. Models trained with weighting χ angles in the loss function based on their frequency. We use the following weights for χ_1 to χ_4 , which are derived from how many residues have which χ angle: [1.0, 18.0/14.0, 18.0/5.0, 18.0/2.0]. We then re-scale them to make sure the absolute scale of the loss function is the same as it was in the un-weighted case.

	Method	Angle MAE ° ↓				Angle Accuracy % ↑			Atom RMSD Å ↓		
		χ_1	χ_2	χ_3	χ_4	All	Core	Surface	All	Core	Surface
CASP13	H-Packer ₀ $\ell_{\max} = 4$	27.60	32.31	47.87	52.77	48.3	59.1	40.5	0.979	0.771	1.136
	H-Packer ₀ $\ell_{\max} = 5$	26.89	31.95	47.51	52.75	49.3	61.5	40.4	0.961	0.726	1.131
	H-Packer ₂ $\ell_{\max} = 4$	24.36	29.94	46.01	52.89	53.0	67.7	42.5	0.884	0.610	1.083
	H-Packer ₂ $\ell_{\max} = 5$	23.64	29.47	45.17	53.26	54.4	69.9	43.6	0.863	0.575	1.070
	H-Packer ₅ $\ell_{\max} = 4$	24.27	29.76	46.32	52.56	53.5	68.8	43.0	0.878	0.594	1.081
	H-Packer ₅ $\ell_{\max} = 5$	23.60	29.40	44.91	52.91	54.7	70.7	43.7	0.858	0.564	1.067
	H-Packer _{up} $\ell_{\max} = 4$	20.78	27.48	44.58	51.97	56.8	72.7	45.6	0.790	0.514	0.999
	H-Packer _{up} $\ell_{\max} = 5$	20.03	26.88	42.74	52.07	58.4	75.4	46.4	0.765	0.483	0.980
CASP14	H-Packer ₀ $\ell_{\max} = 4$	33.14	36.10	49.30	50.22	39.8	55.9	30.7	1.102	0.780	1.307
	H-Packer ₀ $\ell_{\max} = 5$	32.31	35.90	49.05	50.34	40.8	57.0	31.6	1.087	0.762	1.297
	H-Packer ₂ $\ell_{\max} = 4$	30.43	34.49	48.13	50.39	43.3	63.0	32.7	1.027	0.658	1.265
	H-Packer ₂ $\ell_{\max} = 5$	29.96	34.32	47.46	50.50	45.0	65.1	34.1	1.011	0.629	1.250
	H-Packer ₅ $\ell_{\max} = 4$	30.36	34.38	48.76	50.62	43.5	64.1	32.5	1.024	0.648	1.260
	H-Packer ₅ $\ell_{\max} = 5$	29.61	34.03	46.72	50.35	45.2	65.5	34.0	1.002	0.626	1.244
	H-Packer _{up} $\ell_{\max} = 4$	26.84	32.10	46.90	50.03	46.7	68.1	35.0	0.934	0.557	1.178
	H-Packer _{up} $\ell_{\max} = 5$	26.58	31.54	45.67	49.46	48.1	69.5	36.2	0.915	0.534	1.160

Table B.2.2: Ablation in $\ell_{\max}$ of H-Packer, comprehensive of all metrics.

Method	Atom RMSD Å ↓					
	CASP13			CASP14		
	All	Core	Surface	All	Core	Surface
H-Packer ₀ $\ell_{\max} = 4$	0.943	0.733	1.099	1.065	0.743	1.273
H-Packer ₂ $\ell_{\max} = 4$	0.848	0.577	1.046	0.989	0.618	1.230
H-Packer ₅ $\ell_{\max} = 4$	0.841	0.561	1.043	0.986	0.608	1.223
H-Packer _{up} $\ell_{\max} = 4$	0.755	0.483	0.962	0.899	0.526	1.143
H-Packer ₀ $\ell_{\max} = 5$	0.923	0.689	1.092	1.050	0.723	1.262
H-Packer ₂ $\ell_{\max} = 5$	0.826	0.540	1.032	0.972	0.591	1.214
H-Packer ₅ $\ell_{\max} = 5$	0.821	0.530	1.029	0.964	0.589	1.208
H-Packer _{up} $\ell_{\max} = 5$	0.730	0.452	0.941	0.880	0.504	1.124

Table B.2.3: Reconstruction RMSD when considering the additional (non-natural) symmetries used by AttnPacker.

Method	Atom RMSD Å↓					
	CASP13			CASP14		
	All	Core	Surface	All	Core	Surface
H-Packer ₀ $\ell_{\max} = 4$	0.964	0.745	1.129	1.086	0.754	1.297
H-Packer ₂ $\ell_{\max} = 4$	0.863	0.576	1.074	1.010	0.628	1.255
H-Packer ₅ $\ell_{\max} = 4$	0.858	0.560	1.071	1.007	0.617	1.249
H-Packer _{up} $\ell_{\max} = 4$	0.767	0.477	0.987	0.915	0.526	1.167
H-Packer ₀ $\ell_{\max} = 5$	0.943	0.697	1.123	1.070	0.735	1.286
H-Packer ₂ $\ell_{\max} = 5$	0.842	0.540	1.060	0.992	0.598	1.239
H-Packer ₅ $\ell_{\max} = 5$	0.837	0.529	1.057	0.984	0.596	1.232
H-Packer _{up} $\ell_{\max} = 5$	0.741	0.444	0.968	0.895	0.500	1.147

Table B.2.4: **Reconstruction RMSD when standardizing the true structure by reconstructing it using our data-derived constant redundant internal coordinates.**

Method	Atom RMSD Å↓					
	CASP13			CASP14		
	All	Core	Surface	All	Core	Surface
H-Packer ₀ $\ell_{\max} = 4$	0.927	0.708	1.092	1.049	0.717	1.263
H-Packer ₂ $\ell_{\max} = 4$	0.828	0.543	1.038	0.973	0.588	1.220
H-Packer ₅ $\ell_{\max} = 4$	0.821	0.527	1.033	0.969	0.577	1.213
H-Packer _{up} $\ell_{\max} = 4$	0.733	0.447	0.950	0.881	0.494	1.132
H-Packer ₀ $\ell_{\max} = 5$	0.906	0.661	1.084	1.034	0.696	1.251
H-Packer ₂ $\ell_{\max} = 5$	0.805	0.506	1.023	0.953	0.560	1.202
H-Packer ₅ $\ell_{\max} = 5$	0.800	0.496	1.019	0.945	0.558	1.195
H-Packer _{up} $\ell_{\max} = 5$	0.706	0.414	0.930	0.860	0.471	1.111

Table B.2.5: **Reconstruction RMSD when considering both the additional (non-natural) symmetries used by AttnPacker, and when standardizing the true structure by reconstructing it using our data-derived constant redundant internal coordinates.**

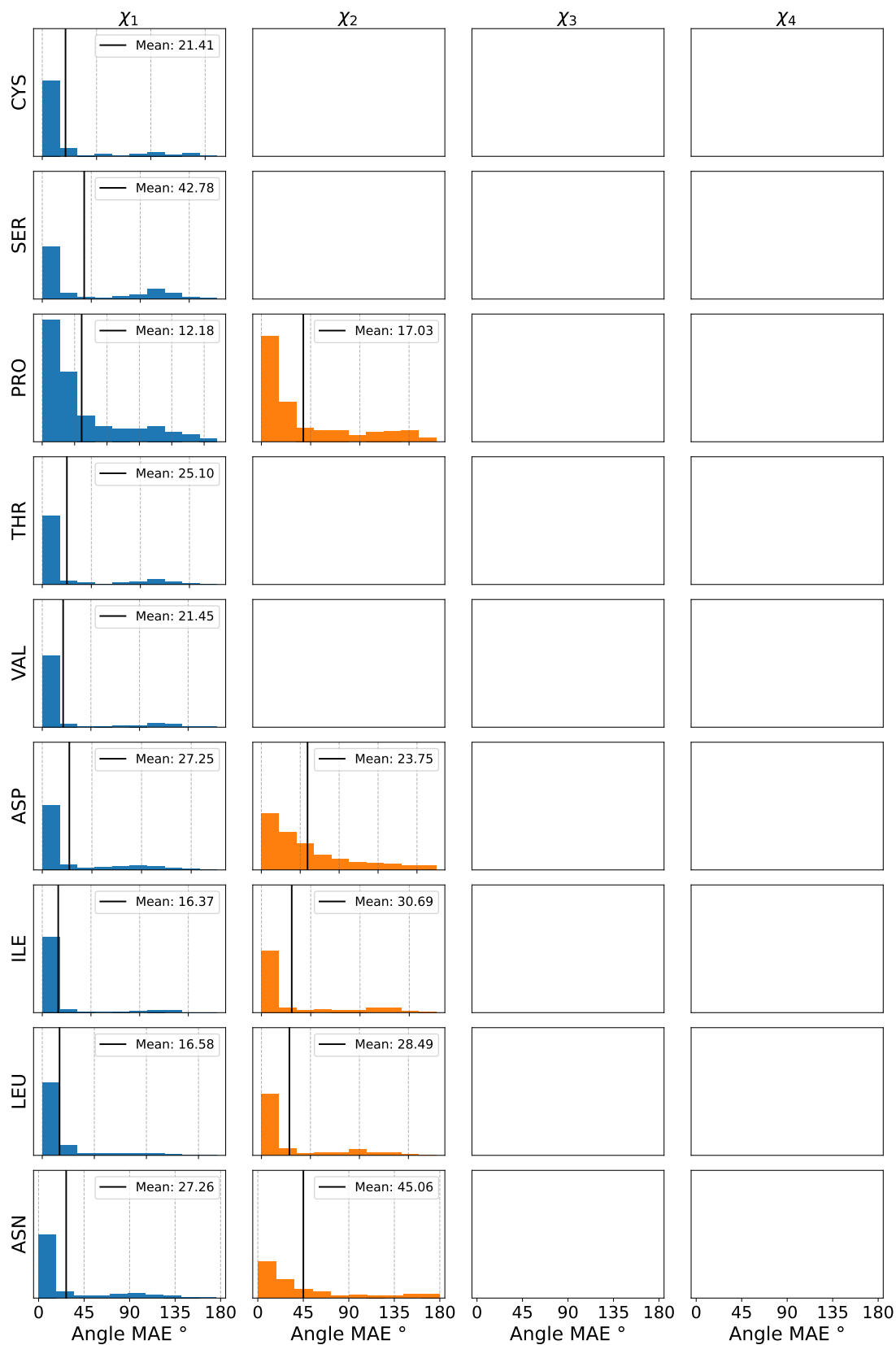


Figure B.2.5: Angle Error distribution on CASP13 for H-Packer $^{\ell_{\max}=5}_5$, part 1.

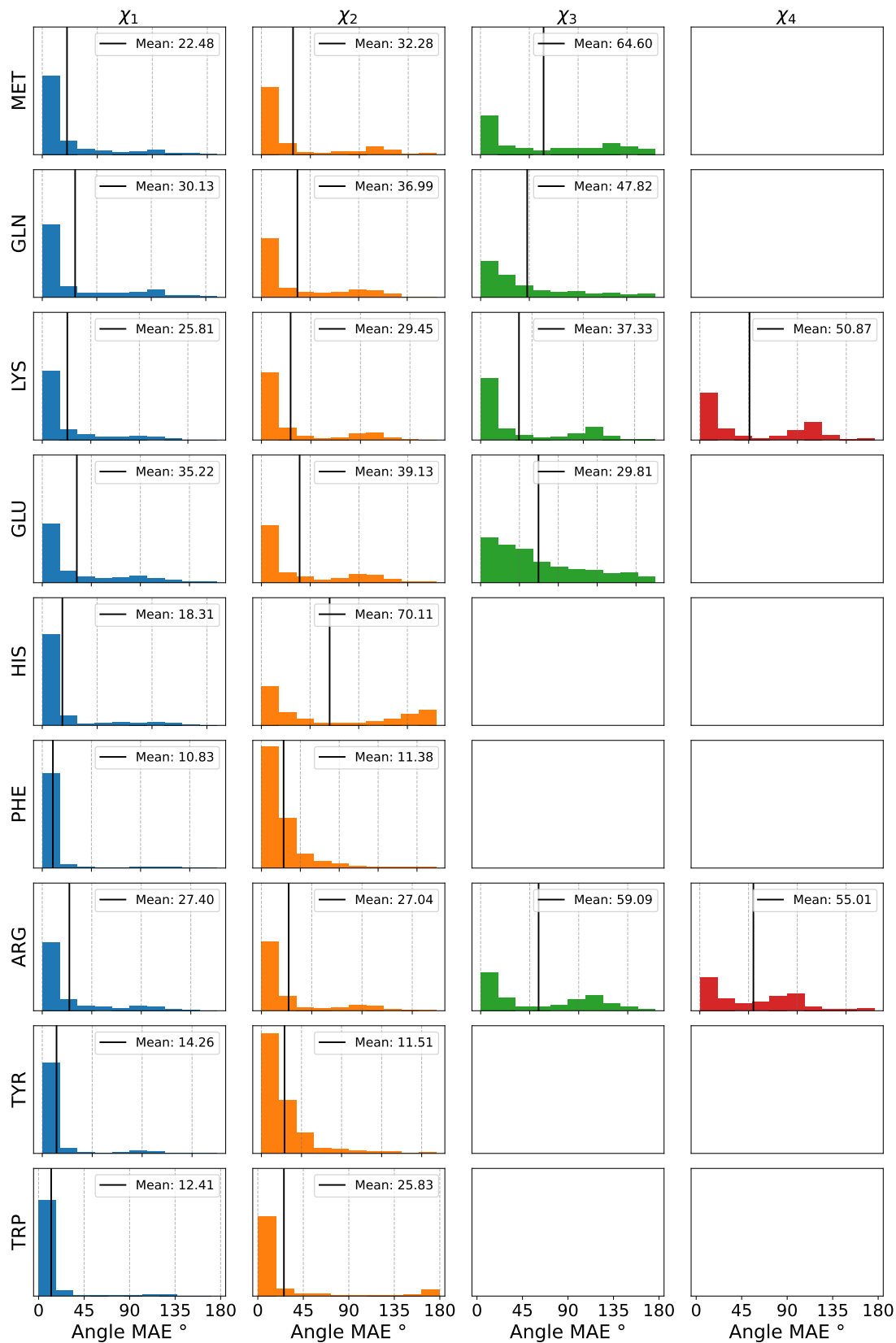


Figure B.2.6: Angle Error distribution on CASP13 for H-Packer $_{\ell_{\max}=5}^{\ell_{\max}=5}$, part 2.

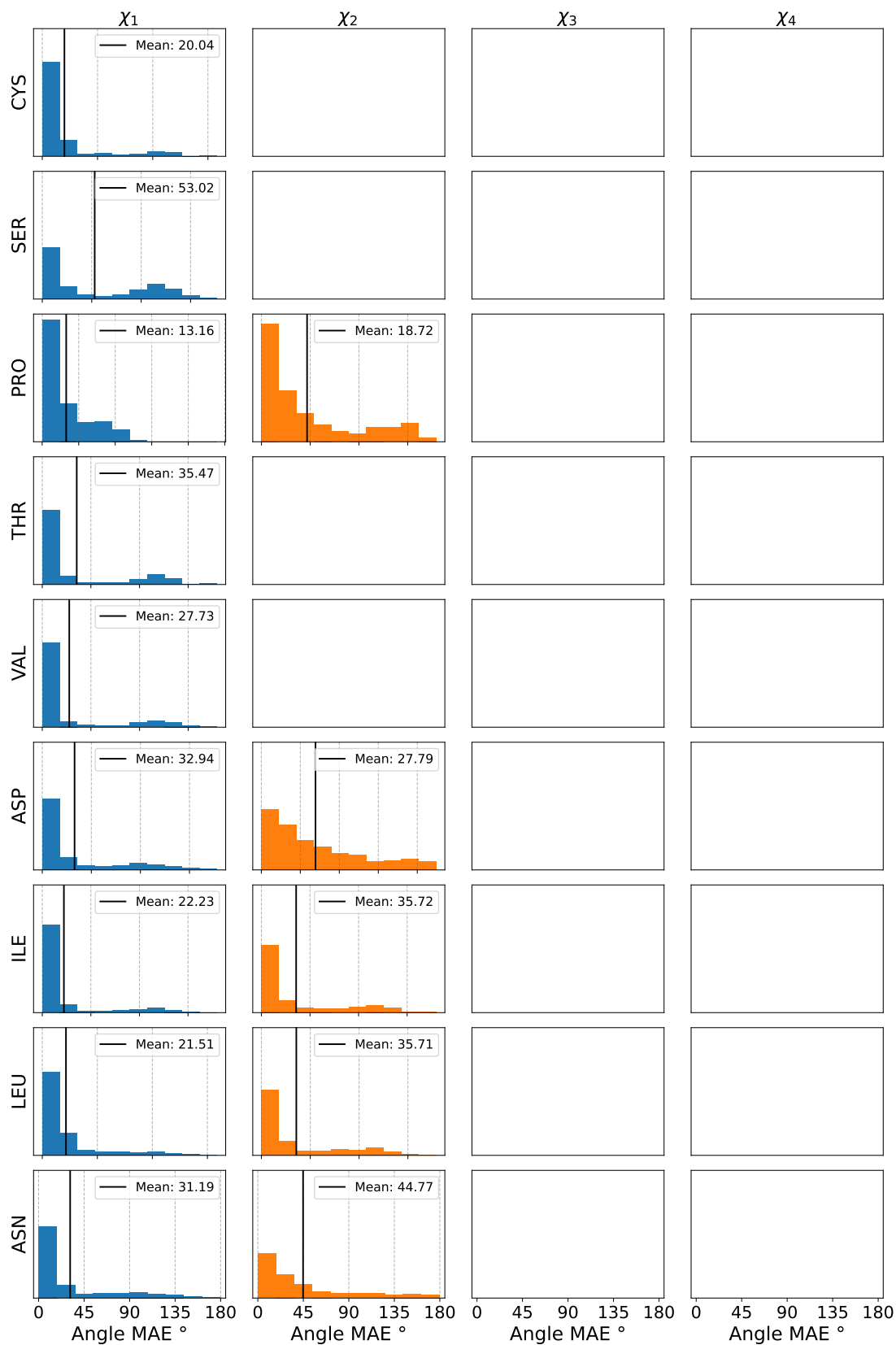


Figure B.2.7: Angle Error distribution on CASP14 for H-Packer $_{\ell_{\max}=5}^{\ell_{\max}=5}$, part 1.

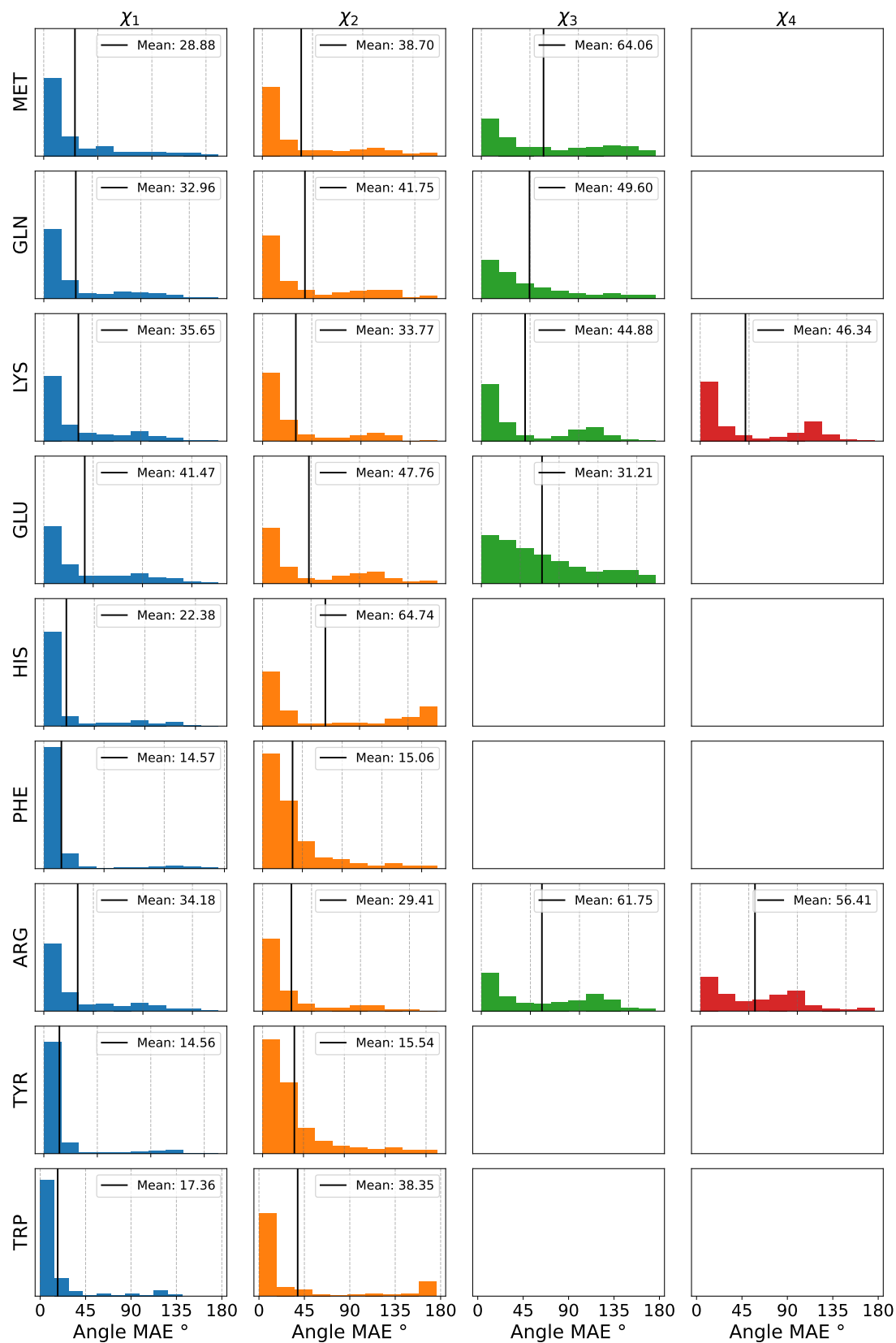


Figure B.2.8: Angle Error distribution on CASP14 for H-Packer $_{5}^{\ell_{\max}=5}$, part 2.

SUPPLEMENT FOR CHAPTER 5

C.1 Extended methods**C.1.1 Pre-processing of protein structure data**

To pre-process protein structure data, we developed two distinct pipelines based on either (i) PyRosetta [39] or (ii) Biopython [45] together with other open-source tools, with code adapted from [20]. The PyRosetta-based pipeline is considerably faster but requires a license, whereas the Biopython-based pipeline is fully open source. We train models using data generated by both pipelines, with the constraint that the Pre-processing pipeline used at inference must match that used during training. Performance differences between the two pipelines are minor (Fig. C.1.6 and C.1.18, and Table C.1.1); unless otherwise stated, we report results obtained with the PyRosetta pipeline, which yields slightly better performance and has faster Pre-processing runtime.

For the PyRosetta workflow, we use PyRosetta functionalities for all Pre-processing steps: repairing PDB files by adding missing residues, adding hydrogen atoms, assigning partial atomic charges, computing solvent-accessible surface areas (SASA); we ignore non-canonical amino acids. For the open-source Pre-processing workflow ("Biopython" pipeline), we proceed as follows. First, we use OpenMM [64] to repair PDB files by adding missing residues and substituting non-canonical residues with their canonical counterparts. Hydrogen atoms are then added using the reduce program [254]. Partial atomic charges are assigned from the AMBER99sb force field [180], and SASA are computed using Biopython [45].

Both pre-processing pipelines retain atoms belonging to non-protein residues and ions, in contrast to the RaSP pre-processing procedure [20]. Notably, the PyRosetta-based pipeline does not replace non-canonical residues.

C.1.2 HERMES pre-training

We pre-trained HERMES using an inverse folding objective, in which the model predicts the identity of a masked focal amino acid (i.e., the native residue in the structure) given its surrounding atomic neighborhood. This training task is analogous to that used for H-CNN [185]. We adopted the same data splits as in H-CNN: 10,957 structures for training, 2,730 for validation, and 212 for testing. These splits are derived from ProteinNet's [8] 30% sequence-identity clustering of PDB structures available at the time of CASP12.

Model parameters were optimized for 10 epochs using the Adam optimizer [112] with a learning rate of 10^{-3} . We selected the model checkpoint with the lowest validation loss at the end of each epoch. A single HERMES model is implemented as an ensemble of 10 independently trained neural network instances, whose predictions are averaged at inference time. Pre-training a single network instance required approximately 40 minutes per epoch on a single NVIDIA A40 GPU.

C.1.3 Rosetta relaxation of mutant structures for the HERMES-relaxed protocol

For the HERMES-relaxed protocol, we use PyRosetta [39] to generate and locally refine mutant protein structures. Specifically, the focal residue is first substituted with the desired mutant amino acid, after which all side-chain atoms within 12 Å of the focal residue's C- α atom are relaxed using the FastRelax protocol with the ref2015_cart energy function and a single relaxation cycle.

We performed a targeted ablation study to select the relaxation parameters. These experiments indicate that allowing backbone atoms to relax slightly degrades performance, and that ensembling over multiple independent relaxation runs does not improve accuracy (Fig. C.1.19A), while substantially increasing computational cost (Fig. C.1.19B). Thus, all results for the HERMES-*relaxed* protocol are reported using side-chain-only relaxation with a single FastRelax cycle.

C.1.4 Training HERMES-*amortized* to implicitly learn about relaxation

We fine-tuned HERMES to align HERMES-*fixed* predictions to those of HERMES-*relaxed*, on a subset of the pre-training protein sites. Specifically, we uniformly sampled 10% of the proteins pre-training training set (1,284), and from those we uniformly sampled 5% of the sites. We similarly sampled 1% of the sites from the proteins of the pre-training validation set, and 5% of the pre-training testing set.

C.1.5 HERMES fine-tuning for downstream tasks

In all analyses, we fine-tuned HERMES models under a Huber Loss objective with hyperparameter $\delta = 1.0$, for 15 epochs using the Adam optimizer [112] with a learning rate of 10^{-3} and a batch size of 128 mutations, selecting the model checkpoint with the lowest validation loss at the end of each epoch. The only exception is the models without pre-training, which are trained for 25 epochs.

By convention, we define model outputs such that higher predicted values correspond to more favorable mutation effects. Accordingly, when fine-tuning on experimental $\Delta\Delta G$ measurements—where lower values indicate greater stability—we trained the model to predict $-\Delta\Delta G$. This sign convention ensures consistent interpretation of model outputs across tasks and is fixed throughout all reported analyses.

To speed-up convergence during fine-tuning, we first rescale the weight matrix and bias vector of the network’s output layer so that the mean and variance of the output logits match those of the validation scores. This initialization step requires a forward pass through the validation data to estimate the mean and variance, but it makes the model outputs immediately on the same scale as the experimental scores. Thus, fine-tuning avoids spending early epochs merely adjusting output magnitudes.

Overall, fine-tuning is computationally efficient. Training a single neural network instance requires approximately 2.5 minutes per epoch on the cDNA117k dataset (~117,000 mutations) and about 4 minutes per epoch on the Megascale training dataset (~217,000 mutations) on a single NVIDIA A40 GPU.

C.1.6 Fine-tuning datasets

Stability effect prediction. For protein stability prediction, we fine-tuned and evaluated the performance of HERMES on the same datasets of three recent structure-based predictors of mutational stability effects—RaSP [20], Stability-Oracle [59], and ThermoMPNN [60]—so results are comparable. The data from these models are used in following way:

- *RaSP dataset.* We used the RaSP dataset [20] as provided by the authors on their github repository (https://github.com/KULL-Centre/_2022_ML-ddG-Blaabjerg). As in the original RaSP paper, the target values are Rosetta-computed stability changes ($\Delta\Delta G$), which are known to be reliable primarily within the range $[-7, 1]$ kcal/mol. To account for this, RaSP applies a sigmoid transformation to the raw $\Delta\Delta G$ values prior to training, effectively saturating the targets outside this interval.

We adopt the same sigmoid transformation, with one modification: we center the transformed values such that $\Delta\Delta G = 0$ maps to zero after transformation. This centering is required by the HERMES

output parameterization, in which the predicted stability change for a mutation to the same amino acid is constrained to be zero (i.e., $\Delta\Delta G_{aa_i \rightarrow aa_i} = 0$). While this property holds for physical $\Delta\Delta G$ values, it is not preserved by the uncentered sigmoid transform used in RaSP. Our centered transformation therefore ensures consistency between the model’s output space and the physical interpretation of stability changes, and it follows,

$$F(\Delta\Delta G) = \frac{1}{1 + e^{-\beta(\Delta\Delta G - \alpha)}} - \frac{1}{1 + e^{\beta\alpha}} \quad (\text{C.1.1})$$

- *Stability-Oracle datasets.* Stability-Oracle introduces two curated datasets that we use in this work [59]:

(i) cDNA117k (training set): derived from the Megascaple cDNA display proteolysis dataset # 1 [232]. The original Megascaple dataset reports approximately 850,000 thermodynamic folding stability measurements (ΔG) across 354 natural and 188 de novo mini-protein domains (40–72 amino acids in length). Following the Stability-Oracle protocol, mutations are filtered to enforce at most 30% sequence identity with the test set, yielding a reduced set of approximately 117,000 mutation-induced stability changes ($\Delta\Delta G$), referred to as cDNA117k.

(ii) T2837 (test set): assembled by combining several commonly used benchmarking datasets of experimentally measured $\Delta\Delta G$ values, including S669 [172], myoglobin [126], ssym [181], and p53 [33]. The resulting test set comprises 2,837 mutations.

We obtained the cDNA117k and T2837 datasets from the Stability-Oracle [59] github repository (<https://github.com/danny305/StabilityOracle/tree/master>). At the time of access, the residue indices provided in the dataset did not correspond to the residue numbering in the original PDB files, but instead to an intermediate, post-processed representation that was not documented in sufficient detail to allow straightforward recovery of the original numbering. To ensure consistency with structural data, we manually corrected the residue numbers in the CSV files to match those in the corresponding PDB structures. The corrected versions of these datasets are included in our repository.

- *ThermoMPNN dataset.* ThermoMPNN also leveraged Megascaple [232], but used datasets # 2 and # 3, curated and split by the authors at a 25% sequence-identity cutoff into 216k training and 28k test mutation effects [60]. We used the train, valid, and test splits of the Megascaple dataset as curated in ThermoMPNN, and provided on their github repository (<https://github.com/Kuhlman-Lab/ThermoMPNN>). Throughout the text, we refer to these datasets as the Megascaple training (train + valid) and testing sets. We downloaded the structures’ .pdb files from <https://zenodo.org/records/7992926>. We note that the Megascaple training set was not controlled for maximum similarity with the T2837 dataset, and six of the T2837 proteins ($\sim 5\%$) have sequence homologs in the Megascaple training set with sequence similarity above 90%.

Binding effect prediction. For predicting mutational effects on binding affinity, we fine-tuned HERMES on the SKEMPI v2.0 dataset [103], using 3-fold cross-validation.

- *SKEMPI dataset*: After removing duplicate experimental entries, SKEMPI v2.0 contains 5,713 measurements of binding free-energy changes ($\Delta\Delta G^{\text{binding}}$) spanning 331 protein–protein complex structures. We further restrict the dataset to complexes with at least 10 annotated mutations, resulting in 116 structures and 5,025 mutations. Finally, we retain only single-point mutations, yielding 93 structures and 3,485 mutations.

SKEMPI provides metadata explicitly designed to support leakage-aware evaluation via two fields: *hold-out type* and *hold-out proteins* (see SKEMPI documentation: https://life.bsc.es/pid/skempi2/info/faq_and_help). Briefly, *hold-out type* groups complexes into broad categories (e.g., protease-inhibitor, antibody-antigen, and pMHC-TCR), while *hold-out proteins* lists PDB identifiers and/or hold-out types that should be co-held-out to avoid training on closely related binding sites. Using this information, we define three split strategies:

- (i) *Easy*: random splitting without using hold-out metadata.
- (ii) *Medium*: we enforce that all entries linked via a mutation’s *hold-out proteins* annotation are assigned to the same split (but we do not additionally group by *hold-out type*).
- (iii) *Difficult*: we enforce that all complexes sharing the same *hold-out type* are assigned to the same split, producing a stringent evaluation of generalization across interaction classes.

In cases where a complex is associated with multiple hold-out types, we assign it to a single type by randomly selecting one of the available labels.

C.1.7 Baseline models

Here, we describe the baseline models used to benchmark HERMES on mutational effect prediction.

ProteinMPNN [52]. ProteinMPNN is an inverse-folding model that samples amino-acid sequences conditioned on a protein backbone (optionally with a partial sequence fixed). Because it also outputs per-site amino-acid probabilities, we used it to score mutational effects via the log-likelihood ratio in Eq. 5.1. As for HERMES, we evaluate ProteinMPNN models trained with two noise levels: 0.02 Å (virtually no noise) and 0.30 Å. We used, and provide, scripts to infer mutation effects built upon a public fork of the ProteinMPNN repository (<https://github.com/gvisani/ProteinMPNN-copy>).

ThermoMPNN [60]. ThermoMPNN is a thermodynamic stability predictor built on top of ProteinMPNN. For a given structure, it extracts the final ProteinMPNN residue embedding at the site of interest and feeds it to a separate head that predicts per-amino-acid ΔG values, from which $\Delta\Delta G$ is computed. Similar to HERMES, this formulation enforces the permutation symmetry of mutational effects by construction, without requiring data augmentation. For our experiments, we used native functionalities in the ThermoMPNN repository (<https://github.com/Kuhlman-Lab/ThermoMPNN>).

Stability-Oracle [59]. Similar to HERMES, Stability-Oracle is trained in two stages. First, a graph-attention network is pre-trained to predict masked amino acids from their local atomic environment (“neighborhood”). Next, the embeddings from the pre-trained model is used to regress over mutation-effect. Specifically, For a target site on a structure, the masked-neighborhood embedding h is extracted from the pre-trained network and concatenated separately with embeddings of the “from” and “to” amino acids. Each concatenated input is

passed through a transformer to produce amino-acid-specific embeddings $e_{aa_{\text{from}}}$ and $e_{aa_{\text{to}}}$, whose difference ($e_{aa_{\text{to}}} - e_{aa_{\text{from}}}$) is fed to a two-layer MLP to predict $\Delta\Delta G_{aa_{\text{from}} \rightarrow aa_{\text{to}}}$. This construction is permutation-symmetric up to the final MLP, since each e_{aa} is computed independently; symmetry is broken only by the MLP, and would have been preserved by a bias-free linear layer. The original method enforces symmetry during training via $19\times$ data augmentation. We report performance scores calculated using predictions provided in the Stability-Oracle’s repository (<https://github.com/danny305/StabilityOracle>).

RaSP [20]. Similar to HERMES, RaSP is trained in two steps. First, a 3D CNN is pre-trained to predict masked amino acids from their local atomic environment (“neighborhood”). Then, a small fully-connected neural network with a single output is trained to regress over mutation effects, using as input neighborhoods’ embeddings from the 3DCNN, the one-hot encodings of wildtype and mutant amino-acids, and the wildtype and mutant amino-acids’ frequencies in the pre-training data. RaSP is fine-tuned on the stability effect of mutations $\Delta\Delta G$, computationally determined with Rosetta [39]. We report performance scores calculated using predictions provided in the authors’ repository (https://github.com/KULL-Centre/_2022_ML-ddG-Blaabjerg).

RDE-Network [143]. RDE stands for Rotamer Density Estimator. This model consists of first a graph neural network encoder that is trained via a normalizing flow objective to predict distributions of side-chain conformations. Then, a prediction head is added, and it is trained on $\Delta\Delta G_{\text{binding}}$ effects from the SKEMPI dataset. We report performance scores as provided in ref. [143].

MIF-Network [143]. This model’s architecture is the same as the encoder in RDE-Network. It is first pre-trained on the task of wild-type amino acid classification. Then, a prediction head is added to the encoder, and it is trained on $\Delta\Delta G_{\text{binding}}$ effects from the SKEMPI dataset [103] in the same manner as RDE-Network. We report performance scores as provided in Ref. [143].

Pythia-PPI [228]. Pythia-PPI uses a graph neural network encoder pre-trained for wild-type amino acid classification, followed by a $\Delta\Delta G$ prediction module with two heads: one for $\Delta\Delta G_{\text{stability}}$ and one for $\Delta\Delta G_{\text{binding}}$. The model is fine-tuned jointly on stability labels from FireProtDB [223] and binding labels from SKEMPI [103], using a validation selected 20:80 stability:binding mixing ratio. The resulting checkpoint (“Vanilla Pythia-PPI”) is then further trained via self-distillation by fine-tuning on its own predictions over all SKEMPI complex structures to obtain the final model (Pythia-PPI). We should note that these two procedures (i.e., joint fine-tuning on stability and binding with a validation selected mixing ratio and self-distillation) could also be applied to HERMES to potentially improve performance. We report performance scores as provided in ref. [228].

C.1.8 ESMFold for computational modeling of protein structures

For the analysis in Fig. C.1.8, we used the ESM Metagenomic Atlas API to fold each sequence individually (<https://esmatlas.com/resources?action=fold>).

C.1.9 Structure-conditioned reversibility analysis

In Figs. 5.5 and C.1.13, we characterized structure-conditioned reversibility on the SSym dataset [181], which contains wild-type and single-mutant structures for 352 mutations across 19 proteins [172].

We assess structure-conditioned reversibility by whether the effects of forward $\alpha \rightarrow \beta$ and reverse $\beta \rightarrow \alpha$ mutations are equal, using the outgoing structural context to compute mutational effects. Specifically, the forward mutational effect $\Delta\hat{F}_{\alpha \rightarrow \beta}^{(\text{model})}$ is computed by conditioning on the structure X_α , and the reverse effect

$\Delta \hat{F}_{\beta \rightarrow \alpha}^{(\text{model})}$ is computed using X_β ,

$$\begin{aligned}\Delta \hat{F}_{\alpha \rightarrow \beta}^{(\text{model})} &= \log P(\beta|X_\alpha) - \log P(\alpha|X_\alpha) \\ \Delta \hat{F}_{\beta \rightarrow \alpha}^{(\text{model})} &= \log P(\alpha|X_\beta) - \log P(\beta|X_\beta)\end{aligned}\tag{C.1.2}$$

With this definition, we do not expect reversibility, as forward and reverse transitions are conditioned on different structures.

