Nano-scale to Meso-scale: Practice and Pedagogy of Deep Learning Applied to Molecular Systems

Evan Komp

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2024

Reading Committee:

David A. C. Beck, Chair

Elizabeth Nance

Zachary Sherman

Program Authorized to Offer Degree:

Chemical Engineering

University of Washington

Abstract

Nano-scale to Meso-scale: Practice and Pedagogy of Deep Learning Applied to Molecular Systems

Evan Komp

Chair of Supervisory Committee:

David A. C. Beck

Department of Chemical Engineering

Deep learning has found its way into the chemical sciences, and brought with it successes that have been observed in "traditional" applications such as computer vision and sentiment analysis.[1–3] The integration of these methods into modern industry, pharma, and materials discovery makes it clear that the next generation of chemical and biological scientists and engineers will be operating at the intersection of data science and their respective fields. Unfortunately, nuances in these chemical systems mean that often methods leveraged in computer science applications are not out-of-the box translatable; expert knowledge is required to design a machine learning solution at the intersection of these fields.[4–6] Here, this nuanced design process is detailed for two deep learning applications to the prediction of reaction rate constants, critical for understanding and design of reactive systems.[7] Additionally, an optional learning event, designed to give undergraduate students an opportunity to learn and apply modern methods at this intersection, is detailed. Finally, work to leverage deep neural machine translation to improve protein thermal stability is presented, and the nuances highlighted. The translator has the ability to produce thermally stable variants of existing proteins or score already generated variants, allowing them to be leveraged at high temperatures where many proteins lose their applicability compared to other materials.[8]

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Acronym
%id	Percent Identity
AA	Amino Acid
E_A	Activation Energy
AF2	AlphaFold2
ChE	Chemical Engineering
DFT	Density Functional Theory
DL	Deep Learning
DMS	Deep Mutational Scan
DNN	Dense Neural Network
DVC	Data Version Control
En-HD	Engrailed Homeodomain
HMM	Hidden Markov Model
MLP	Multi Layer Perceptron
MEP	Minimum Energy Path
MSA	Multiple Sequence Alignment
MSE/MAE	Mean Squared/Absolute Error
NMT	Neural Machine Translation
OGT	Optimal Growth Temperature
PDB	Protein Data Bank, also refers to the file type
PES	Potential Energy Surface
RE	Reaction Energy
(S)ML	(Supervised) Machine Learning
SST	Self Supervised Training
TS(T)	Transition State (Theory)

LIST OF TABLES AND FIGURES

ACKNOWLEDGEMENTS

Some work presented in this dissertation was not conducted by the myself alone. For the work regarding prediction of quantum reaction rate constant products, Stéphanie Valleau, my first advisor, generated the hypothetical reaction barriers and computed quantum rate constants, while the I conducted downstream machine learning and analysis. C-HACK was the work of a number of organizers, as discussed in the document. For the work on protein thermal stability, Humood Alanzi helped to produce the code to run HMMER against Pfam in validating learn2therm protein pairs as well as run NOMELT against ProteinGym LipA benchmark set zero-shot, Chau Vuong drafted code to query AlphaFold Database and run FATCAT on protein pairs, and Ryan Francis created Figures 3-3 and 3-6. All MD simulations were prepared and executed by Christian Phillips. Additional intellectual contributions from a number of people during brainstorming sessions not specifically identified.

The work to predict partition functions (Chapter 1) was funded in part by the Clean Energy Institute via the DIRECT fellowship. The work on protein thermal stability (Chapter 3) is funded by the NSF grant HDR:EDSI 1934292.

Special, super, major, maximum acknowledgements are also made to the author's advisor Dave Beck, whose expertise in the space and support on learning was absolutely critical for working on the topic of protein thermal stability. He is an exemplary case of a PI who is an amazing *advisor* of students in preparation for their careers, as opposed to just a good *producer* of papers.

Finally, gratitude for my family, with unwavering if somewhat questioning support, and for the lifelong friends with whom I shared the grad school experience. You know who you are.

INTRODUCTION

Trailing the computational tech revolution has been a large-data, high-power explosion in algorithms enabled by these resources, namely machine learning.[1] Extremely deep models that have become routine over the last decade have upturned both forward facing and back end tools throughout daily life. Trained machine learning (ML) models are guiding stock trading, driving targeted ads based on a user's interest, automating surveillance through object and facial recognition, providing virtual assistance, analyzing healthcare data, among an uncountable pool of recent applications.[9–12] Most recent ML successes are attributed modern "deep learning" (DL) methods. DL methods have also begun to become routine in the sciences; a number of exciting advances prove that there is headway to be made in leveraging modern DL for chemical, biological science and engineering applications. DL has been used for chemical property prediction in the discovery of lifesaving drugs, critical industrial processes, and revolutionary material discovery throughput, to name a few.[13–18] Unfortunately the tools and methods used in "traditional" DL applications do not always translate out of the box - nuances associated with the subject matter must be built into the problem in order to produce a robust model.[3,6] For example, deep learning has become integrated within the field of computational chemistry, however the ML solutions there have significant differences from architectures and strategies in e.g. computer vision.[19] With advanced ML typically being taught or leveraged in Computer Science programs and roles, its intersection with chemistry and related fields calls for an important focus: practicing and teaching the decision making and design process for applying deep learning to chemical and biological systems.

At the core of any ML model is a goal centered around a subject, also called an example. It is of the utmost importance to define the subject and build in assumptions of the system in such

a way that it can be interpreted by the machine.[5,20] ML models are not artificial general intelligence, they do not think or reason, they simply construct a complex system of mathematical operations that mindlessly optimize a goal function. A failure to capture nuances in the design problem can be detrimental. Due to this, the construction and interpretation of a deep learning problem can be a monumental task with a myriad of pitfalls. A failure to appropriately define the system of interest, or align the learning goal given to the model with the real world application goal leads to an unsuccessful model, or worse, one that we think is successful but crumbles in practice.[4,21] This dissertation describes work both to exemplify and to teach the nuances at the intersection of machine learning and systems involving molecules.

In "METHODS BACKGROUND AND SPECIFICS" I aim to provide context and justification for specific methodology used in the following work. Readers not already comfortably familiar with deep learning, molecular systems, protein primary and tertiary structure, and bioinformatics tasks such as homology search can be selectively brought up to speed on the literature surrounding the fundamentals of preparing for deep learning on these systems. Following, the first chapter of this dissertation focuses on the design process and downstream analysis of a deep learning application to prediction of quantum reaction rate constants.[22] These rate constants are necessary for the study and design of systems involving chemical reactions, such as reaction networks occurring in combustion, a product yielding industrial reactor, or distant galactic gas clouds.[7,23–25] Here, DL was leveraged to predict rate constants with quantum accuracy and with less cost that rigorous *ab initio* calculations or experiment. A different set of DL models was created to predict molecular partition functions, including partition functions of unknown reaction intermediates.[26] In conjunction with existing tools, these partition function models reduce the cost of estimating reaction rate constants (albeit without quantum accuracy) down to negligible.

In the second chapter, I discuss the creation and administration of an optional learning event for undergraduate chemical engineering students to advance their coding and data science.[27] This work aimed to help fill the need for instruction at the intersection of machine learning and chemical sciences, mentioned above. Finally, in the third chapter I outline progress made to translate proteins into thermostable variants using neural machine translation (NMT). In this pursuit, a large and novel dataset was created.[28] Proteins serve a swath of functionalities that we can leverage in our lives and processes, but can lose their abilities at high temperature.[8,29,30] This work aimed to use deep learning to better design primary structure for thermal stability, to complement recent DL advances in protein engineering like predictors that can aid with directed evolution campaigns, or structure-to-sequence models that can produce sequences expected to achieve a target fold. Along with capturing general thermophilic trends in proteins, the model is able to target mutations, including insertions and deletions, that confer improved thermal stability. It proposes stabilizing mutations for the test case Engrailed Homeodomain (En-HD) of *Drosophila melanogaster* and qualitatively ranks existing variant libraries for acetyltransferase LovD and Lipase A.

METHODS BACKGROUND AND SPECIFICS

Deep learning on molecular systems

Training a deep learning model

The supervised machine learning problem (SML) is defined as follows. For an example of interest, there is a relationship that relates some set of features *X* to a target (also called label) *y*:

$$y = f(X) + \epsilon \qquad (Eq.\ 0\text{-}1)$$

Here, ϵ is noise on the target *y* unable to be captured by the features. An SML model seeks out an estimation of this relationship $\hat{y} = \hat{f}(X)$ such that $\hat{f} \approx f$ and so predictions $\hat{y}$ of the target are close to the actual value of the target. This estimator is advantageous when features are easily acquired and the target is difficult to determine, making the estimator an alternative to significant effort. There are many architectures, e.g. functional forms of $\hat{f}$, but each needs to be trained to capture the relationship between features and target, thus a set of data with known targets is required. This is typically the bottleneck in SML; estimators benefit from more data to learn from, but the effort that the model seeks to circumvent must be given for each example used to train the model. An estimator that is unable to capture the complexity of the true function and so predicts with high error is said to have high bias, whereas an estimator that is complex enough to pick up on the noise resulting in poor predictions for future examples (overfitting) is said to have high variance. An optimal SML model balances bias and variance, making low error predictions both for the data it was trained on as well as unseen examples.

The supervised machine learning problem can be attacked with a model architecture composed of artificial "neurons" wherein neurons are organized in layers and layers are interconnected, canonically referred to as a dense neural network (DNN) or multi-layer perceptron (MLP).[31] A model with many layers of interconnected neurons is referred to as a DL model. While there are neuron-based model architectures that leverage mathematical operations that differ from the strict formulation of an MLP, such as graph networks and transformers, even these more complex architectures use MLPs as functional units. MLPs require a fixed length set of input features, which are mathematically transformed into one or more targets. A diagram of an MLP and its atomistic unit, the neuron, is shown in Figure 0-1.

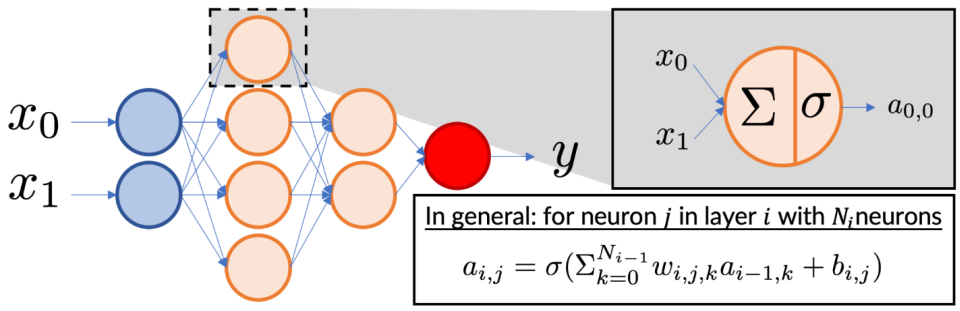

Figure 0-1: left) A small multi-layer perceptron with two input features x, followed by a hidden layer with 4 neurons, and another hidden layer with 2 neurons, and finally an output layer with one supervised target y. right) A single neuron conducts a linear operation on all outputs of the previous layer, a weighted sum with learned parameters w and b. The output a of the neuron is determined by applying a non-linear activation function σ to the linear portion of the neuron.