Reversibility score. For the analysis in Figs. 5.5 and C.1.13 we construct a reversibility score as a mean squared error normalized to be between -1 and 1,

$$\text{rev. score} = 1 - \frac{\text{mean}((\Delta \log p_{fwd} + \Delta \log p_{rev})^2)}{\text{mean}(\Delta \log p_{fwd}^2) + \text{mean}(\Delta \log p_{rev}^2)}\tag{C.1.3}$$

where $\Delta \log p_{fwd} = \log p(\text{mt}|X_{\text{wt}}) - \log p(\text{wt}|X_{\text{wt}})$ is the forward mutational effect computed by conditioning on the wild type structure, and $\Delta \log p_{rev} = \log p(\text{wt}|X_{\text{mt}}) - \log p(\text{mt}|X_{\text{mt}})$ is the reverse mutational effect computed by conditioning on the mutants structure. The averages (mean) are computed over the 352 mutations in the SSym dataset, or stratified as needed by the desired criteria. With this definition, a larger value of **rev. score** implies more degree of reversibility between forward and reverse mutations.

Sequence similarity calculation between the Ssym dataset and pre-training proteins. We stratified the data based on their similarity to the training data. To do so, we used BLASTp via NCBI BLAST+ [35] with individual chains in the SSym structures as queries [181], and individual chains in the pre-training set as the database; for each SSym chain, we then considered its maximum similarity to any sequence in the pre-training set. We found that 9 Ssym proteins have similarity below 70% to the pre-training proteins (only 5 are below 50%, none are below 40%), and 10 proteins have similarity above 70% (comprising the two panels in Fig. 5.5B).

C.1.10 Statistical comparison of model performances on classification metrics

Comparing models' performances on the same tasks. Figures 5.2 and 5.3 report classification performance (precision, recall, F1, etc.) for predicting the stability effects of mutations across models. Figures C.1.5 and C.1.3 report pairwise significance tests (p-values) for differences in these metrics using permutation tests, with the null hypothesis that two models' predictions are exchangeable. To compute these p-values, for each model pair, we generated permuted prediction sets by swapping paired prediction between models with probability 0.5, and repeat this procedure for 1000 random seeds. The p-value is the fraction of permutations in which the metric difference between the permuted sets is at least as large ($\geq$) as the observed difference. We correct the computed p-values for multiple comparisons using Holm–Bonferroni across all model pairs within each metric (i.e., within each panel in Figs. C.1.5, C.1.3).

Comparing a model's performance across tasks. Figure 5.3 reports classification performance (precision, recall, F1, etc.) for predicting mutation stability effects across mutation size categories (tasks). Figure C.1.4 reports, for each model, pairwise significance tests for differences in these metrics between tasks. We estimate p-values via bootstrap resampling: for each task pair, we repeatedly (1000 seeds) draw bootstrap samples of predictions within each task, recompute the metric difference, and define the p-value as the fraction of bootstrap replicates in which the resampled metric difference is at least as large ($\geq$) as the observed metric

difference. We apply Holm–Bonferroni correction across all task pairs within each metric.

Significance of the number of retrieved antigen-stabilizing mutations by different models. In Figures 5.7, C.1.16 and Table C.1.4, we report, for each model, the number of antigen-stabilizing mutations retrieved (x) out of a total of $N = 33$ experimentally verified such mutations. Retrieval is based on amino-acid ranking: a mutation is counted as retrieved if (i) the mutant amino acid is ranked better than the wild type ($r_{\text{mt}} < r_{\text{wt}}$), and (ii) that its rank is below or equal to a certain threshold R , with $R = 3$ for “strongly suggested” and $R = 6$ for “moderately suggested.” We assess significance in the number of retrieved antigen-stabilizing mutations by a given model with a one-sided binomial test: for each model, the p-value is the probability under a null model of recovering at least x successes in N trials with per-mutation success probability p_{null} , i.e., $p = 1 - \text{BinomCDF}(x - 1, N, p_{\text{null}})$. We adjust p-values for multiple testing using Holm–Bonferroni, and report the resulting p-values in Figs. C.1.15 and C.1.15.

To compute the p-values, we consider two null models:

- (i) **Random null.** For each mutation, we sample ranks for the mutant and wild type uniformly without replacement: $r_{\text{mt}} \sim \text{Unif}\{1, \dots, 20\}$ and $r_{\text{wt}} \sim \text{Unif}\{1, \dots, 20\} \setminus r_{\text{mt}}$. A mutation is a “success” if $r_{\text{mt}} < r_{\text{wt}}$ and $r_{\text{mt}} \leq R$, which occurs with probability $p_{\text{null}} = \frac{1}{19 \times 20} \sum_{i=1}^R (20 - i)$.
- (ii) **BLOSUM62 null.** We simply set $p_{\text{null}} = x_{B62}/N$, where x_{B62} is the number of mutations with BLOSUM62 rank $r_{\text{mt}}^{B62} \leq R$. We note that the rank of the wild-type is always 1 for BLOSUM62, so we omit the condition that the rank of the mutant has to be lower than the rank of the wild-type, which is instead applied to all other models, including the random null model.

C.1.11 Using Rosetta to score antigen-stabilizing mutations

In Figures 5.7, C.1.16, C.1.14 and Tables C.1.4, C.1.5, we reported the Rosetta scores for verified antigen-stabilizing mutations. In Figures 5.8 and C.1.17 we used Rosetta scores as oracles for mutations suggested by other models. To compute these scores, we used the PyRosetta software [39] to model protein structures, with wild-type structures serving as template. For each target sequence containing a single point mutation, we threaded the mutant sequence onto all chains of the oligomeric template. Each resulting model was then subjected to a high-resolution refinement protocol using Rosetta’s FastRelax application. This protocol involved five cycles of side-chain rotamer repacking followed by gradient-based energy minimization of backbone (ϕ, ψ) and side-chain (χ) torsion angles, guided by the ref2015_cart all-atom energy function. To improve conformational sampling, we repeated the threading-and-relaxation procedure 20 times per mutation for use in the analysis against validated mutations, and 50 when used as an oracle. For each mutant, we report the mean Rosetta Energy Unit (REU) score of the 20% lowest-energy (most favorable) relaxed models (5 for the analysis against validated mutations, 13 when used as an oracle).

Supplementary Information

Extended analysis of antigen stabilization

Here, we present a more detailed analysis of our antigen stabilization predictions from Figures 5.7, C.1.16 and Table C.1.4. For each antigen, we contextualize the candidate mutations by their local structural environment and examine the characteristics and physico-chemical features of the HERMES-predicted top-ranked antigen-stabilizing substitutions (Fig. C.1.14). This analysis is intended to help practitioners incorporate these models into design workflows and to motivate quantitative, context-dependent success

criteria for real-world stabilization tasks. As demonstrated in the main text (Figs. 5.7, C.1.16), we find that comparing the rank of the wild-type residue to that of putative stabilizing substitutions is an informative diagnostic of model behavior and performance.

RSV-F. RSV-F Cav1 stabilizing mutations S190F and V207L are well-predicted by different HERMES models (the stability-fine-tuned models trained on +cDNA117k and +Megascale, and HERMES-*amortized*), whereas Rosetta and ProteinMPNN struggle to identify these mutations. We therefore focus on these two residues to analyze, post hoc, the structural context underlying HERMES' predictions, with the goal of clarifying which classes of stabilizing mutations HERMES is best suited to characterize.

Both S190F and V207L enhance hydrophobic packing in an underpacked region near the trimer apex (Fig. C.1.20). V207L is a significantly more anticipated mutation than S190F, indicated by a positive BLOSUM62 score; this is reflected in HERMES assigning a higher rank to the wild-type Val207 relative to the mutant Leu (Fig. C.1.14 and Table C.1.4). Notably, the highest-ranking residue predicted by HERMES models at position 207 is Ile (Fig. C.1.14). Inspection of the native wild-type structure (PDB ID: 4JHW [147]) suggests that an Ile mutation would pack exceptionally well in the hydrophobic pocket surrounding residue 207, suggesting V207I might outperform the identified V207L mutation (Fig. C.1.20A).

At position 190, the top-ranked substitutions from the wild-type Ser are Val or Ile. Structural examination suggests both residues can be accommodated within the existing 4JHW structure pocket without requiring the repacking of surrounding residues (Fig. C.1.20B). Conversely, the known stabilizing mutation, Phe, appears to slightly overpack the region in the 4JHW crystal structure (Fig. C.1.20B). While the HERMES-*fixed* model does not strongly prioritize Phe, both HERMES stability-fine-tuned models rank Phe third, behind the smaller Val and Ile. This indicates that fine-tuning on $\Delta\Delta G$ datasets enhances implicit reasoning regarding local repacking and structural relation possibilities, particularly for larger residues. The HERMES-*amortized* model also outperforms the HERMES-*fixed* model, likely due to the model's de-emphasis on strict steric constraints (Table C.1.4).

Universal-HA. We observe robust performance in recovering Universal-HA pH switch mutations across all tested models. All three mutations carry "unanticipated" BLOSUM62 scores.

The mutational-effect heatmaps (Fig. C.1.14) show that, for HA, ProteinMPNN's preferences are narrowly concentrated, strongly favoring the reported stabilizing substitutions, whereas HERMES-*amortized*, HERMES-*fixed* (0.50 + Megascale), and Rosetta yield substantially broader preference profiles. For H355W, all HERMES models correctly indicate that the wild-type His is disfavored, but they rank other bulky aromatics (Tyr or Phe) above the stabilizing Trp. Despite being a known stabilizing mutation, Trp does not appear to fit in the 7VDF crystal structure without clashes (Fig. C.1.21). This discrepancy may stem from the limitations of rigid-body mutagenesis (e.g., PyMOL's wizard), as subtle backbone movements are likely required to accommodate the mutation. This finding cautions against over-interpreting apparent clashes in a single experimentally determined static crystal structure, especially for bulky substitutions, and motivates follow-up evaluation with relaxation-aware structure prediction models or physics-based repacking/scoring tools to capture local flexibility and repacking that may render seemingly sterically forbidden mutations feasible.

hMPV-F. We first analyze stabilizing mutations in the M-104 hMPV variant [80]. The A159L substitution requires the nearby Ile-137 to adopt an alternative rotamer [80]. The two $\Delta\Delta G$ -fine-tuned HERMES models prioritize Val (V) and Ile (I) over Leu (L) at position 159. We note that mutation to either Val or Ile at position 159 would require both Ile-137 and Leu-141 to adopt different rotamers to accommodate their beta-branched side chains (Fig. C.1.22C). Nonetheless, A159V and A159I remain plausible stabilizing mutations and therefore, warrant experimental screening. ProteinMPNN identifies Ala as the most favorable

residue at this position, indicating limited sensitivity to larger hydrophobic substitutions. By comparison, the $\Delta\Delta G$ -fine-tuned HERMES models rank the wild-type Ala substantially lower (ordinal rank 5–7), while consistently favoring larger hydrophobic residues. Low rankings for buried hydrophobic positions may therefore indicate suboptimal core packing, particularly when alternative residues with greater side-chain volume are predicted. Although Leu is not the top-ranked substitution, the $\Delta\Delta G$ -fine-tuned HERMES models correctly capture the preference for a residue larger than Ala to improve packing within this pocket.

The V203I substitution was recovered by nearly all models. A Val to Ile substitution is highly anticipated, with a BLOSUM62 score of 3. Notably, the $\Delta\Delta G$ -fine-tuned HERMES models rank Ile as more favorable than the wild-type Val, whereas ProteinMPNN, ThermoMPNN, HERMES-*fixed*, and HERMES-*amortized* prefer the wild-type Val. In accordance with this mutation being more highly anticipated, it is likely that the pocket occupied by V203 is less underpacked relative to A159L in the M-104 variant, with the wild-type residue still being quite highly-favored. The subtle nature of Val to Ile mutations may result in less signal overall when comparing amino acid ranks as we do in this work.

The V449D substitution replaces a surface-exposed hydrophobic residue with a polar side chain and introduces hydrogen-bonding contacts to Asn298 (Fig. C.1.22A). ProteinMPNN, ThermoMPNN, and all HERMES variants instead rank Glu as the top substitution; structural inspection suggests that Glu could form similar hydrogen bonds, and may therefore also be stabilizing (Fig. C.1.22A). Overall, the models performed well in identifying mutations that enable additional hydrogen bonding.

For V430Q, the predicted $\Delta\Delta G$ reported in the original study is small in magnitude (albeit negative), suggesting that it should not be classified as strongly stabilizing, as suggested in the original study [80]. Consistent with this, neither of the $\Delta\Delta G$ -fine-tuned HERMES models prioritize V430Q. Interestingly, the only model that ranks V430Q as the top substitution is HERMES-*fixed*.

In the MPV-2c variant [15], recovery of V112R depends on providing the model with the trimeric PDB. The introduced Arg forms interprotomer van der Waals contacts and intraprotomer hydrogen bonds that stabilize the prefusion trimer (Fig. C.1.22B). Because this mechanism is inherently multimeric and not captured by a strictly local energetic proxy, it may explain why the $\Delta\Delta G$ -fine-tuned HERMES models do not prioritize V112R. In contrast, HERMES-*amortized* ranks Arg second, slightly favoring the more conservative Leu, which could plausibly adjust hydrophobic interprotomer contacts.

D209E is correctly predicted by ProteinMPNN, HERMES-*fixed*, and HERMES-*amortized*, whereas the $\Delta\Delta G$ -fine-tuned HERMES models perform poorly in proposing this mutation. In particular, the $\Delta\Delta G$ -fine-tuned models rank bulky hydrophobics (Leu, Met, Ile, Val) ahead of the charge-conserving Glu. Although the charged substitution D209E plausibly stabilizes the prefusion state via favorable polar interactions, the stabilizing potential of these alternative hydrophobic substitutions remains untested.

E453P replaces a glutamate with proline, a substitution that can be strongly stabilizing by rigidifying locally flexible regions. Among the evaluated methods, only HERMES-*amortized* predicts this mutation. ProteinMPNN ranks the wild-type Glu as most favored and Pro as most disfavored (rank 1 vs. 20), while ThermoMPNN partially recovers the substitution but still favors the native Glu. Consistent with these mixed signals, the original study [15] reports that E453P improves stability but reduces expression, and that E453Q yields weaker stabilization with improved expression. This highlights the poorly understood coupling between stability and expression and suggests that some models may systematically downweigh proline substitutions, potentially because prolines are underrepresented in natural sequences, and they appropriately learned to rarely recommend them. The fact that HERMES-*amortized*—but not HERMES-*fixed*—recovers E453P suggests that modeling relaxed neighborhoods helps identify sites that can accommodate Pro's constrained Ramachandran geometry. Finally, we suspect that the utility and stabilizing ability of proline substitutions in antigen design is not fully captured by HERMES when fine-tuned with the cDNA117k or Megascale datasets,

as the proteins represented in those datasets are much smaller, and lack the interplay of conformational switching between prefusion and post-fusion.

DENV-E. We observe mixed performance on DENV-E stabilizing mutations. S29K is recovered only by ProteinMPNN and ThermoMPNN. Although S29K, T33V, and A35M (“PM4” mutations as a group) are spatially proximal and may act synergistically, they were originally identified by Rosetta site-saturation mutagenesis, suggesting they should be accessible to single point-mutation scoring schemes [177]. In the 1OAN structure [158], introducing Lys at position 29 appears to create substantial steric clashes across rotamers (Fig. C.1.23A). S29K and A35M are not anticipated by BLOSUM62, whereas T33V is neutral according to BLOSUM62 (score of 0, Table C.1.4) but is strongly preferred by all models tested (Fig. 5.7 and Table C.1.4). Structurally, T33V lines a hydrophobic pocket, making the isosteric Val substitution easier to predict. Notably, the PM4 mutations are not present in the best-available SC12 structure (PDB: 6WY1) [121].

A major challenge in stabilizing DENV-E is strengthening homodimer interactions. A259W and T262R were mutations made at the dimer interface with the intention of enhancing the strength of the homodimer. A259W is highly unanticipated (BLOSUM62 = -3) but, together with T262R, can form a favorable cation- π interaction while improving packing against the neighboring protomer (Fig. C.1.16A.4). Notably, the HERMES model fine-tuned on Megascale favors A259W despite its apparent synergy with T262R; this could reflect sensitivity to the interprotomer packing geometry, although a chance effect cannot be excluded. We also expect the generally weak recovery of T262R across models to improve when Trp is present at position 259, consistent with the coupled nature of this interface motif.

Like E453P in hMPV-F, the proline substitution T280P is recovered only by HERMES-*amortized*, further highlighting the model’s ability to reason about proline accommodation. F279W fills an underpacked region. However, comparison of the stabilized 6WY1 structure with the native 1OAN crystal structure suggests that backbone movement (residues 269–281) is required to optimally fit Trp, potentially aided by T280P (Fig. C.1.23B). ThermoMPNN correctly reasons regarding the Trp introduction, and HERMES-*fixed* + Megascale ranks Trp second. Other stability-fine-tuned HERMES models prefer Phe, Ile, or Leu, likely adhering more strictly to the steric limitations of the 1OAN backbone.

We observe strong reasoning across almost all HERMES models, ProteinMPNN, and Rosetta in identifying G106D, which introduces stabilizing polar and electrostatic contacts (Fig. C.1.23C). Curiously, ThermoMPNN struggles with this specific mutation.

SARS-CoV-2 Spike. The spike protein of SARS-CoV-2 was initially stabilized through the introduction of two proline mutations (“S-2P”), and subsequent work identified four additional prolines that further improve stability and expression of the full-length spike [96]. Here we evaluate recovery of these four additional proline sites (excluding the original S-2P mutations). Consistent with its strong performance on proline substitutions, HERMES-*amortized* ranks Pro as the top choice at all four positions. ProteinMPNN also performs well, recovering Pro at 3/4 sites.

Summary. Our analysis across multiple antigens reveals several consistent themes regarding model performance and decision-making logic.

First, we observe distinct behaviors regarding hydrophobic packing. The HERMES models fine-tuned on stability datasets consistently demonstrate enhanced reasoning for hydrophobic core packing. These models frequently suggest larger hydrophobic residues (e.g., Ile or Phe) to fill underpacked cavities where models without explicit stability training might prefer the native residue or smaller conservative substitutions. However, this sensitivity can sometimes lead to “confusion” among similar hydrophobic amino acids (e.g., Val vs. Ile vs. Leu), where the models correctly identify the chemical property needed but may rank several

hydrophobic options similarly.

Second, the interplay between structural rigidity and model flexibility is evident in the prediction of proline mutations. HERMES-*amortized* consistently outperforms other models in identifying stabilizing proline substitutions. This suggests that the model's training, which incorporates relaxed neighborhoods, allows it to better identify backbone locations capable of accommodating the steric constraints of proline, whereas other methods more often penalize these stabilizing mutations due to perceived steric incompatibilities of the provided protein structure.

Third, we note differences in the mutational landscape profiles predicted by different architectures (Fig. C.1.14). ProteinMPNN tends to produce narrow substitution profiles, often assigning very low probabilities to non-native residues unless the signal is overwhelmingly strong. In contrast, HERMES models—particularly HERMES-*amortized* and those fine-tuned on Megascale stability data—exhibit broader mutational profiles. This broader landscape may be more advantageous for design applications, as it provides a richer set of plausible candidate hypotheses for experimental validation and may guide a practitioner's intuition regarding the nature or predicted effects of various substitutions in a more granular manner.

Finally, while ProteinMPNN and Rosetta remain powerful tools for sequence recovery, the HERMES-*amortized* and stability-fine-tuned models demonstrate a unique capacity to prioritize mutations that improve local packing density even when such mutations appear sterically challenging in the provided structure. This highlights the utility of using an ensemble of models to capture different modes of stabilization, from electrostatic optimization to hydrophobic core repacking and backbone rigidification.

Model	Accuracy	
	Pyrosetta pre-processing	Biopython pre-processing
HERMES- <i>fixed</i> 0.00	0.73	0.75
HERMES- <i>fixed</i> 0.50	0.64	0.65
HERMES- <i>amortized</i> 0.00	0.55	0.47
HERMES- <i>amortized</i> 0.50	0.50	0.44
HERMES- <i>fixed</i> 0.00 + Ros	0.41	0.40
HERMES- <i>fixed</i> 0.50 + Ros	0.38	0.37
HERMES- <i>fixed</i> 0.00 + cDNA117k	0.47	0.45
HERMES- <i>fixed</i> 0.50 + cDNA117k	0.39	0.38
HERMES- <i>amortized</i> 0.00 + cDNA117k	0.37	-
HERMES- <i>amortized</i> 0.50 + cDNA117k	0.34	-
HERMES- <i>fixed</i> 0.00 + cDNA117k train ESMFold	0.46	0.49
HERMES- <i>fixed</i> 0.50 + cDNA117k train ESMFold	0.40	0.40
HERMES- <i>fixed</i> Untr. 0.00 + cDNA117k	0.09	-
HERMES- <i>fixed</i> Untr. 0.50 + cDNA117k	0.08	-

Table C.1.1: Accuracy of HERMES models on wildtype amino-acid classification on all sites across 40 CASP12 test proteins. Accuracy is defined as proportion of sites for which the wild-type amino acid is predicted with the highest probability among all 20 canonical amino acids. Model names indicate the architecture, the coordinate-noise amplitude used, and when applicable, the fine-tuning dataset (listed after “+”); *Untr.* is short for *Untrained*, indicating models that had no pre-training and were instead only trained on stability effects. Accuracy is reported for the two pre-processing schemes (with PyRosetta and Biopython) used in HERMES.

Model	hh:mm:ss
HERMES- <i>fixed</i> 0.50	00:11:23
HERMES- <i>amortized</i> 0.50	00:11:23
HERMES- <i>relaxed</i> 0.50	12:13:20

Table C.1.2: Inference speed of HERMES models on the T2837 dataset. Runtimes are reported in hours (hh), minutes (mm), and seconds (ss) for inference on the T2837 dataset, which comprises 2837 mutation effects across 129 proteins. For HERMES-*fixed* and HERMES-*amortized*, the script ‘mutation_effect_prediction_with_hermes.py’ was used; for HERMES-*relaxed*, the script ‘mutation_effect_prediction_with_hermes_with_relaxation.py’ was used. Both scripts, along with the dataset ‘csv’ file, are available in our GitHub repository. All models were executed using a single CPU and a single A40 GPU.

Category	Description
Hydrophobic Property	1. Retention coefficient in TFA
	2. Free energy of solution in water
	3. Solvation free energy
	4. Melting point
	5. Number of hydrogen-bond donors
	6. Number of full nonbonding orbitals
	7. Partition energy
	8. Hydration number
	9. Retention coefficient in high performance liquid chromatography (HPLC), pH 7.4
	10. Retention coefficient in HPLC, pH 2.1
	11. Partition coefficient in thin-layer chromatography
	12. Retention coefficient at pH 2
	13. R_f for 1-N-(4-nitrobenzofurazono)-amino acids in ethyl acetate/pyridine/water
	14. ΔG of transfer from organic solvent to water
	15. Hydration potential or free energy of transfer from vapor phase to water
	16. R_f , salt chromatography
	17. $\log D$, partition coefficient at pH 7.1 for acetamide derivatives of amino acids in octanol/water
	18. $\Delta G = RT \log f$, f = fraction buried/accessible amino acids in 22 proteins
Steric Property	19. Average volume of buried residue
	20. Residue accessible surface area in tripeptide
	21. Graph shape index
	22. Normalized van der Waals volume
	23. STERIMOL length of the side chain
	24. STERIMOL minimum width of the side chain
	25. STERIMOL maximum width of the side chain
	26. Average accessible surface area
	27. Distance between C_α and centroid of side chain
	28. Side-chain angle θ
	29. Side-chain torsion angle ϕ
	30. Radius of gyration of side chain
	31. Van der Waals parameter R_0
	32. Van der Waals parameter ε
	33. Refractivity
Electronic Property	34. Value of θ (i)
	35. Substituent van der Waals volume
	36. α CH chemical shifts
	37. α NH chemical shifts
	38. A parameter of charge transfer capability
	39. A parameter of charge transfer donor capability
	40. Nuclear magnetic resonance (NMR) chemical shift of α carbon
	41. Localized electrical effect
	42. Positive charge
	43. Negative charge
	44. Polarity
	45. Net charge
	46. Amphipathicity index
	47. Isoelectric point
	48. Electron-ion interaction potential values
	49. pK_{NH_2} (NH_2 on C_α)
	50. pK_{COOH} ($COOH$ on C_α)

Table C.1.3: **Table of amino-acid properties used for comparison with substitution matrices.** Values associated with each amino acid are listed in ref. [150].

antigen PDB id	stabilized name	mutation	BLOSUM62 score ; rank	Rosetta	Protein MPNN 0.30	Thermo MPNN + Megascale	HERMES -fixed 0.50	HERMES -amortized 0.50	HERMES -fixed 0.50 + cDNA117k	HERMES -fixed 0.50 + Megascale
				$r_{wt} \rightarrow r_{mt}$	$r_{wt} \rightarrow r_{mt}$	$r_{wt} \rightarrow r_{mt}$	$r_{wt} \rightarrow r_{mt}$	$r_{wt} \rightarrow r_{mt}$	$r_{wt} \rightarrow r_{mt}$	$r_{wt} \rightarrow r_{mt}$
RSV-F 4JHW	Cav1 [147]	S190F V207L	-2 ; 16 1 ; 3	19 → 11 9 → 4	4 → 7 2 → 7	14 → 3 6 → 5	4 → 8 2 → 3	11 → 4 3 → 2	12 → 3 3 → 2	13 → 3 3 → 2
		Uncl. [123]	S215P	-1 ; 13	13 → 1	1 → 14	10 → 20	10 → 15	6 → 3	16 → 19
	TriC [147]	D486H	-1 ; 7	9 → 2	1 → 10	1 → 8	5 → 4	10 → 4	20 → 9	18 → 9
		E487Q	2 ; 3	2 → 9	1 → 13	1 → 12	2 → 3	4 → 5	11 → 12	14 → 12
		F488W D489H	1 ; 3 -1 ; 7	15 → 17 5 → 2	1 → 14 3 → 14	3 → 1 5 → 12	2 → 6 11 → 8	1 → 13 18 → 15	1 → 3 20 → 11	1 → 2 17 → 12
HA 7VDF	Universal-HA [154]	H355W K380I	-2 ; 12 -3 ; 19	5 → 1 17 → 1	5 → 1 9 → 1	4 → 2 11 → 1	4 → 3 8 → 4	7 → 2 14 → 3	5 → 3 13 → 2	8 → 3 17 → 2
		E432I	-3 ; 19	14 → 9	12 → 2	13 → 2	9 → 4	15 → 5	12 → 3	14 → 3
		hMPV-F 5WB0	M104 [80]	L130D	-4 ; 19	15 → 3 [†]	8 → 9	4 → 10	5 → 9	12 → 2
A159L	-1 ; 10			6 → 1 [†]	1 → 12	3 → 5	1 → 7	2 → 4	7 → 3	5 → 3
V203I	3 ; 2			4 → 3 [†]	1 → 2	1 → 2	1 → 2	2 → 1	3 → 1	2 → 1
V430Q	-2 ; 10			12 → 3 [†]	5 → 4	12 → 6	8 → 1	10 → 5	5 → 12	10 → 8
V449D	-3 ; 16			13 → 8 [†]	13 → 3	16 → 2	11 → 2	15 → 2	18 → 2	19 → 6
	MPV-2cREKR [15]	V112R	-3 ; 19	6 → 13 [†]	8 → 1	2 → 1	2 → 9	8 → 3	4 → 11	5 → 10
		D209E	2 ; 2	8 → 1 [†]	11 → 1	12 → 4	3 → 1	14 → 3	15 → 6	16 → 7
		V231I	3 ; 2	3 → 1 [†]	3 → 1	3 → 1	2 → 1	2 → 1	3 → 1	3 → 1
		E453P	-1 ; 10	10 → 2 [†]	1 → 20	1 → 2	8 → 19	13 → 1	17 → 16	19 → 6
	Uncl. [123]	E80D V155P	2 ; 2 -2 ; 13	18 → 13 10 → 20	3 → 1 4 → 20	13 → 18 1 → 20	1 → 12 1 → 19	7 → 17 1 → 20	15 → 19 2 → 20	14 → 18 4 → 20
DENV-E 1OAN	SC12 [177]	S29K	0 ; 6	4 → 3 [†]	3 → 1	7 → 2	1 → 12	2 → 5	6 → 12	9 → 14
		T33V	0 ; 3	9 → 4 [†]	3 → 1	6 → 1	3 → 1	5 → 1	10 → 1	10 → 1
		A35M	-1 ; 13	11 → 4 [†]	6 → 1	10 → 8	1 → 12	1 → 11	1 → 7	6 → 7
		G106D	-1 ; 5	18 → 1 [†]	11 → 1	6 → 11	8 → 1	6 → 2	20 → 1	18 → 1
		A259W	-3 ; 20	7 → 1 [†]	1 → 3	4 → 11	1 → 20	1 → 13	1 → 2	2 → 1
		T262R	-1 ; 11	15 → 3 [†]	1 → 15	18 → 7	3 → 14	10 → 3	16 → 3	17 → 4
		F279W	1 ; 3	3 → 1 [†]	4 → 11	4 → 1	1 → 6	3 → 7	1 → 4	1 → 2
		T280P	-1 ; 13	12 → 2 [†]	7 → 10	9 → 14	4 → 19	6 → 2	2 → 20	8 → 19
SARS-Cov-2 6VSB	hexapro [96]	F817P	-4 ; 20	3 → 1	16 → 1	1 → 12	2 → 16	3 → 1	2 → 18	2 → 19
		A892P	-1 ; 14	16 → 3	5 → 1	2 → 4	2 → 1	4 → 1	4 → 1	5 → 1
		A899P	-1 ; 14	8 → 19	6 → 8	7 → 18	2 → 6	4 → 1	8 → 15	5 → 17
		A942P	-1 ; 14	5 → 12	3 → 1	2 → 8	1 → 4	2 → 1	2 → 1	2 → 1
	proportion of correctly and strongly suggested mutations			20/33	15/33	11/33	8/33	19/33	15/33	13/33
	proportion of correctly and at least moderately suggested mutations			23/33	16/33	14/33	11/33	23/33	16/33	17/33
	proportion of correctly and at least weakly suggested mutations			27/33	16/33	16/33	12/33	24/33	19/33	22/33

Table C.1.4: Predicting antigen-stabilizing mutations with HERMES: extended results. Recall for different models (columns) is evaluated on 33 previously reported antigen-stabilizing mutations (rows) spanning five viral antigens. For each antigen, we list the PDB structure used for scoring and the publication(s) that originally reported the mutation. Mutations are specified as wild-type→mutant substitutions at the annotated site. Seven models are compared (columns). We additionally report the BLOSUM62 substitution score for each mutation and the mutant’s rank among the 20 possible amino-acid substitutions for the wild-type residue (per BLOSUM62). For each model and mutation, predicted ranks of the wild-type and mutant amino acids are shown as $r_{wt} \rightarrow r_{mt}$. Dagger symbols (†) indicate mutations originally proposed as stabilizing by Rosetta-based pipelines in the source reference. When a model ranks the mutant better than the wild type ($r_{mt} < r_{wt}$), the cell is shaded by the prediction strength based on the value of r_{mt} : dark green, strongly suggested ($r_{mt} \leq 3$); light green, moderately suggested ($4 \leq r_{mt} \leq 6$); light yellow, weakly suggested ($r_{mt} \geq 6$). Column summaries report counts of strongly, at least moderately, and at least weakly suggested mutations (out of 33); **bold** indicates significance (p-value < 0.05) for the number of recalled mutations relative to a random null model (see Fig. C.1.15 for p-values and Methods for details.) All structures were scored in their native multimeric states, generating symmetric partners when needed. ThermoMPNN’s native mode predicts mutation effects only for monomers, ignoring multimeric assemblies even when present in the input structure. “Uncl.” stands for uncleaved prefusion-closed state.

pdbID	# of monomer sites	HERMES GPU	HERMES CPU	Rosetta	
		[seconds] all sites	[seconds] all sites	[CPU-hours] one site	[CPU-hours] all sites
4JHW	449	57	112	150	67,350
7VDF	485	64	118	164	79,540
5WB0	442	43	154	151	66,742
1OAN	394	32	151	59	23,246
6VSB	968	69	265	156	151,008

Table C.1.5: Execution times for saturation mutagenesis predictions on the viral antigens considered in this study. Executions times (in seconds) of HERMES apply to HERMES-*fixed* and HERMES-*amortized* models, regardless of whether zero-shot or fine-tuned. Times were computed when running the script `run_hermes_on_pdbfiles.py` providing as input the pdbfile as well as a single monomeric chain. A single CPU core with 64GB of memory was used, and a NVIDIA A40 GPU when applicable. For Rosetta, we computed times (in CPU-hours) for a single CPU core with 4 GBs of memory, and averaging 10 relaxation instances, which we consider the minimum number of instances for robust results. Times for all sites in the structure were extrapolated by multiplying the calculated average time for a single mutation by the number of monomeric sites.

antigen PDB id	stabilized name	mutation	mutation type	is synergistic	notes
RSV-F 4JHW	Cav1 [147]	S190F	cavity-filling	False	
		V207L	cavity-filling	False	
	Uncl. [123]	S215P	proline	False	
	TriC [147]	D486H	electrostatic	True	
		E487Q	electrostatic	True	
F488W		cavity-filling	True		
		D489H	electrostatic	True	
HA 7VDF	Universal-HA [154]	H355W	cavity-filling	False	
		K380I	cavity-filling	False	
		E432I	cavity-filling	False	
hMPV-F 5WB0	M104 [80]	L130D	electrostatic	False	
		A159L	cavity-filling	False	
		V203I	cavity-filling	False	
		V430Q	electrostatic	False	
		V449D	electrostatic	False	
	MPV-2cREKR [15]	V112R	electrostatic	False	same charge, slightly different size: unclear
		D209E		False	
		V231I	cavity-filling	False	
		E453P	proline	False	
	Uncl. [123]	E80D		False	same charge, slightly different size: unclear
V155P	proline	False			
DENV-E 1OAN	SC12 [177]	S29K	electrostatic	False	
		T33V	cavity-filling	False	
		A35M	cavity-filling	False	
		G106D	electrostatic	False	
		A259W	cavity-filling	True	
		T262R	electrostatic	True	
		F279W	cavity-filling	False	
		T280P	proline	False	
SARS-Cov-2 6VSB	hexapro [96]	F817P	proline	False	
		A892P	proline	False	
		A899P	proline	False	
		A942P	proline	False	

Table C.1.6: Characteristics of antigen-stabilizing mutations. Hand-curated mutation types are listed for antigen-stabilizing mutations reported in Fig. 5.7 and Table C.1.4. Cavity-filling mutations are defined as substitutions to hydrophobic residues that are larger than the wild-type when the wild-type is also hydrophobic. Electrostatic mutations are substitutions that change the residue's net charge. Proline mutations correspond to substitutions to proline. Synergistic mutations were identified through structural reasoning based on the spatial arrangement of mutations within the corresponding structure; see ref. [32] for a breakdown of mutation types considered in the structure-based vaccine design literature. "Uncl." stands for "Uncleaved Prefusion-Closed".

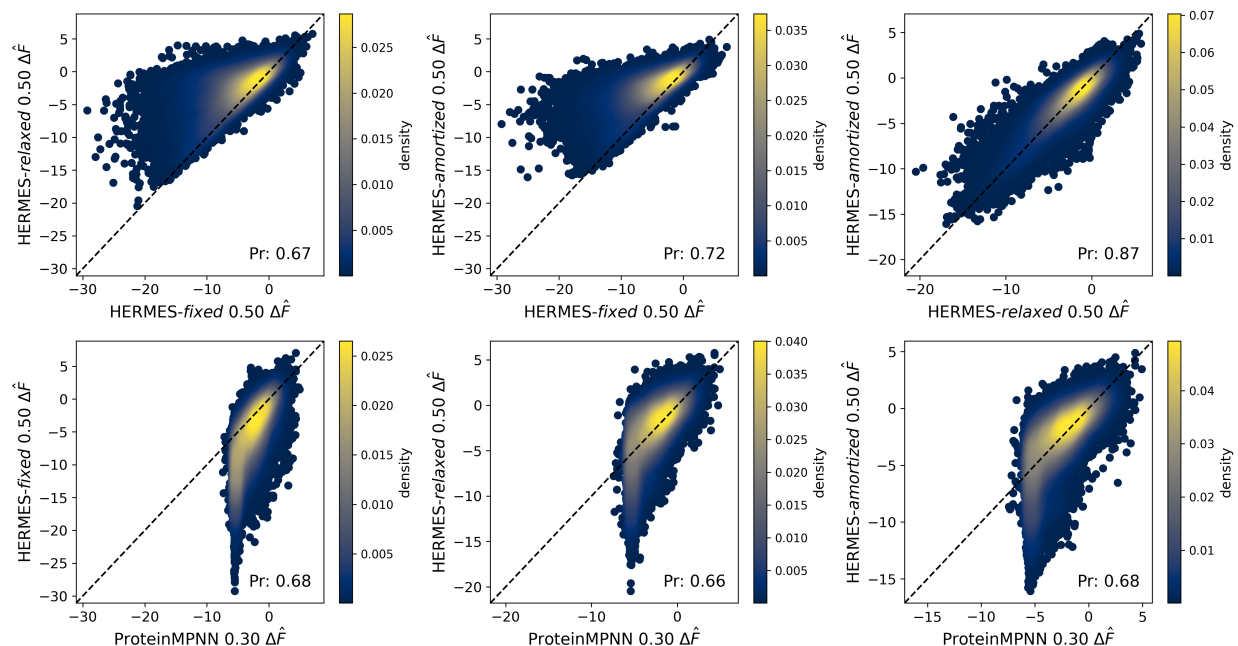


Figure C.1.1: Comparison of zero-shot model predictions on the Megascale test set. For each substitution in the Megascale test set, the predicted change in amino acid propensity upon mutation ($\delta \log p$) is compared between two models in each panel. Color indicates local point density (blue denotes low density and yellow denotes high density). The reported “Pr” in each panel corresponds to the Pearson correlation coefficient between the predictions of the model pair. Model names indicate the architecture and the coordinate-noise amplitude used. $\Delta \hat{F}$ indicates the model’s prediction, following Equations 5.1 and 5.2.