As MLPs grow deeper, they can capture arbitrarily complex functions,[32] becoming unbiased. The values of the parameters (See Figure 0-1) within the model must be determined through training via gradient descent of the loss surface. The capability of the model is evaluated according to the loss function calculated for training data given to the model. Critically, this loss function is differentiable with respect to each parameter within the model, thus at any place on the loss surface, the loss gradient can be computed in order to update model parameters in the direction

of lower loss. The amount that the parameters change is attenuated by a scalar factor called the learning rate. Unfortunately, the number of parameters within the model grows as $(N_{i-1}+1)N_i$ for each layer of neurons i, where N_i is the number of neurons in the layer. Thus, the dimensionality of the loss can become millions to billions as the model gets deeper and deeper, so the parameter optimization process can be extremely computationally intensive.[33] All ML applications described in Chapters 1 and 3 are neural deep learning models. Models in Chapter 1 are DNNs, while models in Chapter 3 are Transformers, see "**Attention based "transformers" for arbitrary sets**".

There are a number of tricks of the trade used in the pursuit of a fast parameter optimization towards an effective model with low bias and variance, some of which are summarized in Table 0-1 below. Each of these methods involve choices made by the model developer called hyperparameters.

Table 0-1: Summary of techniques and their hyperparameters used to arrive at effective parameters for a deep learning model.[34]

Technique	Hyperparameters	Description
Mini-batch training	batch size	Update model parameters more frequently by using loss gradients computed for small batches of the training data
Loss optimization	Loss function, type of optimizer (eg. SGD, RMSProp, Adam), learning rate, momentum rates	Method for determining how much and in what direction to change parameters on the loss surface, given the loss gradient
Regularization	Type (eg. L1, L2, dropout), strength	Dampens overfitting of model parameters
Normalization	Type (data, layer, batch)	Keeps neuron activations in regions of the activation function with high gradients, keeping the loss surface from plateauing

The learning hyperparameters above (non-exhaustive) as well as architectural hyperparameters such as number of layers, layer width, activation functions used, among others, results in an extremely large hyperparameters space. Unfortunately, poor choices in these hyperparameters can make the difference between an effective, generalizable model and a useless one. In conjunction with the computational cost of training deep models, this means that finding the best hyperparameters (hyperparameter optimization or hyperopt) is an extremely time, compute, and carbon intensive process. The most basic strategy for finding effective values for M target hyperparameters is a search over an M dimensional grid, discretized with some granularity. This is a rigorous strategy, but does not address the energy and carbon cost problem of model

hyperopt at all. It is preferable to use informed sampling strategies,[35,36] or evolutionary algorithms,[37] for hyperopt, reducing carbon cost. Hyperparameter optimization/affect is discussed for each ML application presented in Chapters 1 and 3. There is also a significant recent push to estimate and report the carbon cost of machine learning training, both to highlight the cost of the rise of deep learning as well as inspire strategies to reduce it.[38–40] This was done for the work described in Chapter 3.

Self supervised pre-training

As mentioned above, one of the bottlenecks of SML is the availability of labeled data. For some applications with large (>100k) and rich unlabeled data, schemes can be developed to learn the structure of the input examples in a self-supervised manner (SST). This generally looks like masking or hiding some random portion of each example, and optimizing parameters on a loss that measures the ability of the model to reconstruct the missing parts. Autoencoding is another strategy, where the model must compress and then reconstruct the input. SST has been shown to capture rich example structure, and the model can be used to create features for SML or directly "fine-tuned" for a downstream supervised task with increased performance over direct SML.[41–44] The pretrained "foundational" models can also, for some applications, provide zero-shot capabilities, e.g. effectiveness at a task without downstream fine tuning.[45,46] A selection of examples where self-supervised learning techniques have created rich foundational models in chemistry and biology is given in Table 0-2 below.

Table 0-2: Selection of self-supervised pretrained models for molecules and proteins.

Model	Input Example Structure	Training Data Size	Downstream Tasks
MolGNet[47]	Molecular Connectivity Graph	11 mil	QSAR for drugs, small molecules
ChemBERTa[48]	Molecular string (SMILES, SELFIES)	77 mil	QSAR for drugs, small molecules
Vagrant[49]	Molecular string and 3D atomistic structure	130 k	QSAR and inverse design for drugs, small molecules
ESM2[50]	Protein Primary Structure	617 mil	Binding, stability, activity
ProtTrans (ProtBERT, ProtT5, etc.)[41]	Protein Primary Structure	2.5 bil	Binding, stability, activity
STEPS[51]	Protein Primary + Tertiary Structure	40 k	Binding, stability, activity

Nearly all highly successful pretrained models, including the ones above, are based on a transformer-like neural architecture that leverages an attention mechanism for arbitrarily sized sets representing a single example, and are trained on very large datasets. Refer to "**Attention based "transformers" for arbitrary sets**" for an overview of this architecture. SST foundational models are leveraged for all DL applications in Chapter 3.

Representation of atoms, molecules, and reactions to the machine

One of the largest hurdles with deep learning applied to molecular structure is the process of featurization. Inputs for neural layers must be fixed length vectors such that matrix operations can be conducted on them in a forward pass. Thus, either the molecule must be encoded as a fixed

length vector, or cleverly decomposed into units that can be. The featurization hyperparameter can be debilitating, as it defines the starting information available to the model; if the features given are not rich in information on the target, a deep learning model cannot create a reasonable relationship regardless of its complexity. Effective featurization methods has been a widely studied topic.[7,52–54] This problem can be decomposed into two possibilities: one example is one molecule represented by atom connectivity without any resolution on atomistic positions, or one example has atomistic 3D resolution. The first is sufficient when the target of the model is a quantity that is an average over the ensemble of states available to the system, such as solubility. The second is necessary when the target is something that must be studied as a function of atomistic positions, such as molecular partition functions. A depiction of different encoding methods and models is given in Figure 0-2 and discussed following.

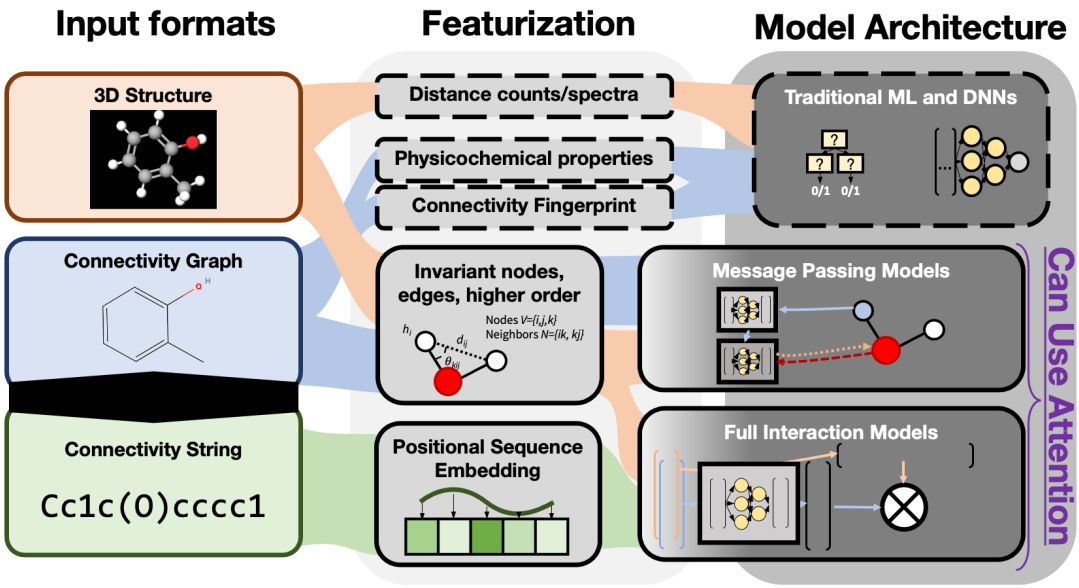

Figure 0-2: Overview of encoding and machine learning strategies for molecules. Molecules can be viewed by their connectivity, or by their 3D atomic resolution, impacting the methods of encoding. For both cases, fixed length feature vectors can be developed by expert design, such as physicochemical properties, bonding fingerprints, or distance matrix counts. These can then be given to traditional ML architectures. Both can also be decomposed into graph or cloud like representations, and neural operations that pass information among neighboring atoms can learn to encoding molecules without any human design, directly from the data. Fully interacting models that consider information from all atoms at pairwise or even higher order, such as E3NN, can operate directly on 3D space without decomposition into graphs. Finally, strings of the molecule like SELFIES can be considered for generative sequence models like canonical transformers, that treat all tokens (letters) in the sequence pairwise.

For connectivity-based features, one often considers computing a vector of "descriptors," which are generally human interpretable exact quantities about the structure such as molar mass or are estimates of properties such as polarizability. Descriptor techniques have been used for chemical property prediction for decades,[13] and tools exist to rapidly produce large vectors of common descriptors.[55] Alternatively, one can determine "fingerprint" based features, which are not as human interpretable but encode the molecular graph into graph distance counts centered around each atom in the structure. They have also seen heavy use in property prediction.[56] Another option seeing recent focus is learned representations, which can be applied to both connectivity and 3D cases.[57–61] Here, the molecule is decomposed into atoms, which are featurized by encoding

atom types, and operations over atom connections are used to produce a fixed length feature vector. For this strategy, atoms represent nodes in a graph and edges between nodes are bonds. Neural units then convolve information across edges in a process called "message passing." Learned features are advantageous because the feature vector is part of the training task instead of designed by humans; the learned feature vector is produced specifically to best make predictions of downstream tasks. Unfortunately, this also means that a significant amount of data are required to also train the featurizer end-to-end with the predictor, so it is difficult to leverage for tasks involving small datasets.[62] One may also leverage sequence-based models, canonical to natural language processing (NLP), to directly encode a molecular string like SELFIES.[46,63] These learned featurizers also require significant data, and are typically used for tasks involving molecular generation.[49,64–66]

For problems requiring 3D structure, there are a wide variety of methods, and learned representations have also been explored in this space. This is done by representing distances in graph edges and drawing edges between atoms that are not necessarily "bonded". This is challenging because the number of edges in this case grows as N^2 compared to the connectivity case above where the number of edges is approximately N. Given that each message passed is effectively a forward pass through a small MLP, and usually multiple messages are passed for each edge, this results in more than N^2 times the cost compared to representing a molecule as a fixed vector and passing it to an MLP. In practice, this extreme expense means that edges are not drawn between very distant atoms, but this produces a challenge unique to the 3D problem: equivariance.[67] It is possible that two different 3D structures map to the same graph if distances alone are considered. To circumvent this, clever modifications allowing message passing over higher order interactions to occur are required, such as the e3 equivariant neural network

framework.[68,69] There are also many human designed featurization techniques for 3D structure including Coulomb Matrix,[70] Autocorrelation functions,[71] and EncodedBonds.[72] These techniques necessarily involve encoding the distances between an arbitrary number of atoms of different types into a fixed length vector.

Encoded Bonds, is effectively a histogram of atom distances of certain pairing types, and is leveraged in Chapter 1 (See section "Predictions of stable and transition state partition functions"). For example, one of the resulting features using encoded bonds may be "number of carbon-nitrogen distances between 2.0 and 2.2 Angstrom". This is advantageous over something like a coulombic interaction matrix as it does not require padding the feature vector with zeros, known to have a negative effect on learning, yet the size of the system is still inherently captured by the bin heights.[72]

In order to represent reactions, multiple molecular representations can be combined either by concatenation or subtraction. For example, the modeling of reaction energy prediction can be done by using features both from stable reactants and stable products.[20,73] One can also consider using features associated with intermediate structures, the energy barrier separating reactants and products, or graph representations of the entire reaction.[22,60,73,74] Features are engineered for reaction barriers in Chapter 1, see "Creating a dataset of hypothetical quantum rate constants".