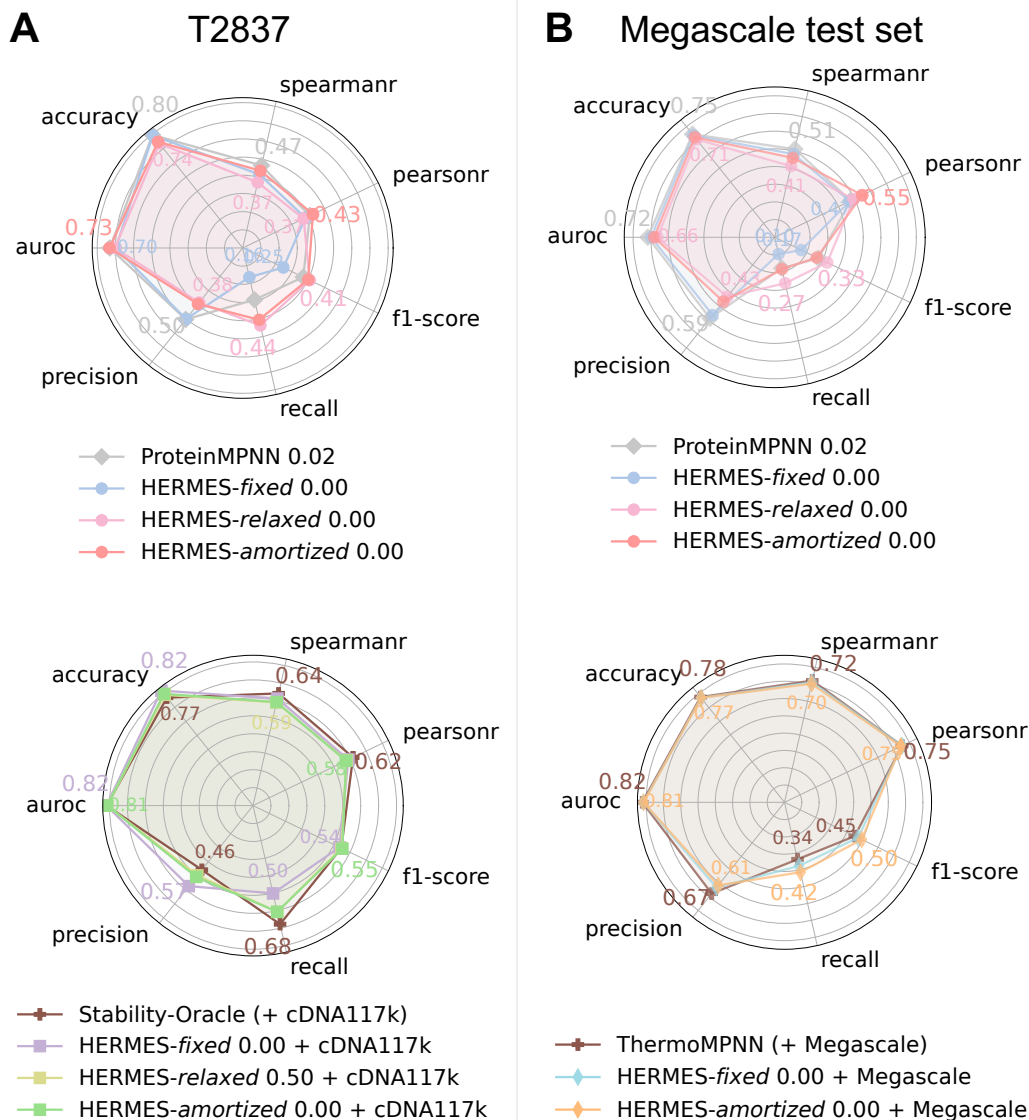


Figure C.1.2: Predicting mutational effects on thermodynamic folding stability. Stabilizing-versus-destabilizing classification metrics are computed using $\Delta\Delta G < 0$ (experimental) and $\Delta \log p > 0$ (predicted) as cutoffs for stabilizing mutations. **(A)** Evaluation on the T2837 results: zero-shot models (top) and models fine-tuned on cDNA117k (bottom). **(B)** Evaluation on Megascale test set results: zero-shot models (top) and models fine-tuned on the Megascale training set (bottom). Model names indicate the architecture, the coordinate-noise amplitude used, and when applicable, the fine-tuning dataset (listed after “+”); *Untr.* is short for *Untrained*, indicating models that had no pre-training and were instead only trained on stability effects. Only models trained without coordinate noise are shown; the noise amplitude is indicated within each model name.

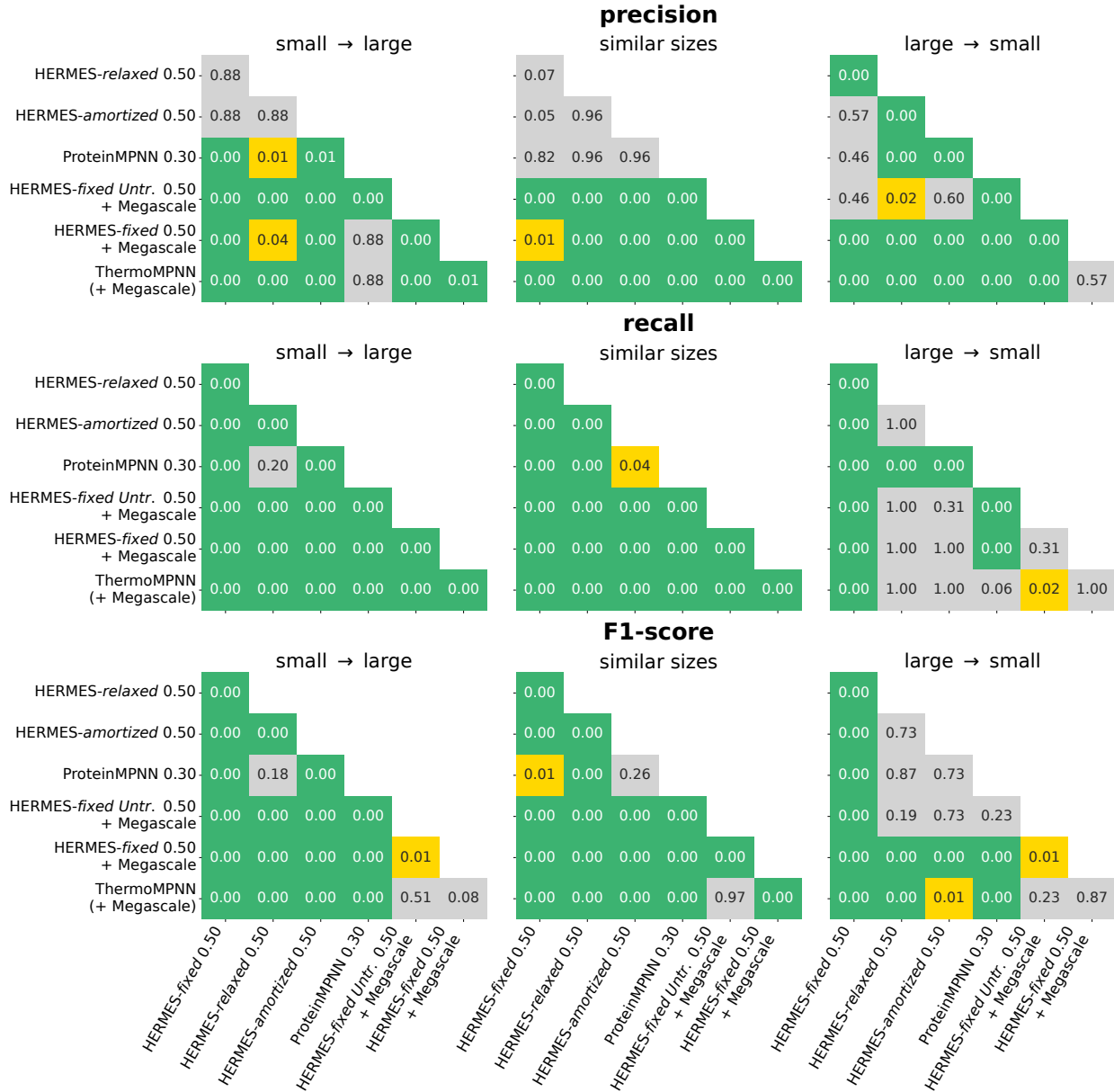


Figure C.1.3: Statistical significance of model performance differences on stabilizing mutation identification. Shown are two-tailed p-values for differences in model performance on identifying stabilizing mutations across Megascale test set subsets. P-values correspond to the performance comparisons shown in Fig. 5.3. Green indicates strong statistical significance ($p < 0.01$), while yellow indicates weaker significance ($0.01 \leq p < 0.05$). P-values were computed using a permutation test and corrected for multiple comparisons using the Holm–Bonferroni procedure within each performance metric (see Methods for details).

	p-values small->large vs. large->small		
HERMES-fixed 0.50	1.000	0.000	0.000
HERMES-relaxed 0.50	0.000	0.000	0.000
HERMES-amortized 0.50	1.000	0.000	0.000
ProteinMPNN 0.30	0.534	0.000	0.016
HERMES-fixed Untr. 0.50 + Megascale	0.000	0.000	1.000
HERMES-fixed 0.50 + Megascale	0.000	0.068	1.000
ThermoMPNN (+ Megascale)	0.000	0.722	1.000
	precision	recall	F1-score

Figure C.1.4: Statistical significance of within-model performance differences in identifying stabilizing mutations across mutational size classes. Shown are two-tailed P-values for within-model differences in performance when identifying stabilizing mutations from the small→large vs. large→small mutational subsets of the Megascale test set. P-values correspond to the performance comparisons shown in Fig. 5.3. Green indicates strong statistical significance ($p < 0.01$), while yellow indicates weaker significance ($0.01 \leq p < 0.05$). P-values were computed using a bootstrap test and corrected for multiple comparisons using the Holm–Bonferroni procedure within each performance metric (see Methods for details).

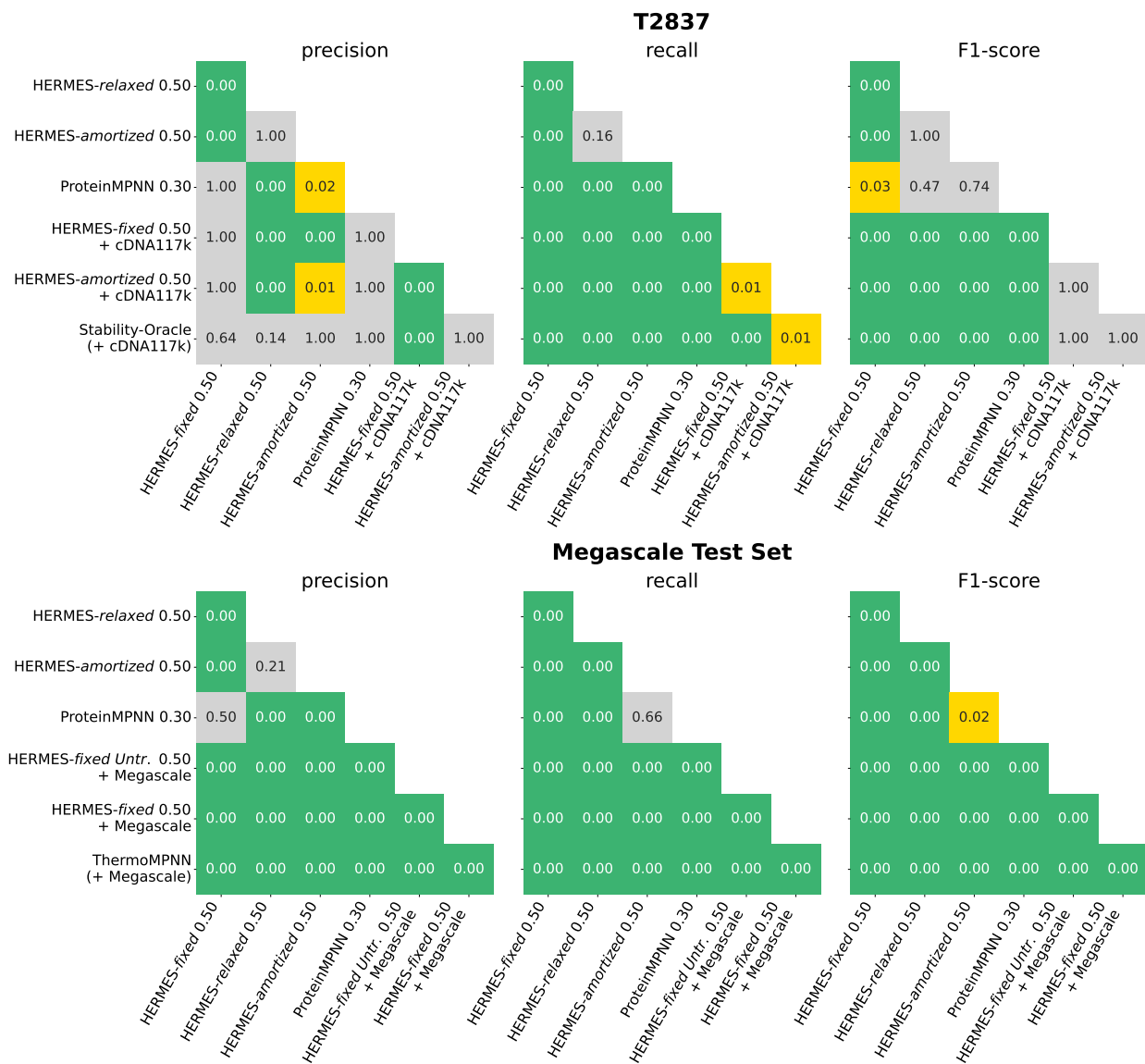


Figure C.1.5: Statistical significance of model performance differences in identifying stabilizing mutations on the full test set. Shown are two-tailed p-values for differences in model performance when identifying stabilizing mutations on the full test set. P-values correspond to the performance comparisons shown in Fig. 5.2. Green indicates strong statistical significance ($p < 0.01$), while yellow indicates weaker significance ($0.01 \leq p < 0.05$). P-values were computed using a permutation test and corrected for multiple comparisons using the Holm–Bonferroni procedure within each performance metric (see Methods for details).

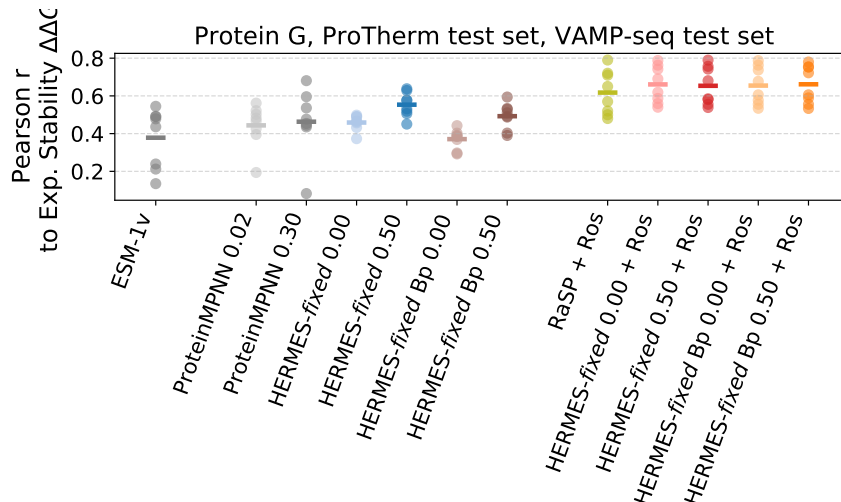


Figure C.1.6: Pearson correlation between model predictions and experimental stability effects on the RaSP test set (8 proteins) [20]. Each dot represents one protein, and the horizontal bar indicates the mean correlation across proteins. Model labels specify the architecture, the coordinate-noise amplitude, and, when applicable, the fine-tuning dataset (denoted after “+”). “Bp” indicates the use of our open-source Biopython-based protein Pre-processing. See Methods for details on the RaSP dataset.

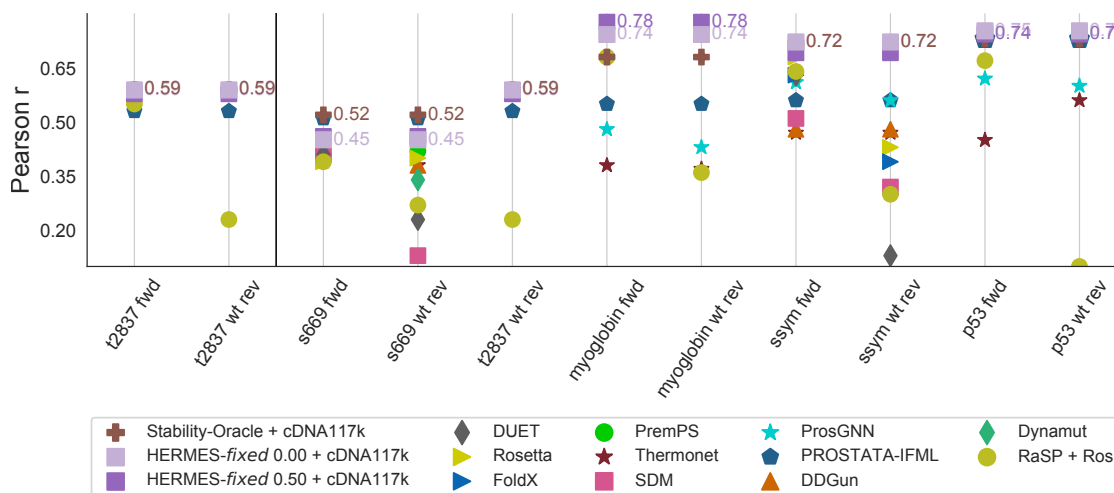


Figure C.1.7: Pearson correlation between model predictions and experimental stability effects on the T2837 dataset and its subsets. Pearson correlation values for all models other than HERMES are taken from [59]. This figure closely replicates a figure from ref. [59], with the key difference that predictions for “reverse” mutations are computed here by conditioning on wild-type structures (denoted as “wt rev”). This distinction is made to avoid confusion with “reverse” mutation predictions computed on mutant structures in the Ssym dataset (Fig. 5.5). For each dataset (x-axis), we denote in text the performance of the HERMES models as well as that of the best-performing model overall.

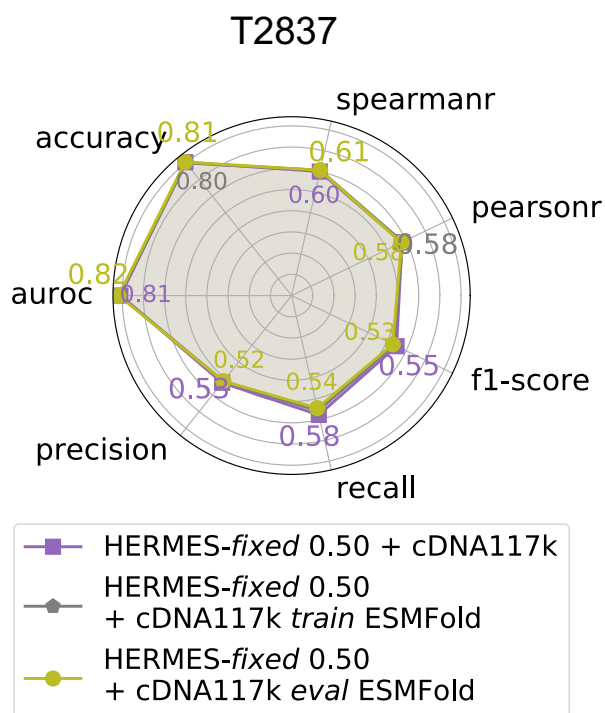


Figure C.1.8: Predicting mutational effects on thermodynamic stability using ESMfold predicted structures for fine-tuning or testing. Stabilizing-versus-destabilizing classification metrics are computed using $\Delta\Delta G < 0$ (experimental) and $\Delta \log p > 0$ (predicted) as cutoffs for stabilizing mutations. We report results on the T2837 dataset, after fine-tuning models on cDNA117k. We consider models fine-tuned and evaluated on crystal structures (purple), models fine-tuned on ESMfold predicted structures and evaluated on crystal structures (grey, “*train* ESMfold” in the model name), and models fine-tuned on crystal structures and evaluated on ESMfold predicted structures (olive, “*eval* ESMfold” in the model name).

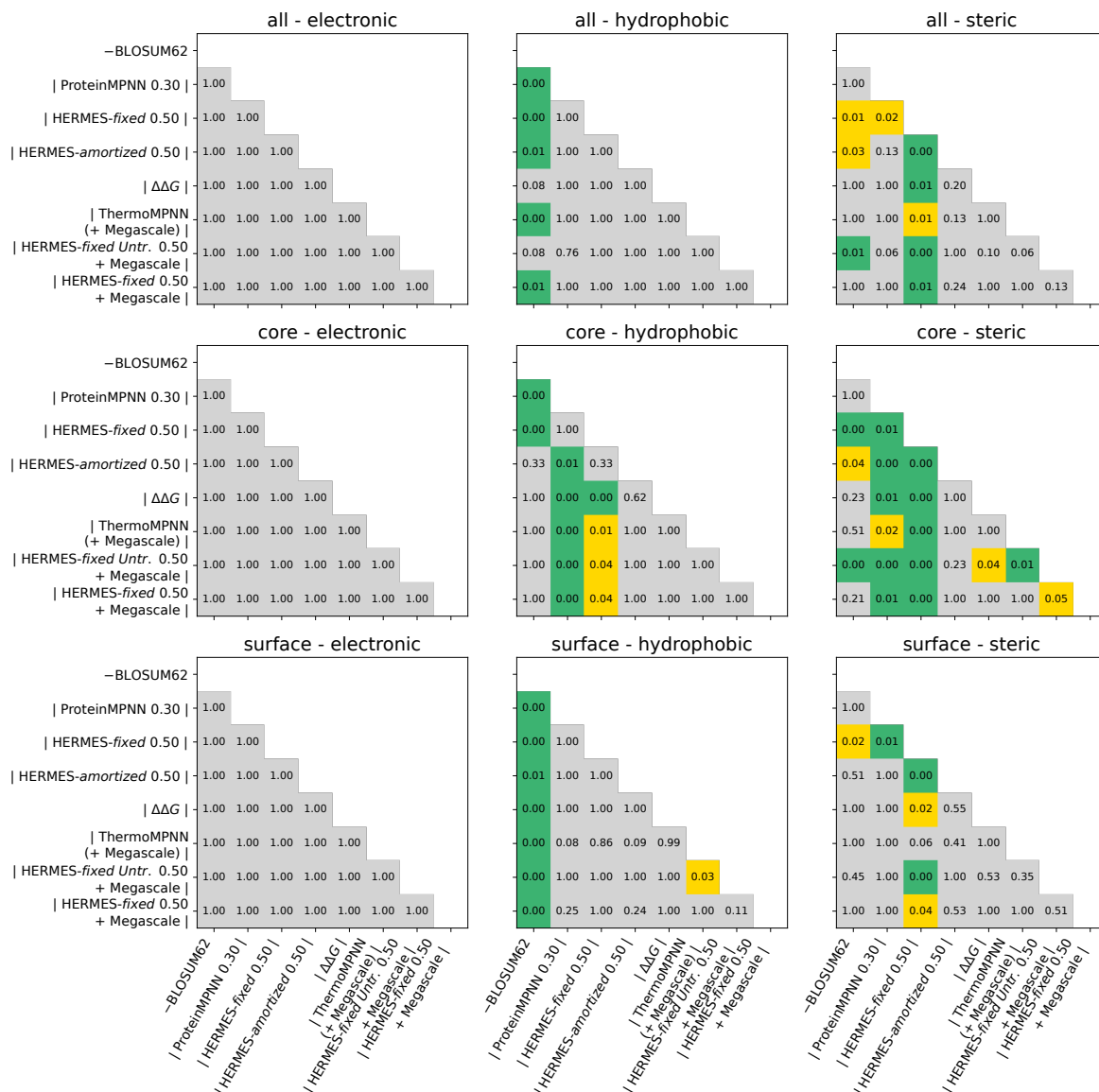


Figure C.1.9: Significance testing results for Fig. 5.4B. Shown are p-values of two-tailed t-tests comparing the distributions of spearman correlations between each model's substitution matrix, and amino-acid properties of a particular class (electronic, hydrophobic, steric). The Holm-Bonferroni method was used to correct p-values for multiple testing error. Entries corresponding to pairs of distributions with p-value ≤ 0.01 are colored in green, p-value ≤ 0.05 are colored in yellow, and p-value > 0.05 are colored in gray.

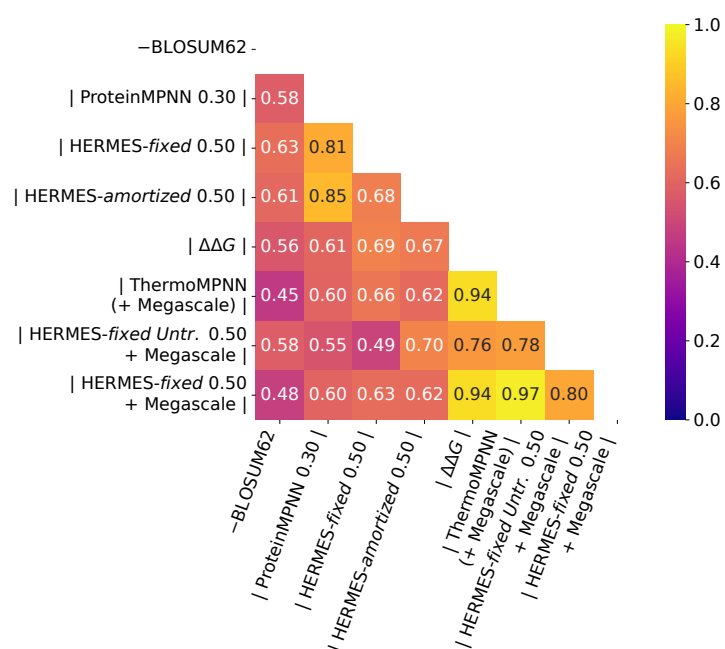


Figure C.1.10: **Spearman correlations between pairs of model-average substitution matrices M^{model} .** The heatmaps for the underlying model-predicted substitution matrices are shown in Fig. 5.4.

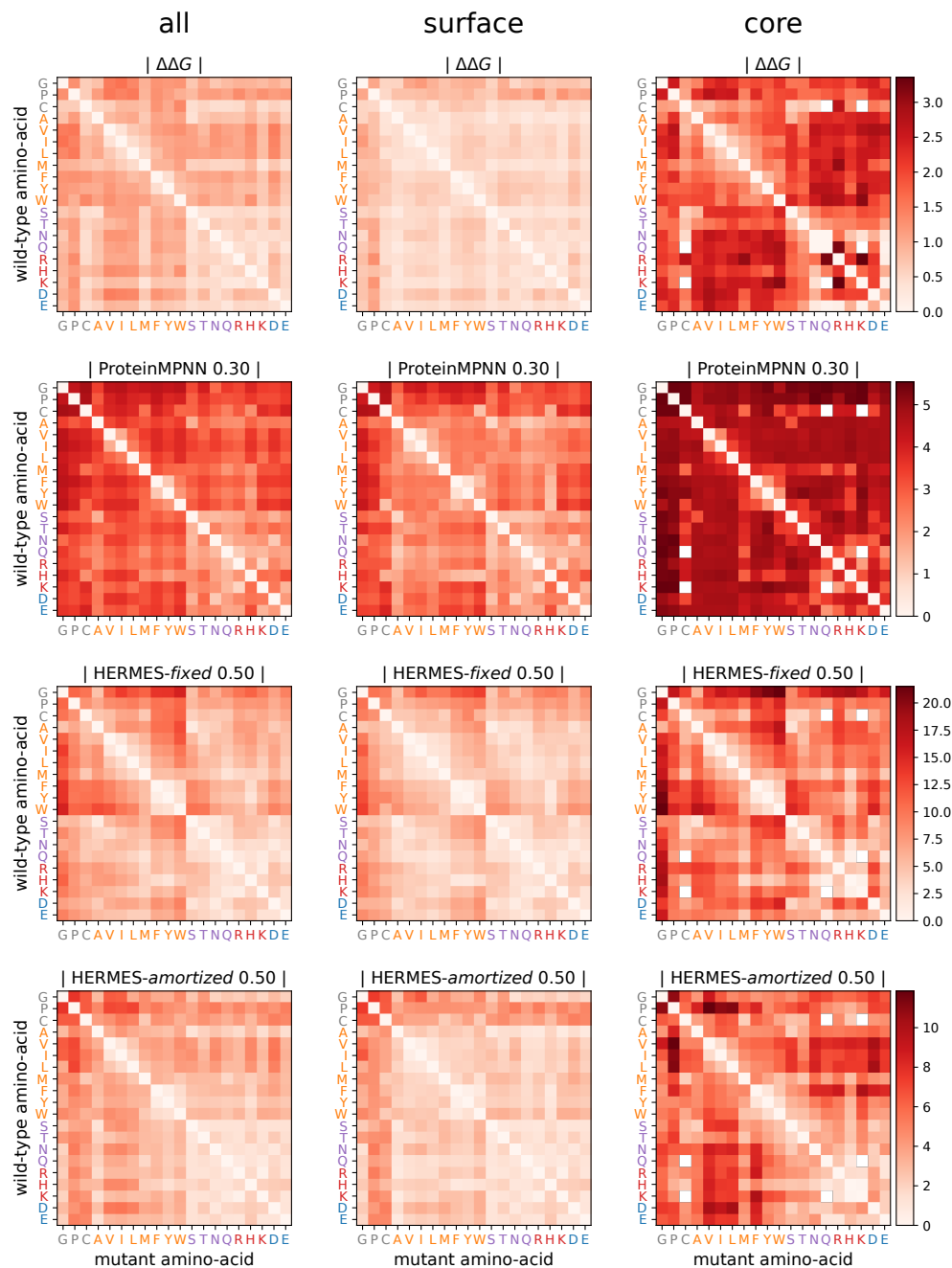


Figure C.1.11: **Model-averaged substitution matrices stratified by protein core and surface residues for zero-shot models.** Shown are model-averaged substitution matrices M^{model} for different zero-shot models (rows 2-4), computed from subsets of sites in the Megascale test set. The first row shows the experimental matrices for mean $|\Delta\Delta G|$ values across mutation subsets. Columns correspond to all residues (left), core residues with solvent-accessible surface area $\text{SASA} < 1^2$ (center), and surface residues with $\text{SASA} > 3^2$ (right).

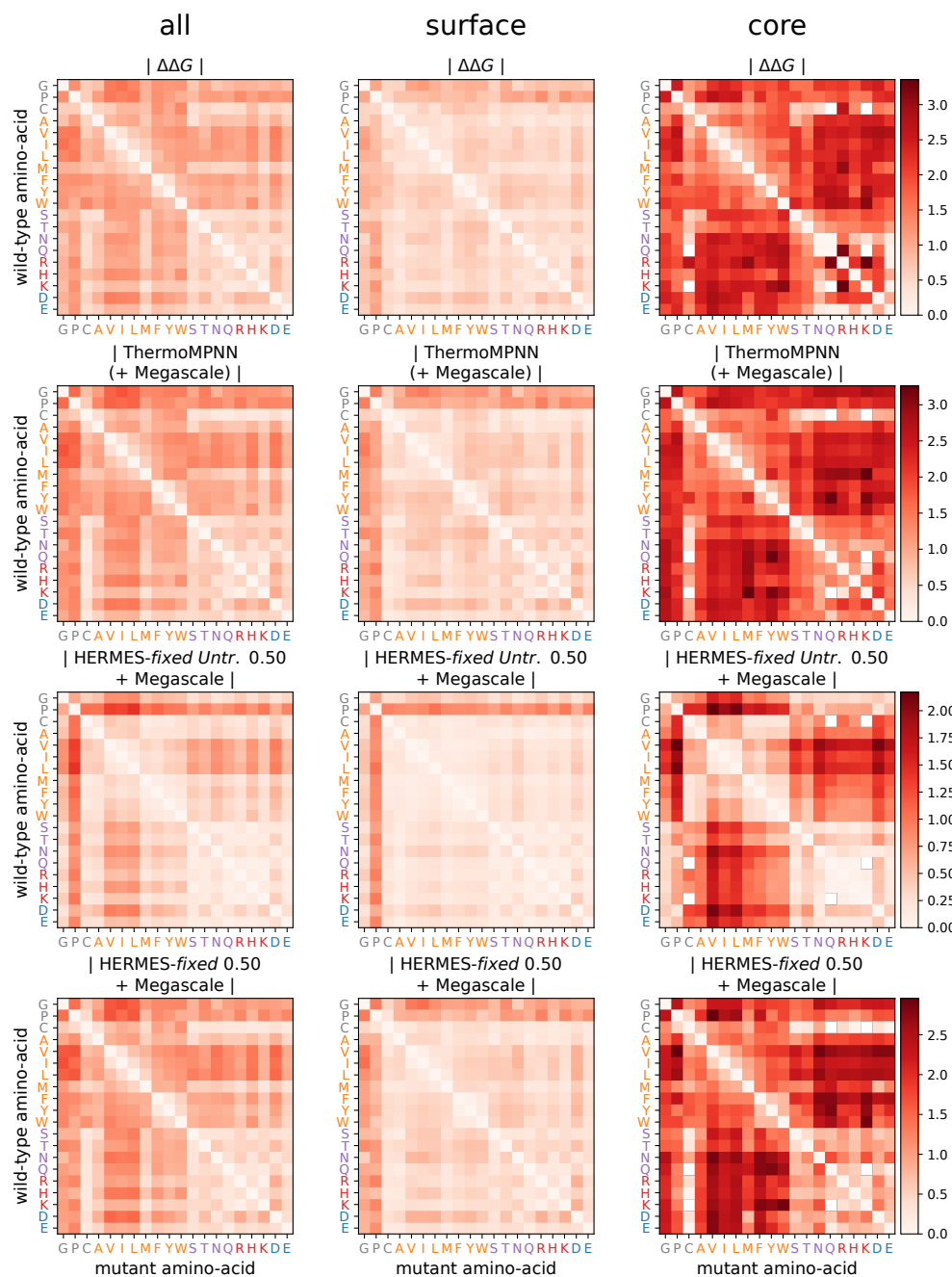


Figure C.1.12: **Model-averaged substitution matrices stratified by protein core and surface residues for stability fine-tuned models.** Similar to Fig. C.1.11 but for models fine-tuned on the Megascale dataset.

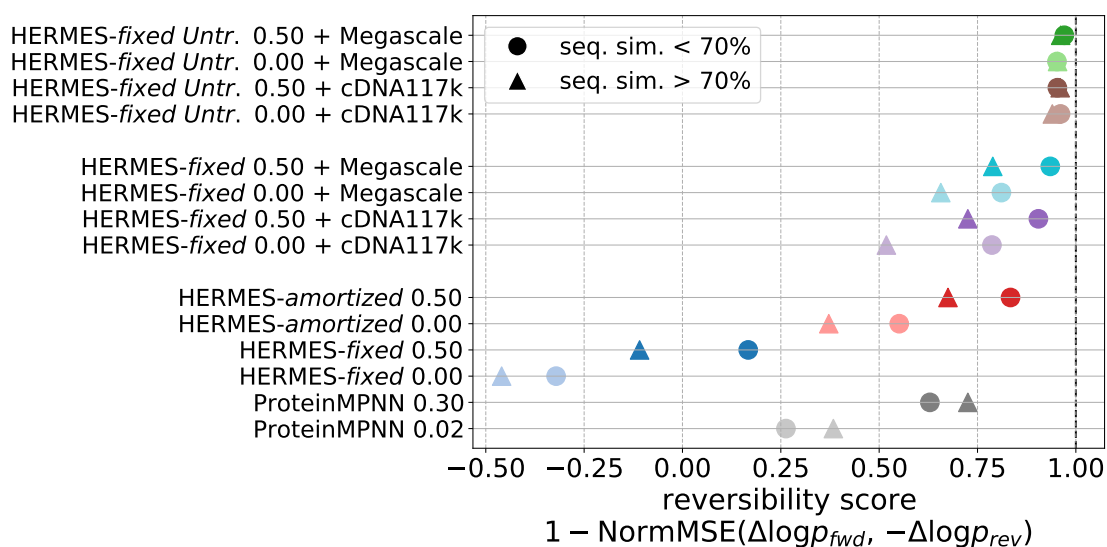


Figure C.1.13: Model reversibility scores on the Ssym dataset. Reversibility is quantified as one minus the mean squared error between the forward mutational effect $\Delta \log p_{fwd}$ and the negated reverse effect $-\Delta \log p_{rev}$ for different models (rows), where model predictions are conditioned on the protein structure containing the outgoing amino acid; The resulting score is normalized to lie between -1 and 1, with higher values indicating a greater degree of reversibility: $1 - \text{mean}((\Delta \log p_{fwd} + \Delta \log p_{rev})^2) / (\text{mean}(\Delta \log p_{fwd}^2) + \text{mean}(\Delta \log p_{rev}^2))$.