Representation of protein primary and tertiary structure

Proteins have been the subject of supervised machine learning tasks for decades. Major topics include activity prediction, binding affinity, stability, and both the forward and reverse protein 3D structure engineering problem.[75,76] Much like small molecule systems (see section "Representation of atoms, molecules, and reactions to the machine") a protein can be represented

to a ML model starting from the 3D structure, or the bonding graph, which in this case is a linear sequence of amino acids. A depiction of different strategies is given in Figure 0-3. The 3D case can be treated identically to the 3D case of molecular systems - distances and atom and/or residue types are encoded into a fixed length distance map, or a representation is learned by equivariant neural networks which leverage an attention mechanism to embed structural context. There is some variability in resolutions considered here, one can model just alpha and beta carbon atoms and side chain, or all atoms. In the case of primary sequence, proteins differ somewhat from small molecular systems. One can still compute a vector of descriptors or fingerprints describing the protein such as k-mer embeddings or amino acid properties, however as opposed to graph based message passing, learned representations for proteins have been developed by leveraging techniques directly from natural language processing.[41,77,78]

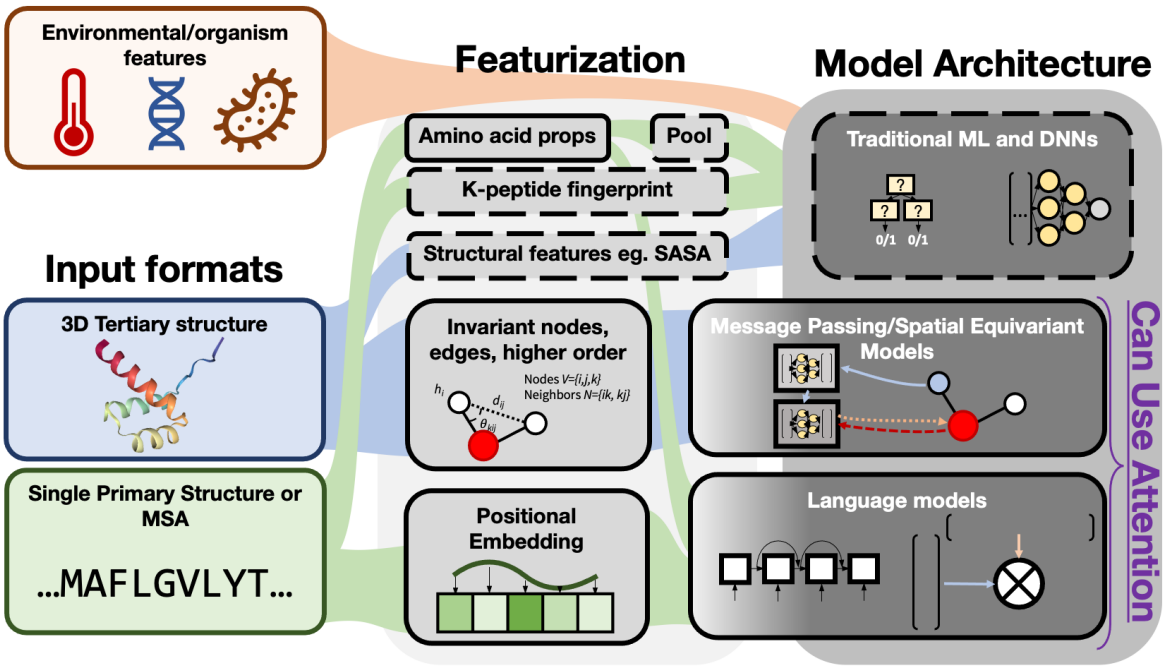

Figure 0-3: Overview of encoding and machine learning strategies for proteins. Proteins can be viewed by their primary or tertiary structure, impacting the methods of encoding. For both cases, fixed length feature vectors can be developed by expert design, such as solvent accessible surface area, percentage of buried residues and other structure features, counts of protein k-mers, and aggregation over amino acid properties. These can then be given to traditional ML architectures. Tertiary structure can also be embedded by message passing over e.g. bonds, neighbor atoms, or neighbor residues, up to fully equivariant 3D neural networks. Language models such as LSTM or transformers can be used to embed primary structure. Note that often methods will modularly combine strategies, e.g. to use a language model on amino acids, amino acid types, and combine the results with protein or organism level properties like amino acid compositions and environmental conditions.

Much like a sentence or paragraph can be decomposed into word or sub-word tokens from a fixed vocabulary, primary amino acid sequence can be decomposed into amino acid tokens, and transformer architecture seen in NLP can be directly applied. In this case, each of the 20 amino acids are assigned a one-hot embedding on a length 20 vector, and the resulting l x 20 matrix can be positionally embedded and passed into a standard transformer-based model. Note the encoding may be longer than 20 in order to accommodate additional tokens like stop and start, and in some instances, developers have chosen to assign rare amino acids (U, O) to the same token.[41] There are subtle differences to the protein language as to a written one, for example the vocabulary is ~20 amino acids as opposed to thousands of letters and the "sentence" length can be hundreds of amino

acids as opposed to dozens of words. This does have a minor effect on the model, for instance the longer sentence length adds to the memory required to input data, generally limiting training batch size. Regardless, treating primary sequence as a *natural* language problem is clearly viable with modern attention-based architectures, and is leveraged in Chapter 3 for all ML applications.

Data splitting for effective evaluation of trained models

In ML it is necessary to reserve a set of data to be held out to final model evaluation, as an approximation of how well the model will perform on future examples, which is the primary goal of a ML model. Critically, this test set of data must not contain examples or information from the data that is seen by the model, referred to as the development set.[79] A model with low bias that can effectively learn from the development data is likely to overfit, and do poorly on future examples, which can be caught using a properly designed test set: a model that is overfit will perform poorly on the test set since the data were not seen during the training process. When the test set is compromised with dev set information, it is referred to as data leakage and any claims made about the model performance are invalid.[80] To ensure that the test data is a rigorous evaluation of the model goal, it is extremely important to take care in this splitting process.

A very common strategy is to randomly split examples in a dataset into development and test. This ensures that no one example exists in both datasets, assuming the initial data was unique, but it does not prevent more insidious forms of data leakage. For example, if one's goal is to make predictions about the properties of molecules, and the test data contains molecules whose molecular structure is highly similar to training data, then the test set may not be a quality indicator of prediction capability on new molecules. Random splitting is not sufficient when the canonical ML assumption that the data are Independent and Identically distributed (IID) does not hold. A

rigorous split necessitates expert knowledge of the data space. A few of the data splitting strategies leveraged across molecular scale-domains in this work are described below.

For systems that make predictions involving molecules, such as quantitative structure-activity predictors or kinetic predictors, a good portion of predictive capability can be captured by simple physicochemical properties, for which very similar molecules exhibit very similar properties.[81] For this reason, a model that has seen Benzyl chloride would likely perform very well on Benzyl bromide, simply by virtue of having seen the actual experimental/measured value for a nearly identical molecule. This is a poor indicator of how well the model would perform on more distant molecular systems, e.g. linear Alkanes. A more rigorous approach to splitting this type of data is based on the molecular scaffold,[82] ensuring that test and development sets do not contain data with the same carbon backbone structure. Specifically, every small molecule in the dataset is grouped by its Murcko carbon scaffold. Groups remain together for data splitting such that no single scaffold will occur across train/test splits or data folds. This strategy is leveraged for partition function prediction (reactant used when considering reactions) in Chapter 1.

For systems that operate on protein amino acid sequence, a few rigorous strategies are used in Chapter 3. The strategy used is dependent on the protein task at hand, but in some cases is extremely critical, as the IID assumption does not hold due to evolutionary relationships and vast regions of empty protein space.[83] For example, it is known that amino acid distributions leveraged by an organism are dependent on the environment and evolutionary history; this distribution alone has some predictive power over a protein's source organism.[84] For the model trained to predict host organism growth temperature from sequence alone, discussed in Chapter 3, this is a detrimental attribute as it allows the model to learn a relationship between sequence and *organism* instead of sequence and *temperature*, then memorize which organisms occur at which temperature.

For this reason, a strategy leveraged here is to split by the source organism.[85] One might also split proteins temporally, where novel contributions to a database are held out and older ones used for training. This is true for the task of structure prediction with e.g. the CASP or CAMEO hold out sets, though they apply some additional filtering.[86] For tasks that are protein specific, e.g. fitness landscapes around a starting protein where random mutagenesis or recombination of natural sequences is used, random splitting is perfectly rigorous, or sometimes splitting based on natural vs designed sequence.[87,88] For tasks that involve a general capability across diverse protein families/folds, splitting becomes significantly more difficult. For example, proteins with >50%id often have same-or-similar function, yet there are plenty of cases where protein pairs with <30%id retain the same function.[89,90] Often we consider sequence homology instead of identity when comparing homologs, which is a better metric for remote evolutionary distance. Yet here as well, one might have close homologs with different functions or distant ones with the same function. Protein functional/similarity is highly diffuse and continuous; it is not possible to draw perfect discrete barriers over protein space. Due to this, there is no perfect or otherwise standard splitting strategy for protein sequences that guarantees proteins across splits contain zero mutual information available in the literature. This is also undesirable, as information of evolutionary history is rich and can help make effective models.[91,92] Yet, it is still desirable to split proteins with some amount of rigor such that models cannot perform well on test sets by memorizing examples. Common strategies all involve applying some sort of similarity metric to pairwise proteins, and constructing clusters of similar proteins.[83,86,93,94] Sequence identities of 30-50%, UniProt cluster labels, and alignment E-values of about $1e^{-4}$ have been used in the past. If possible, more sensitive hidden Markov model (HMM) scans are considered more rigorous than sequence identity.[95] Sequence identity is leveraged for NMT dataset creation in Chapter 3 due to the cost of creating

multiple sequence alignments (MSAs) for the novel dataset used. Clusters are kept together during data splitting, approximating a split on the protein family, superfamily, or subfamily, depending on the method.

Another form of data leakage is the use of testing data during hyperparameter optimization.[80] If capability metrics on the test set are used to optimize the model architecture, then naturally the model will perform well on that particular test set, but is not necessarily evidence of a model accurate on new examples. For this reason, the development set is further split into training and validation sets using the same splitting strategies as above. The training set is used for determining model parameters, while the validation set is used to score model capability on new data. Thus, the validation data is an approximation of test set performance, which is itself an approximation of future performance. The validation data is then used to guide hyperparameter optimization without incorporating data from the test set. The gold standard validation strategy is to split the development set up into *K* folds, as opposed to a single train-validation split. *K* models are then trained, each using one data fold as validation and the remaining development data as training. Validation scores of the *K* models are averaged. This strategy prevents bias in a single validation set from misguiding hyperparameter search, but also requires approximately *K* times the computational cost.[5] Typically, the number of folds is 5 or 10 when the training cost of the model allows, but sometimes the model is too large to conduct multi-fold validation considering the cost of model training.[33] K-fold validation is leveraged in Chapter 1, where the ML models are small enough to incur the computational cost.

Attention based “transformers” for arbitrary sets

Transformer based neural networks have seen recent success in a variety of applications, originally applied to the field of neural machine translation.[96,97] While there are many variations of transformers, they all are composed of MLP units and rely on a neural mechanism coined “attention.” Attention as a method is not unique to transformers; attention is often used to weigh message passing in learned molecular representations (See section “Representation of atoms, molecules, and reactions to the machine”). Here, attention is discussed in the context of transformers. The original transformer architecture, composed of an encoder and an autoregressive decoder, along with a depiction of the attention mechanism is shown below in Figure 0-4. Attention produces two practical side effects that have driven this architecture to the forefront of deep learning: 1) they can effectively act on variable length inputs, where MLPs require a fixed length vector or zero padding, which impacts performance, and 2) they can naturally address masked machine learning problems, where unlabeled data can be used for learning by asking the model to fill in masked portions of the input. These two practicalities mean that vast quantities of unlabeled data can be leveraged for SST, and are discussed below.