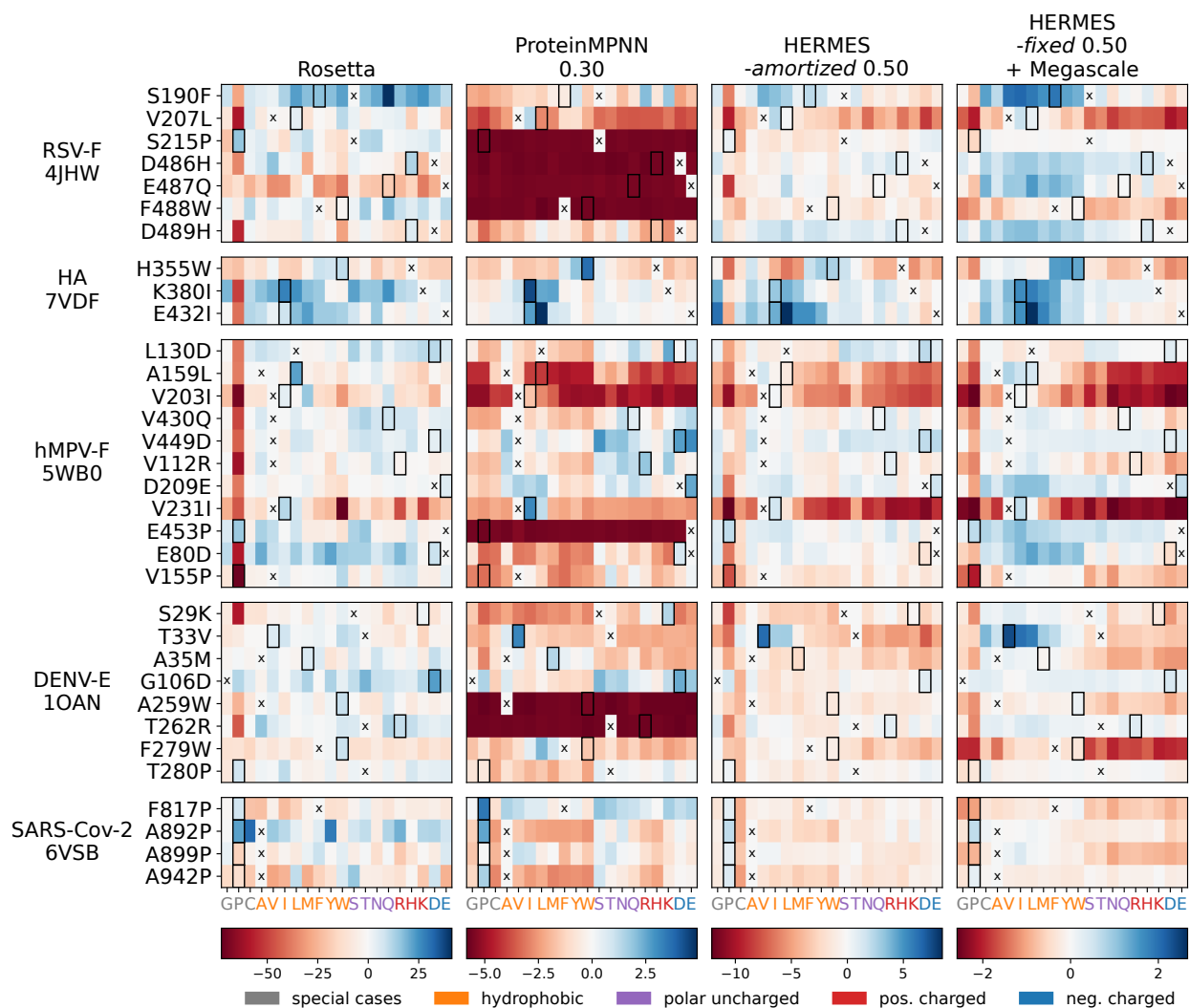


Figure C.1.14: Model predictions for amino acid preferences at sites with known antigen-stabilizing mutations. Predictions from different models (columns) are shown for sites with known antigen-stabilizing mutations specified in Fig. 5.7 and Table C.1.4. For each antigen (rows), predictions are computed using the protein structure corresponding to the PDB ID indicated on the left. Predictions are reported as changes in Rosetta Energy Units (REU) for Rosetta, and $\Delta \log p$ for ProteinMPNN and HERMES models. Wild-type amino acids are marked with centered crosses, while stabilizing mutant amino acids are indicated by dark borders. Amino acids are grouped by broad biochemical class (see legend at the bottom) and, within each class, ordered by increasing size (number of atoms).



Figure C.1.15: **Statistical significance for the number of retrieved antigen-stabilizing mutations.** Shown are p-values from binomial tests comparing the number of antigen-stabilizing mutations retrieved by each model (rows) against random expectation (top) and the BLOSUM62 predictions (bottom). These significance tests correspond to the results reported in Figs. 5.7, C.1.16 and Table C.1.4; see Methods for details of p-value computation.

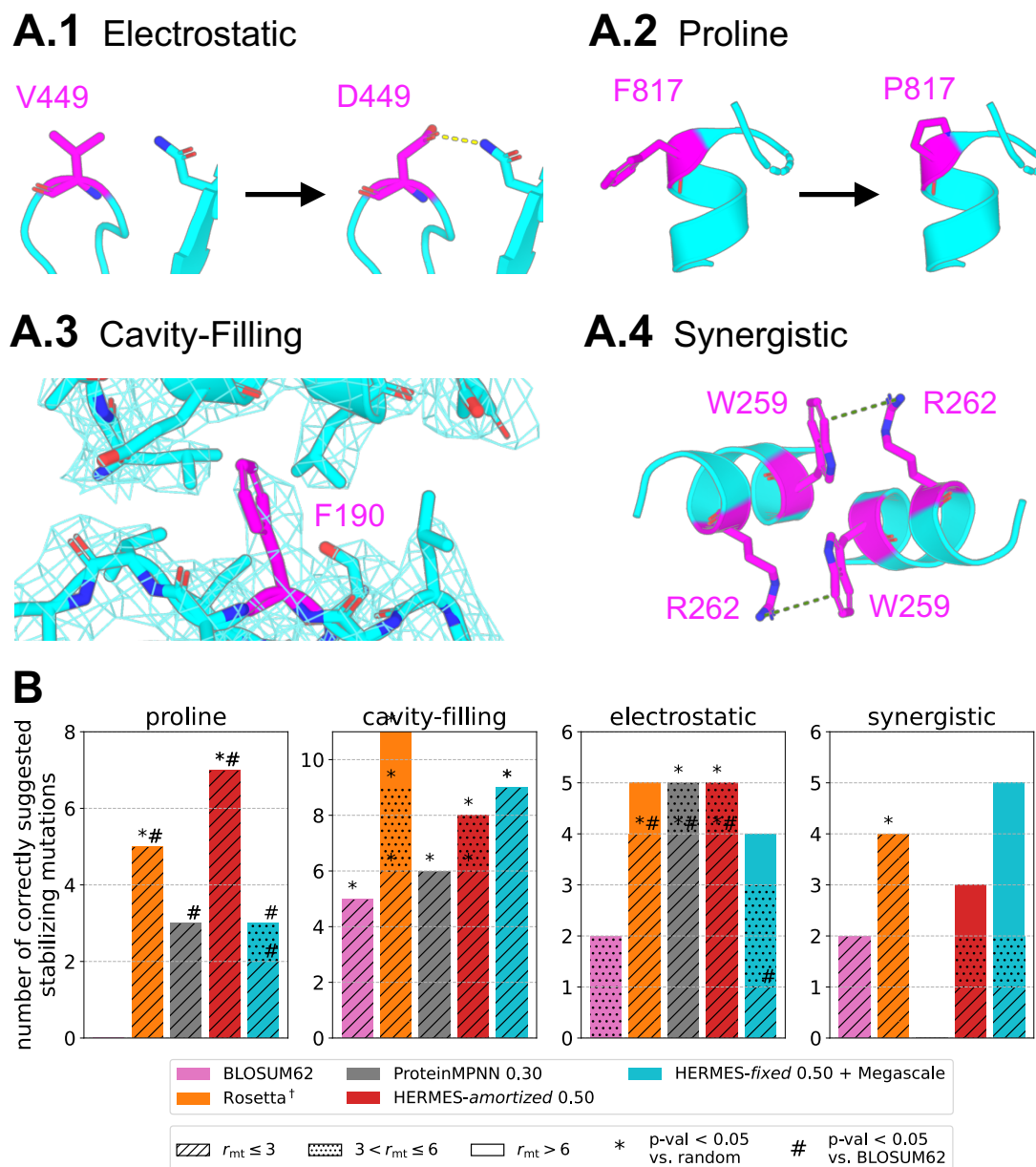


Figure C.1.16: Model ability to recover different mutation patterns that stabilize antigens. (A) Representative examples of mutation types commonly seen in antigen stabilization [32], and analyzed in this study. The mutated residue(s) are shown in magenta. (A.1) *Electrostatic mutation* in hMPV-F: wild-type structure (PDB ID 5WB0, left), in-silico mutant generated with PyMOL's mutagenesis wizard; right. The introduced Aspartic acid forms an electrostatic interaction with a nearby Asparagine (dashed yellow line). (A.2) *Proline mutation* at the N-terminal of a α -helix cap in SARS-CoV-2 spike: wild-type structure (PDB ID 6VSB; left), in-silico mutant generated with PyMOL's mutagenesis wizard; right. (A.3) *Cavity-filling mutation* in RSV-F: mutant structure (PDB ID 4MMS). A bulky hydrophobic substitution packs a previously underfilled region: the *2Fo-Fc* electron density map is shown as a thin mesh. (A.4) *Synergistic dimer-stabilizing mutations* in DENV-E: A259W and T262R in the dimer mutant structure (PDB ID 6WY1). introduced in both chains, create a stabilizing cation- π interaction (dashed green lines). (B) Number of stabilizing mutations recovered by each model, stratified by mutation class, reported as in Fig. 5.7B.

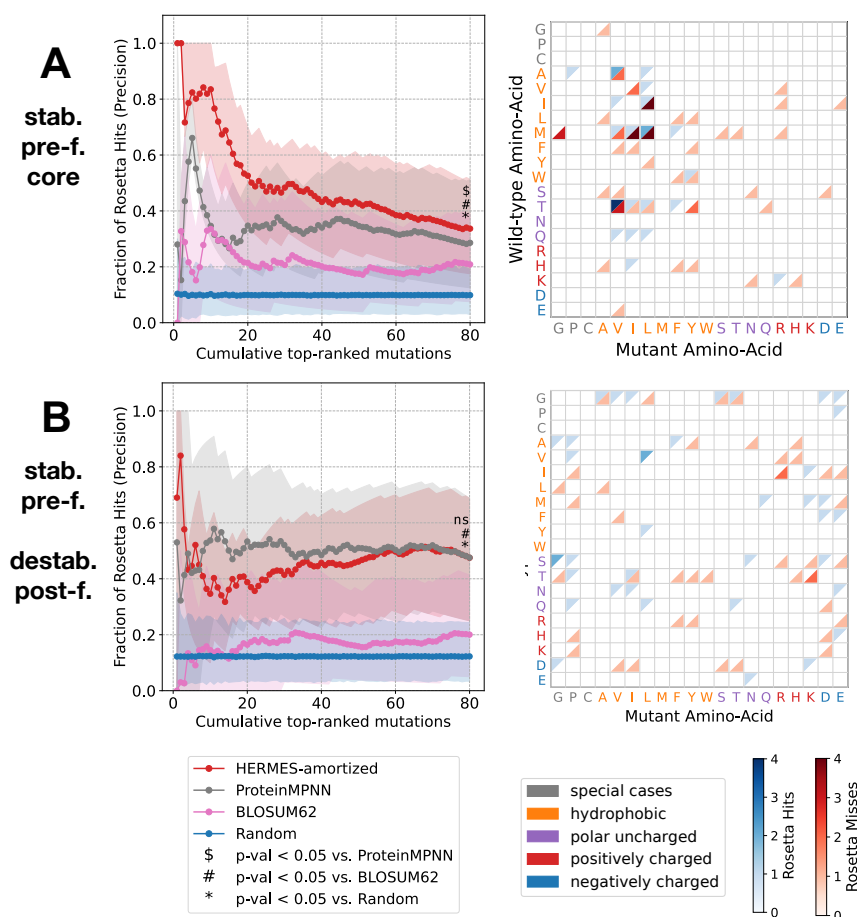


Figure C.1.17: Identifying new antigen-stabilizing mutations with HERMES for DENV-E. Panel **A** shows results using the “stabilize pre-fusion core” pipeline, whereas panel **B** shows results using the “stabilize pre-fusion and destabilize post-fusion” pipeline. Each panel is split in three sub-panels. *Left*: proportion of Rosetta hits as a function of number of top- k suggested mutations (for all methods except for Random, for which k mutations are sampled at random from the pool of 160 randomly-selected mutations). Rosetta hits were computed multiple times via 200 bootstrap samples with replacement of the 50 REU scores. Solid lines and markers denote the mean across bootstrap samples, whereas shaded regions denote the 5-95 percentile range. We note that all BLOSUM62 suggestions had a score ≥ 1 . P-values were computed via Wilcoxon signed-rank test and corrected for multiple testing via Bonferroni within each panel. *Right*: heatmap showing number of Rosetta hits (blue) and misses (red) for each amino-acid mutation pair suggested by HERMES-amortized. Here, Rosetta hits were computed using all 50 REU scores instead of bootstrap samples.

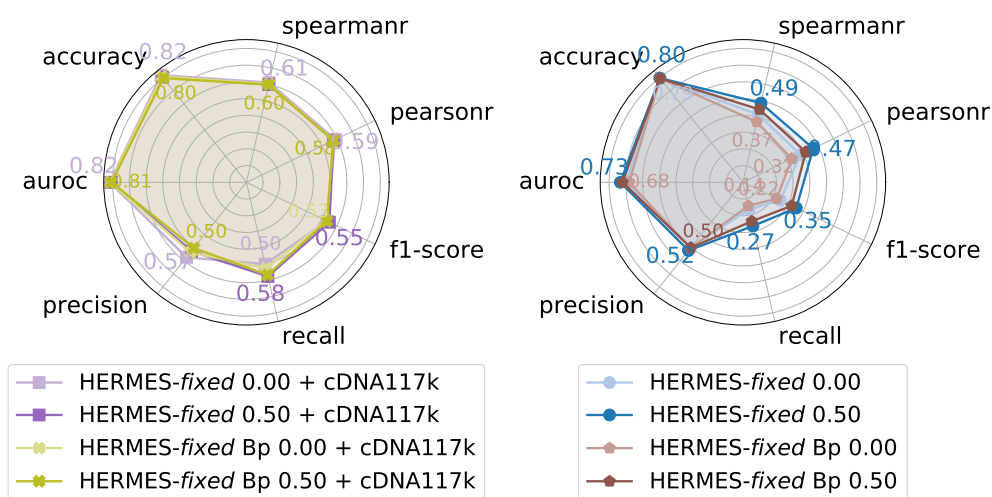


Figure C.1.18: Comparison of PyRosetta and Biopython Pre-processing pipelines for predicting mutation stability effects on T2837. Classification accuracy metrics, analogous to those shown in Fig. 5.2, are reported for fine-tuned models (left) and zero-shot models (right). In each case, models trained using PyRosetta-based Pre-processing are compared with those using Biopython-based Pre-processing (denoted by “Bp” in the model name). Model labels specify the architecture, the coordinate-noise amplitude, and, when applicable, the fine-tuning dataset (listed after “+”). Consistent with results on the RaSP dataset (Fig. C.1.6), Biopython-Pre-processed models show slightly reduced performance relative to PyRosetta-Pre-processed models; however, this difference becomes statistically insignificant after fine-tuning.

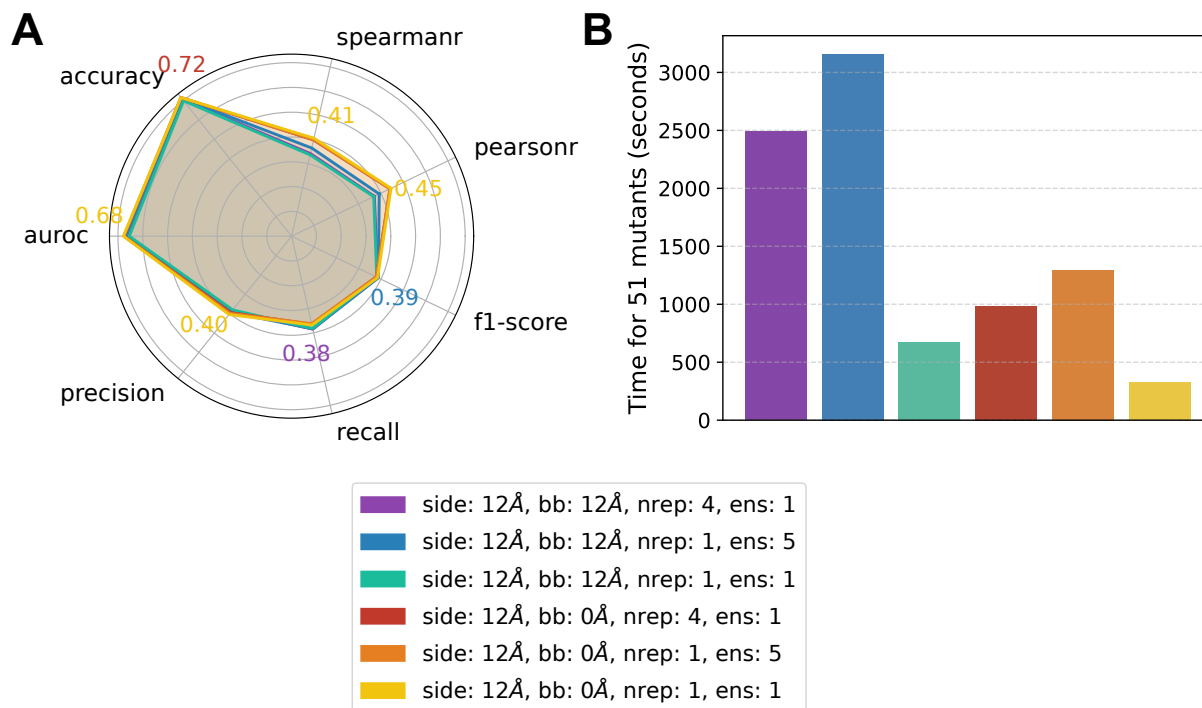


Figure C.1.19: Ablation of PyRosetta fastrelax parameters for HERMES-relaxed 0.50 on the cDNA117k dataset. (A) Classification accuracy metrics, analogous to those shown in Fig. 5.2, are reported for HERMES-relaxed models evaluated on relaxed mutant structures using different PyRosetta fastrelax parameters (indicated by color). HERMES-relaxed scores a mutation as the log-probability difference between the mutant and wild-type amino acids. The wild-type log-probability is evaluated on the wild-type structure, while the mutant log-probability is evaluated on the wild-type structure after introducing the mutation and performing local relaxation. We use the PyRosetta fastrelax protocol and vary the following parameters, noting that the procedure is stochastic: (1) **side**, the distance cutoff for side-chain relaxation; (2) **bb**, the distance cutoff for backbone relaxation; (3) **nreps**, the number of protocol repetitions, with the lowest-energy conformation retained; (4) **ens**, the ensemble size, where predictions are averaged over relaxations obtained with different random seeds. (B) Inference speed for predicting mutational effects on 51 mutants across the same PyRosetta fastrelax parameters as in (A) (colors). A single NVIDIA A40 GPU and a single CPU with 64G of memory were used for all parameters.

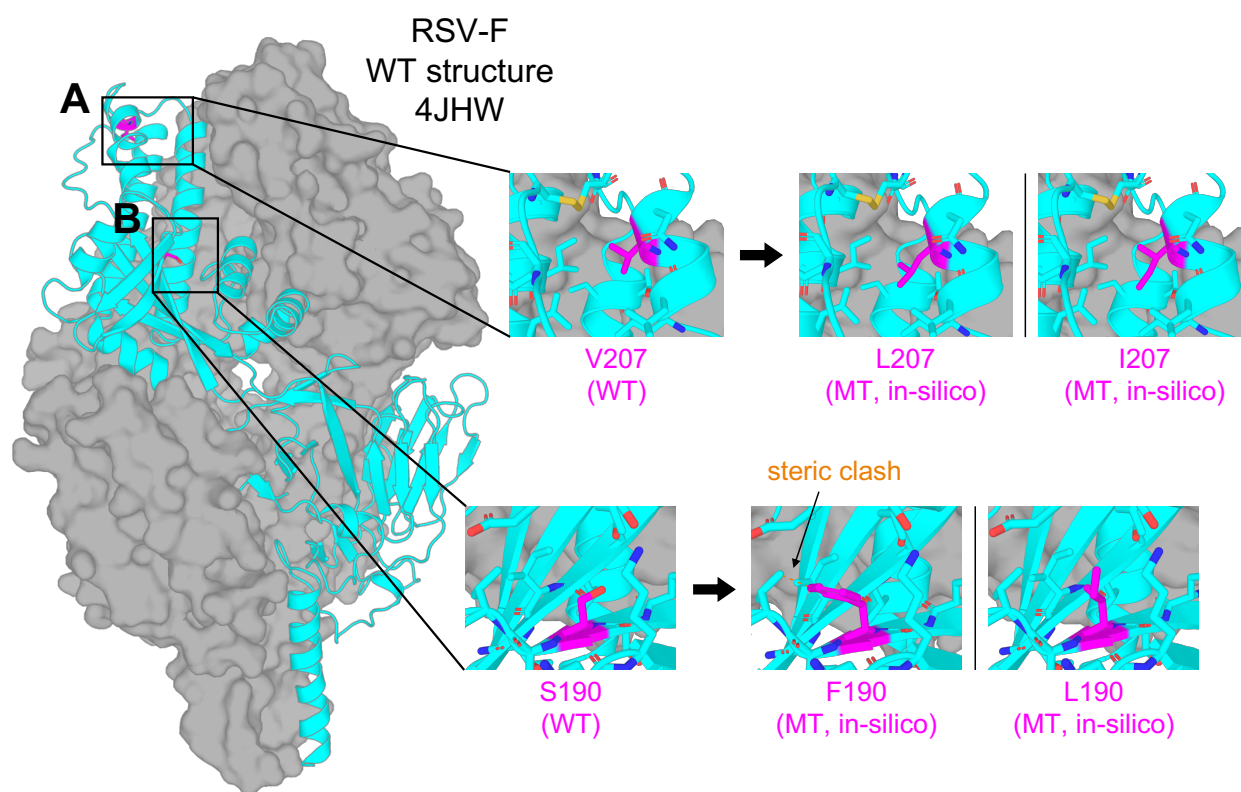


Figure C.1.20: Structure of the wild-type pre-fusion RSV-F antigen highlighting observed and candidate antigen-stabilizing mutations. The wild-type structure is shown on the left (PDB ID: 4JHW). A single protomer of the trimer is displayed as a cyan cartoon, with the remaining protomers shown as a grey surface. WT denotes wild type and MT denotes mutant. Mutations labeled as “in silico” were introduced using PyMOL’s Mutagenesis Wizard starting from the wild-type structure. Steric clashes (orange dashed lines) were identified using PyMOL’s “find clashes” command, and polar contacts (yellow dashed lines) were identified using the corresponding PyMOL command. **(A)** The L207 mutant has been experimentally shown to stabilize the pre-fusion conformation [147] and appears to enhance intraprotomer packing. We speculate that the I207 mutant, which HERMES-*amortized* predicts to have a comparable ranking to L207 in Fig. C.1.14, would pack similarly and may therefore represent an additional stabilizing mutation worth screening. **(B)** Mutant F190 is observed to be stabilizing [147], though it appears to slightly over-pack the region. We speculate that L190, which HERMES-*amortized* predicts to have a comparable ranking to F190 in Fig. C.1.14, would provide a similarly stabilizing effect without over-packing the region.

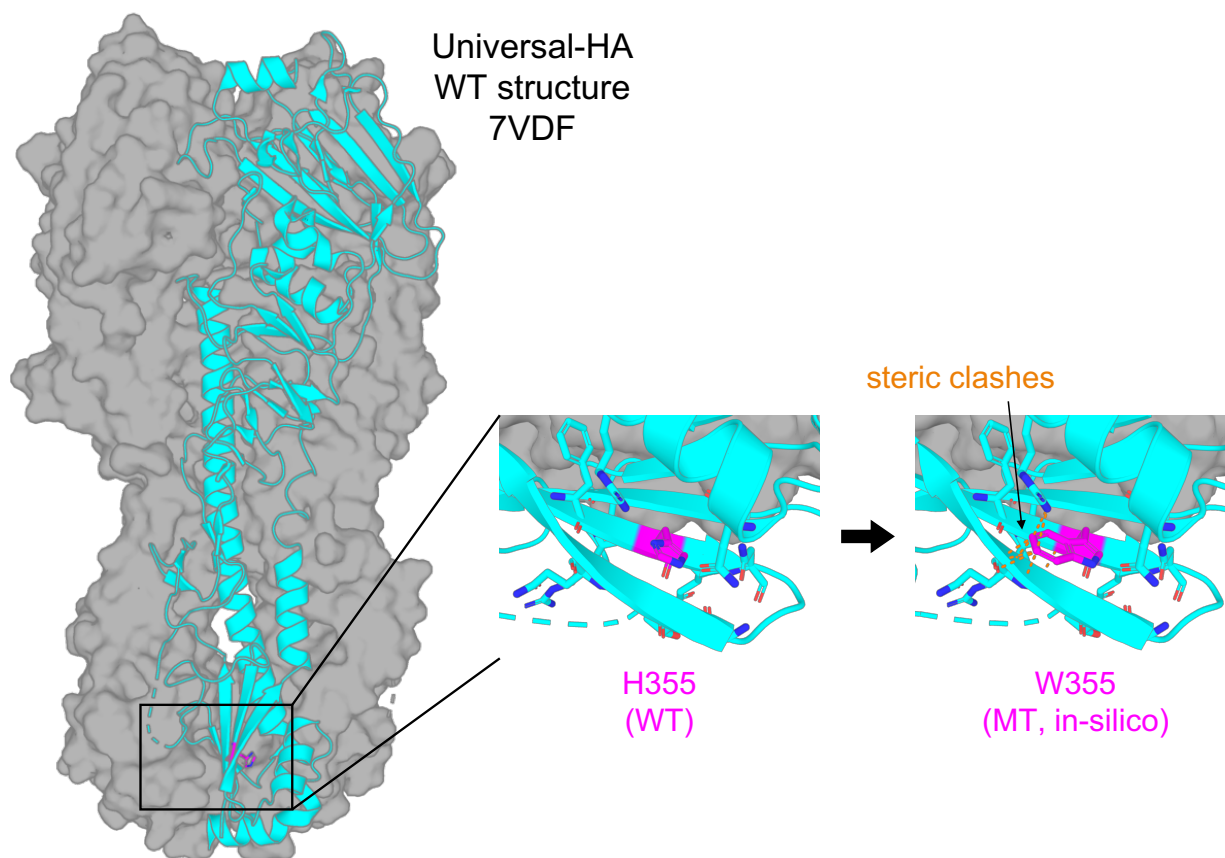


Figure C.1.21: Structure of the wild-type pre-fusion Universal-HA antigen, with highlighted observed and candidate antigen-stabilizing mutations. The wild-type structure is shown on the left (PDB ID 7VDF). A single protomer of the trimer is shown as cyan cartoon; the other copies are shown as grey surface. WT denotes wild type and MT denotes mutant. Mutations labeled as “in silico” were introduced using PyMOL’s Mutagenesis Wizard starting from the wild-type structure. Steric clashes (orange dashed lines) were identified using PyMOL’s “find clashes” command, and polar contacts (yellow dashed lines) were identified using the corresponding PyMOL command. We highlight the H355W mutation which, despite stabilizing the pre-fusion conformation, exhibits substantial steric clashes in the wild-type structural context.

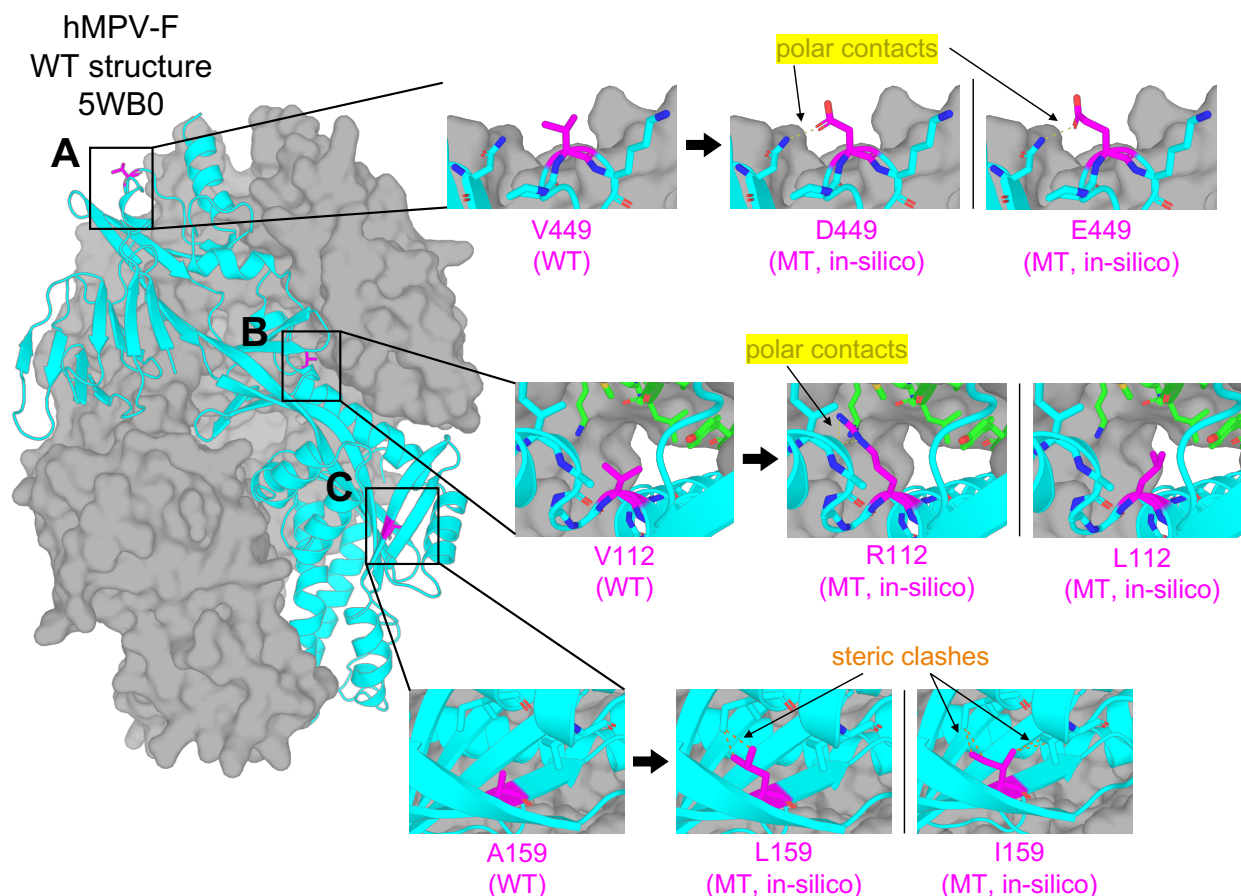


Figure C.1.22: Structure of the wild-type pre-fusion hMPV-F antigen, with highlighted observed and candidate antigen-stabilizing mutations. The wild-type structure is shown on the left (PDB ID 5WB0). A single protomer of the trimer is shown as cyan cartoon; the other copies are shown as grey surface. WT denotes wild type and MT denotes mutant. Mutations labeled as “in silico” were introduced using PyMOL’s Mutagenesis Wizard starting from the wild-type structure. Steric clashes (orange dashed lines) were identified using PyMOL’s “find clashes” command, and polar contacts (yellow dashed lines) were identified using the corresponding PyMOL command. **(A)** The proposed V449D mutation is highlighted, which likely stabilizes the complex by removing a surface-exposed hydrophobic residue and introducing an intraprotomer polar contact. We speculate that substitution with glutamic acid, which HERMES-*amortized* predicts to have a comparable ranking to D449 in Fig. C.1.14, would produce a similar stabilizing effect. **(B)** Introduction of an Arginine in place of a Valine at position 112 likely introduces an intraprotomer polar contact, as well as packing against the adjacent protomer (shown in green); we believe L112, which HERMES-*amortized* predicts to have a comparable ranking to R112 in Fig. C.1.14, would also pack against the adjacent protomer better than Valine. **(C)** A159L stabilizes the complex likely due to a cavity-filling effect, albeit necessitating nearby I137 to adopt a different rotamer; we believe I159, which HERMES-*amortized* predicts to have a comparable ranking to L159 in Fig. C.1.14, would also fulfill a similar role on the condition of L141 also adopting a different rotamer.

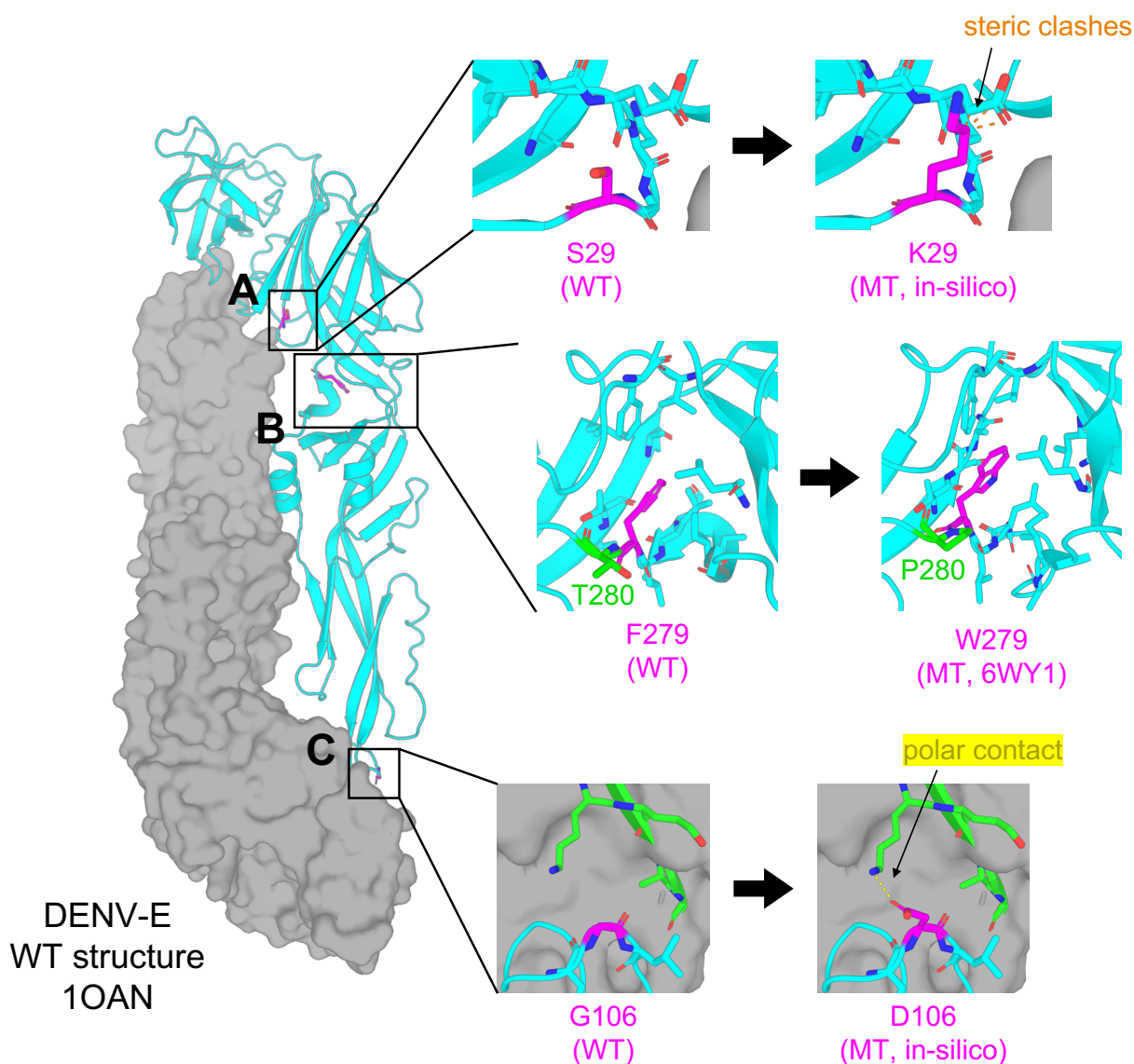


Figure C.1.23: Structure of wild-type pre-fusion DENV-E antigen, with highlighted observed and candidate mutations. The wild-type structure is shown on the left (PDB ID 1OAN). A single protomer of the trimer is shown as cyan cartoon; the other copies are shown as grey surface. WT denotes wild type and MT denotes mutant. Mutations labeled as “in silico” were introduced using PyMOL’s Mutagenesis Wizard starting from the wild-type structure. Steric clashes (orange dashed lines) were identified using PyMOL’s “find clashes” command, and polar contacts (yellow dashed lines) were identified using the corresponding PyMOL command. **(A)** The S29K mutation is experimentally observed to be stabilizing [177], despite appearing to introduce substantial steric clashes when inspected in the wild-type structure, suggesting that stabilization requires a shift in backbone conformation. **(B)** The F279W mutation is experimentally observed to be stabilizing [177], likely by filling an under-packed cavity. A substantial backbone rearrangement is also observed in the mutant structure (residues 269–281), potentially facilitated by the T280P mutation. **(C)** The G106D mutation is experimentally observed to be stabilizing [177], likely through the introduction of a polar contact with the adjacent protomer (shown in green) and/or interactions with surrounding water molecules.

Appendix D

SUPPLEMENT FOR CHAPTER 6

D.1 TCR-pMHC binding affinity and T-cell activity prediction

Peptide energy calculation with HERMES. To predict the affinity of TCR-pMHC complexes or the activity of TCRs induced by their interactions with a pMHC complex, we introduce the predicted energy for a peptide σ , given its surrounding TCR and MHC,

$$E_{\text{peptide}}(\sigma; \text{TCR}, \text{MHC}) = \sum_{i=1}^{\ell} E(\sigma_i; \text{TCR}, \text{MHC}, \sigma_{/i}). \quad (\text{D.1.1})$$

We assume that each peptide residue contributes linearly to the total energy by the amount $E(\sigma_i; \text{TCR}, \text{MHC}, \sigma_{/i})$, where σ_i is the amino acid at position i of the peptide. A residue's energy contribution E is evaluated as its associated logit value from HERMES, taking in as input the atomic composition of the surrounding TCR, MHC and the rest of the peptide $\sigma_{/i}$ (Fig. 1A). This method is akin to predicting the effect of a single amino-acid substitution on a protein's function by using the difference in logits - or equivalently of log-probabilities - corresponding to the mutant and wild-type amino-acids [238].