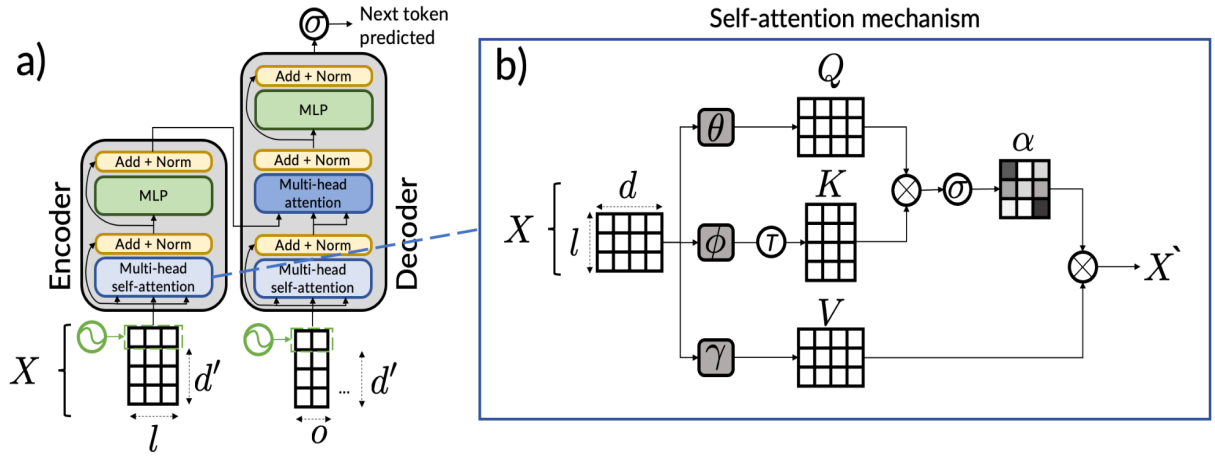

Figure 0-4: a) A vanilla transformer neural network composed of an encoder and a decoder. All matrix shapes are representative. An input sequence of l parts each with d' features is concatenated with a positional encoding to produce an input matrix X. The model can be used to generate a new sequence of o parts. The encoder embeds the input sequence into a fixed length vector using self-attention. The decoder does the same with an output sequence, which is then used along with the input encoding in a cross attention mechanism to generate the next portion of the output sequences. b) The self-attention mechanism. Learnable linear transformations θ, ϕ, γ act on the input to produce queries (Q), keys (K) and values (V). Queries and keys are multiplied and softmaxed to produce a normalized "attention matrix" α that is a pairwise expression of the weight that information at some position in the length l sequence should be considered at some other position in the sequence. The new state of information at each position in the sequence is then updated using the attention weights and the values. In the case of cross attention, instead of the same input producing Q, K, and V, a different matrix is used to produce K and V. Note that generally a "Multi-attention head" is used, here the input is sliced along d features and each slice is used in a different attention head before concatenating the results back together.

In a transformer model the input is decomposed into a size l set, and each portion of the sequence must be embedded into a vector of d' features. For sets that have an ordering, e.g. sequences, a positional embedding is added to each vector to provide information on the part of the sequence the vector applies to. In the NLP setting, the sequence is a series of "tokens" each representing a word or sub-word. The equation for embedding an item in this setting is given in Eq. 0-2, where I is a one-hot-encoding over the vocabulary of size d' and S is a vector of k periodic functions evaluated at i. The resulting token embedding is then of size $d' + k = d$. The transformer is not exclusive to ordered sequences described here, however, and is viable for any set as long as each portion of the set can be represented by a fixed vector. A transformer may be composed of one or both of an encoder and a decoder. The output of each is a vector of some size for each input

token. In the encoder case, these can be considered a condensed representation of the item in its context, while in the decoder case, this is usually the softmax (sum=1.0) distribution over all possible labels e.g. the vocabulary *d'* given some set of previous tokens. The decoder can generate sequences autoregressively by iteratively selecting the most likely item. This makes transformers highly modular and flexible for NLP: encoder-only models can be used to embed arbitrary sequences for downstream tasks like sentiment analysis on words or whole passages, while encoder-decoder and decoder-only models can be used to generate new sequences either spontaneously or when conditioned on some input, e.g. NMT, summarization, or joke generators.[98,99] In these generation use cases, the output probabilities are generally attenuated with a "temperature" and can be sampled, making generation a stochastic process, or searched over using BEAM search for highly probable sequences without getting greedily trapped by early choices.[100]

$$x_i = I(y_i) \rightarrow \{0,1\}^{d'} || S(i) \rightarrow \mathbb{R}^k \qquad (Eq.\ 0\text{-}2)$$

In many applications, which portions of a sequence have a large impact on other portions is highly context dependent. Attention addresses this by providing a way for information at some location in the sequence to provide context to another portion of the sequence in a sequence dependent manner, thus for a given sequence, the "interacting" portions of the sequence are identified. This makes attention a favorable solution for applications with context dependent sequences without the drawbacks of other strategies such as recurrent neural networks, discussed elsewhere.[101] As an additional benefit of the attention mechanism, inputs of arbitrary length can be used as opposed to an MLP where inputs must be padded with 0s, hindering learning.[72] With attention, a sequence that is shorter than the positional encoding input size is still padded with zeros, but the attention mechanism applies 0 weight to those indexes, thus ignoring the padding.

In the attention mechanism for transformers, all items in the set are attending to each other pairwise.

Transformers are pretrained – and finetuned depending on the downstream task – using masked or causal language modeling and the categorical cross entropy loss function. In masked language modeling, parts of the target sequences are masked and the model predicts a distribution of tokens at the masked positions. In the causal language modeling case, the probability of tokens over the "next" position in a sequence given all previous positions is predicted. Typically, teacher forcing is used, where the actual labels are used to condition the prediction of the next token, regardless of what the causal model would have predicted for a previous token. These predicted probabilities are compared to the true token at each position via categorical cross entropy, typically later averaged over all positions:

$$CCE_i = \sum_j^{d'} -\mathbb{I}(j = y_i) \ln P_i(j) \qquad (Eq.\ 0\text{-}3)$$

where *d'* is the set of possible tokens, or the vocabulary. The true token label is y_i.

The result of this training strategy is that transformers can evaluate the likelihood of parts or a whole sequences in addition to their intended tasks. For a causal model, like the NOMELT model trained in Chapter 3, the output softmax probability over tokens at some position i in a sequence $P_i(\cdot\,|x_0, x_1, \ldots, x_{i-1})$ is given as P_i where x_i is the token identity at position i. The normalized probability of a particular token j from the vector at that position is written as $P_i(j)$. The log likelihood of a sequence or a set of tokens at specific positions can be measured according to Eq. 0-4 below, where M is some set of positions in the sequence, from a single one up to the whole sequence[88,92,102]:

$$L(\boldsymbol{x}) = \frac{1}{M}\sum_{i \in M} \ln P_i(x_i) \qquad (Eq.\ 0\text{-}4)$$

Further, the likelihood of a specific token change at a particular position given a fixed set of all previous tokens, the combined probability of a set of token changes, or the difference in causal likelihood of entire sequences of arbitrary length can be measured. This quantity is useful when comparing sequences, such as evaluating the likelihood of a mutation on a protein sequence occurring relative to the wild type, WT, in Eq. 0-5 below:

$$\Delta L(\boldsymbol{x}_{variant}) = L(\boldsymbol{x}_{variant}) - L(\boldsymbol{x}_{WT}) \qquad (Eq.\ 0\text{-}5)$$

A disadvantage of transformer based models is the cost of training and predictions. Composed of often billions of parameters, training transformers on a single PC is not possible. Instead, often multiple GPUs are required in parallel on a computing cluster. Not only does this training requirement limit the number of researchers with the resources to train them, it is also more difficult for an ML practitioner to train a transformer over a smaller DNN, requiring novel and finicky hyperparameters such as learning rate schedules, batch normalization, label, smoothing, and specialized software environments meant to speed up model training at scale like DeepSpeed.[103] Given this, only researchers with exclusive access to hundreds of GPUs can feasibly optimize transformers over many hyperparameters. All of this of course condenses into a significant energy cost.[39,104]

Homology modeling

One major task in the study of proteins and a critical step in the work conducted in "Developing a database of low and high temperature paired proteins" is the labeling of two distinct proteins as same-or-similar function. It is important to acknowledge immediately the nuances

associated with the term “same function.” For example, is a trypsin the same function as a chymotrypsin? Both are proteases and have similar structure and mechanism, but different substrate specificity.[105] In conjunction with this nuance, it is worth noting that the only way to identify a protein function with full confidence is to observe it in its environment, meaning that the inference of same-or-similar function directly from primary and tertiary structure is not exact. Regardless, the identification and comparison of evolutionarily related sequences - homology modeling - allows us to approximate if two proteins occupy the same niche.[106] This process has two facets, comparison of primary amino acid sequence, and the comparison of 3D structure.

Sequence alignment

Homology modeling typically begins with local sequence alignment, where a protein is compared to every other protein in a defined database of proteins, and portions of the proteins are compared in order to produce an alignment. The alignment is scored, and one uses the likelihood that a score were to occur by random chance to infer that portions of the two proteins are homologous.[107] Optimal local sequence alignment algorithms have been available for some time, and include BLAST,[108] and FASTA.[109] An overview of the BLAST algorithm is given in Figure 0-5 below.

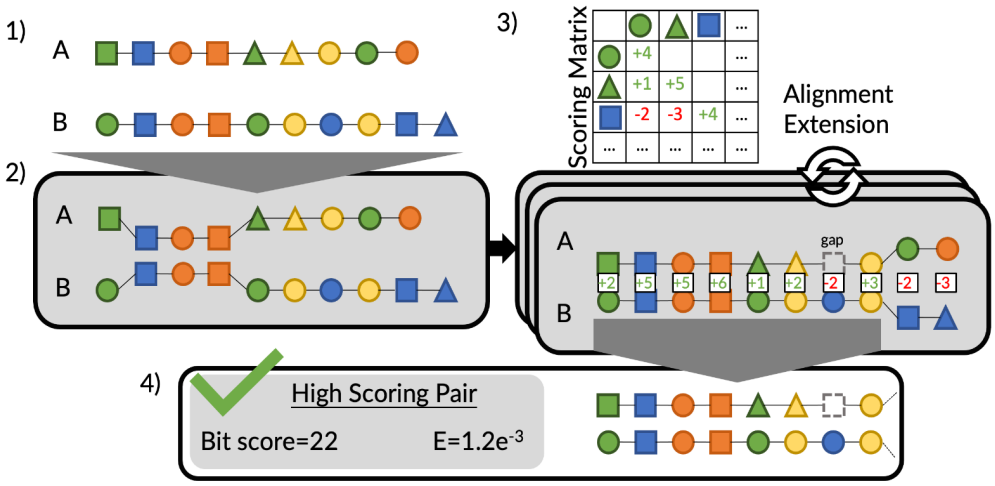

Figure 0-5: Depiction of the BLAST algorithm for a single query sequence A versus one sequence B in a target database. This process is repeated for protein A against all other proteins in the database. 1) Protein A is compared to protein B. Colored shapes are unique amino acid types, where same color indicates that two amino acids are phenotypically similar e.g. a green square may be Alanine while a green circle is Valine. 2) Exact matches in amino acids of some minimum length (3 is depicted) are identified as the start of an alignment. 3) A scoring matrix which defines whether an amino acid substitution is likely to be non disruptive (positive score) or disruptive (negative) to the protein structure is used to score each pair of amino acids in the alignment. The alignment is then iteratively extended: as pairs of amino acids are added to the alignment, they add their score according to the scoring matrix. The algorithm is usually configured to accept gaps (e.g. an amino acid was inserted or deleted in one sequence compared to the other sequence) with some penalty to the score. 4) When extension causes the score to go down for some number of steps in a row, the algorithm stops and the alignment is considered a scoring pair.