To compute peptide energies, HERMES operates on the structural composition of the protein complex. However, data on the impact of substitutions on the TCR-pMHC structure is limited, and computational models often fail to capture subtle conformational changes in the structure due to amino acid substitutions in a peptide. To address this issue, we adopt two protocols to estimate energy for a diverse array of peptides for a given TCR-MHC complex:

- **HERMES-*fixed*:** In the simplest approach, we select the TCR-pMHC structure with matching TCR and MHC to our query and with the peptide sequence that most closely matches the peptide of interest and directly input it into HERMES. The peptide energy is subsequently calculated as described in eq. D.1.1 (Fig. 1B). This method does not modify the underlying structure, effectively assuming that any amino acid substitutions do not significantly alter the protein conformation.
- **HERMES-*relaxed*:** Since amino acid substitutions can locally reshape the protein complex, we introduce a more involved protocol to account for these structural changes. We begin with the available crystal structure of the TCR-pMHC complex that closely matches our query as template. We then mutate the original peptide to the desired state and relax the structure using PyRosetta's `cartesian_ddg_fastrelax` protocol [38]. During relaxation, side chain and backbone atoms of the peptide are allowed to move, while only the side chain atoms of the MHC and TCR chains within a $10\text{\AA}$ of the peptide are flexible. We then calculate the peptide energy with HERMES (eq.D.1.1), using the relaxed structure of the mutated variant as input (Fig. 1B). Since PyRosetta's relaxation procedure is stochastic in nature, we run 100 realizations of the relaxation procedure and average the peptide energies across them. For ablation purposes, we also use an alternative protocol, whereby we use the HERMES peptide energy from the conformation with the minimum (most favorable) Rosetta energy computed on the relaxed atoms; we call this protocol *relaxed-min_energy*.

Overall, we use four protocols for scoring peptides that differ in their restrained protocol (HERMES-*fixed* or HERMES-*relaxed*), and the amount of noise injected during training of the underlying HERMES model. Specifically, we train HERMES models with no added noise ($\sigma = 0.00$) and another with moderate Gaussian noise with standard deviation $\sigma = 0.50$ applied to the atomic coordinates of the protein structure data during training. We refer to the resulting peptide scoring protocols as HERMES-*fixed* 0.00, HERMES-*fixed* 0.50, HERMES-*relaxed* 0.00, HERMES-*relaxed* 0.50, specifying both the model and the noise amplitude.

Evaluation of EC₅₀ from dose response measurements against peptide libraries. Ref. [140] presents dose response measurements for fraction of 4-1BB⁺ CD8⁺ T-cells across varying peptide concentrations. These measurements are done for peptide libraries of single point mutants from a given wild-type peptide, across 7 different TCR-MHC systems: three TCRs recognizing variants of the highly immunogenic HLA-A02:01-restricted human cytomegalovirus (CMV) epitope NLVPMVATV (NLV); three TCRs recognizing variants of the weaker HLA-A02:01-restricted melanoma gp100 self-antigen IMDQVPFSV; and one TCR recognizing variants of the weakly immunogenic HLA-B*27:05-restricted pancreatic cancer neo-peptide GRLKALCQR [140].

The half-maximal effective concentration (EC₅₀) from the dose response curves of 4-1BB⁺ CD8⁺ T-cell fractions can quantify the level of T-cell activity in response to each peptide. We used our own fits to estimate the EC₅₀ associated with T-cell responses. Specifically, we fitted a Hill function with background to the dose response measurements for each peptide from ref. [140],

$$f(x) = \frac{(A - A_0)x^n}{x^n + K^n} + A_0 \quad (\text{D.1.2})$$

where $f(x)$ is the fraction of 4-1BB⁺ CD8⁺ T-cells pulsed by a peptide at concentration x , A is the response amplitude, A_0 is the background activation, n is the Hill coefficient, and K is the half-maximal effective concentration EC₅₀. Similar to ref. [140], we regularized our fitted Hill functions, whereby the deviation of the amplitude A and the Hill coefficient n from 1, and the background activation A_0 from 0 are penalized. The code to fit these curves is available in our github repository.

D.2 De-novo peptide design

Peptide design algorithm. Given a TCR and an MHC, we use HERMES to design de novo peptides that can form a stable TCR-pMHC complex, and potentially, elicit a T-cell response. Our approach starts with an existing template TCR-pMHC structure. We refer to the peptide in the template as wild-type; see Table D.7.5 for the templates we use in our analyses. Similar to evaluating the peptide energy, we use two pipelines to sample novel peptides:

- **HERMES-*fixed*:** This is our basic pipeline, in which we generate novel peptide sequences by conditioning on the fixed TCR-pMHC template structure. Specifically, we sequentially mask the atoms of each amino acid along the peptide and then use HERMES to compute the probability $P(\sigma_i | \mathbf{x}_i)$ of different amino acid types σ_i at the masked residue i , conditioned on the surrounding atomic neighborhood $\mathbf{x}_i$, within a 10 Å radius of the masked residue’s C- α . The neighborhood $\mathbf{x}_i$ can include atoms from the TCR, the MHC, and the other amino acids in the peptide. Repeating this procedure for every residue along the peptide allows us to construct a Position Weight Matrix (PWM) that represents the probabilities of different amino acids at each peptide position, conditioned on the surrounding structural neighborhood. We generate peptides using multinomial sampling from the PWM; see the schematic in Fig. 1C.
- **HERMES-*relaxed*:** This pipeline incorporates simulated annealing and Markov chain Monte Carlo (MCMC) sampling to account for structural changes resulting from peptide substitutions. We start with

the template TCR-MHC structure and an initial random peptide sequence $\sigma^{(0)}$, which is *in-silico* packed via PyRosetta, followed by the Relax protocol, yielding a relaxed structure $\mathbf{x}^{(0)}$. During relaxation, both side chain and backbone atoms of the peptide are allowed to move, while only the side chain atoms of the MHC and TCR within 10 Å of the peptide are flexible. Following the HERMES-*fixed* protocol, we then evaluate the PWM associated with this relaxed structure and use it to generate a new peptide sequence, $\sigma^{(1)}$. This sequence is again packed and relaxed, producing a new structure, $\mathbf{x}^{(1)}$. Using eq. D.1.1, we evaluate the energies of the peptides in their respective (relaxed) structures as $E(\sigma^{(0)}; \mathbf{x}^{(0)})$ and $E(\sigma^{(1)}; \mathbf{x}^{(1)})$, and apply the Metropolis-Hastings criterion to accept the transition from $\sigma^{(0)}$ to $\sigma^{(1)}$ with probability:

$$P_{\sigma^{(0)} \rightarrow \sigma^{(1)}} = \min \left(1, e^{-\frac{E(\sigma^{(1)}; \mathbf{x}^{(1)}) - E(\sigma^{(0)}; \mathbf{x}^{(0)})}{T}} \right) \quad (\text{D.2.1})$$

where T is the temperature. This process is iterated, incrementally reducing T at each step, until convergence. The peptide obtained at the final iteration represents the protocol's de novo design. Repeating this procedure many times produces a diverse set of peptides spanning various distances from the wild-type sequence; see the schematic in Fig. 1C and Algorithm 1 for more details, and the the convergence curves in Fig. D.7.6C,F,I.

Algorithm 1: HERMES-relaxed protocol for generating TCR-MHC-specific peptide sequences. For brevity, we define $\mathbf{x}_i \equiv (\text{TCR}, \text{MHC}, \sigma_i)$ and drop the arguments of E_{peptide} .

Input: TCR-pMHC structure $\mathbf{x}$, maximum number of iterations N ;

Output: peptide sequence $\sigma^{(N)}$;

$T_0 \leftarrow 10$;

$T_f \leftarrow 0$;

$a \leftarrow 0.95$;

$T^{(0)} \leftarrow T_0$;

$\sigma^{(0)} \leftarrow \text{random sequence}$;

$\mathbf{x}^{(0)} \leftarrow \text{RosettaMutate}(\mathbf{x}^{(0)}, \sigma^{(0)})$;

$\mathbf{x}^{(0)} \leftarrow \text{RosettaFastRelax}(\mathbf{x}^{(0)})$;

$E_{\text{peptide}}^{(0)} \leftarrow \sum_{i \in \mathcal{R}_{\text{peptide}}} E(\sigma_i^{(0)}; \mathbf{x}_i^{(0)})$;

for $k = 1, 2, \dots, N$ **do**

for $i \in \mathcal{R}_{\text{peptide}}$ **do**

$\sigma_i^{(\text{temp})} \sim \text{Multinomial}(\text{softmax}([E(\text{aa}; \mathbf{x}_i^{(k-1)})/T^{(k)}]_{\text{aa} \in \text{amino-acids}}))$;

end

$\mathbf{x}^{(\text{temp})} \leftarrow \text{RosettaMutate}(\mathbf{x}^{(k-1)}, \sigma^{(\text{temp})})$;

$\mathbf{x}^{(\text{temp})} \leftarrow \text{RosettaFastRelax}(\mathbf{x}^{(\text{temp})})$;

$E_{\text{peptide}}^{(\text{temp})} \leftarrow \sum_{i \in \mathcal{R}_{\text{peptide}}} E(\sigma_i^{(\text{temp})}; \mathbf{x}_i^{(\text{temp})})$;

$p \sim \text{Uniform}(0, 1)$;

if $p < \exp(-(E_{\text{peptide}}^{(\text{temp})} - E_{\text{peptide}}^{(k-1)})/T^{(k)})$ **then**

$E_{\text{peptide}}^{(k)} \leftarrow E_{\text{peptide}}^{(\text{temp})}$;

$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(\text{temp})}$;

$\sigma^{(k)} \leftarrow \sigma^{(\text{temp})}$;

else

$E_{\text{peptide}}^{(k)} \leftarrow E_{\text{peptide}}^{(k-1)}$;

$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k-1)}$;

$\sigma^{(k)} \leftarrow \sigma^{(k-1)}$;

end

$T^{(k+1)} \leftarrow (T_0 - T_f)a^k + T_f$;

end

Similar to scoring, we use four protocols for peptide design that differ in their restrained protocol (HERMES-*fixed* or HERMES-*relaxed*), and the amount of noise injected during training of the underlying HERMES model. Specifically, we train HERMES models with no added noise ($\sigma = 0.00$) and another with moderate Gaussian noise with standard deviation $\sigma = 0.50$ applied to the atomic coordinates of the protein structure data during training. We refer to the resulting peptide design protocols as HERMES-*fixed* 0.00, HERMES-*fixed* 0.50, HERMES-*relaxed* 0.00, HERMES-*relaxed* 0.50, specifying both the model and the noise amplitude.

Filtering metrics for peptide design with HERMES and benchmarking of different design approaches.

For each system, we first selected designs predicted by NetMHCpan to be at least weak binders to their target MHCs ($EL_rank \leq 2.0$) [188]. We then used TCRdock [26] to filter peptides for reactivity. TCRdock is a structure prediction algorithm built on top of AlphaFold2 [108], and refined to model TCR-pMHC interactions. Its PAE score for the TCR-pMHC interface can be used to distinguish reactive from decoy peptides [26], and therefore, can provide a metric for filtering our designed peptides; other work has similarly used the PAE of structure-prediction algorithms to filter designed protein binders [134]. Using TCRdock, we retained those peptides whose PAE scores at the TCR-pMHC interface fell below a threshold, as illustrated in Figs. D.7.13, D.7.14, and D.7.15. The distribution of PAE scores is highly system specific [134], so we set each system's threshold near the TCRdock PAE value of its *wild-type* peptides in complex with the respective TCR-MHC, as the wild types are known binders. Among the filtered peptides, we selected 93 for experimental validation, ensuring that designs from both HERMES-*fixed* and HERMES-*relaxed*, as well as from models with 0.0 and 0.5Å noise levels were represented. We also enforced a minimum difference of 2-3 amino acids (depending on the system) between the designs and the wild-type peptides.

It should be noted that the AF3 PAE score can also discriminate between reactive and decoy peptides (Figs. D.7.8, D.7.16), making it also suitable for filtering. However, a full-access implementation of AF3 was unavailable before our experimental validations. Therefore, we used AF3 only as a secondary filter for designs in the EBV and MAGE systems, and for benchmarking against TCRdock's PAE score for the designed peptides (Figs. D.7.8); see details below.

Design parameters for different TCR-pMHC systems. Here, we provide the details on the specifics of peptide selection for each TCR-MHC system. It should be noted that as we progressed from NY-ESO to EBV and MAGE, we chose to explore the sequence space farther from the wild-type peptides, leading to more challenging designs.

- *NY-ESO*: The wild-type peptides (NYESO: SLLMWITQC and NYESO V9C: SLLMWITQV) have TCRdock PAEs of 5.17 and 5.18, respectively. As positive designs, we selected 58 peptides with the lowest TCRdock PAE values (all below 5.35) and a Hamming distance of at least two amino acids from their respective wild-type sequences. For negative designs, we uniformly sampled 35 peptides among those designed by HERMES whose TCRdock PAE values fell between 5.5 and 7.0. We did not enforce all HERMES models to be represented equally among these designs, nor did we enforce equal representation of the template structures. However, we note that all models are represented among positive designs (Fig. D.7.9).
- *EBV*: The wild-type peptides (HPVG: HPVGEADYFEY and HPVG E5Q: HPVQGADYFEY) have TCRdock PAEs of 4.90 and 4.82, and AF3 PAEs of 1.15 and 1.13, respectively. We selected 93 positive peptides using a TCRdock's PAE score cutoff of 5.1 and AF3 PAE score cutoff of 2.0. Specifically, we sampled uniformly below these cutoffs with the following design protocols: HERMES-*fixed* at both noise levels using the HPVG structure (3MV7) as template (24 peptide), HERMES-*fixed* at both noise

levels, using the HPVG E5Q structure (4PRP) as template (23 designs); HERMES-*relaxed* at both noise levels, using the HPVG structure (3MV7) as template (24 peptides), and HERMES-*relaxed* at both noise levels using the HPVG E5Q structure (4PRP) as template (23 designs); see Fig. D.7.10.

- *MAGE*: The wild-type peptides (MAGE-A3: EVDPIGHLIY and Titin: ESDPIVAQY) have TCRdock PAEs of 5.21 and 5.03, and AF3 PAEs of 1.00 and 0.98, respectively. We selected 93 positive peptides using a TCRdock's PAE score cutoff of 5.2; the AF3 PAE scores for all of these designed peptide are comparable to that of the wild-types. For these designs, we sampled uniformly below the PAE cutoffs with the following design protocols: HERMES-*fixed* at both noise levels using MAGE-A3 structure (5BRZ) as template (15 peptides); HERMES-*fixed* at both noise levels using the MAGE-A3 structure (5BRZ) as template with E₁ fixed (15 peptides); HERMES-*relaxed* at both noise levels using the MAGE-A3 structure (5BRZ) as template (16 peptides); HERMES-*relaxed* at both noise levels using the MAGE-A3 structure (5BRZ) as template with E₁ fixed (15 peptides); HERMES-*fixed* at both noise levels and using the Titin structure (5BS0) as template (16 peptides); HERMES-*relaxed* at both noise levels using the Titin structure (5BS0) as template (16 peptides); see Fig. D.7.11.

Characterizing the size of the sequence space for design. The size of the sequence space to explore grows exponentially with increasing distance from the wild-type peptide. If no restriction is imposed (i.e., all amino acids are permissible with equal probabilities), the size of the sequence space follows,

$$\Omega_0(d) = \binom{\ell}{d} 19^d \quad (\text{D.2.2})$$

where ℓ is the length of the peptide, and d is the distance from the wild-type sequence.

MHC presentation imposes constraints on amino acid usages in peptides, especially at the anchor residues. Let $p_i(a)$ denote the probability of amino acid a at position i of the peptide, e.g., set by the sequence-specific preferences of the MHC allele. The probability of introducing a mutation at position i depends on the prominence of the wild-type amino acid, and is equal to $q_i = 1 - p_i(\text{wt})$. The typical number of non-wildtype amino acids realized at position i can be estimated by $n_i = 2^{S_i}$, where $S_i = -\sum_{a \neq \text{wt}} \hat{p}_i(a) \log_2 \hat{p}_i(a)$ is the entropy associated with the non-wildtype amino acid frequencies, and $\hat{p}_i(a) = \frac{p_i(a)}{1 - p_i(\text{wt})}$ is the normalized probability of amino acid a , obtained by excluding the wild-type from the set. Therefore, the typical size of the sequence space at Hamming distance d from the wild-type peptide—subject to constraints imposed by the amino acid frequencies $p_i(a)$ (e.g., due to MHC restriction)—is given by,

$$\Omega(d) = \sum_{\{i_1, \dots, i_d\}} (q_{i_1} n_{i_1}) \dots (q_{i_d} n_{i_d}) \equiv \sum_{\{i_1, \dots, i_d\}} 2^{\hat{S}_{i_1} + \dots + \hat{S}_{i_d}} \quad (\text{D.2.3})$$

where the summation runs over all $\binom{\ell}{d}$ ways to choose d distinct sites at which mutations may occur, and $q_i n_i$ is the expected number of substitutions associated with site i —evaluated as the likelihood of mutating site i times the typical number of viable amino acid replacements at that site—and $\hat{S}_i = S_i + \log_2 q_i$ is the resulting adjusted entropy at site i .

In principle, evaluating $\Omega(d)$ in eq. D.2.3 involves summation over a combinatorially large number of terms, which can become computationally prohibitive. In practice, however, many of the sites have the same adjusted entropy, which simplifies the evaluations of $\Omega(d)$. For example, the adjusted entropies for mutations away from the 9 amino acid NY-ESO peptide restricted by HLA-A*02:01 yield the following (rounded) values: [4, 2, 4, 4, 4, 3, 4, 4, 2] in bits, indicating six sites with 4 bits of entropy (or $2^4 = 16$ typical substitutions), one site with 3 bits (8 typical substitutions), and two sites with 2 bits (4 typical substitutions). To demonstrate, in this case, the number of sequences at distance $d = 5$ from the wild-type follows,

$$\begin{aligned}
\Omega_{\text{NYESO}}(d=5) &= \binom{6}{5}16^5 + \binom{6}{4}16^4 \times \left[\binom{2}{1}4^1 + 8^1 \right] + \binom{6}{3}16^3 \times \left[\binom{2}{1}4^1 \times 8^1 + \binom{2}{2}4^2 \right] + \binom{6}{2}16^2 \times \binom{2}{2}4^2 \times 8^1 \\
&= 29,065,216
\end{aligned}
\tag{D.2.4}$$

Similar calculations can be done to characterize the size of the sequence space at varying distances from the wild-type, as shown in Fig. 4.

D.3 Alternative computational methods for scoring peptides and design

We compare our approach to alternative methods (listed below), based on which we characterize peptide scores akin to HERMES's peptide energy, and when possible, use them to design novel peptides. A detailed comparison between the accuracy of different methods in predicting the affinity of TCR-pMHC interactions and the peptide-induced activation of T-cells is presented in Tables D.7.2 and D.7.3. Comparison between the fidelity of the designs—only according to computational metrics—is shown in Figures D.7.13, D.7.14, and D.7.15. All the designed peptides as well as the code to generate samples are available in our github repository.

- *BLOSUM62 substitution matrix* [87]: The BLOSUM62 substitution matrix characterizes the favorability of substitutions between all pairs of amino acids, and has been used widely to characterize the mutational effects in proteins. As our first (and simplest) benchmark, we used BLOSUM62 to define a peptide energy (score). Specifically, we score a peptide by summing over the BLOSUM62 scores associated with the amino acid substitutions from the wild-type peptide σ^{wt} (the one present in the template structure) to the query peptide sequence σ that we are interested in,

$$E_{\text{peptide}}^{(\text{B62})}(\sigma; \sigma^{\text{wt}}) = \sum_{i=1}^{\ell} B62[\sigma_i^{\text{wt}} \rightarrow \sigma_i] \tag{D.3.1}$$

Notably, the BLOSUM62 predicted peptide energy only depends on the substitutions in the peptide relative to the wild-type amino acid, and does not depend on the structural context of these substitutions (Fig. D.7.2).

For designs with BLOSUM62, we independently sample amino acids along the peptide based on their favorability with respect to the wild-type amino acid at each position. Specifically, at a given position i we interpret the substitution score $B62[\sigma_i^{\text{wt}} \rightarrow \sigma_i]$ from the wild-type to any of the other 19 amino acids as the energy difference between the two states, and by applying a softmax function with temperature T , we evaluate the probability of these possible substitutions. We then use multinomial sampling to draw new amino acids from this probability distribution at each position,

$$\sigma_i \sim \text{Multinomial} \left(\text{softmax} \left[\frac{B62[\sigma_i^{\text{wt}} \rightarrow \sigma]}{T} \right]_{\sigma \in \text{amino-acids}} \right) \tag{D.3.2}$$

For our analyses, we use temperatures $T = 1.0, 2.0, 3.0$ to modulate the variability of the designed peptides.

- *Łuksza et al. M and $d \cdot M$ substitution matrices* [140]: Here, M is an amino acid substitution matrix, and d is a vector of position-specific scale factors for 9-mer peptides. Briefly, they were constructed to

fit the EC_{50} measurements on seven saturation mutagenesis experiments of three different wild-type peptides, against a total of seven TCR-MHC pairs—the very same T-cell activity data that we used to test HERMES in Fig. 3. Specifically, d and M were constructed such that:

$$d_i \cdot M[\sigma_i^{\text{wt}} \rightarrow \sigma_i^{\text{mt}}] \approx \log \frac{EC_{50}^{\text{mt}}}{EC_{50}^{\text{wt}}} + R_{\text{B62}}[\sigma_i^{\text{wt}} \rightarrow \sigma_i^{\text{mt}}] \quad (\text{D.3.3})$$

where the mutant and wild-type peptides only differ at position i , and R_{B62} is a regularization term reflecting the BLOSUM62 substitution matrix; we refer the reader to the Supplementary Methods of ref. [140] for details. Since all peptides considered when constructing d and M are 9-mers, the position scale factors can only be used on 9-mer peptides. We use d and M for scoring in a manner analogous to BLOSUM62:

$$E_{\text{peptide}}^{(d \cdot M)}(\sigma; \sigma^{\text{wt}}) = \sum_{i=1}^{\ell} d_i \cdot M[\sigma_i^{\text{wt}} \rightarrow \sigma_i] \quad (\text{D.3.4})$$

and

$$E_{\text{peptide}}^{(M)}(\sigma; \sigma^{\text{wt}}) = \sum_{i=1}^{\ell} M[\sigma_i^{\text{wt}} \rightarrow \sigma_i] \quad (\text{D.3.5})$$

Similar to BLOSUM62, the predicted peptide energies $E_{\text{peptide}}^{(d \cdot M)}$ and $E_{\text{peptide}}^{(M)}$ only depend on the type substitutions, and not on their structural contexts.

- *MHC position weight matrices* [226]: MHC alleles exhibit distinct sequence-specific preferences for peptide presentation, most notably in the peptide's amino acid composition at anchor positions. Computational models have characterized these preferences with position weight matrices (PWMs) that reflect the the probability of different amino acids at each position for peptides presented by a given MHC allele [142, 141, 11, 226]. As a benchmark for our designs, we use the PWMs from the MHC Motif Atlas [226] to independently sample amino acids at each site along the peptide. The resulting peptide sequences should reflect biases imposed by MHC presentation but not interactions with TCRs.
- *ProteinMPNN* [51]: ProteinMPNN is a commonly used tool for inverse-folding (i.e., designing protein sequences consistent with the backbone of a structure). It is primarily used to sample amino acid sequences conditioned on a protein's backbone structure, and optionally with partial sequence information (i.e., amino acids specified at some of the residues). As ProteinMPNN also outputs probability distributions of amino acids at different sites of the protein, it can also be used to score peptides (i.e., evaluate peptide energy) in a manner similar to HERMES, as

$$E_{\text{pep 1-by-1}}^{(\text{MPNN})}(\sigma; \text{TCR}, \text{MHC}) = \sum_{i=1}^{\ell} \log P^{(\text{MPNN})}(\sigma_i; \text{backbone}, \sigma_{\text{TCR}}, \sigma_{\text{MHC}}, \sigma_{/i}) \quad (\text{D.3.6})$$

where $P^{(\text{MPNN})}(\sigma_i; \text{backbone}, \text{TCR}, \text{MHC}, \sigma_{/i})$ is the ProteinMPNN estimate for the probability of amino acid σ_i at position i of the peptide, given the backbone structure of the whole protein complex, and the amino acid sequence of the MHC σ_{MHC} , TCR σ_{TCR} , and the rest of the peptide $\sigma_{/i}$. We term this scoring scheme as peptide 1-by-1 since we mask the peptide's residues one at a time.

We explored two additional ProteinMPNN scoring schemes that differ in the residues they mask and score. As our second scoring scheme, we simultaneously mask the entire peptide's sequence σ , instead

of one amino-acid at a time, and evaluate the score for the whole peptide $E_{\text{pep}}^{(\text{MPNN})}(\sigma; TCR, MHC)$:

$$E_{\text{pep}}^{(\text{MPNN})}(\sigma; TCR, MHC) = \sum_{i=1}^{\ell} \log P^{(\text{MPNN})}(\sigma_i; \text{backbone}, \sigma_{\text{TCR}}, \sigma_{\text{MHC}}) \quad (\text{D.3.7})$$

For the third scoring scheme, we consider masking and scoring the TCR residues that make contact with the peptide—defined as those whose C- α lie within 12 Å of any peptide residue’s C- α . We call this set **TCR-int** and its corresponding amino acid composition as $\sigma_{\text{TCR-int}}$. In this scheme, both the peptide and the interacting TCR residues are masked simultaneously and scored together:

$$\begin{aligned} E_{\text{pep and tcr}}^{(\text{MPNN})}(\sigma; TCR, MHC) &= \sum_{i=1}^{\ell} \log P^{(\text{MPNN})}(\sigma_i; \text{backbone}, \sigma_{\text{TCR} / \text{TCR-int}}, \sigma_{\text{MHC}}) \\ &+ \sum_{j=1}^n \log P^{(\text{MPNN})}(\sigma_{\text{TCR-int}_j}; \text{backbone}, \sigma_{\text{TCR} / \text{TCR-int}}, \sigma_{\text{MHC}}) \end{aligned} \quad (\text{D.3.8})$$

Similar to HERMES, we considered ProteinMPNN models trained with two noise levels: 0.02 Å (very little noise) and 0.20 Å.

For designs with ProteinMPNN, we use the algorithm’s native functionalities, keeping the TCR and MHC structures fixed (from the design template) and prompting the model to design only the peptide. We use two ProteinMPNN models (trained with 0.02 Å and 0.20 Å noise), and use a temperature of 0.7 for all systems.

- *ESM-IF1* [99]: ESM-IF1, like ProteinMPNN, is an inverse-folding model, but it lacks native support for scoring or designing proteins when only partial sequence information is available.

To access the ESM-IF1 model, we used the code in the esm repository at https://github.com/facebookresearch/esm/tree/main/examples/inverse_folding.

For scoring a peptide sequence given a TCR-MHC context, we used the script `score_log_likelihoods.py` with the `-multichain-backbone` flag, with the same TCR-pMHC structure inputs as for HERMES and ProteinMPNN. This script computes a score for the peptide sequence conditioned on the backbone coordinates of the entire TCR-pMHC complex, but *not* on the sequence of the TCR and the MHC:

$$E_{\text{peptide}}^{(\text{ESM-IF1})}(\sigma; TCR, MHC) = \log P_{\text{peptide}}^{(\text{ESM-IF1})}(\sigma; \text{backbone})$$

To design peptide sequences given a TCR-MHC context, we used the script `sample_sequences.py` with the `-multichain-backbone` flag, and a temperature of 0.7 for all systems, with the same TCR-pMHC structure inputs as for HERMES and ProteinMPNN. Similar to scoring, this script samples peptide sequences conditioned on the backbone coordinates of the entire TCR-pMHC complex, but *not* on the sequences of the TCR and MHC.

- *TCRdock* [26]: TCRdock is a structure prediction algorithm built on top of AlphaFold2 [108], and refined to predict the structure of TCR-pMHC complexes. The algorithm models the structure of TCR-pMHC complexes by leveraging structural templates provided by its curated database, as well as the knowledge of TCR-pMHC canonical binding poses. Crucially, TCRdock’s Predicted Alignment Error (PAE) for TCR-pMHC interfaces was found to well discriminate between true peptide binders

and random decoys [26]. Thus, to benchmark against TCRdock, we interpret this PAE score as peptide energy, and use it to predict TCR-pMHC binding affinities and peptide-induced T-cell activities.

TCRdock takes as input sequences of a TCR, MHC, and peptide, and identifies TCR-pMHC structures in its database with similar amino acid sequences to the query to serve as templates to model the structure of the input TCR-pMHC complex. To examine how template-query sequence similarity influences TCRdock’s predictive accuracy, we ran the algorithm under two conditions: the default mode, which uses any closest-matching templates available, and the *benchmark* mode, which restricts the maximum sequence similarity of templates with the input TCR-pMHC.

We use TCRdock to characterize peptide energies, and to filter the sequences designed by HERMES. However, TCRdock does not provide a natural way to sample peptides for design, and because of its high computational cost, we did not include TCRdock PAE calculations in our design pipeline (e.g., for MCMC sampling).

- *TULIP* [152]: TULIP is a transformer-based model trained to generate either TCR or peptide sequences conditioned on the other binding partners in a TCR-pMHC complex. The conditional likelihood function learned by TULIP can be used to score peptides within a TCR-pMHC context. Specifically, this scoring scheme was used in ref. [152] to predict the T-cell activity measurements from ref. [140], and we benchmarked our predictions against TULIP’s for this data. In Table D.7.3, we present the Spearman correlations between TULIP predictions and the fitted EC_{50} values as reported in ref. [140], which are the same correlations reported in the TULIP paper [152]. Additionally, we provide correlations with the fitted EC_{50} values obtained using the procedure outlined in Section D.1 and eq. D.1.2, consistent with correlations reported in Fig. 3.
- *NetTCR2.2* [104]: NetTCR2.2 consists of a Convolutional Neural Network classifier to predict TCR-peptide binding from the amino acid sequences of the peptide and of all six CDRs from the α and the β chains of a TCR. Because the training set contains many examples of different TCRs binding to the same peptide, but far fewer cases of multiple peptides tested against a given TCR, the model performs best when ranking candidate TCRs for a fixed peptide, as reported in ref. [104]. By contrast, we aim to address the complementary task for ranking peptides for a fixed TCR.

We query NetTCR2.2 through the authors’ web server (<https://services.healthtech.dtu.dk/services/NetTCR-2.2/>). CDR boundaries are assigned by aligning each TCR sequence to the IMGT reference sequences, using code from the TCRdock repository [26]. Of the TCR-pMHC pairs examined in this work, only one system (TCR1, Table D.7.1) appears in the NetTCR2.2 training set, and even then with just a single annotated positive peptide. Therefore, this small overlap is unlikely to bias our predictions.

- *TAPIR* [68]: Similar to NetTCR-2.2, TAPIR consists of a Convolutional Neural Network trained to predict TCR-peptide-MHC binding. As input, TAPIR considers the CDR3’s amino acid sequences, the V and J genes, the peptide’s amino acid sequence, and the MHC allele. We use TAPIR for scoring via the web interface hosted by the authors <https://vcreate.io/tapir/>. We extract V/J genes and CDR3 from a TCR sequence by aligning it to the IMGT reference sequences, using code from the TCRdock repository [26]. We find that five out of nine of the TCR-pMHC systems in this study are present in the training data of TAPIR, but each only with a single annotated positive peptide (1G4 TCR, TCR1, TCR4, TCR5, TCR6, Table D.7.1), and therefore, this small overlap is unlikely to bias our predictions.

Computational modeling of TCR-pMHC templates with AlphaFold3. For six out of seven systems with T-cell activity measurements from ref. [140] (TCRs 2-7), no experimentally-determined structure is available. For these, we generated structures using the AlphaFold3 (AF3) model [2]. Furthermore, to probe how sensitive HERMES is to AF3 modeled inputs, we produced AF3 predictions for every TCR-pMHC complex we studies in this work—both for scoring and design design targets—using the wild-type sequences of all the chains as inputs; see Dataset ?? for these sequences. Each target was folded using both the AF3 server’s option with template search (with default cutoff date of September 9th 2021), and without template, the latter approximating the scenario in which no homologous structures exist. We note, however, that several of TCR-pMHC complexes we analyzed here were likely present in the AF3 training data, so the “template-free” predictions can only serve as an approximate test of true de-novo performance.

We fold TCR-pMHC systems for which we have a structure using restricted amino-acid sequences of the TCR and MHC chains that were extracted with functionalities in the TCRdock codebase, and are equivalent to the sequences that TCRdock would use to predict the structure [26]. Specifically, we first extract the amino-acid sequences from the experimental structure. Then, we extract V/J genes, CDR3 and MHC allele from them using ‘parse_tcr_pmhc_pdbfile.py’, and input them to ‘setup_for_alphafold.py’ to get the restricted amino-acid sequences. For the bispecific Fab antibody-mediated T-cell therapy from ref. [97], we used AF3 to fold and dock the Fab and the pMHC by extracting the amino-acid sequences from the experimental structure. We found that AF3 displayed high uncertainty (as measured by pLDDt) only at the Fab-pMHC interface, and not in the folds of individual chains. The sequences we used as input to AF3 are listed in Dataset ??.

Benchmarking HERMES predictions for T-cell affinity and activity. We compared the performance of HERMES models at predicting the binding affinity and activity of T-cells against substitution matrices (BLOSUM62 [87], and Łuksza *et al*’s TCR-specific $d \cdot M$ and M matrices [140]), the TCR-pMHC structure prediction algorithm TCRdock [26], the sequence-based machine learning algorithms for TCRs (TAPIR [68], NetTCR [104], and TULIP [152]), and the structure-based inverse folding models (ESM-IF1 [99] and ProteinMPNN [51]). Since all of these models are unsupervised in nature (except for Łuksza *et al* activity predictions for TCRs 1-7), we do not expect predictions to be in the same units as the affinity and activity values, and thus, we use Spearman correlation between data and predictions as a metric for model performance.

For predicting TCR-pMHC binding affinities (Table D.7.2), BLOSUM62 leads for the 1G4 TCR system, while template-guided TCRdock tops on the A6 TCR systems. HERMES-*relaxed* with no noise (0.00) achieves slightly lower Spearman correlation but is statistically indistinguishable from these best performers (p-value > 0.05 associated with the difference in the Fisher’s z-transformed Spearman correlations). For 1G4, ESM-IF1 and the Łuksza *et al*’s TCR-specific $d \cdot M$ and M substitution matrices are similarly competitive. HERMES-*relaxed* 0.00 consistently outperforms the sequence-based models (TAPIR, NetTCR-2.2), ProteinMPNN, and TCRdock-*no template* (the TCRdock model without using structural templates similar to the query data). Lastly, choosing the minimum-Rosetta-energy conformation for HERMES-*relaxed* yields correlations with the data that are comparable to the baseline HERMES-*relaxed* model that averages over the energies of 100 different realizations of relaxed peptides (Fisher’s z-test p-value > 0.05). Table D.7.2 shows these benchmarking results in detail.

For predicting T-cell activity (Table D.7.3), HERMES-*fixed* without noise (0.00) achieves the best performance for TCR1, TCR2, and TCR3, and proteinMPNN-0.02 shows competitive performances for TCR1 and TCR3. ProteinMPNN-0.02 leads for TCR6 and H2-scDb, while HERMES-*fixed* 0.00 achieves slightly lower but statistically indistinguishable Spearman correlations (Fisher’s z-test p-value > 0.05). All models struggle to provide reliable predictions for TCR7, with the exception of ProteinMPNN 0.02, which attains a moderate Spearman correlation $\rho = 0.31$ when scoring one peptide amino acid at a time (eq. D.3.6), and $\rho = 0.29$,

when scoring the peptide and the TCR together (eq. D.3.8). For TCR4, which has a poor structural model (AF3 PAE = 2.07), all structure-based models fail to produce reliable predictions, whereas BLOSUM62 achieves a moderate Spearman correlation $\rho = 0.33$. The Łuksza *et al*'s $d \cdot M$ substitution matrix overall performs well on the TCR 1-7 benchmark; however, because it was obtained by fitting on these same data, its predictions are not strictly zero-shot. The structure-based models HERMES and ProteinMPNN consistently outperform the sequence-based models (TAPIR, NetTCR-2.2, and TULIP), as well as the structure-based ESM-IF1. Lastly, choosing the minimum-Rosetta-energy conformation for HERMES-*relaxed* model yields a similar performance to the baseline HERMES-*relaxed* model that averages over the energies of 100 different realizations of relaxed peptides. Table D.7.3 shows these benchmarking results in detail.

Our results do not indicate a clear preference between HERMES-*fixed* and HERMES-*relaxed*: HERMES-*fixed* predicted T-cell activity better, while HERMES-*relaxed* predicted binding affinity between TCR and pMHC better. However, it is important to note that in all systems tested here, the scored peptides are highly similar to the wild-type peptides, with a maximum Hamming distance of just one amino acid from a wild-type peptide found in one of the available structural templates. As a result, relaxation may not be essential for achieving high performance. Interestingly, noised HERMES models underperform compared to the un-noised counterparts. This is in contrast to prior observations, where noised models achieve higher performances in predicting the stability effect of mutations in proteins [238]. Moreover, we observe that ProteinMPNN predictions are robust to the choice of scoring scheme, i.e., scoring peptide amino acids one at a time $E_{\text{pep 1-by-1}}^{(\text{MPNN})}$ (eq. D.3.6), the whole peptide simultaneously $E_{\text{pep}}^{(\text{MPNN})}$ (eq. D.3.7), or peptide and the TCR interface together $E_{\text{pep and tcr}}^{(\text{MPNN})}$ (eq. D.3.8); see Fig. D.7.1.