The E-value is a rigorous expression for the probability that a given alignment score occurred by chance for the search space used, and is defined below.

$$E = \frac{m \cdot n}{2^B} \qquad \textit{(Eq. 0-6)}$$

Where m is the length of the query protein, n is the total length of all proteins in the search space, and B is the bit score for some high scoring pair alignment identified. This value can be used to infer homologs, where values closer to 0.0 indicate a local alignment that is unlikely to occur by chance, and so is likely evolutionarily related. Note that the E-value is normalized by the sequence length and the database size, thus to infer homology for a longer protein or searching in a larger database, a larger bit score is required as both increase the probability of finding an alignment by chance. It is critical to note that the coverage of a local alignment need not be the entire query or

hit sequence; local portions of the amino acid sequence can be identified. An example of this would be a pair of two-domain proteins that share a single domain. Local alignment would identify the domain as homologous but the alignment would not cover the second, differing domain. This would be reflected as an alignment with less than 100% "coverage" of the constituent strands.

Searching for local alignment homologs among very large sets of proteins can be somewhat expensive.[110] Due to this cost, modifications to the algorithm have been made in order to speed up search time with limited impact on number and quality of homolog hits, such as with the DIAMOND software.[111] Typically, after running local alignment for a protein against a large database, the resulting hits whose E-value is small enough to infer a desired evolutionary proximity are considered homologs.

It is also possible to conduct Multiple Sequence Alignment (MSA) as opposed to single vs single pairwise alignment.[112] Algorithms such as PSI-BLAST[113] and JACKHMMER[114] conduct MSA. This strategy has advantages: it is able to identify much more distant homologs by considering multiple potentially intermediate sequences, and further it provides substantially more information on structure and function. MSA searches are the most reliable way to infer structure and function for a new unstudied protein sequence using primary sequence alone; sequences with known structure and function in an MSA that is aligned to the new sequence provide statistics on the conserved portions of the sequence. This is significantly more trustworthy than assuming a similar structure by comparison to a single pairwise alignment, where differences in primary sequence may or may not substantially affect the structure.[115] Indeed, the inclusion of a protein's MSA is a critical portion of the strategy that allowed the state-of-the-art protein structure prediction AlphaFold2 achieve accuracy.[91] Unfortunately, scanning a new sequence against known

MSA families with e.g. HMMs to find homologs is sensitive and fast, building MSAs for many different proteins from scratch using is extremely expensive.

Structural alignment

Protein 3D structures can be aligned pairwise or in multiple alignments analogous to sequence alignment, however it is a very different problem to score or evaluate a 3D structure alignment compared to a sequence alignment.[116] Firstly, once can consider the "degree of similarity" using a variety of different quantities. Some directly compare 3D structural elements such as RMSD in common residue positions between two structures. However, one can also consider extracting a similarity based on two dimensional features: instead of comparing distances, do common residues have the same contact map/tertiary interactions? Examples include the elastic Dali score.[117] There are a very large number of proposed metrics meant to score the similarity of two structures, and regardless of the choice, they are all predicated on aligning the two structures such that they can be faithfully compared.[116] One can treat the structures as rigid bodies and identify the rotation and translation that minimizes the RMSD of common residues while simultaneously maximizing the number of residues considered equivalent. Given that proteins are flexible and bodies that occupy different conformations, it is popular to instead compare RMSD of substructure elements and allow for flexible linkers during alignment.[118] These so-called flexible aligners such as FATCAT score the quality of the alignment according to the RMSD of substructure elements while penalizing large differences in linker orientation.[119] Considering that it is common for amino acid differences to cause conformational changes but are still highly similar structures, flexible aligners can find alignments where rigid body methods would yield extremely high RMSD.

The gold standard for inferring function of a protein is the structural alignment (requiring the subject's full 3D structure to have been measured) to structures in a primary sequence MSA. Specifically, the target protein's structure is measured along with its primary sequence. The primary sequence is used to search for an MSA using e.g. HMMER, and structures within the MSA are aligned to the target protein structure. A high degree of alignment leads to confidence that the new protein occupies the same or similar function as the known proteins in the MSA.

Protein Structure and Thermal Stability Estimation

Structure prediction

AlphaFold2 (AF2) has been extremely impactful on the community for its ability to model the three dimensional structure of protein sequences.[91,120] It leverages a novel architecture and a new and very large dataset to do this. There is a fairly involved data preparation step at prediction time. A target protein sequence is run against a large dataset of unlabeled sequences via jackhammer in order to produce a MSA. The MSA is given to a two dimensional "EvoFormer," which is a novel transformer that has column wise (evolutionary diversity) attention in additional to the standard positional attention of the transformer, see "**Attention based "transformers" for arbitrary sets**" for an overview of the transformer and attention mechanism. Optionally, AF2 also initializes with a series of 3D positional templates found by querying a large structure database for exact matches on k-mers in the input sequence. Information is exchanged in multiple layers before outputting a predicted 3D structure. The output structure can be optionally minimized using Amber force fields.[121] AF2 prediction is a stochastic process due primarily to input MSA subsampling at inference time. Due to this, each inference run will produce slightly different structures. The differences tend to be more pronounced for protein regions with low pLDDT, e.g. the model

confidence.[122] The developers provide 5 different trained model parameter sets. AF2 predictions average on the order of 1.0 Å RMSD, close to the resolution of structure measuring instrumentation. Note that for all AF2 predictions made locally in this work, model checkpoint 4 was used and the Amber relaxation was not conducted due to the downstream uses of the structures leveraging their own relaxation procedures.

Given the expensive of running AF2 predictions, with high CPU requirements for MSA building and GPU requirements for large model predictions, there has also been work to create a model that does not require evolutionary variability inputs to predict protein structure. ESM2 is a large encoder-only pretrained language model trained on billions of protein sequences in self-supervised fashion.[50,123] By training a small projection of the model residue embeddings to predict a distance matrix, the authors of this work were able to predict structure with surprising average accuracy at 1.9 Å RMSD. Unlike AF2, this method is quite fast, does not require and MSA, and is not stochastic, in exchange for the loss in accuracy.

Rational design methods can also be used to predict structure.[124,125] Here, templates from the PDB and an energy function fit using Bayesian statistics on PDB structure, are used to iteratively insert and perturb 3D structure until a final structure is achieved. With a longer runtime and poorer accuracy on average, these methods have generally been overshadowed for single pass structure prediction by the accuracy of AlphaFold2.

Prediction of protein stability

Accurate prediction of a proteins stability has been a desire for decades, and a host of different methods (and targets) exist.[126–129] Firstly, it is worth differentiating thermodynamic and thermal stability. Thermodynamic stability, often described by the folding free energy change, ΔG,

describes the equilibrium between native folded state and unfolded states, usually at ambient or natural temperatures. This is given in Eq. 0-7 below. Here, H is the system enthalpy in energy per mol, and S is the system entropy in energy per mol-temperature. Note that each term is a function of temperature.

$$\Delta G = \Delta H - T\Delta S \qquad (Eq.\ 0\text{-}7)$$

Astonishingly, this quantity is approximately the same order of magnitude across evolution among organisms living in different environments.[130,131] On the other hand, the thermal stability, described by the melting temperature T_M, describes some elevated temperature after which the equilibrium shifts and the protein loses function. See Figure 0-6B. These quantities can be described in both absolute terms, or relative to some wild type or starting protein sequence e.g. upon mutation. Further, predictors of these quantities fall generally into methods that leverage statistical potentials or first principles upon the protein structure, and those that leverage machine learning upon the protein sequence or structure or both. See Figure 0-6A for a breakdown of prediction method inputs and outputs.

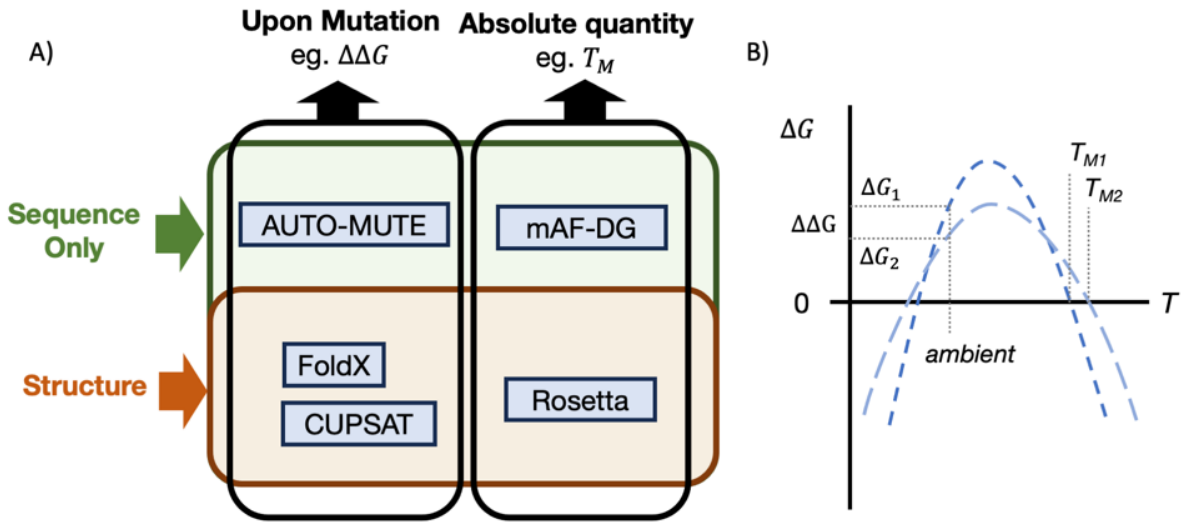

Figure 0-6: Breakdown of methods for the prediction of protein stability. A) Targets can be either absolute quantities of the protein, like melting temperature, or relative quantities to some wild type/parent sequence, such as change in folding free energy change. Methods to predict these quantities often operate on the 3D structure of the protein, applying statistical potentials, first principles, or machine learning. There are some methods that can make predictions with sequence alone, which leverage machine learning and sometimes environmental features. Examples are given for each category.[124,132–135] B) The thermodynamic curves for two variants of a protein. The folding free energy change, $\Delta\Delta G$, at ambient or environmental conditions, describes the difference in thermodynamic stability between the proteins. The melting temperature describes the thermal stability as the temperature at which equilibrium shifts to the unfolded state.