Lastly, we benchmarked all four HERMES models and two ProteinMPNN models (using peptide-scoring schemes eq. D.3.7 with different noise levels) on AF3-predicted structures, for both activity and affinity predictions (Fig. D.7.4). We generated the AF3 structures both with and without template guidance, and the resulting predictions made with these AF3 structures were compared to performances using the corresponding experimental structures. AF3 produces accurate models for all complexes except the H2-scDb Fab bound to pMHC, for which the docking orientation and conformation were incorrect (Table D.7.4). Expectantly, we saw a significant drop in model performances for both HERMES and ProteinMPNN when using the inaccurate AF3 predictions for H2-scDb (Fig. D.7.4). For both HERMES and ProteinMPNN, replacing experimental structures with AF3 predictions altered predictions for the 1G4 and A6 TCRs binding to pMHC variants, but the changes were not statistically significant (p-value > 0.05 associated with the difference in the Fisher's z-transformed Spearman correlations); see Fig. D.7.4. Finally, for TCR1, we observe no differences in performance at predicting T-cell activity between using the AF3 structures vs. the experimental structures. Comparisons for TCRs 2-7 show insignificant differences between predictions using AF3 models with and without templates.

In Table D.7.4 we report, where available, the C_{α} RMSD between AF3-predicted and experimentally determined structures, together with AF3 confidence metrics computed with and without template guidance. The confidence metrics include the TCR-pMHC interface predicted alignment error (PAE), the interchain predicted TM-score (ipTM), and the predicted TM-score (pTM; not interface-specific). We found ipTM and pTM to be closely related (Table D.7.4). Because structure-based models such as HERMES and ProteinMPNN rely on accurate input structures, Fig. D.7.4B plots their best-performing variants against AF3 confidence. The AF3 model for TCR4—on which both HERMES and ProteinMPNN perform poorly—exhibits the second-lowest ipTM confidence (after H2-scDb). However, the limited range of available AF3 confidence values prevents definitive conclusions about how these metrics impact the accuracy of predictions for peptide mutation-effects on T-cell responses.

performance for the structure-based HERMES and ProteinMPNN models. Specifically, we hypothesized that predictions made on low-confidence structures should be expected to be of poor quality, since structure-based models assume the use of accurate crystal-structure-like structures. To test this, we plotted the models' performance on AF3-folded structures as a function of AF3's confidence metrics (Fig. D.7.4). We considered PAE and interchain predicted Template Modeling (ipTM) scores, two functionally different scores describing the models' confidence in its prediction of the structure at the interface between chains. While PAE was computed specifically on the interface between TCR/Ab and pMHC, we considered the overall ipTM score provided by AF3 to increase the amount of complementary information between the two metrics. We also report the predicted Template Modeling (pTM) scores - which are not specific to interfaces - in Table D.7.4, and find them to be very closely correlated with ipTM scores. Unfortunately, lack of datapoints with very low confidence metrics prevents the drawing of any significant conclusions. In fact, all systems considered except for the H2-scDb Fab and TCR4 have an ipTM score above 0.80 - with TCR4 being right at the boundary - a value that is generally considered to be the threshold for high-quality predictions [2].

Benchmarking HERMES designs against alternative models. Although we did not experimentally test the designs by the different models, we used both peptide presentation by MHC predicted by NetMHCpan and TCRdock's PAE as benchmarking metrics across different design methods. We note that these are only computational metrics and should not be taken as ground truth; this is particularly true for results using TCRdock PAE as a proxy for binding, which are much more likely farther from the ground truth than the easier-to-predict MHC presentation.

Figs. D.7.13, D.7.14, and D.7.15 show the fraction of peptides meeting each criterion for designs produced by various HERMES models, ProteinMPNN, BLOSUM62, and the system's MHC peptide preferences. Consistently, across the three systems, and for all models, the fraction of designs passing the PAE threshold decreases with increasing Hamming distance from the wild-types' sequences (Figs. D.7.13, D.7.14, and D.7.15; panels E, and G). Also consistently, but more surprisingly, models trained with Gaussian noise applied to the structure—for both HERMES and ProteinMPNN—perform worst than their noise-less counterparts (Figs. D.7.13, D.7.14, and D.7.15; panels E, and G).

HERMES models are consistently the best at designing peptides that are scored well for presentation by the respective MHC alleles (panel C of Figs. D.7.13, D.7.14, and D.7.15). No single model stands out as being consistently best across the three systems at designing peptides above the TCRdock PAE threshold: HERMES outperforms ProteinMPNN in the NY-ESO system (Fig. D.7.13), performs similarly in the EBV system (Fig. D.7.14), and performs worse in the MAGE system (Fig. D.7.15). BLOSUM62 makes worse designs compared to HERMES and ProteinMPNN based on TCRdock PAE scores. Surprisingly, ESM-IF1 designs have the lowest success in MHC presentation, but show better TCRdock PAE scores compared to HERMES-*relaxed* for the MAGE system (Figs. D.7.13, D.7.14, and D.7.15). Designs sampled from the MHC motif, while scoring well on MHC presentation, consistently score poorly for TCRdock PAE. We expected HERMES-*relaxed* to design better binders than HERMES-*fixed* at greater sequence distances from the wild-types. However, this holds only for the NY-ESO system (Fig. D.7.13) and not for EBV or MAGE (Figs. D.7.14, D.7.15). Overall, a broader benchmarking study, ideally with experimental validation, is needed to rigorously evaluate the effectiveness of different peptide design methods.

To assess how using AF3 structures as templates influence design, we ran the HERMES-*fixed* design protocol using experimental and AF3 structures as templates and compared the resulting peptide likelihood matrices (as position weight matrices or PWMs) for the three systems (Fig. D.7.12). Across NY-ESO, EBV, and MAGE, the resulting PWMs were largely consistent across the two template choices: most peptide positions showed the same preferences, and when variability did appear it was typically confined to chemically similar residues. There are two noteworthy exceptions: in NY-ESO, the top choice in position 7 of the peptide was flipped from Threonine and Aspartic Acid when AF3 template was used (Fig. D.7.12A). In MAGE, using

the AF3 model introduced a pronounced preference for glutamic acid at position 1 (Fig. D.7.12C), a residue that we have already shown to be important for eliciting T-cell activity (Fig. 4). Aside from these cases, AF3-derived structures preserve the peptide specificity landscape learned by HERMES.

D.4 Experimental protocol to measure peptide activities

Artificial antigen-presenting cell lines. We obtained the full-length coding sequences of HLA-A*02:01, HLA-A*01:01, and HLA-B*35:01 from the IMGT/HLA database [193]. Gene fragments were synthesized by Genscript and were cloned into the lentiviral pLVX-EF1 α -IRES-Puro (Clontech) vector. We used Lipofectamine 3000 Transfection Reagent (Invitrogen) to transduce the 293T packaging cell line (ATCC CRL-3216) with the psPAX2 packaging plasmid (Addgene plasmid #12260), the pMD2.G envelope plasmid (Addgene plasmid #12259), and the HLA-encoding lentiviral vector. The lentivirus-containing supernatant was collected after 24 and 48 hours post-transfection and was filtered through a 0.45 μ m SFCA filter (Thermo Scientific). The lentivirus was concentrated using Lenti-X Concentrator (Takara Bio) according to the manufacturer's protocol, resuspended in DPBS (Gibco), and stored at -80C for future use. To generate monoallelic HLA cell lines for HLA-A*02:01 and HLA-B*35:01, HLA-deficient K562 cells (ATCC CCL-243) were transduced with lentivirus. Since K562 cells have a high expression of both MAGE-A3 and Titin antigens, we transduced murine EL4 cell lines (ATCC TIB-39) with HLA-A*01:01 lentivirus to produce HLA-A*01:01 presenters lacking the natural expression of the tested antigens. At 48 hours post-transduction, K562 cells were transferred to Iscove's Modified Dulbecco's Medium containing 10% FBS, 2mM L-glutamine, 100 U/mL penicillin/streptomycin, and 2 μ g/mL puromycin (Gibco). EL4 cells were transferred to RPMI media containing 10% FBS, 2mM L-glutamine, 100 U/mL penicillin/streptomycin, and 2 μ g/mL puromycin (Gibco). The antibiotic selection continued for one week before transferring cells to the antibiotic-free media.

TCR-expressing Jurkat 76.7 cell lines. Nucleotide sequences of TCR α and TCR β chains were generated using stitchr [86]. Both chains were modified to incorporate murine constant segments (TRAC*01 and TRBC2*01) to enhance TCR surface expression. For NYESO and EBV-specific TCRs, the TCR α and TCR β chains were connected through a P2A ribosomal skip motif, synthesized as a single bicistronic gene fragment by Genscript, and cloned into the lentiviral pLVX-EF1 α -P2A-mCherry-IRES-Puro (Clontech). For MAGE-specific TCR, gene fragments for the TCR α and TCR β chains were separately synthesized by Genscript and cloned into either the lentiviral pLVX-EF1 α -P2A-mCherry-IRES-Puro (Clontech) or pLVX-EF1 α -IRES-G418 (modified in-house from pLVX-EF1 α -IRES-Puro). The 293T packaging cell line (ATCC CRL-3216) was transfected with TCR-encoding lentiviral vectors, psPAX2 packaging plasmid (Addgene plasmid #12260), and pMD2.G envelope plasmid (Addgene plasmid #12259) using Lipofectamine 3000 Transfection Reagent (Invitrogen). The lentivirus-containing supernatant was collected after 24 and 48-hours after transfection, filtered through a 0.45 μ m SFCA filter (Thermo Scientific), concentrated using Lenti-X Concentrator (Takara Bio) according to the manufacturer's protocol, resuspended in DPBS (Gibco), and stored at -80C for future use. To generate Jurkat cells expressing the paired TCR, TCR-null CD8-positive Jurkat 76.7 cells with NFAT-eGFP reporter were transduced with lentivirus. At 48 hours post-transduction, cells were transferred to RPMI media containing 10% FBS, 2mM L-glutamine, 100 U/mL penicillin/streptomycin, and 1 μ g/mL puromycin (Gibco). For MAGE-specific cell line generation Jurkat cells were co-transduced with lentiviruses encoding TCR α and TCR β chains and cultured with both 1 μ g/mL puromycin and 500 μ g/mL Genectin (Gibco). The antibiotic selection continued for one week. The abTCR expression on the surface was confirmed by mCherry expression and staining with a 1:100 dilution of anti-mouse TCR β APC-Fire750 (clone H57-597, 109246, Biolegend).

Specificity validation of TCR-expressing Jurkat. Original cognate peptides and their variants were synthesized as a crude peptide library with by Genscript, and then diluted to a 4mM stock solution in

DMSO (Sigma). To examine the specificity of the TCR-expressing Jurkat cell lines, we co-cultured 100,000 Jurkat cells with equal amounts of artificial APCs expressing the corresponding HLA allele in a 96-well round-bottom tissue culture plate. We tested each specificity—MAGE, EBV, NYESO—under 96 different conditions. These included one or two positive control original peptides and an unstimulated “no peptide” control. The cells were cultured in RPMI media supplemented with 10% FBS, 2mM L-glutamine, 100 U/mL penicillin/streptomycin, and 1 μ g/ml each of anti-human CD28 (BD Biosciences, 555725) and CD49d (BD Biosciences, 555501). The co-cultures were pulsed with 1 μ M peptide in triplicate. We added DMSO to unstimulated control wells (no peptide) at concentrations equivalent to those in the peptide-stimulated wells. After incubating the cells for 24 hours at 37°C and 5% CO₂, we washed them once in DPBS (Gibco) and resuspended them in 150 μ l DPBS for flow cytometry analysis using a custom-configured BD Fortessa with FACSDiva software (Beckton Dickinson). We determined GFP percentages using FlowJo version 10.8.1 software (BD Biosciences). Alternatively, we measured the GFP percentage in wells of a 96-well round-bottom plate using Celcyte X (Echo), using 4x objective in spheroid mode, acquisition in green channel (800 ms exposure 3dB gain). Image analysis was done with CellCyte Studio software with total spheroid area recipe and standard defaults (50 a.u. Contrast Sensitivity, 2 a.u Smoothing, 100 μ m² filled hole size 100 μ m Min. object size), relative GFP+ spheroid area was used as output. For NY-ESO, we compared the T-cell activity level measured using flow cytometry with fluorescence microscopy, and show consistency between the two (Fig. D.7.7). Given the consistency between the two experimental approaches, we relied on fluorescence microscopy only to measure GFP levels induced by different peptides for MAGE and EBV.

Selecting the GFP expression threshold for T-cell activity. Experimental measurements are done across three replicates, and the reported GFP levels in Figs. 4, D.7.8 for each condition are averaged over the three replicates. We consider a peptide to have activated T-cells if its induced replicate-averaged GFP level met two criteria: 1) it exceeded 0.5%, and 2) it surpassed the mean plus three standard deviations of the GFP level negative controls (no-peptide). According to these criteria, the GFP activation cutoffs were 0.5% for the NY-ESO system, 0.64% for the EBV system, and 0.76% for the MAGE system.

D.5 Experimental activities of the designed peptides

Comparing design success rates across the three TCR-pMHC systems. Among the designed peptides, 50% in the NY-ESO system (Fig. 4A), 16% in the EBV system (Fig. 4B), and 26% in the MAGE system induce significant T-cell activities. However, it should be noted that the designed NY-ESO peptides were closer in sequence to their wild-type templates (2-7 amino acid differences) than the peptides designed for the EBV (3-9 differences) or MAGE (3-8 differences) systems. We find that the activity of designs decays with increasing distance from the respective wild-type, with almost no designs beyond 5 amino-acid distance from the wild-types activating their respective T-cells (Fig. 4, D.7.9, D.7.10, D.7.11). This underscores the difficulty of exploring peptide variants far from the wild-type sequences. If any positive designs do exist at these greater distances, they represent an exponentially smaller fraction of the sequence space, and when in complex with the TCR-MHC, are likely to adopt a different binding mode compared to that of the wild-type.

Minor adjustments to the design protocols, aimed at better exploring the design space, may have also contributed to the observed differences in performance across the systems. For NY-ESO, which achieved the highest design success rate, peptides with highest PAE scores were chosen for experimental validation, whereas in EBV and MAGE, peptides were sampled uniformly above the set PAE thresholds. In MAGE, fixing the amino-acid at position 1 to a Glutamic Acid (E) for the one-third of the designs led to an elevated success rate in this subset (Fig. D.7.11D); E₁ is shared between the two wild-type peptides and this design choice allowed us to assess the importance of this amino acid for function.

The design potential of different HERMES models. In panel A of Figures D.7.9 (NY-ESO), D.7.10 (EBV), and D.7.11 (MAGE), we show the fidelity of designs using the four HERMES models: HERMES-*fixed* with no noise, HERMES-*fixed* with 0.5 Å noise, HERMES-*relaxed* with no noise, and HERMES-*relaxed* with 0.5 Å noise. Although PyRosetta relaxations allows HERMES-*relaxed* to explore the sequence space more broadly, the resulting designs that are made at greater distances from the wild-types seldom succeed in activating T-cells. This pattern is also observed with using models trained with 0.5 Å Gaussian noise. Indeed, successful designs lie within five amino acid substitutions from the wild-types, and for these designs, the un-noised HERMES-*fixed* model demonstrates the highest overall fidelity. To better explore the peptide energy landscape and to design sequences further from the wild-types, it may be necessary to adopt a more specialized relaxation model that accounts for the intricacies of TCR-pMHC poses, while being consistent with the neural network's model of protein environments.

Comparing TCRdock and AF3 as design filters. Using our T-cell activity measurements as the ground truth, we sought to quantify how well the TCRdock and AF3 PAE scores can be used as predictors of T-cell activation and filtering mechanisms for peptide design. For this purpose, we focus primarily on the NY-ESO system, for which we validated a balanced mix of positive and negative designs, chosen based on their TCRdock PAE scores. Here, we found that TCRdock PAE could moderately distinguish true positive from true negative designs (AUROCs = 0.91 and 0.88, Fig. D.7.16A). Notably, using TCRdock PAE resulted in a very low false negative rate (0.03%) but a considerably higher false positive rate (50%); see Fig. D.7.8A top. AF3 PAE scores performed equivalently to TCRdock PAE in differentiating positives from negatives (AUROCs = 0.89 and 0.94, Fig. D.7.16A), though they correlated only moderately with TCRdock's PAE values predictions (Spearman correlation $\rho = 0.5$, Fig. D.7.8C). Indeed, some peptides deemed positive by TCRdock were classified as negative by AF3, and vice versa (Fig. D.7.8C).

When restricting the analysis to peptides that were deemed as positive based on their TCRdock and AF3 PAE scores, we did not expect either TCRdock or AF3 PAE to effectively discriminate between peptides that did or did not elicit a T-cell response in our experimental assay (i.e., true vs. false positives). However, we found that both TCRdock PAE and AF3 PAE scores retained significant predictive power in the NY-ESO system (AUROCs = 0.81 and 0.77, and AUROCs = 0.80 and 0.89, Fig. D.7.16A). In the EBV and MAGE systems, the TCRdock and AF3 PAE values for peptides that we classified positive showed lower predictive power in distinguishing true from false positives compared to the NY-ESO system (Fig. D.7.16B-C). The PAE scores of TCRdock and AF3 differed most markedly in the MAGE system (Spearman correlation $\rho = 0.09$ for MAGE), with TCRdock assigning relatively unfavorable PAE scores to the wild-type peptides, whereas AF3 ranked those same peptides near the top (Fig. D.7.8I).

The PAE values reported by the structure prediction algorithms have been shown to be noisy indicators of protein folding fidelity and design quality [196]. While larger experimental libraries will be needed to draw definitive conclusions, it appears that PAEs from TCRdock and AF3 have some complementary predictive strengths, and therefore, combining their information could lead to more robust design decisions. Moreover, both of these scores appear to be uncorrelated or only weakly correlated with HERMES scores across systems (Figs. D.7.8, D.7.13H, D.7.14H, D.7.15H), suggesting that AF3 and TCRdock PAE's can serve as complementary filters to HERMES during design.

HERMES energy as design filter. We further sought to assess whether HERMES energy score itself could serve as a predictor of T-cell activation and as a filtering criterion for design. For each of the three systems, we scored all peptides we designed with all four combinations of HERMES protocols, regardless of which protocol was originally used for design. For each system, we then computed two sets of ROC curves, separately for peptides designed with HERMES-*fixed* and those designed with HERMES-*relaxed*.

We found that scores from the HERMES-*relaxed* models consistently performed poorly in predicting T-cell activation, with only one out of six cases achieving an AUROC better than random; HERMES-*relaxed* 0.00 as scoring criterion on the peptides designed with HERMES-*fixed* in the EBV system showing AUROC = 0.66 (Fig. D.7.16B). In contrast, HERMES-*fixed* 0.00 was consistently the best filtering criterion, outperforming even TCRdock and AF3 PAE score in several cases: in the EBV system, HERMES-*fixed* 0.00 scores on peptides designed by HERMES-*fixed* reached AUROC = 0.74; in the MAGE system, AUROC were 0.65 and 0.89 for and peptides designed with HERMES-*fixed* and HERMES-*relaxed*, respectively (Fig. D.7.16). These results highlight two main takeaways: first, that HERMES-*fixed* 0.00 is the most reliable model for both scoring and designing peptides to elicit T-cell response; second, selecting high-scoring peptides using this model can improve hit rates, though potentially at the cost of reduced diversity in the peptide pool.

D.6 T-cell and MHC specificity as the entropy of the associated peptide distribution

We characterize the specificity of a TCR-MHC complex and that of the MHC by the diversity of peptides that they can recognize. For TCR-MHC specificity, we starting with a TCR-pMHC structural template, and use HERMES-*fixed* model to generates a peptide position weight matrix (PWM) that captures the ensemble of peptides recognized by the TCR and presented by the MHC. In Figure 5, we analyzed 105 human and 36 mouse MHC class I TCR-pMHC structures—curated by the TCRdock template database [26]—which included only MHC alleles represented in at least two structures. For each MHC allele in this set, we then characterized the distribution of peptides that the MHC can present (independently of the TCR) using PWMs inferred by the MHC Motif Atlas [226]. In both cases, we use the entropy of these PWMs, $S = -\sum_{\alpha,i} p_i(\alpha) \log_2 p_i(\alpha)$, as a proxy for degeneracy of the target (TCR-MHC or MHC), where $\alpha \in \{1, \dots, 20\}$ denotes the amino acid types, and $i \in \{1, \dots, \ell\}$ are positions along the peptide.

D.7 Predicting MHC-I peptide presentation preferences with HERMES.

HERMES is designed to be broadly applicable to diverse protein-interaction contexts. As such, we attempted to use HERMES-*fixed* 0.00 model to recapitulate the ensemble of peptides presented by an MHC-I allele. To do so, we gathered 416 peptide-MHC-I crystal structures from the TCR3D database [83] (Table D.7.7). For each structure, we computed a peptide position position weight matrix (PWM) with the HERMES-*fixed* 0.00 design protocol, conditioning on the surrounding MHC structure and the masked wild-type peptide as the template. The resulting PWMs were then averaged across complexes that shared the same MHC allele and peptide length to produce an allele-specific peptide motif (for a given peptide length).

We compared these HERMES-generated PWMs with the corresponding PWMs from the MHC Motif Atlas [226], which we consider as ground truth; we used relative entropy (KL-Divergence), $D_{KL} = \sum_{\alpha,i} p_i^{\text{motif}}(\alpha) \log_2 \frac{p_i^{\text{motif}}(\alpha)}{p_i^{\text{HERMES}}(\alpha)}$ as well as the entropy difference between the two PWMs as metrics; here, $p_i^{(\cdot)}(\alpha)$ is the probability of amino acid α at position i of a peptide, given the morel specified in the superscript. When structural coverage is extensive (roughly more than 30 structures of distinct peptides with a given MHC allele), HERMES-derived motifs align closely with those from MHC Motif Atlas, as quantified by the low KL-divergence and comparable entropy values (Fig. D.7.17). Under sparse coverage, HERMES motifs begin to underestimate the peptide diversity that can be presented by an MHC allele (measured by motif entropy), a limitation we attribute to incomplete sampling of peptide backbone conformations by HERMES-*fixed* (Fig. D.7.17C). Overall, we see a clear pattern that the KL-Divergence increases with decreasing number of structural templates for a given MHC allele (Fig. D.7.17B). Importantly, even with only a handful of templates, HERMES still captures substantially more diversity than a naïve baseline PWM generated by simply aligning the template peptides themselves (Fig. D.7.17A). Moreover, in almost all cases, HERMES

generated PWMs are more similar (lower KL-Divergence) to that of the MHC Motif Atlas than the naïve baseline PWM are (Fig. D.7.17A).

These results suggest that accurate modeling of MHC presentation with HERMES-*fixed* design protocol requires both sufficient structure templates and peptide diversity. It should be noted that MHC Motif Atlas's allele-specific motifs—derived from the large body of peptide binding data curated for individual MHC-I alleles—are very reliable, so we continue to use them as the benchmark for assessing motif specificity. Developing design protocols that better sample peptide backbone conformations remains an important avenue for future work.

TCR / Ab	(PDBid, WT peptide)	MHC allele
1G4 TCR	(2BNR [42], SLLMWITQC); (2BNQ [42], SLLMWITQV)	HLA-A*02:01
A6 TCR	(1AO7 [75], LLFGYPVYV); (1QSF [61], LLFGYPVAV)	HLA-A*02:01
TCR1	(5D2N [258], NLVPMVATV)	HLA-A*02:01
TCR2	(AF3 model, NLVPMVATV)	HLA-A*02:01
TCR3	(AF3 model, NLVPMVATV)	HLA-A*02:01
TCR4	(AF3 model, IMDQVPFSV)	HLA-A*02:01
TCR5	(AF3 model, IMDQVPFSV)	HLA-A*02:01
TCR6	(AF3 model, IMDQVPFSV)	HLA-A*02:01
TCR7	(AF3 model, GRLKALCQR)	HLA-B*27:05
H2-scDb (Fab)	(6W51 [97], HMTEVVRHC)	HLA-A*02:01

Table D.7.1: Structure templates used for the binding affinity and T-cell activity analyses. The first two rows contain information related to the analyses on binding affinity data, the rest are related to T-cell activity data. Every structure identified by a 4-letter PDBid was downloaded from the Protein Data Bank. The raw PDB file for 5D2N [258] has two copies of the TCR-pMHC complex, so we deleted one before running any analysis. Similarly, the raw PDB file for 6W51 [97] has the Antibody and pMHC coordinates of two different copies within the crystal, so we expanded the crystal using PyMOL's `symexp` command and kept a single bound Antibody-pMHC complex. The sequence inputs used to fold TCR-pMHC complexes with AlphaFold3 [2] can be found in Dataset ??.

Model type	Model	1G4 TCR	A6 TCR
Substitution Matrix	BLOSUM62	0.83 (0.00)	0.56* (0.02)
	Luksza et al. M	0.44* (0.12)	0.24 (0.38)
	Luksza et al. $d \cdot M$	0.48* (0.08)	0.29 (0.27)
Structure Predictor	TCRdock	0.62* (0.02)	0.88 (0.00)
	TCRdock- <i>no template</i>	0.25 (0.39)	-0.14 (0.62)
Sequence-based	TAPIR	0.29 (0.32)	0.35 (0.19)
	NetTCR-2.2	-0.25 (0.40)	-0.27 (0.36)
Structure-based	ESM-IF1	0.38* (0.17)	0.17 (0.53)
	ProteinMPNN 0.02 Pep 1-by-1	-0.01 (0.97)	0.49 (0.05)
	ProteinMPNN 0.20 Pep 1-by-1	0.12 (0.69)	0.29 (0.27)
	ProteinMPNN 0.02 Pep	-0.01 (0.97)	0.49 (0.06)
	ProteinMPNN 0.20 Pep	0.14 (0.63)	0.29 (0.27)
	ProteinMPNN 0.02 Pep and TCR	0.07 (0.81)	0.54 (0.03)
	ProteinMPNN 0.20 Pep and TCR	-0.08 (0.79)	0.26 (0.33)
Structure-based	HERMES- <i>fixed</i> 0.00	0.29 (0.31)	0.63* (0.01)
	HERMES- <i>fixed</i> 0.50	0.19 (0.51)	-0.28 (0.29)
	HERMES- <i>relaxed</i> 0.00	0.72* (0.00)	0.56* (0.03)
	HERMES- <i>relaxed</i> 0.50	0.65* (0.01)	0.06 (0.83)
	HERMES- <i>relaxed</i> -min_energy 0.00	0.58* (0.03)	0.71* (0.00)
	HERMES- <i>relaxed</i> -min_energy 0.50	0.52* (0.06)	0.51 (0.04)

Table D.7.2: Benchmarking of models for predicting TCR-pMHC binding affinities. The table lists the Spearman correlation ρ (p – value) of models’ predictions against experimentally measured binding affinities for (i) 1G4 TCR in complex with HLA-A*02:01 binding to the variants of the NY-ESO-1 peptide, and (ii) the A6 TCR in complex with HLA-A*02:01 binding to the variants of the Tax peptide. The HERMES model names specify the scoring scheme (fixed, relaxed, and relaxed with peptide energy computed from the relaxed conformation with the minimum Rosetta energy), and the noise amplitude (in Å) injected to the coordinates of the structure data during training. Similarly, the ProteinMPNN model names capture scoring scheme (1-by-1 (eq. D.3.6), the whole peptide (eq. D.3.7), and the peptide and TCR interface together (eq. D.3.8)), and the noise amplitude injected to the training data. We show in **bold** the best model for each dataset, and any model whose performance is not significantly different from the best model (p-value > 0.05 for the difference in the Fisher’s z-transformed Spearman correlations) is indicated with an asterisk (*).

Model type	Model	H2-scDb (Fab)	TCR1	TCR2	TCR3	TCR4	TCR5	TCR6	TCR7
Sub. Matrix	BLOSUM62	-0.05 (0.52)	-0.07 (0.47)	0.14 (0.20)	0.38 (0.00)	0.33* (0.01)	0.42* (0.00)	0.55* (0.00)	-0.01 (0.89)
	Luksza et al. M	0.19 (0.02)	†0.25 (0.01)	†0.39 (0.00)	†0.53* (0.00)	†0.25* (0.03)	†0.27 (0.00)	†0.36* (0.00)	†0.01 (0.94)
	Luksza et al. $d \cdot M$	0.46* (0.00)	†0.50* (0.00)	†0.66* (0.00)	†0.40 (0.00)	†0.38 (0.00)	†0.55 (0.00)	†0.42* (0.00)	†0.27* (0.00)
Struc. Pred.	TCRdock	N/A	0.02 (0.83)	0.13 (0.24)	0.21 (0.03)	0.18 (0.15)	-0.02 (0.83)	0.19 (0.04)	0.06 (0.50)
	TCRdock-no template	N/A	-0.03 (0.77)	0.44 (0.00)	0.11 (0.27)	-0.22 (0.08)	0.28 (0.00)	0.14 (0.14)	0.06 (0.50)
Seq.-based	TAPIR	N/A	0.36 (0.00)	0.36 (0.00)	-0.00 (0.97)	0.12 (0.32)	0.39* (0.00)	0.08 (0.42)	-0.01 (0.94)
	NetTCR-2.2	N/A	-0.05 (0.59)	0.20 (0.06)	0.36 (0.00)	-0.21 (0.09)	0.23 (0.01)	0.20 (0.03)	-0.22 (0.02)
	TULIP	N/A	0.12 (0.24)	0.18 (0.09)	-0.12 (0.20)	0.04 (0.73)	0.24 (0.00)	0.29 (0.00)	N/A
	TULIP* (from [152])	N/A	0.47* (0.00)	0.44 (0.00)	0.14 (0.07)	0.23* (0.00)	0.20 (0.01)	0.09 (0.25)	N/A
Struc.-based	ESM-IF1	0.12 (0.13)	0.43* (0.00)	0.41 (0.00)	0.19 (0.05)	-0.15 (0.22)	0.28 (0.00)	0.25 (0.01)	0.08 (0.40)
	ProteinMPNN 0.02 1-by-1	0.53* (0.00)	0.46* (0.00)	0.53 (0.00)	0.49* (0.00)	-0.23 (0.06)	0.45* (0.00)	0.42* (0.00)	0.31 (0.00)
	ProteinMPNN 0.20 1-by-1	0.03 (0.65)	0.27 (0.01)	0.46 (0.00)	0.40 (0.00)	-0.23 (0.06)	0.26 (0.00)	0.49 (0.00)	0.02 (0.79)
	ProteinMPNN 0.02 Pep	0.55 (0.00)	0.46* (0.00)	0.52 (0.00)	0.49* (0.00)	-0.28 (0.02)	0.45* (0.00)	0.41* (0.00)	0.31 (0.00)
	ProteinMPNN 0.20 Pep	0.04 (0.64)	0.27 (0.01)	0.44 (0.00)	0.40 (0.00)	-0.23 (0.05)	0.25 (0.00)	0.48* (0.00)	0.02 (0.83)
	ProteinMPNN 0.02 Pep and TCR	0.50* (0.00)	0.44* (0.00)	0.53 (0.00)	0.34 (0.00)	-0.24 (0.05)	0.51* (0.00)	0.46* (0.00)	0.29* (0.00)
	ProteinMPNN 0.20 Pep and TCR	-0.02 (0.84)	0.20 (0.04)	0.47 (0.00)	0.30 (0.00)	-0.23 (0.06)	0.32 (0.00)	0.48* (0.00)	-0.02 (0.81)
Struc.-based	HERMES-fixed 0.00	0.46* (0.00)	0.56 (0.00)	0.71 (0.00)	0.59 (0.00)	-0.06 (0.60)	0.37 (0.00)	0.42* (0.00)	-0.01 (0.95)
	HERMES-fixed 0.50	0.26 (0.00)	0.42* (0.00)	0.61* (0.00)	0.54* (0.00)	0.03 (0.83)	0.26 (0.00)	0.31 (0.00)	0.02 (0.82)
	HERMES-relaxed 0.00	0.29 (0.00)	0.31 (0.00)	0.57 (0.00)	0.55* (0.00)	-0.20 (0.11)	0.37 (0.00)	0.42* (0.00)	0.14 (0.15)
	HERMES-relaxed 0.50	0.16 (0.04)	0.37 (0.00)	0.50 (0.00)	0.45* (0.00)	-0.16 (0.20)	0.23 (0.00)	0.06 (0.49)	0.08 (0.37)
	HERMES-relaxed-min_energy 0.00	0.18 (0.02)	0.32 (0.00)	0.56* (0.00)	0.50* (0.00)	-0.10 (0.39)	0.37 (0.00)	0.45* (0.00)	0.04 (0.65)
	HERMES-relaxed-min_energy 0.50	0.13 (0.09)	0.39 (0.00)	0.51 (0.00)	0.46* (0.00)	-0.12 (0.34)	0.15 (0.07)	0.17 (0.07)	0.08 (0.38)

Table D.7.3: Benchmarking of models for predicting peptide-induced T-cell activities. The table lists the Spearman correlation ρ (p – value) of models’ predictions against experimentally measured T-cell activities. The correlations for TCR1-7 are reported based on the EC₅₀ values obtained using the procedure outlined in Section D.1 and Eq. D.1.2, consistent with correlations reported in Fig. 3. In addition, we present the Spearman correlations between TULIP predictions and the fitted EC₅₀ values as reported in ref. [140], which are the same correlations reported in the TULIP paper [152] (TULIP* row). The TULIP* correlations also include datapoints for which T-cell responses were below the detection limit, and which we exclude from our analysis (gray points in Fig. 3). We note that the substitution matrix - with (C) and without (C/d) the position-based scaling d - by Luksza et al. [140] was constructed precisely using experimental activity data from TCR1-7; we note predictions made on these measurements with a dagger† to indicate that they are not representative of the matrix’ ability to predict T-cell activity in new settings. The HERMES model names specify the scoring scheme (fixed, relaxed, and relaxed with peptide energy computed from the relaxed conformation with the minimum Rosetta energy), and the noise amplitude (in Å) injected to the coordinates of the structure data during training. Similarly, the ProteinMPNN model names capture scoring scheme (1-by-1 (eq. D.3.6), the whole peptide (eq. D.3.7), and the peptide and TCR interface together (eq. D.3.8)), and the noise amplitude injected to the training data. We show in **bold** the best model for each dataset, and any model whose performance is not significantly different from the best model (p-value> 0.05 for the difference in the Fisher’s z-transformed Spearman correlations) is indicated with an asterisk (*).

TCR / Ab	with templates				without templates			
	RMSD	PAE	ipTM	pTM	RMSD	PAE	ipTM	pTM
1G4 TCR	0.72	1.06	0.94	0.94	0.82	1.08	0.93	0.94
1G4 TCR w/ 9V on peptide	0.72	1.07	0.94	0.94	0.82	1.08	0.93	0.94
A6 TCR	0.87	0.91	0.95	0.95	0.83	0.91	0.95	0.95
A6 TCR w/ 8A on peptide	0.67	1.00	0.94	0.94	0.73	1.03	0.94	0.95
TCR1	0.92	1.52	0.88	0.91	1.12	2.46	0.82	0.86
TCR2	N/A	1.63	0.86	0.89	N/A	1.86	0.84	0.87
TCR3	N/A	0.84	0.95	0.95	N/A	0.86	0.95	0.95
TCR4	N/A	2.07	0.80	0.83	N/A	2.01	0.84	0.86
TCR5	N/A	1.17	0.91	0.92	N/A	1.19	0.91	0.93
TCR6	N/A	1.14	0.92	0.94	N/A	1.16	0.91	0.93
TCR7	N/A	1.37	0.84	0.87	N/A	1.42	0.89	0.91
H2-scDb (Fab)	13.89	19.44	0.48	0.54	11.57	19.2	0.47	0.55
eng. TCR w/ MageA3 peptide	0.62	1.03	0.92	0.94	0.69	1.03	0.92	0.94
eng. TCR w/ Titin peptide	0.64	1.05	0.92	0.94	0.65	1.03	0.92	0.94
TK3 TCR w/ HPVQ peptide	1.05	1.11	0.92	0.94	1.05	1.05	0.93	0.94
TK3 TCR w/ HPVQ EQ5 peptide	1.01	1.07	0.92	0.94	1.06	1.07	0.92	0.94

Table D.7.4: **AlphaFold3 folding metrics for structures used in our analyses.** Root-mean-square deviation (RMSD) is computed between the α -C's of AF3 predicted structures and the corresponding experimentally determined structure. The reported PAE values are averaged across the interfaces of TCR (α and β chains) and pMHC. ipTM and pTM values are reported from the AF3's confidence output. For template-guided predictions, we used the default template cutoff in the AlphaFold Server (September 9th 2021). If an AF3 prediction incorrectly docked the TCR on the opposite side of the pMHC, the run was discarded and repeated. In the case of the H2-scDb bi-specific Fab-pMHC complex, although AF3 placed the Fab on the correct side of the pMHC, the docking pose was consistently inaccurate, resulting in high RMSD and PAE values. This was the only system for which AF3 failed to produce a realistic structure model. Additional details on experimental structures, peptide, TCR, and MHC sequences used in AF3 runs can be found in Tables D.7.1, D.7.5, Dataset ???. Example AF3-generated structures are shown in Figure D.7.5.

System	TCR	(PDBid, WT peptide)	MHC allele
NY-ESO	1G4 TCR	(2BNR [42], SLLMWITQC);	HLA-A*02:01
		(2BNQ [42], SLLMWITQV)	
EBV	TK3 TCR	(3MV7* [85], HPVGEADYFEY);	HLA-B*35:01
		(4PRP* [136], HPVQGADYFEY)	
MAGE	eng. TCR	(5BRZ* [187], EVDPIGHLY);	HLA-A*01:01
		(5BS0* [187], ESDPIVAQY)	

Table D.7.5: **Structure templates used for peptide design.** For the PDB entries marked with an asterisk (*), we used the curated structures from the TCRdock database [26] instead of the PDB entries.

MHC allele	(PDBid, WT peptide)
HLA-A*01:01	(5BS0, ESDPIVAQY); (5BRZ, EVDPIGHLY)
HLA-A*02:01	(4FTV, LLFGYPVYV); (5D2L, NLVPMVATV); (2F53, SLLMWITQC); (5TEZ, GILGFVFTL); (6VQO, HMTEVVRHC); (3HG1, ELAGIGILTV); (2P5E, SLLMWITQC); (7N6E, YLQPRTFLL); (6RP9, SLLMWITQV); (5C08, RQWGPDPAAV); (3H9S, MLWGYLQYV); (5C0B, RQFGPDFPTI); (6EQA, AAGIGILTV); (5EUO, GILGFVFTL); (3D39, LLFGYPVYV); (6VRN, HMTEVVRHC); (5YXU, KLVALGINAV); (5EU6, YLEPGPVTV); (1BD2, LLFGYPVYV); (1OGA, GILGFVFTL); (2VLK, GILGFVFTL); (5MEN, ILAKFLHWL); (3D3V, LLFGYPVYV); (1AO7, LLFGYPVYV); (5ISZ, GILGFVFTL); (5JHD, GILGFVFTL); (6DKP, ELAGIGILTV); (1QSE, LLFGYPVYV); (3QDM, ELAGIGILTV); (5JZI, KLVALGINAV); (2P5W, SLLMWITQC); (5C0C, RQFGPDWIVA); (7N1E, RLQSLQTYV); (4QOK, EAAGIGILTV); (5E6I, GILGFVFTL); (2VLR, GILGFVFTL); (6RSY, RMFPNAPYL); (3QEQ, AAGIGILTV); (3O4L, GLCTLVAML); (5C0A, MVWGPDPPLYV); (5C07, YQFGPDFPIA); (5D2N, NLVPMVATV); (2F54, SLLMWITQC); (2VLJ, GILGFVFTL); (5NHT, ELAGIGILTV); (6TRO, GVDYDGREHTV); (7N1F, YLQPRTFLL); (3GSN, NLVPMVATV); (6D78, AAGIGILTV); (3QDG, ELAGIGILTV); (5NQK, ELAGIGILTV); (5HHO, GILEFVFTL); (2PYE, SLLMWITQC); (5HHM, GILGLVFTL); (2GJ6, LLFGKPVYV); (5NMF, SLYNTIATL); (3UTS, ALWGPDPAAA); (5E9D, ELAGIGILTV); (6RPA, SLLMWITQV); (1QSF, LLFGYPVAV); (4EUP, ALGIGILTV); (5YXN, KLVALGINAV); (2BNQ, SLLMWITQV); (6RPB, SLLMWITQV); (5HYJ, AQWGPDPAAA); (3QFJ, LLFGFPVYV); (3UTT, ALWGPDPAAA); (3QDJ, AAGIGILTV); (4L3E, ELAGIGILTV); (5C09, YLGGPDFPTI); (6VRM, HMTEVVRHC); (1QRN, LLFGYAVYV); (2BNR, SLLMWITQC); (3PWP, LGYGFVNYI); (6AMU, MMWDRGLGMM); (6AM5, SMLGIGIVPV); (4MNQ, ILAKFLHWL); (5NME, SLYNTVATL); (5NMG, SLFNTIAVL)
HLA-A*11:01	(5WKH, GTSGSPIINR); (5WKF, GTSGSPIVNR)
HLA-A*24:02	(3VXR, RYPLTFGWCF); (3VXM, RFPLTFGWCF); (3VXS, RYPLTLGWCF)
HLA-B*07:02	(6VMX, RPPIFIRRL); (6AVF, APRGPHGGAASGL); (6AVG, APRGPHGGAASGL)
HLA-B*08:01	(1MI5, FLRGRAYGL); (3FFC, FLRGRAYGL); (4QRP, HSKKKCDEL); (3SJV, FLRGRAYGL)
HLA-B*27:05	(4G8G, KRWIILGLNK); (4G9F, KRWIIMGLNK)
HLA-B*35:01	(6BJ2, IPLTEEAEI); (4PRP, HPVGQADYFEY); (3MV7, HPVGEADYFEY); (3MV9, HPVGEADYFEY); (3MV8, HPVGEADYFEY)
HLA-B*35:08	(4PRH, HPVGDADYFEY); (4PRI, HPVGEADYFEY)
HLA-B*44:05	(3DXA, EENLLDFVRF); (3KPR, EEYLKAWTF); (3KPS, EEYLQAFY)
H2Db	(3PQY, SSLENFRAYV); (5WLK, SLLNAYYL); (5TJE, KAVYNFATM); (7JWI, ASNENMETM); (5SWS, ASNENMETM); (5TIL, KAPYNFATM); (7JWJ, ASNENMETM); (6G9Q, KAPYDYAPI); (5M02, KAPFNATM); (5M00, KAVANFATM); (5M01, KAPANFATM)
H2Kb	(3PQY, SSLENFRAYV); (5WLK, SLLNAYYL); (5TJE, KAVYNFATM); (7JWI, ASNENMETM); (5SWS, ASNENMETM); (5TIL, KAPYNFATM); (7JWJ, ASNENMETM); (6G9Q, KAPYDYAPI); (5M02, KAPFNATM); (5M00, KAVANFATM); (5M01, KAPANFATM)
H2Ld	(2OI9, QLSPFPFDL); (4NHU, GGGAPWNPAMMI); (4MVB, QPAEGGFQL); (4MXQ, SPAPRPLDL); (2E7L, QLSPFPFDL); (4N5E, VPYMAEFGM); (6L9L, SPSYAYHQF); (3E3Q, QLSPFPFDL); (3E2H, QLSPFPFDL); (3TPU, FLSPFWFDI); (3TFK, QLSDVPMDL); (4MS8, SPAEAGFFL); (4N0C, MPAGR-PWDL); (3TJH, SPLDSLWWI)

Table D.7.6: Structure templates to compute TCR-MHC specificities. The table list the structures used to compute TCR-MHC specificities in Fig. 5 for different MHC class I alleles in humans and mice. Structures were collected from the TCRdock curated database [26].

MHC allele	Peptide Length	PDBid
HLA-A*01:01	9	3BO8; 4NQV; 4NQX
HLA-A*01:01	10	6AT9; 6MPP
HLA-A*02:01	9	1B0G; 1DUZ; 1EEY; 1EEZ; 1HHG; 1HHI; 1HHJ; 1HHK; 1I1F; 1I1Y; 1I7R; 1I7T; 1I7U; 1JHT; 1QEW; 1QR1; 1S9W; 1S9X; 1S9Y; 1TVB; 2C7U; 2GTW; 2GTZ; 2GUO; 2V2W; 2V2X; 2VLL; 2X4O; 2X4S; 2X4U; 3BGM; 3FT2; 3FT3; 3FT4; 3GSO; 3GSQ; 3GSR; 3GSU; 3GSV; 3GSW; 3GSX; 3H7B; 3H9H; 3HPJ; 3I6G; 3IXA; 3KLA; 3MR9; 3MRB; 3MRC; 3MRD; 3MRE; 3MRF; 3MRG; 3MRH; 3MRI; 3MRJ; 3MRK; 3MRL; 3MYJ; 3PWJ; 3PWL; 3PWN; 3QFD; 3TO2; 3V5D; 3V5H; 3V5K; 4I4W; 4K7F; 4L29; 5E00; 5EU3; 5HHN; 5HHP; 5HHQ; 5MEO; 5MEP; 5N6B; 5WSH; 6SS7; 6SS8; 6SS9; 6SSA; 6VR1; 6VR5; 6Z9W; 7EU2; 7LG2; 7LG3; 7MJ6; 7MJ9; 7MKB; 7N1A; 7N1B; 7RTD; 7SA2; 7SIS; 7U1R; 7U21; 7UM2; 7ZUC; 8FU4; 8RBV
HLA-A*02:01	10	1HHH; 1I4F; 1JF1; 2CLR; 2GT9; 3BH8; 3GIV; 3I6K; 3MGO; 3MGT; 3MRM; 3MRN; 3MRO; 3MRP; 3MRQ; 3MRR; 3UTQ; 4GKN; 4GKS; 4JFO; 4JFP; 4JFQ; 5C0D; 5C0G; 5C0I; 5FDW; 5N1Y; 6AMT; 6G3J; 6G3K; 6TRN; 7M8S; 7Q98; 7UR1
HLA-A*02:01	11	5D9S; 5EOT; 7MJ7
HLA-A*02:06	9	6UJO; 6UJQ
HLA-A*03:01	9	3RL1; 7L1B; 7L1C; 7MLE; 7UC5
HLA-A*03:271	10	8JHV; 8JHW
HLA-A*11:01	9	1Q94; 7S8R; 8I5E
HLA-A*11:01	10	1QVO; 5WJN; 7OW3; 7S8Q; 7S8S; 8RBU; 8RH6; 8RHQ
HLA-A*11:01	11	4MJ5; 4MJ6
HLA-A*24:02	8	4F7T; 4WU5; 5HGA; 5HGB
HLA-A*24:02	9	2BCK; 3I6L; 6XQA; 7F4W; 7JYV; 7JYW; 7MJA; 7NMD
HLA-A*24:02	10	3NFN; 3VXN; 3VXO; 3VXP; 3WL9; 3WLB; 4F7M; 5HGD; 5HGH; 7JYU
HLA-A*26:01	9	8XES; 8XFZ; 8XKC
HLA-A*68:01	10	1TMC; 6EI2
HLA-B*07:02	9	4U1K; 5E00; 5E01; 5WMN; 7LFZ; 7LG0; 7LGD; 7RZD; 7RZI; 7S7D; 7S7E; 7S7F; 7S8A; 7S8E; 7S8F
HLA-B*08:01	8	8E13; 8E8I
HLA-B*08:01	9	3SKM; 3SKO; 4QRQ; 4QRT; 5WMQ; 5WMR; 7NUI
HLA-B*15:01	9	5TXS; 6UZS; 6VB3; 7XF3; 8ELG; 8ELH
HLA-B*15:02	9	6UZM; 6UZN; 6UZO; 6VB0; 6VB7; 6VIU
HLA-B*18:01	8	4XXC; 8RNG; 8ROO; 8ROP
HLA-B*18:01	9	6MT3; 8RNH
HLA-B*27:05	9	1JGE; 1UXS; 1W0V; 2A83; 2BSR; 2BST; 3B6S; 3BP4; 3DTX; 3LV3; 5IB1; 5IB2; 6PYW; 6Y2A; 7CIS
HLA-B*27:05	10	2BSS; 7CIQ; 7CIR; 7DYN
HLA-B*27:09	9	1K5N; 1W0W; 3B3I; 3BP7; 3CZF; 3HCV; 6Y27; 6Y2B
HLA-B*35:01	9	1A9B; 1A9E; 1CG9; 3LKN; 3LKO; 3LKP; 3LKQ; 3LKR; 3LKS; 7SIG; 8EMF; 8EMG; 8EMI; 8EMJ
HLA-B*35:08	13	3VFM; 3VFR; 3VFT; 3VFU; 3VFW
HLA-B*37:01	9	6MT4; 6MT5; 6MT6
HLA-B*44:03	9	1SYS; 3KPO
HLA-B*44:05	9	1SYV; 7TUC
HLA-B*53:01	9	7R7V; 7R7W
HLA-B*57:01	9	6D2T; 7R7X
HLA-B*57:01	10	3VRI; 3VRJ; 6D29; 6V2O; 8F7M
HLA-B*57:03	9	2BVP; 6V2Q
HLA-B*58:01	8	7X1B; 7X1C
HLA-B*58:01	9	5VWH; 7WZZ; 7X00
HLA-B*81:01	9	4U1I; 4U1L; 4U1S
H2-Db	9	1BZ9; 1FFN; 1FFO; 1FFP; 1FG2; 1HOC; 1INQ; 1JPG; 1JUF; 1LD9; 1N5A; 1S7U; 1S7V; 1S7W; 1S7X; 1ZHB; 2F74; 2ZOL; 3BUY; 3CC5; 3CCH; 3CH1; 3CPL; 3FTG; 3QUK; 3QUL; 3TBS; 3TBT; 3TBV; 3TBW; 3TBY; 3WS3; 3WS6; 4HUU; 4HUV; 4HUW; 4HUX; 4HV8; 4IHO; 4L8B; 4L8C; 4L8D; 4NSK; 5E8N; 5E8O; 5E8P; 5JWD; 5MZM; 5WLI; 6G9R; 6L9M; 6L9N; 7N9J; 7P0A; 7P0T
H2-Db	10	1N3N; 1WBX; 1WBY; 1YN6; 1YN7; 3L3H; 5JWE; 6H6D; 6H6H; 7N5Q
H2-Db	11	1JPF; 6X00
H2-Dd	9	5KD7; 5T7G; 8D5E; 8D5F; 8FHL; 8FHU
H2-Dd	10	1RII; 3E6H; 3FCR; 5KD4; 5WEL; 6NPR

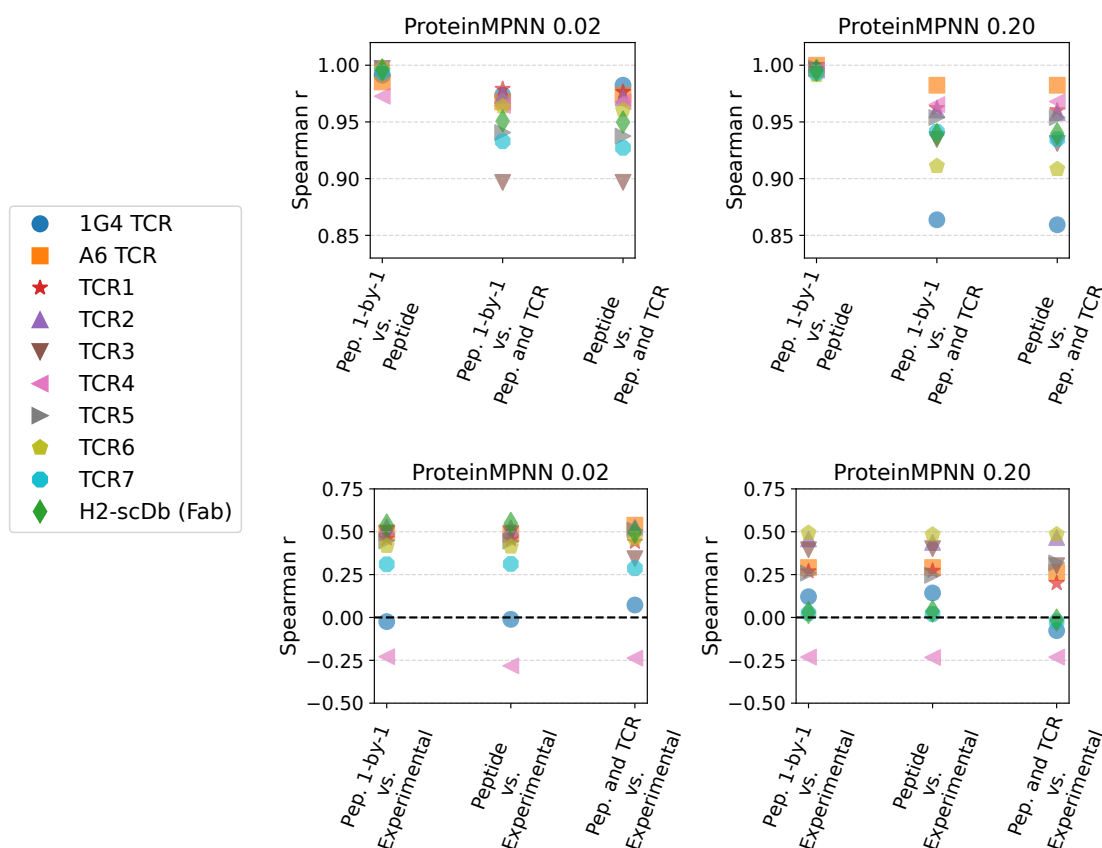


Figure D.7.1: Robustness of ProteinMPNN performance across scoring schemes . The performance of ProteinMPNN across different systems (legend) is shown for models trained with varying noise amplitudes (specified above each panel), and using different scoring schemes. We consider three scoring schemes: (i): “Peptide 1-by-1” in which residues within the peptide are masked and scored one at a time, using the remainder of the TCR-pMHC complex as structural context (eq. D.3.6), (ii) “Peptide” scheme, in which the entire peptide is masked and scored simultaneously (eq. D.3.7), and (iii) “Pep. and TCR” scheme in which both the peptide and the nearby TCR residues are masked and scored simultaneously (eq. D.3.8). Top panels show the Spearman correlation between predictions from different scoring schemes, while bottom panels show correlations between each scheme and experimental data. Results demonstrate that ProteinMPNN predictions are robust to the choice of scoring scheme.

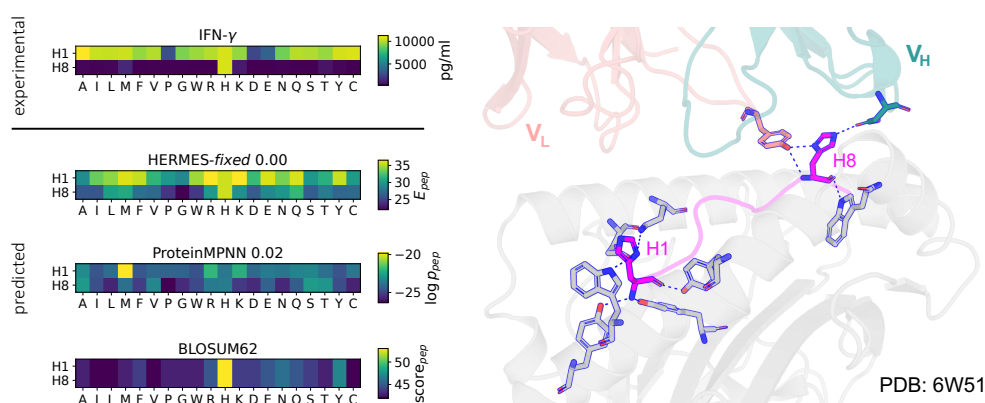


Figure D.7.2: Context-dependent mutational effects on T-cell activity. The heatmaps show both experimental measurements and model predictions of T-cell activities of amino acid variants at positions 1 and 8 of the p53^{R157H} peptide, which we study in complex with the designed bi-specific antibody H2-scDB and HLA-A*02:01 [97]. The wild-type peptide has a histidine at both sites. A standard substitution matrix (such as BLOSUM62) would assign identical scores to mutations at these two sites, because they share the same wild-type residue. However, peptide binding and the subsequent T-cell activity depends on the structural context of each substitution. Here, H1 forms polar contacts only with the MHC (from the sides), while H8 is more tightly constrained by polar contacts with the TCR (right panel). HERMES-fixed correctly leverages structural context to predict T-cell activity as a result of substitutions away from histidines at both positions. Although ProteinMPNN also incorporates structural information, its predictions are not as accurate in this instance.

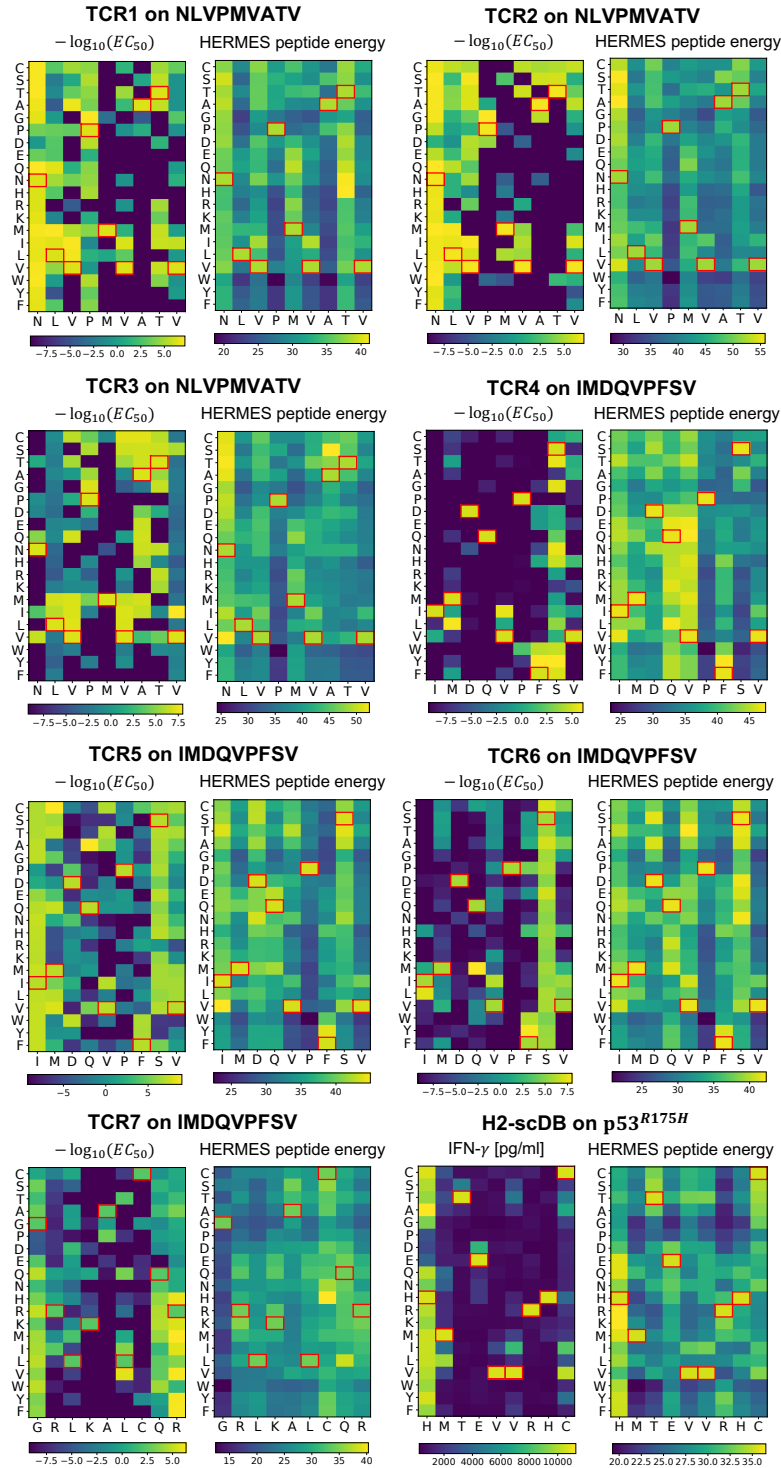


Figure D.7.3: Mutational effect on T-cell activities . The heatmaps show the experimental measurements and the predictions from the HERMES-fixed model for the effect of all single point mutations on peptides interacting with TCR1-7 [140], and the designed bi-specific antibody Fab in the H2-scDB system [97] (similar to the data presented in scatter plots in Fig. 3). In the heatmaps, each column corresponds to a position in the peptide and is labeled with its wild-type amino acid (indicated by a red square on the heatmaps), and the rows show the amino acid substitutions.

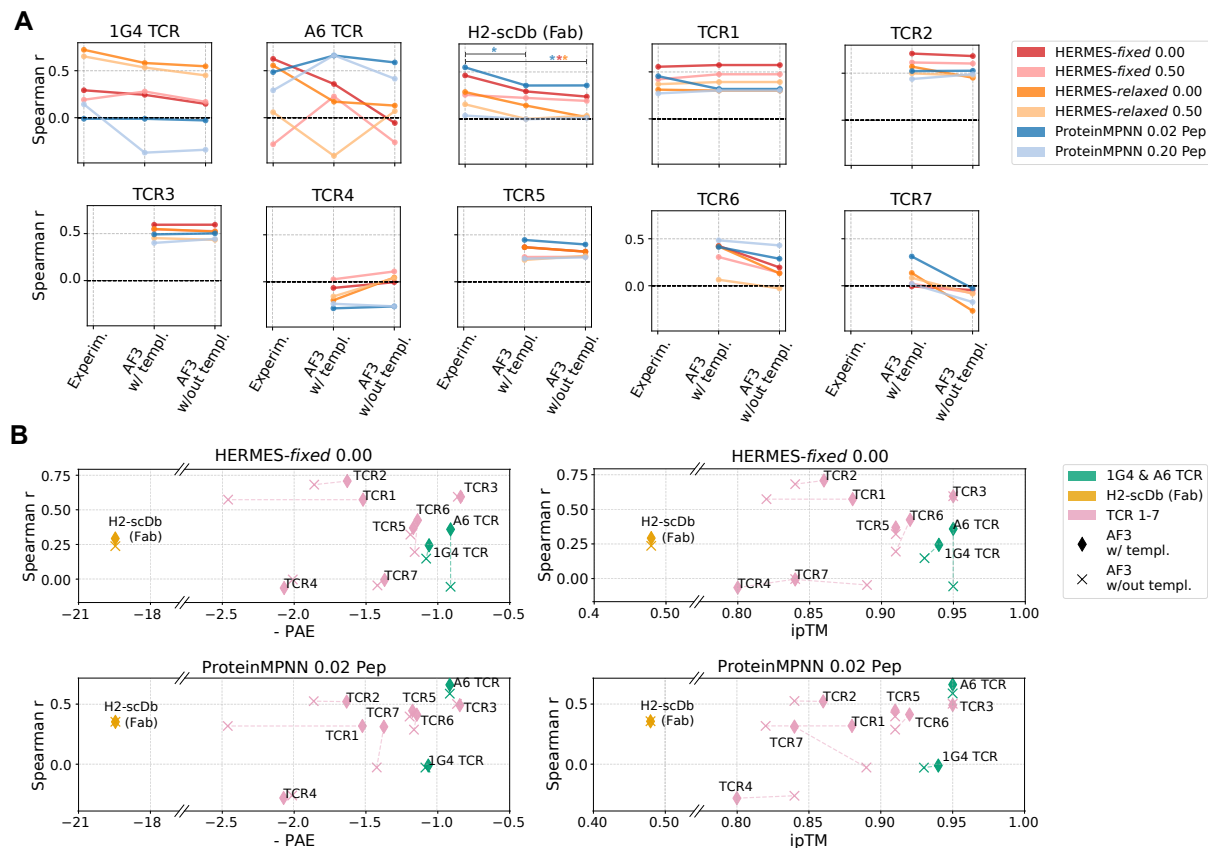


Figure D.7.4: HERMES and ProteinMPNN prediction accuracies for AlphaFold3-predicted structures.

(A) For all systems analyzed, we used the AlphaFold Server to model the structures of complexes with the wild-type peptides, both with and without template guidance; see Table D.7.4 for details on the confidence of these AF3 models. For each system, the Spearman correlation (r) between experimental affinity or activity data and predictions from HERMES and ProteinMPNN models (colors) is shown for three structure inputs: experimental structures (when available), AF3 model with template, and AF3 model without template. Statistically significant drop in accuracy between models using experimental vs. AF3 structures—defined as cases with p -value < 0.05 for difference in the Fisher's z -transformed Spearman correlations—are indicated with an asterisk (*) in the same color as that of the model (from the legend). Such drops are only observed for the H2-scDb Fab-pMHC system, where AF3 failed to generate a realistic structure. (B) Spearman correlation (r) between experimental affinity/activity data and predictions from the overall best HERMES-fixed (top) and ProteinMPNN (bottom) models from (A), plotted against AF3 confidence metrics: negative predicted alignment error ($-PAE$; left) and interchain predicted TM-score (ipTM; right). Results are shown for AF3 structures generated with and without template guidance. The confidence metrics are reported in Table D.7.4. The HERMES model names specify the scoring scheme (fixed, relaxed), and the noise amplitude (in Å) injected to the coordinates of the structure data during training. Similarly, the ProteinMPNN model names capture scoring scheme, i.e., the whole peptide (eq. D.3.7), and the noise amplitude injected to the training data.

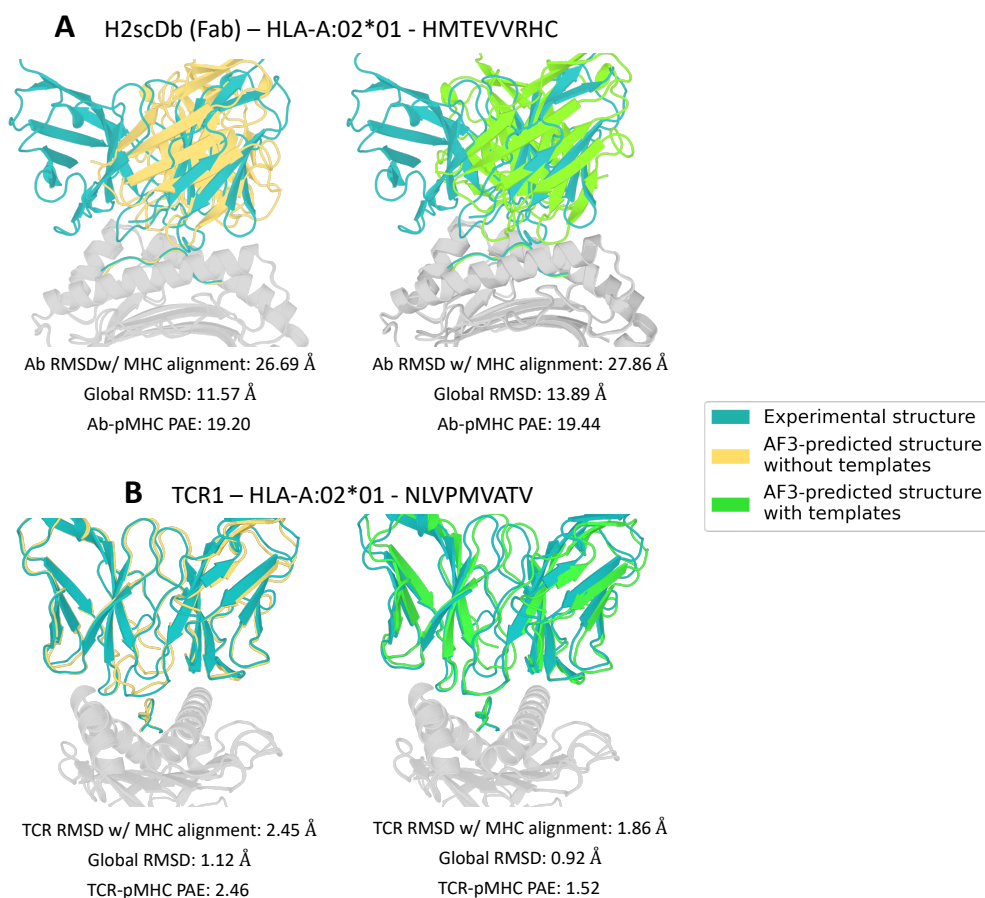


Figure D.7.5: Comparison of experimentally determined structures with AlphaFold3 predicted structures. The experimentally determined structure of (A) H2-scDb Fab-pMHC complex, and (B) TCR1-pMHC complex are shown against the AF3 predicted structures generated without (left) and with (right) template guidance. All structures were aligned on the MHC using PyMOL to enable consistent visualization of TCR/Fab docking geometries. AF3 failed to produce a realistic structure model for the H2-scDb system, as evidenced by the high RMSD and interface PAE values reported in Table D.7.4.

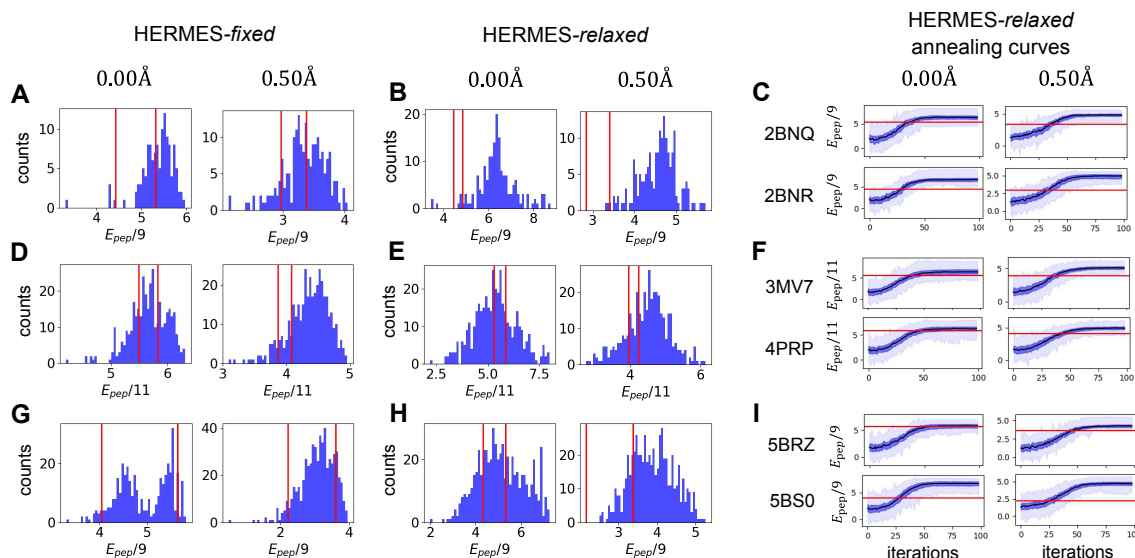


Figure D.7.6: Peptide design energy profiles across systems and HERMES models. For the NY-ESO system, panels (A) and (B) show the distribution of peptide energies (eq. D.1.1) divided by the length, for peptides designed with HERMES-fixed and HERMES-relaxed protocols, respectively. Designs were pooled across the two structure templates used (2BNQ and 2BNR), with the peptide energies computed using their respective template structures. In all panels, the red lines indicates the energies of the wild-type peptides in their corresponding structures. (C) The scaled energy of the designed peptides are shown as a function of the MCMC iterations during the HERMES-relaxed protocol, using the two different structural templates for the NY-ESO system: 2BNQ (top) and 2BNR (bottom). The red line indicates the energy of the wild-type peptide in its respective structure. In panels (A-C), designs are grouped by the noise amplitude applied during HERMES training (no noise: 0.00Å, and moderate noise: 0.50Å), as indicated above each panel. The HERMES-relaxed protocol consistently produces peptides with more favorable (higher) peptide energies than that of the wild-type, indicating that the protocol can successfully optimize within the peptide energy landscapes; models trained with noise are more successful in finding peptides with higher energy values. Panels (D-F) and (G-I) present analogous design statistics for the EBV and the MAGE systems, respectively.

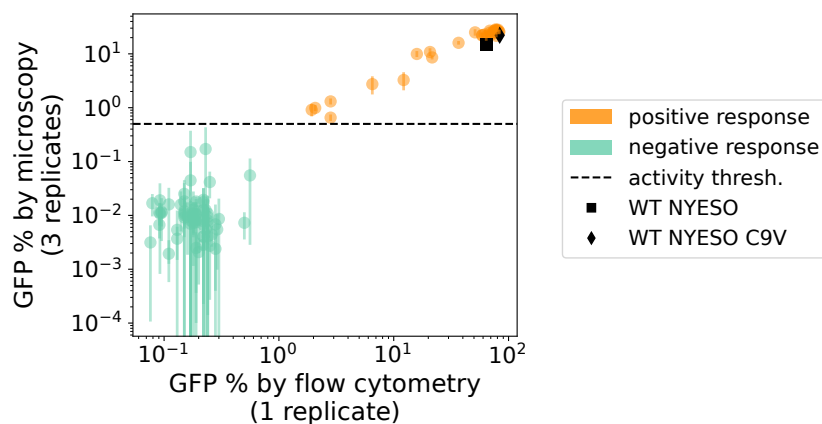


Figure D.7.7: Measurements of GFP levels by flow cytometry and fluorescence microscopy. For the NY-ESO system, we performed measurements of GFP levels from peptide-induced T-cell activities, using both flow cytometry and fluorescence microscopy. Fluorescence microscopy measurements were performed in three independent replicates; markers denote the mean GFP level, and error bars show the standard deviation across the three replicates. Flow cytometry was done only on a single replicate. The two measurement methods demonstrate good agreement, with a clear separation between designs that induced significant T-cell activation (orange) and those that did not (green). The black markers indicate the activities induced by the wild-type peptides.

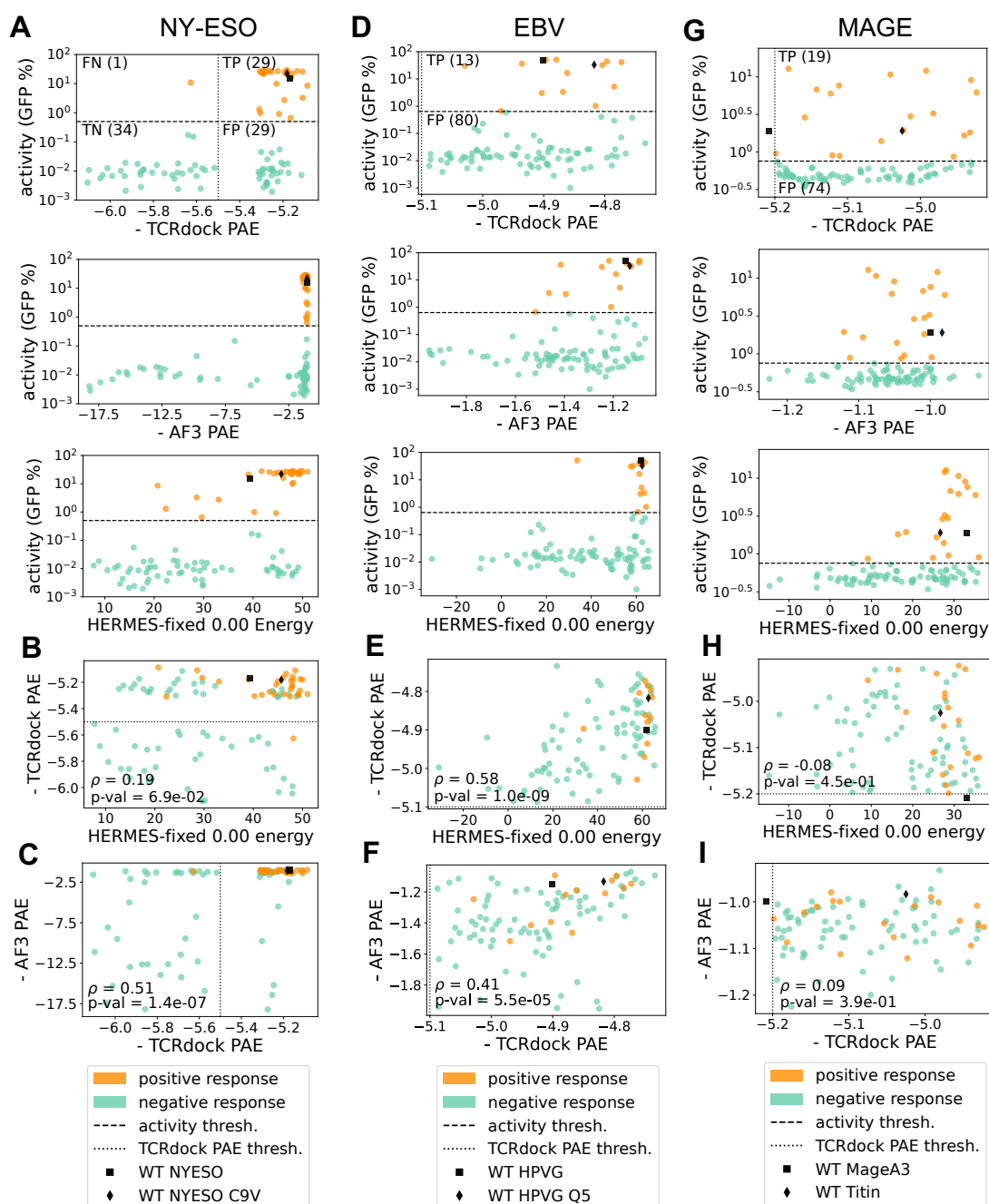


Figure D.7.8: TCRdock and AlphaFold3 PAEs for HERMES-designed peptides . (A) For the NY-ESO system, the induced T-cell activity by the designed peptides is shown against the computed TCRdock negative-PAE (first row), the AlphaFold3 negative-PAE (second row), and the energy score predicted with HERMES-fixed 0.00 (non noise during training) (third row); higher negative-PAE's indicate better designs. For the designed peptides in the NY-ESO system (B) shows the relationship between negative TCRdock PAE values and the peptide energies from the HERMES-fixed 0.00. model, and (C) shows the negative AlphaFold3 PAE versus TCRdock PAE values. Peptides that induced significant T-cell responses are shown in orange, and non-responders are shown in green. The vertical dashed line indicate the filtering threshold for TCRdock PAE, with points to the right representing the negative design set. Panels (D-F) and (G-I) show analogous statistics for the EBV and the MAGE system, respectively. No negative designs (i.e., designs below the respective TCRdock negative-PAE thresholds) were experimentally validated for these systems. We observe that the HERMES scores and TCRdock PAE values are uncorrelated or only weakly correlated across systems, and so are TCRdock and AF3 PAE, suggesting that AF3 and TCRdock PAE's can serve as complementary filters to HERMES during design.