Much of the literature is focused on changes in stability upon one or a few point mutations.[133,134,136–141] Nearly all of them rely on an input 3D structure, and can incorporate statistical potentials, machine learning, environmental conditions, among others. Unfortunately, there are major drawbacks of this strategy, the most critical for the work discussed in Chapter 3 being that they can be poor in generalizability, biased towards destabilizing mutations and towards select protein families, as well as being unable to handle many mutations or insertions and deletions.[142,143] Further, many available methods are concerned with thermodynamic stability, while the work in Chapter 3 is targeting thermal stability. Thus, only methods that predict absolute thermal stability given a standalone protein sequence are considered, arguably the hardest category of prediction outlined in Figure 0-6A. Some of the first work to do this built statistical methods on dipeptide percentiles.[144] Since then, additional strategies consider higher order interactions, leverage machine

learning on amino acid features, and novel neural architectures and have improved performance and generalizability.[145–147] The best of recent predictors can achieve about 6ºC MAE on a blind test set, which while good enough to resolve distant orthologs is unfortunately not precise enough to differentiate between protein variants in small experimental melting temperature studies.[148,149] Additionally, the datasets used to train these models have a limited amount of diversity, coming from a small number of organisms and protein families.[150,151] This makes them unpredictable given a novel protein to study. In order to increase data size (and generalizability), at the cost of more noise, organism optimal growth temperature can be used as a proxy for thermal stability, but models trained in this way are even more imprecise and unable to differentiate between very similar proteins.[85,152] All of these supervised predictors are not precise enough to guide high resolution melting temperature optimization, for example to rank variants during a directed evolution campaign.

The recent mAF-DG is able to qualitatively rank protein variants by melting temperature without explicitly training on T_M targets.[132] Further, it incorporates structural knowledge into the prediction while still only requiring primary sequence like the supervised predictors thus far discussed. The general method leverages the stochastic nature of the state of the art structural predictor, AlphaFold2, which was trained on the sum of human protein structural knowledge: a number of structure replicates are predicted using AF2, and each is relaxed with the Rosetta energy function. The mean of energy values over this ensemble, analogous to the folding free energy change of the protein in Rosetta units, is the predictor output. This output was shown to qualitatively rank variants with 1-29 mutations by melting temperature, among multiple different proteins. None of these variants, or any melting temperatures whatsoever, were used to train the method.

Given the qualitative nature of the mAF-DG method to differentiate between protein variants, it was leveraged in Chapter 3 of this work. AF2 model checkpoint 4 was used, as suggested by the authors. The parameters for AF2 and Rosetta used were exactly as described in the original work, with a few modifications: the AF2 Ensemble size was 25 instead of 100, which the authors concluding was sufficient for prediction with little loss in accuracy compared to 100. Further, in order to account for a small amount of insertions and deletions between two protein variants, the score from mAF-DG was normalized by the sequence length. This is a novel choice, in that previously the method had not been used on libraries with insertions and deletions, and it was noticed that changing the number of residues changes the magnitude of the score strictly due to more or less free energy associated with more or fewer atoms on the system. The method was executed on 5 variants of Ribonuclease with sequences differing in length from 96 to 110 and melting temperatures between 41.1 and 53.2 ^{o}C.[153] When the mAF-DG method was normalized by sequence length, the method had a Spearman's correlation of -0.999 with P-value of $1.4e^{-24}$, confirming that normalization can be used to recover qualitative thermal stability estimation within a small difference in sequence length.

There has also been work to estimate protein thermal stability temperature with molecular dynamics (MD). Many of them use the same case study, Engrailed homeodomain, as is explored in Chapter 3. Unfolding pathways can be intuited by a number of dynamic attributes like core residue RMSD, solvent accessible surface area, constraint network analysis, native contacts, etc. The properties can be combined to define 1D reaction pathways for folding and unfolding, yet even simply observing RMSD of core Cα over the course of long simulations can give insight into whether the protein is unfolded. [154–158] All simulations conducted in this work were set up identically. The protein was placed in a periodic boundary box, solvating and charge balanced with

ions. Data presented is from 1 microsecond long production runs after NPT equilibration. The charmm36m forcefield was used.[159] The full configuration files used, including additional information such as timesteps, constraints, velocity cutoffs, etc. are given in Appendix C.2.5.

CHAPTER 1: SMALL MOLECULE REACTION RATE CONSTANTS

Understanding a system's reactivity is critical for the design of new materials,[160,161] to tune the production of compounds in industry,[162] for drug design[163] and so forth. Specifically, it is necessary to determine the thermal reaction rate constants, *k,* of each elementary reaction occurring within a reaction network, or an aggregation of steps that is descriptive empirically. In lieu of experiment, it is possible to calculate a rate constant estimation using first principles which has been studied extensively over the last century.[164–166] While there are many different levels of theory for the calculation of ranging from collision and transition state theory (relatively cheap) to dynamic coupled cluster calculations (extremely expensive), even the most approximate require a computational cost that can be prohibitive for systems containing many elementary reactions or large molecules. In the field of surface catalysis, this is combated by building scaling relationships that can be used to predict the rate constants of many participating steps from only one or a few "descriptors," reducing the cost of understanding a system by up to an order of magnitude.[167] Unfortunately, these relationships must be built for each system, thus the cost reduction occurs only when studying a known system on new surfaces, but does not allow for the rapid understanding of a new chemical reaction network, and is completely inapplicable for gas phase systems. Given the success of machine learning based approaches for molecular property prediction,[66,168] application to reactions has seen recent focus.[7]

Quantum reaction rates of arbitrary minimum energy paths

The difficulty of computing quantum reaction rate constants

To compute a thermal reaction rate constant, we recall the Arrhenius equation, the microcanonical Rice, Kassel, and Marcus theory (RRKM),[169–171] canonical transition state theory (TST),[172–174] and variational transition state theory (VTST),[175] among others. Here, the reaction is considered classically, and quantum phenomena such as reaction tunneling are not accounted for. More accurate methods such as path sampling,[176] and the flux-flux and flux-side correlation functions,[177] are based on the dynamics of the reaction, adding to the cost substantially. One can also use master equation based approaches,[178] and scattering theories.[179–181] Regardless of the method of choice, each with assumptions impacting the cost-accuracy trade-off, knowledge of the potential energy surface (PES) of the reaction is required. The PES is a surface defined by the potential energy $V(R)$ of the system for any orientation of atoms in 3D space R. The surface has 3N-6 dimensions where N is the number of atoms in the system. The potential energy is the electronic attraction and repulsion of nuclei and electrons, generally under the Born-Oppenheimer approximation where the potential energy parametrically depends on nuclei coordinates R and excited electronic states are assumed to have higher potential energy for all R. The energy of the ground vibrational state is often included in the ground state potential energy. Additional potential energy surfaces are required for non-adiabatic accuracy, where nuclear motion is not assumed to be unaffected by electronic states as in Born-Oppenheimer. Unfortunately, determination of the energy or gradients of the energy in R at any position on a PES requires an *ab initio* calculation.

While there are varying levels of *ab initio* theory from Hartree Fock to Coupled Cluster,[182] the typical choice for application in recent years is density functional theory (DFT) using the Kohn-

Sham equations.[183] While this method is not exact, it is common given its very good cost-accuracy tradeoff. Even still, a single DFT gradient calculation for a system of a dozen atoms can take on the order of minutes using high performance computing. This is prohibitive considering that computing a reaction rate constant dynamically on the potential energy surface for all but very small systems requires thousands of single point *ab initio* calls.[184] Due to this cost, all but the smallest reaction systems are generally considered assuming the reaction occurs over a single one-dimensional minimum energy path (MEP), with the transition state occurring at the highest potential energy on that path. Thus, the reaction must (classically) overcome an activation energy E_A in order to move from the locally stable reactant state to the product state, a larger activation energy will result in roughly exponentially slower kinetics of the reaction. The difference in the reactant and product local minima energy is the reaction energy (RE), which determines the thermodynamic equilibrium of the reaction. A depiction of reactions occurring on the MEP assumption is shown in Figure 1-1.

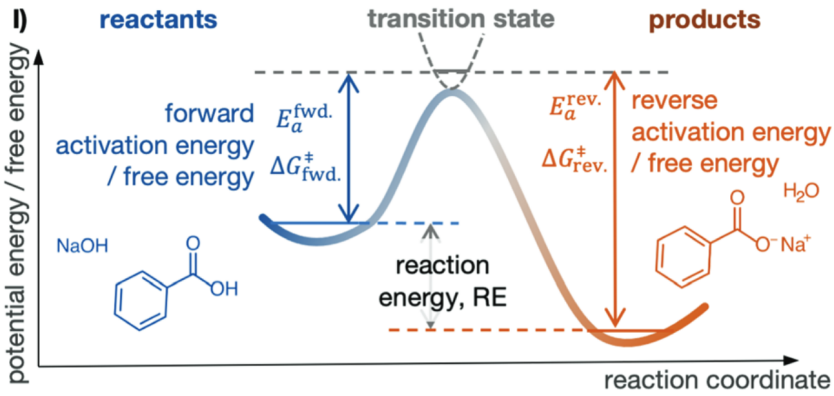

Figure 1-1: Overview of a reaction barrier. Can be represented as potential energy E, or including entropic energy to become Gibb's free energy G. This system must overcome an activation energy to move from locally stable reactants to products or vice versa. The larger this energy, the slower the rate of reaction for some temperature. The difference in energy between reactants and products (RE) determines the state of thermal equilibrium. If one local optimum is significantly lower in energy, the system will be mostly composed of that configuration if given enough time to reach equilibrium.

It is relatively routine to determine a locally stable configuration of atoms and the system's potential energy using gradient descent (~10 DFT calls) which makes determining the thermodynamics of a reaction quite straightforward. Unfortunately, even making the 1D path assumption, hundreds of DFT calls may be required in order to identify the MEP, and by extension the transition state. Common methods for this include Nudged Elastic Band and string methods.[185,186] At this point, one commonly computes the classical transition state theory reaction rate constant with a total cost equal to that of searching for the MEP (hours to days). In some situations the assumptions made in TST and VTST yields a reaction rate constant that is unacceptably far from the non-classical quantum rate constant, particularly at low temperatures.[177,187] This commonly occurs for atoms tunneling through a surface. In such a situation, currently the only option is to conduct a 1D wave packet or flux based propagation,[188] which can add hours to the cost already incurred by determining the MEP.

In this work, a deep learning model was trained to predict the quantum reaction rate constant using the minimum energy path alone, to circumvent this secondary cost of computing the quantum reaction rate constant accounting for effects such as tunneling.[22] First, a large dataset of hypothetical reaction barriers were generated. For each reaction barrier, the rate constants were computed analytically or using a flux derived transmission coefficient method.[188] Then, a dense neural network was trained on features extracted from the MEPs to predict the rate constant product $k(T)Q_R(T)$. Here Q_R is the partition function of the reactant, which was not separated from the rate constant k because the reaction barriers considered hypothetical reactants. The details are found in the next two sections, and the work is summarized in Figure 1-2.

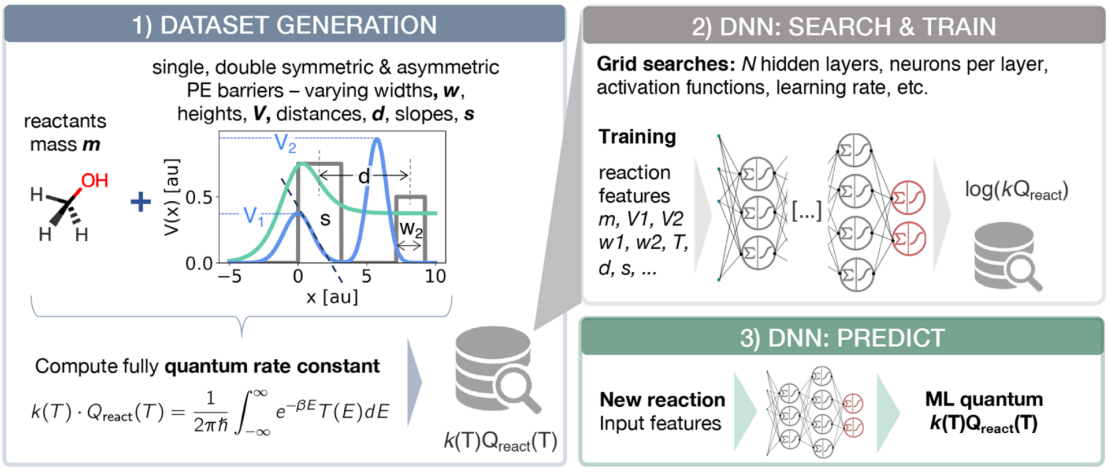

Figure 1-2: An overview of the work to predict quantum reaction rate constants. 1) Hypothetical single and double 1D reaction barriers are generated, and for each a quantum reaction rate constant product was computed analytically (rectangular, gaussian, Eckart, and Peskin type) or by computing asymptotic transmission coefficients in the energy domain. 2) DNNs were optimized in order to make predictions of the logarithm of the rate constant product at some temperature T, using features extracted from the barrier shapes such as widths, heights, and slopes. 3) Optimized predictors can make quantum rate estimations from MEPs without additional expensive propagation calculations.

Creating a dataset of hypothetical quantum rate constants

A database of ~6.9 million quantum reaction rate constant products, $k(T)Q_{react}(T)$, was generated for one-dimensional potentials. The quantum reaction rate constant products were computed in the energy domain from the general Eq. 1.1:

$$k_Q(T) \equiv k(T)Q_{react}(T) = \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} e^{-\beta E} T(E) dE \qquad (Eq.\ 1.1)$$

where $T(E)$ is the transmission coefficient and $\beta = 1/k_B T$.

The transmission coefficients were evaluated following the approach developed in Ref.[188] where it was shown that given a coordinate position, $x^{(1)}$, located far from the reaction barrier, one can write:

$$T(E) = 4\left|\psi_E\left(x^{(1)}\right) + \frac{\hbar}{ip}\partial_x\psi_E\left(x^{(1)}\right)\right|^{-2} \qquad (Eq.\ 1.2)$$

In Eq. 1.2 the coordinate position, $x^{(1)}$, must satisfy the condition $x^{(1)} \gg L$, where L defines the range $x \in [-L, +L]$ within which the potential is non zero. $\psi_E(x)$ is the eigenfunction of the Hamiltonian for the energy eigenvalue $E = \frac{p^2}{2m}$ where p is the linear momentum and m the reactant mass. This approach was successful in reproducing the full quantum reaction rate constant as could be obtained in the time domain using the flux-flux correlation function approach,[189] with the advantage of avoiding the demanding time propagation of the wavefunction. The reactant partition functions were considered as part of the output to be predicted using machine learning. For simplicity, the output product $k_Q(T) \equiv k(T)Q_{react}(T)$ is referred to as the reaction rate constant product. Eq. 1.1 above was solved for single and double symmetric and asymmetric rectangular, Eckart,[190] gaussian and Peskin[191] potentials as shown in panel 1) of Figure 1-2 and defined below in Table 1-1.

Table 1-1: Range of potential energy parameters for the symmetric and asymmetric single and double barriers used to compute reaction rate constants for the dataset. For a symmetric barrier, $\alpha = 1$.

Feature	**Values**
Reactant mass – m	1060 – 1060000 [au]
Barrier heights – V_1, V_2	5.0 – 80.7 [kcal / mol]
Barrier widths – w_1, w_2	0.1 – 6 [au]
Distance – d	0.0 – 5 [au]
Asymmetry – α	0.0 – 1.0
Temperature – T	80 – 2500 [K]

For single rectangular and Eckart potentials exact analytical expressions were used for the transmission coefficient.[166,190,192] More details on the numerical evaluation of the transmission

coefficient can be found in Appendix A.1.1. Depictions of the resulting rate constant dataset can be found in Appendix section A.1.2.

A predictor of kinetic rate constants products

A DNN was trained on the dataset of hypothetical 1D energy barriers. The network's optimal architecture and modeling hyperparameters were identified by carrying out grid searches with five-fold cross validation using Scikit-learn's GridSearchCV[193] and Keras with TensorFlow backend.[194] The first search was conducted over one to three hidden layers of $n \in [1, 6, 12, 24, 32, 64, 128]$ neurons per layer with batch size 32, 64 or 128 and epochs in the set $[50, 100, 200, 300]$. For this grid search, the Adam optimizer[195] was used with learning rate set to 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and the sigmoid and tanh activation functions for the hidden and output layer (See Appendix A.1.3-a for details on the first grid search). Subsequently a second and third grid search were carried out on the best model of the first search for each feature set, to identify the optimal activation functions and learning rate (Appendix A.1.3-b and c). Finally, a fourth grid search was carried out on the optimal model obtained from the third grid search to choose the final batch size (Appendix A.1.3-d). In all searches, the ranking metric was the mean squared error (MSE). The optimal hyperparameters values are discussed in the following paragraph and shown in detail in Appendix A.1.3-e.

From the grid searches, as shown in Table 1-2, it was found that the optimal network architecture consisted of three hidden layers with 128 neurons in the first layer, 24 in the second and 12 in the third layer.

Table 1-2: Optimal DNN hyperparameters found from four grid-searches with five-fold cross validation.

Hyperparameter	**Value**
neuron configuration	(128, 24, 12)
epochs	300
batch size	256
hidden layer activation function	sigmoid
output activation function	linear
learning rate	0.001

The grid search on the activation function indicated the sigmoid function as best for the hidden layers. For the output layer, the top three functions were the softsign, linear and hyperbolic tangent functions. These three combinations had comparable metrics of loss and the linear function was chosen as the output function in order to provide all real numbers as an output range and to lessen prediction cost slightly. Finally, the optimal learning rate was found to be 0.001. The best batch size was determined in the fourth grid search by comparing the moving average validation loss of the optimal model and found to be 256. Note that a grid search for regularization was also carried out but it did not lead to an improvement on the model as overfitting was not observed.

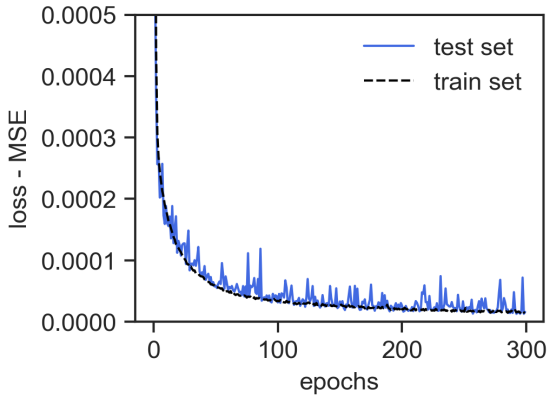

Figure 1-3: Final optimized DNN model training loss (MSE) (black dashed line) and testing loss (blue solid line) for the scaled logarithm of the reaction rate product $log\ (k_Q)$ as a function of epochs.

In Figure 1-3 testing dataset loss (blue solid line) and the training dataset loss (black dashed line) is shown as a function of epochs. The testing loss closely followed the training loss and decreased as training progressed. Overfitting was not observed from the trend of the average testing loss. The fluctuations in the test loss were associated with the batch size in mini batch training. The final value of the training loss expressed in terms of the scaled $log\ (k_Q)$ was equal to $1.55 \cdot 10^{-5}$.

In Figure 1-4 the unscaled predicted DNN values of the $log\ (k_Q)$ with respect to the exact test set values (blue circles) is shown. The red dashed line indicates the identity, i.e. the case where a model would predict perfectly. The linear fit (black line) to the predicted data is very close to the identity with a mean absolute error (MAE) of 0.27 log (1/ps) and standard deviation of 0.86 log(1/ps). This corresponds to a relative error of 1.1% respect to the mean $log\ (k_Q)$ value. The null hypothesis of always predicting the mean of $log\ (k_Q)$ produces a MAE of 24.5 log(1/ps) with standard deviation 34.7 log(1/ps). The small value of the MAE compared to the null indicates that the model was able to predict accurately within the range of values it was trained on.

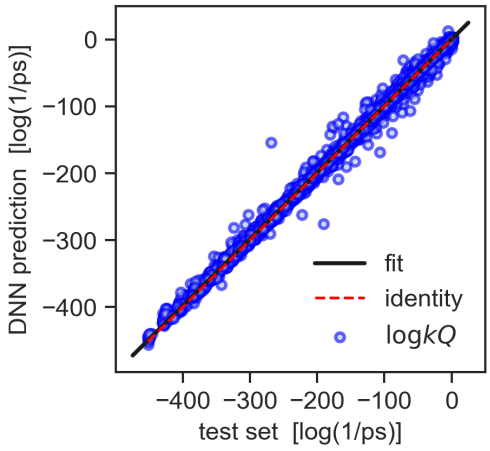

Figure 1-4: Predictions of the optimal DNN model on the test set. Blue circles represent individual examples. A linear fit to the data (black line) is very similar to the identity function (red line). The percent MAE rescaled with respect to the mean is 1.1%. The standard deviations from the mean is 0.86 log(1/ps).

The trained DNN's ability to predict quantum effects was investigated by comparing predicted $log\ (k_Q)$ values to transition state theory rate constant results for the entire test set. In Figure 1-5 panel a) the average difference between the DNN predicted values and the corresponding transition state rate constant, binned by temperature interval, is shown in grey bars with diagonal lines. Specifically, the j-th bin's height h_j was computed as:

$$h_j = 1/N_{test}^{(j)} \cdot \sum_{i=1}^{N_{test}^{(j)}} \left(log\left(k_{Q_i}\right) - log\left(k_{Q_i}^{TST}\right)\right) / |log\left(k_{Q_i}\right)| \qquad (Eq.\ 1.3)$$

The normalization by the absolute value of the quantum rate constant product $log\ (k_Q)$ was carried out so as to even out changes in rate value due to reactant mass and barrier shape and emphasize the difference with respect to the classical rate constant. The same average difference is shown in blue for the test dataset. The DNN values closely followed the test set and for temperatures below 300K, the percent MAE rescaled with respect to the exact mean difference was 38%.

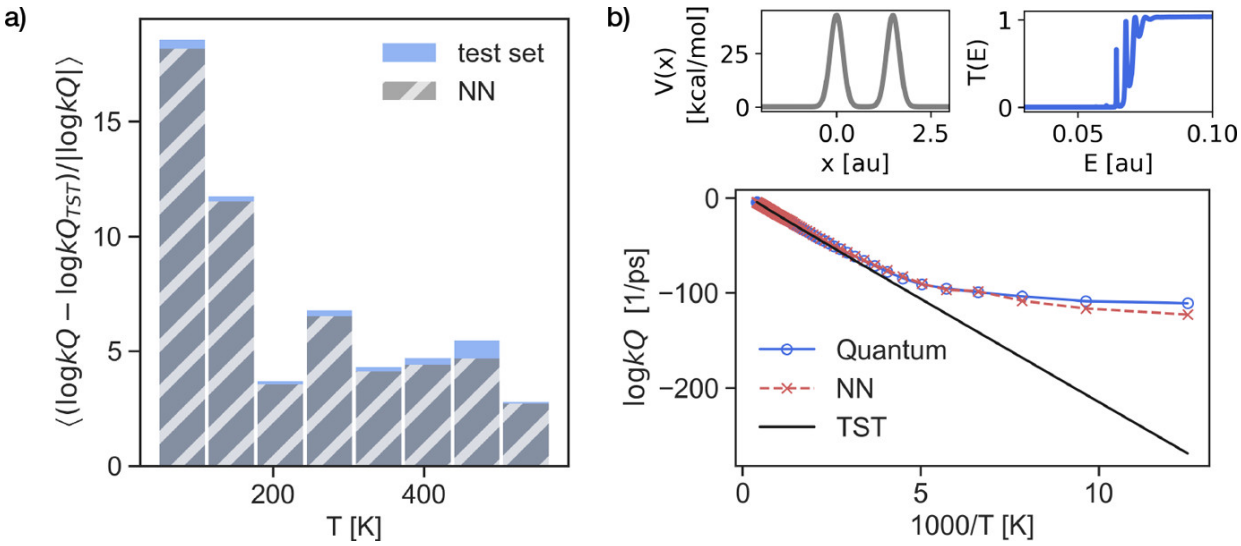

Figure 1-5: Panel a) average prediction of quantum effects (Eq 1-2) as a function of temperature for the test set (light blue) and for the neural network predictions (grey with diagonal lines). As temperature increases the NN correctly predicts a decrease in quantum effects. Panel b) Top plots show, on the left hand side, a double barrier potential taken from the test set with $V_1 = V_2 = 42.89$ kcal/mol, $w_1 = w_2 = 0.5$ au and d = 0.5 au and, on the right hand side, the transmission coefficient for reactants of mass 11.63 g/mol. The lower plot reports the corresponding predicted $log\ (k_Q)$ as a function of the inverse temperature (red dashed line), the exact values (blue line) and a transition state theory estimate based on the first barrier height (black line).