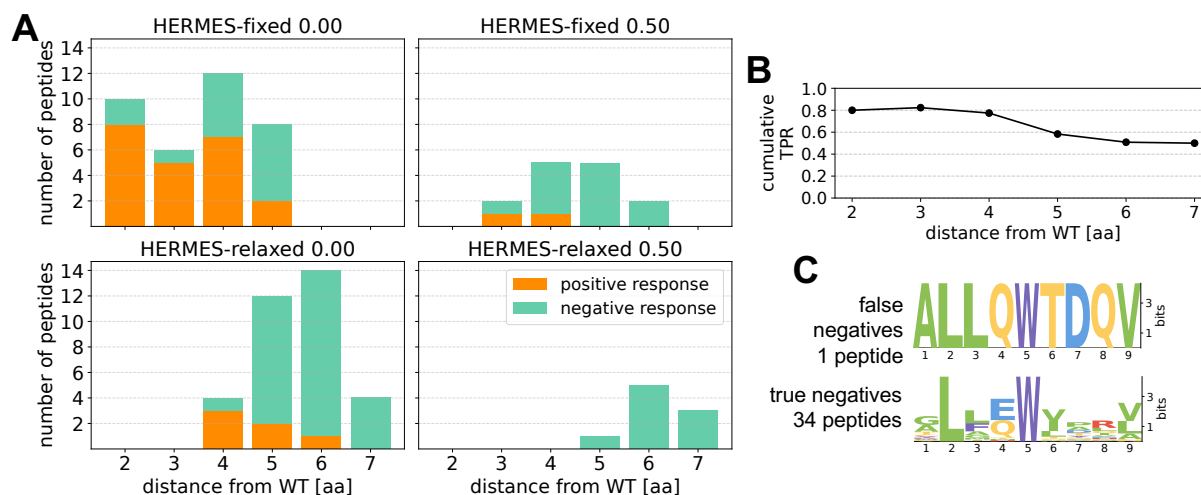


Figure D.7.9: Experimental T-cell activation by HERMES-designed peptides in the NY-ESO system . (A) The number of peptides designed by different HERMES models (panels) that induced GFP expression beyond the T-cell activation threshold (positives; shown in orange) versus those that did not (negatives; shown in green) is shown. The designed peptides are grouped according to their minimum Hamming distance from any of the wild-type sequences used as design templates (Table D.7.5). The HERMES model names specify the scoring scheme (fixed, relaxed), and the noise amplitude (in Å) injected to the coordinates of the structure data during training. **(B)** The cumulative True Positive Rate (TPR) for designed peptides is shown as a function of their Hamming distance from any of the wild-types. **(C)** Logo plots show the sequence composition of the False Negative and the True Negative designs.

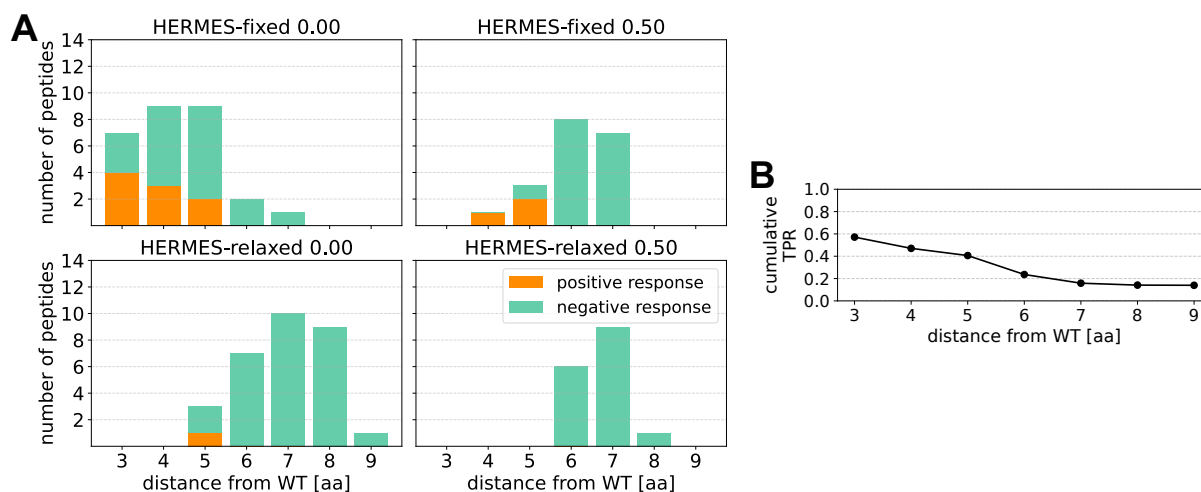


Figure D.7.10: Experimental T-cell activation by HERMES-designed peptides in the EBV system. (A-B) Similar to (A-B) of Fig. D.7.9 but for the EBV system.

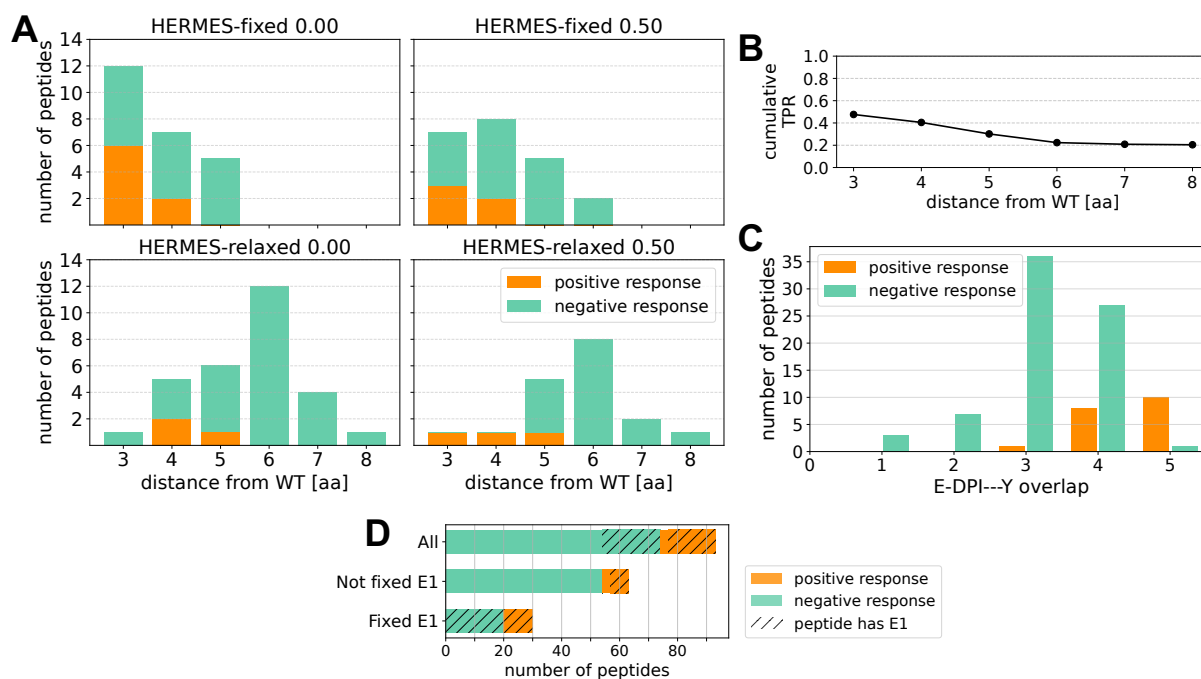


Figure D.7.11: Experimental T-cell activation by HERMES-designed peptides in the MAGE system. (A-B) Similar to (A-B) of Fig. D.7.9 but for the MAGE system. (C) The number of designs that induced GFP expression beyond the T-cell activation threshold (positive response; in orange) versus those that did not (negative response; in green) is shown as a function of the designs' overlap with the E-DPI--Y motif; this motif has been shown to enhance T-cell activation in this system [36]. The overlap is the number of amino acids that a design shares with the E-DPI--Y motif (maximum of 5). The fraction of positive designs increases with greater similarity to the motif. (D) Shown are the numbers of True Positives (orange) and False Positives (green) among designs generated under two protocols: one that enforced glutamic acid at position one (E_1)—labeled “fixed E_1 ”—and one that did not enforce E_1 —labeled “not fixed E_1 ”. For each protocol, the dashed segments indicate the number of designs that contain E_1 . The “All” bar presents the aggregate data from both protocols.

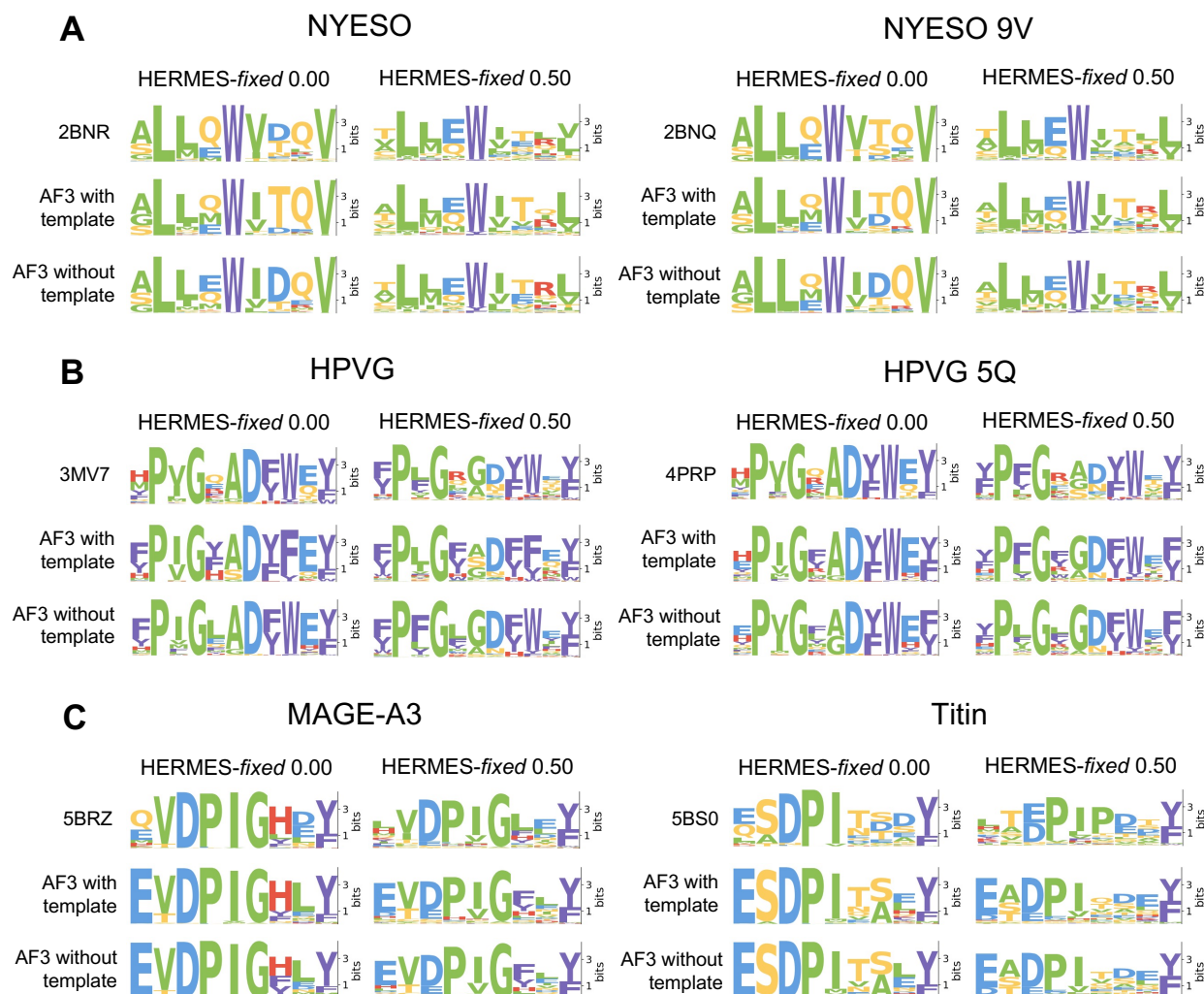


Figure D.7.12: Composition of designed peptides using experimental versus AlphaFold3 structures as design templates. For the (A) NY-ESO, (B) EBV, and (C) MAGE, each panel displays the position weight matrices (PWMs) for peptides designed using the HERMES-fixed protocol. Designs were generated using three types of structural templates as indicated in each panel: the experimentally determined wild-type structure (first row), and the AlphaFold3-predicted models of that structure generated with (middle) and without (bottom) template guidance. PWMs are shown separately for the HERMES-fixed models trained with different noise amplitudes (as indicated above the panels). For each system, results are provided for the two wild-type peptides used as input structural templates (shown in the left and right columns of each panel).

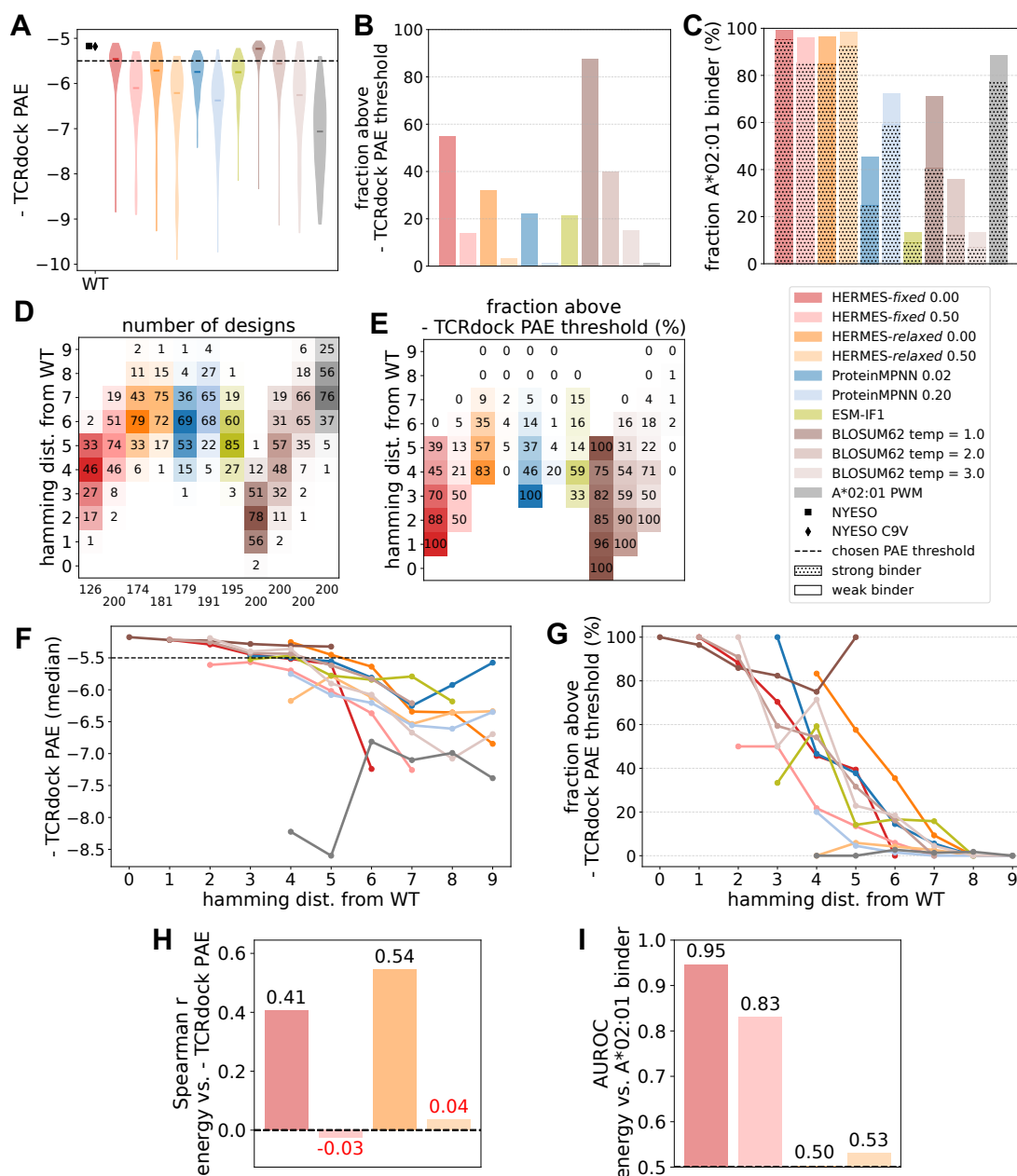


Figure D.7.13: Benchmarking of models for peptide design in the NY-ESO system. (A) The distribution of the TCRdock negative-PAE's for peptides designed by different computational protocols (colors) are shown; in each case the score distribution for 100 independently designed peptides is shown. The horizontal dashed line indicates the PAE threshold chosen to classify positive designs. (B) The fractions of designed peptides that pass the TCRdock PAE threshold is shown for each method. (C) The fraction of designs that are predicted by NetMHCPan [188] to be presented by the template's MHC allele are shown; dotted regions correspond to strong binders (EL_rank ≤ 0.5), while the bar indicates all weak and strong binders (EL_Rank ≤ 2.0). HERMES designs are most consistently predicted to be presented by the MHC. (D) The number of designs, and (E) the fraction of designs that pass the TCRdock PAE threshold are shown for each model (colors) at different Hamming distances from the wild-types; The numbers at the bottom of each column in (D) indicate the total number of unique designs. (F) The median of TCRdock negative-PAE across designs, and (G) the fraction of designs that pass the TCRdock PAE are shown for each model (colors) as a function of the Hamming distance from the wild-types. (H) Spearman correlation between HERMES-predicted peptide energy and TCRdock PAE is shown, with data pooled across design models. In each case, HERMES energies are computed using the same protocol applied during design. Correlations that are not statistically significant (p-value > 0.05) are highlighted in red. (I) Shows the AUROC for classifying whether a designed peptide is presented by the template's MHC allele (i.e., predicted to be a strong binder according to NetMHCPan [188]), using HERMES peptide energy as the classifier. As in (H), for each peptide, HERMES energies are computed via the same protocol used for their design. The HERMES model names specify the design scheme (fixed, relaxed), and the noise amplitude (in Å) injected to the coordinates of the structure data during training. The ProteinMPNN design uses baseline scoring scheme with peptide 1-by-1 (eq. D.3.6), with the training noise amplitude specified in the model name.

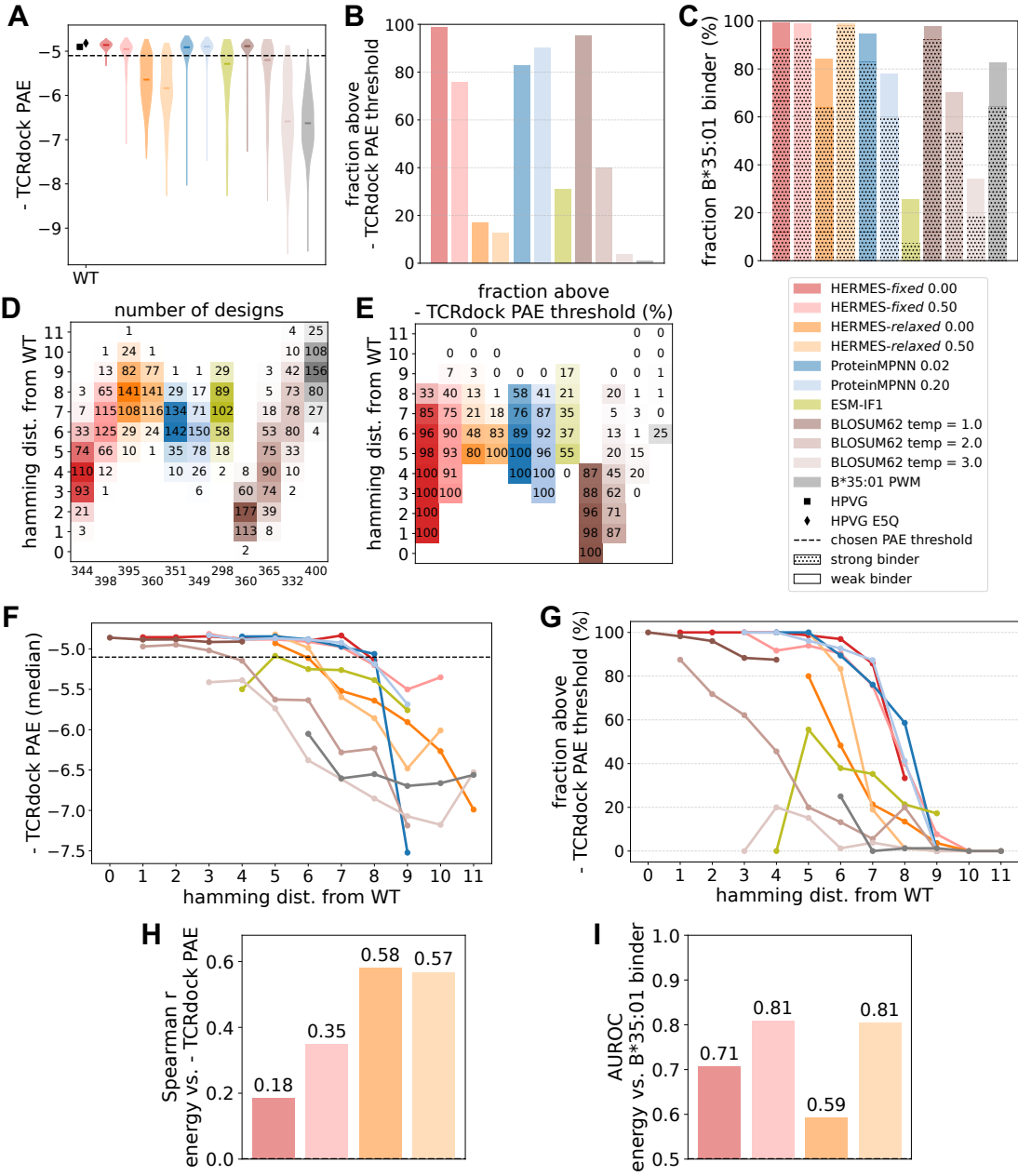


Figure D.7.14: **Benchmarking of models for peptide design in the EBV system.** Similar to Fig. D.7.13 but for the EBV system.

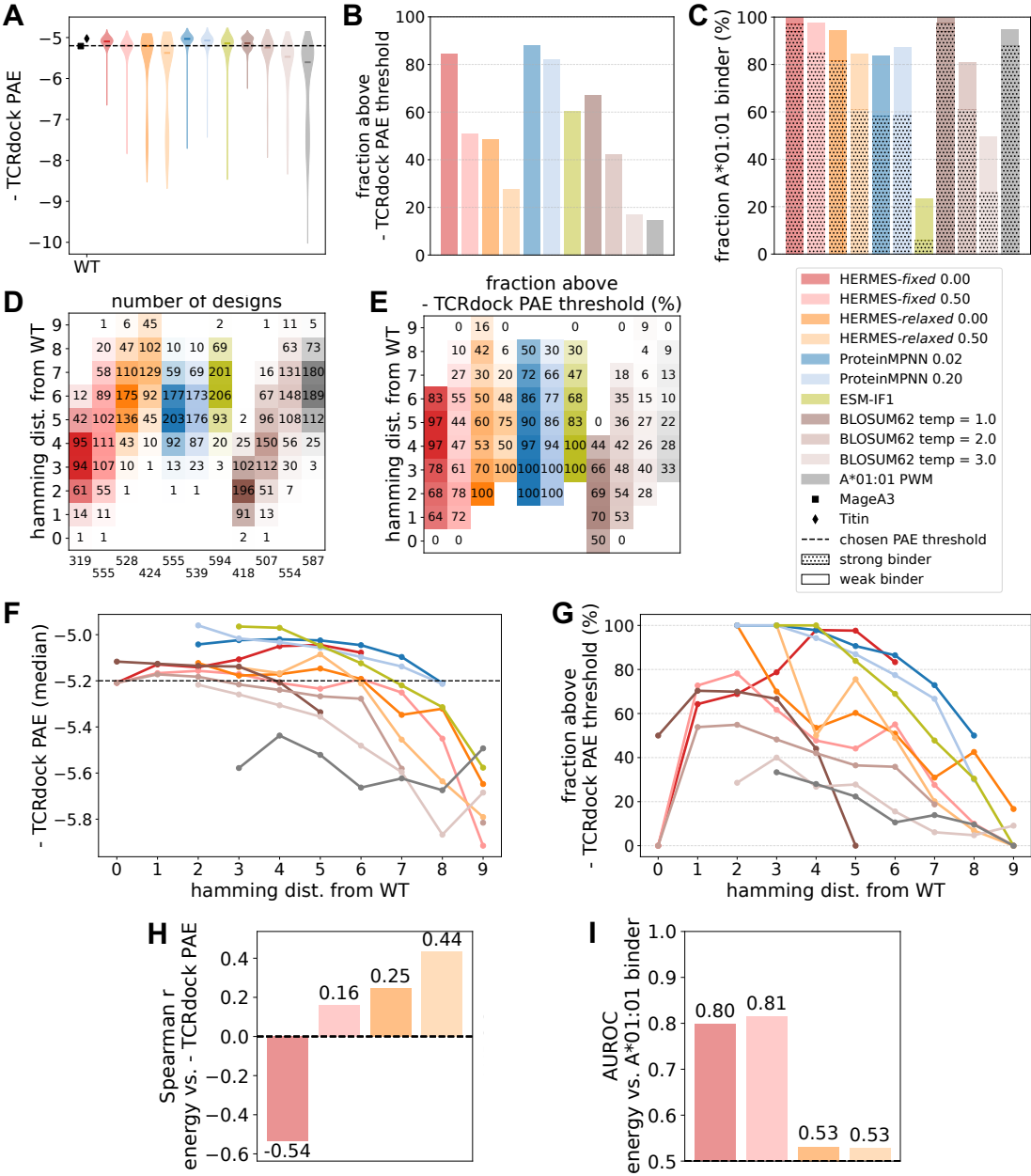


Figure D.7.15: **Benchmarking of models for peptide design in the MAGE system.** Similar to Fig. D.7.13 but for the MAGE system.

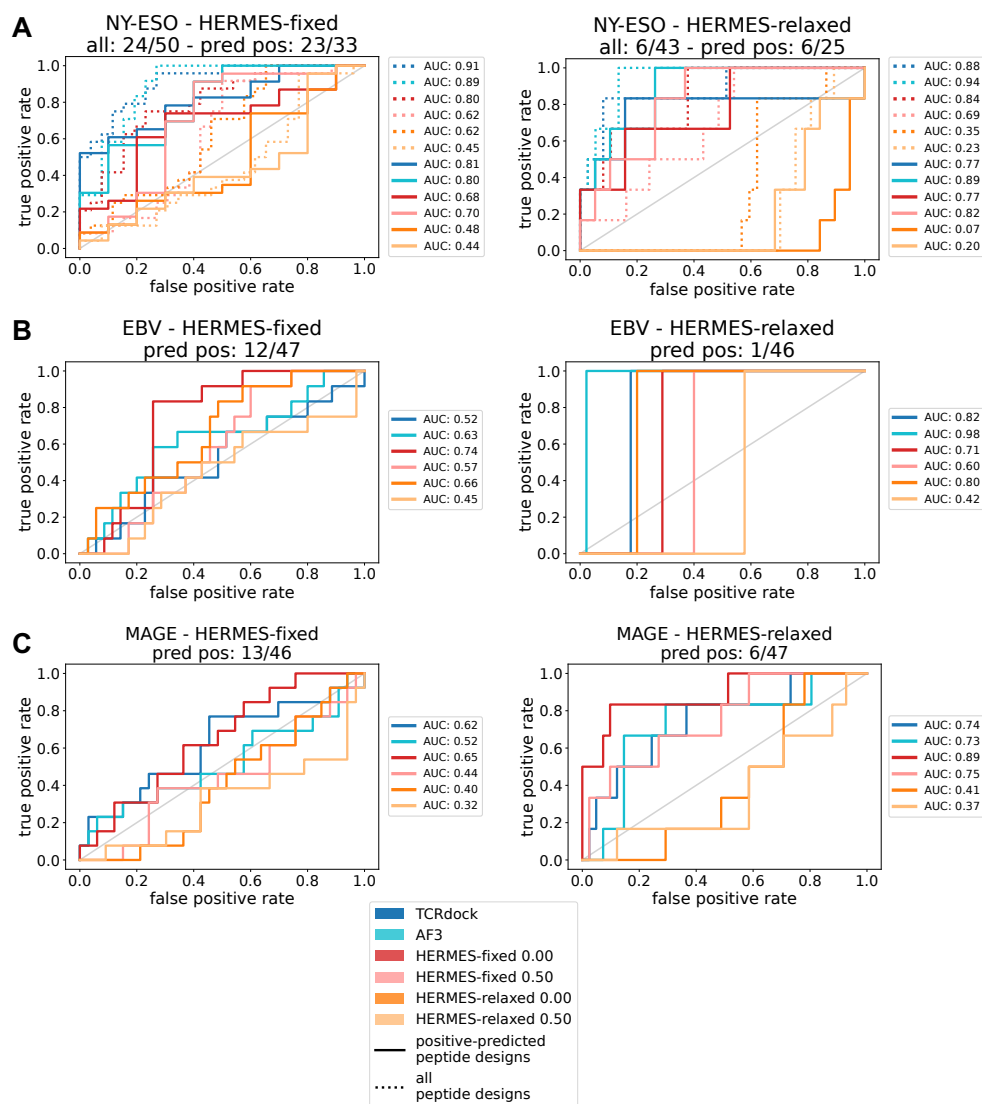


Figure D.7.16: Discrimination ability of different scoring schemes as design filters. The ROC curves show true positive versus false positive rates when using scores from different models—HERMES, AlphaFold3, and TCRdock (color coded as in the legend)—as scoring criteria for peptide design. Results are shown separately for designs generated using the *HERMES-fixed* (left) and the *HERMES-relaxed* (right) protocols, for the (A) NY-ESO, (B) EBV, and (C) MAGE systems. Experimentally measured peptide-induced T-cell activity (GFP%) is used as the ground truth for evaluating the predictive performance in each case. For the NY-ESO system only (A) we show ROC curves computed on both peptides that were predicted to activate T-cells (positives) based on their TCRdock PAE scores (full lines), and the mixture of positive and negative peptides (dotted lines). For EBV and MAGE (B-C), the scoring criterion is only tested for the peptides that we deemed to be activating (positive) based on their TCRdock PAE scores (full lines), as these were the ones that we tested experimentally. The HERMES model names specify the design scheme (fixed, relaxed), and the noise amplitude (in Å) injected to the coordinates of the structure data during training.

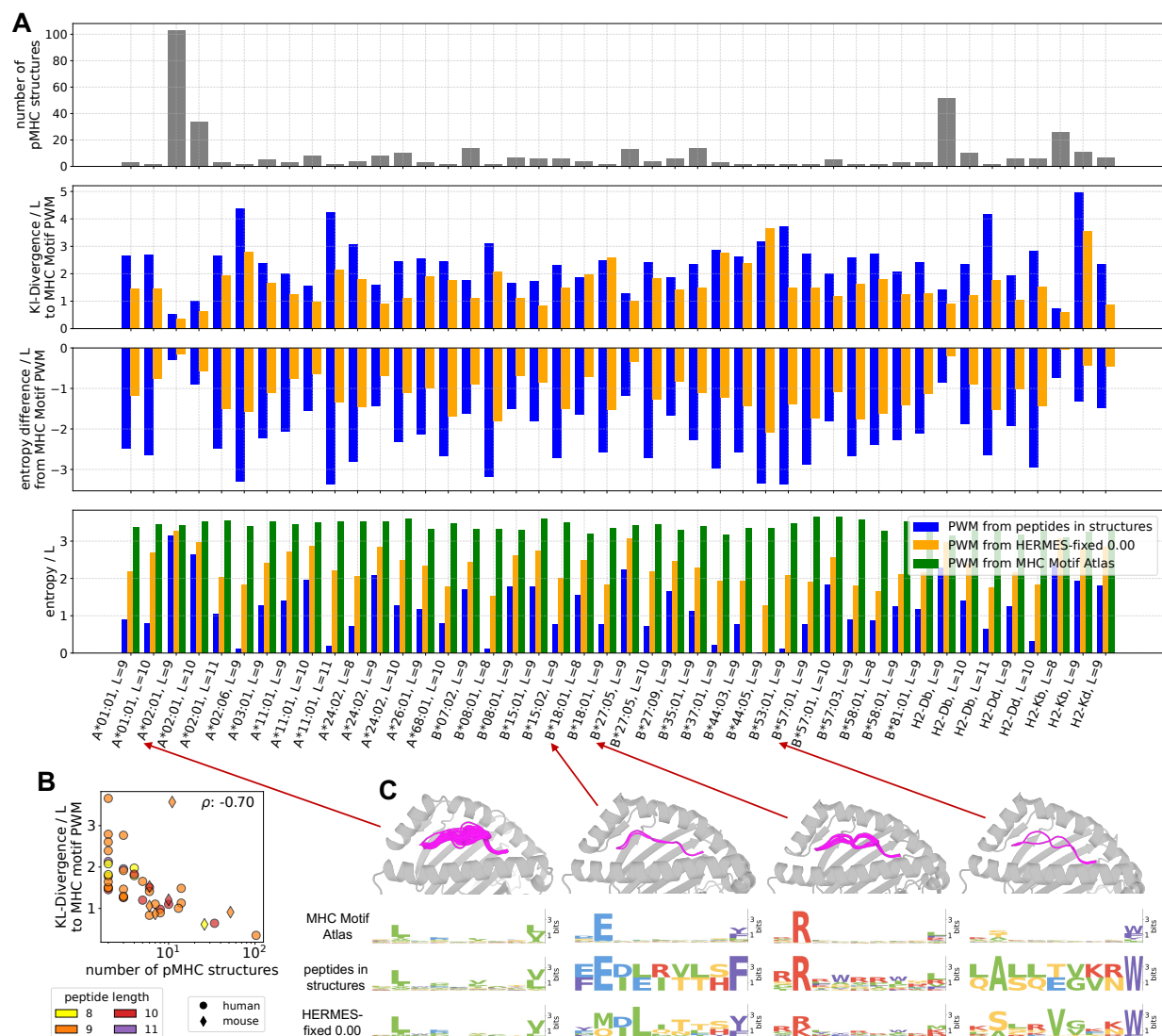


Figure D.7.17: Predicting MHC-I peptide presentation preferences with HERMES. We evaluated HERMES's ability to recover MHC class I peptide presentation preferences by analyzing pMHC structures from the TCR3d database [83], grouped by MHC allele and peptide length (L), retaining only those combinations with at least two structures. As ground truth, we used position weight matrices (PWMs) from the MHC Motif Atlas [226], derived from large datasets of peptide sequences known to be presented by each allele. HERMES-fixed was used to design peptides for each structure in a given allele-length group, and the resulting PWMs were averaged across structures to generate a motif reflecting the peptide preferences for that MHC I allele (for a given length). As the baseline, we used PWMs generated directly from the peptides found in the structural dataset. PWM quality was evaluated using the Kullback-Leibler divergence, and the entropy difference from the ground-truth PWM. **(A)** Shown are the summaries of data and performances across all MHC I allele-length combinations: number of available structures (first row), KL-divergence between the HERMES-derived PWMs (or the baseline) to the ground truth (second row), the entropy difference to the ground truth (third row), and the entropy of the PWMs based on the three approaches (fourth row). **(B)** Shown is the KL-divergence of HERMES-derived PWMs versus the number of available pMHC structures for across different allele-length classes. **(C)** Examples of pMHC structures and the corresponding PWMs are shown for four allele-length combinations, spanning a range of structure counts. In each case, structures are aligned along the MHC chain to show the diversity of peptide poses captured by the ensemble of available structures, used as design templates.