In panel b) of Figure 1-5 an example of the predicted $log\ (k_Q)$ for a symmetric double gaussian barrier taken from the test set is shown. The barrier heights were $V_1 = V_2 = 42.89$ kcal/mol, the widths $w_1 = w_2 = 0.5$ au and the distance between barriers was d = 0.5 au. The reactants mass was 11.63 g/mol which is close to the mass of a carbon atom. As can be seen from the transmission coefficient, panel b) top right-hand side plot, the peaks indicate the presence of quasi bound states which lead to resonant tunneling and to a larger rate constant.[191,196] The presence of quasi bound states increases the duration of the tunneling process and contemporarily the length of time-dependent calculations required to compute the rate constant *ab initio*. In this context, it is promising to see that the predicted DNN $log\ (k_Q)$ values (red) closely followed the exact quantum values (blue) and only start underestimating at temperatures below ~160 K even still maintaining much more accuracy than classical theory.

To evaluate the ability of the DNN model to predict kinetics beyond the test dataset, two additional reactions were considered. The first is the extensively studied H + H_2 reaction.[197] The one-dimensional Eckart barrier approximation to the potential energy surface[198] was used with $V_1 = 9.8$ kcal/ mol, $w_1 = 2.31$ au and $m = 1060$ au to solve for the exact kinetics. The potential is shown in Figure 1-6 panel a) upper plot. In the lower plot, the exact quantum results (blue with circles), the DNN predictions (red with triangles) and transition state theory (black line). The DNN predictions followed the trend of the exact results and recovered most of the quantum effects.

The second process studied was hydrogen diffusion on Ni(100). For this system, a one-dimensional pathway was extracted from the EDIM potential energy surface[199] with EAM-5 parametrization developed by Truong and Truhlar.[200] The potential energy path was determined by first minimizing the potential energy along the z coordinate and then by taking a path symmetrically crossing three hollow sites. The one-dimensional path was then fitted to a double gaussian potential to extract features for the DNN. A plot of the two-dimensional surface and of the one-dimensional double gaussian path is shown in Figure 1-6, panel b) upper section. With this potential we computed the exact logarithm of the reaction rate constant product (Figure 1-6, panel b) lower plot – blue), the transition state theory estimate for the rate constant (black) and the DNN predicted rate constant (red).

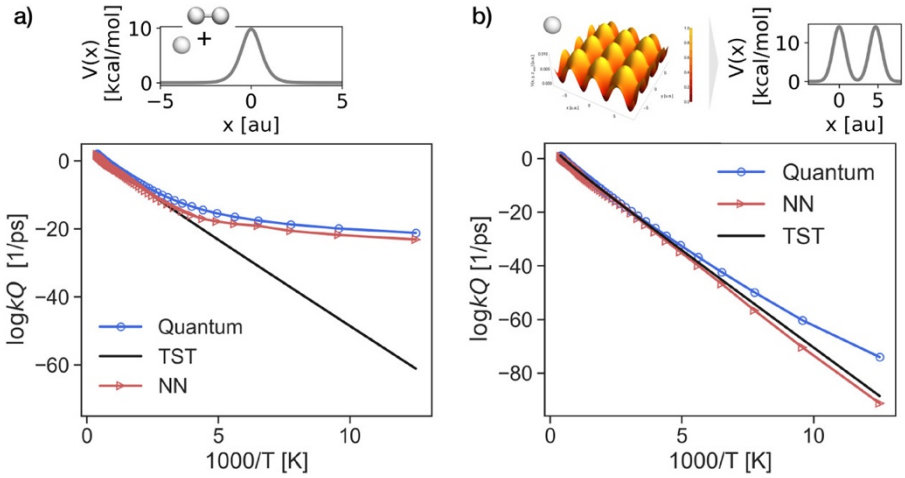

Figure 1-6: Panel a) Upper plot: Eckart barrier for the H + H2 collinear reaction. Lower plot: comparison of the exact $log\ (k_Q)$ for the H+H2 reaction (blue with circles) with the DNN prediction (red with triangles) and transition state theory (black line). Panel b) Upper plots: two-dimensional potential energy surface for Ni(100) and extracted double gaussian fitted one-dimensional barriers. Lower plot: exact quantum $log\ (k_Q)$ (blue with circles) for the one-dimensional diffusion of hydrogen on a Ni(100) surface compared to the DNN prediction (red with triangles) and transition state theory (black line).

The DNN predictions underestimate the rate constant at low temperatures likely due to extrapolation: the example barrier has the minimum mass ever seen by the model, the maximum barrier width, and a very low barrier height compared to the bulk of training examples. Still, the model recovers the classical rate. This shows that with a simple input feature representation, one can obtain qualitative information on reactivity for systems beyond the test set. To predict quantum rates more accurately and obtain quantitative accuracy for a broader set of reactions, providing additional information about the system is required, such as reactant and product geometries, and more resolved barrier features e.g. with end-to-end convolution of the barrier as opposed to engineered features.

Predictions of stable and transition state partition functions

Transition state theory to predict kinetic rates of reaction networks

While the quantum reaction rate constant is necessary to achieve an accurate *ab initio* rate in some circumstances, often transition state theory is enough. TST is used widely in practice such as in catalytic network modeling and other systems involving many reactions.[201–203] While ignoring tunneling effects, it is *usually* sufficient at high temperatures (ambient or above). The TST rate constant is defined as $k^{TST}(T) = \frac{1}{\beta h}\frac{Q_{TS}(T)}{Q_R(T)}e^{-\beta E_a}$, where E_a is the activation energy, $\beta = 1/k_B T$ while Q_{TS} and Q_R are the transition state and reactant partition functions at a given temperature T. Unfortunately, the activation energy and transition state partition functions both require the identification of a transition state from a minimum energy path search. This search is one of the most expensive portions of determining a reaction rate constant within the 1D path assumption, and leaves TST calculations as a bottleneck in computational reaction network study. On the other hand, once input features have been computed, a trained ML estimator may predict chemical properties within seconds.[20,60,70,204–206] Machine learning has successfully been applied to evaluate electronic energies,[207–209] accelerate the search for minimum energy paths,[210–212] and predict kinetic properties[7] such as quantum reaction rate constants for one dimensional minimum energy paths, discussed above.[22,213] Leveraging this success, fast prediction of the activation energy and the transition state partition function would allow for TST rate constants to be approximated rapidly, opening the floodgates for study of massive reaction networks involving dozens or more individual steps.

To meet this need, a dataset of over 1.5 mil partition functions for >30k gas phase organic chemistry species extracted for an existing dataset[214] of reactant, product, and transition state

structures for unimolecular reactions was generated with the goal of training machine learning models.[26] With this data, two DNN partition function estimators were optimized. The first DNN, "Qest", predicts the natural logarithm of the molecular partition function from featurized molecular geometry and inverse temperature, $1/T$. Calculating the partition function of thousands of structures at multiple temperatures can take minutes to hours, which the proposed ML model Qest alleviates. The second DNN, "QesTS", predicts the natural logarithm of the partition function of an unknown transition state, $log\ Q_{TS}(T)$, from the difference between product and reactant featurized geometries, the reactant and product partition functions on the log scale, and $1/T$. QesTS requires no knowledge of the transition state, so avoids the much more significant cost of MEP search. These models were then used to predict partition functions to be used in the computation of transition state theory reaction rate constants. The reaction rate constant predictions were also compared to *ab initio* TST reaction rate constants. The various steps taken to create the data, train, and evaluate the models are described in the following subsections. An overview of this work is shown in Figure 1-7.

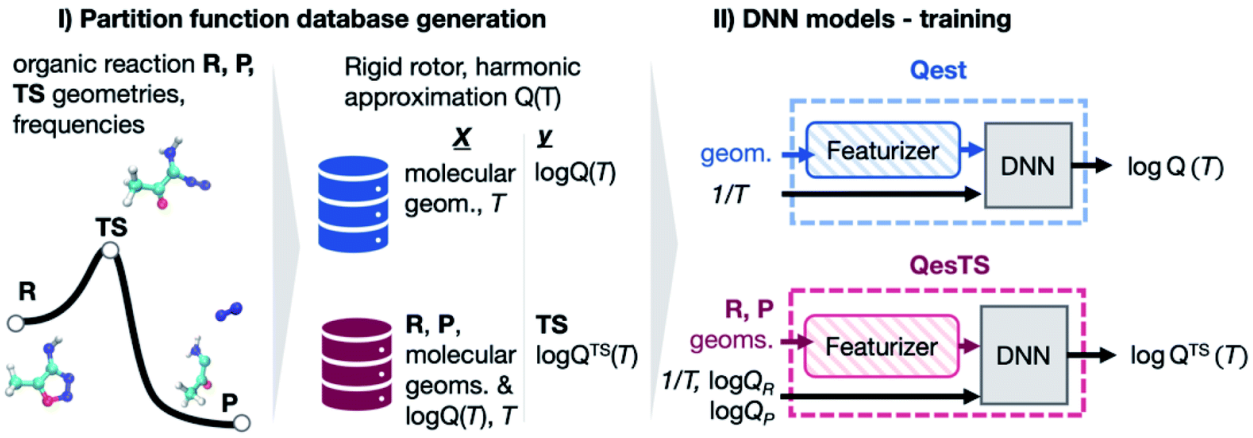

Figure 1-7: Overview of work to predict molecular and transition state partition functions. 1) Partition functions were computed over a range of temperatures for an existing dataset of reactants, products, and transition states. This produced two datasets: one of partition functions of all structures and one of transition state partition functions. 2) Two ML models were optimized and trained. Qest makes predictions of partition function at some temperature based on 3D structure input. QesTS makes predictions of an unknown transition state partition function at some temperature using reactant and product 3D structures.

A dataset of partition functions

Partition functions were computed for 35,883 reactant, product, and transition state geometries taken from a dataset[214] which contained 11,961 gas phase, organic chemistry reactions involving molecules of at most seven C, O and or N atoms. The partition functions were computed under the assumption that electronic, translational, rotational, and vibrational degrees of freedom were separable, and by using the rigid-rotor and harmonic-oscillator approximations:

$$Q(T) \approx Q_{el}(T)\, Q_{trans}(T)\, Q_{rot}(T)\, Q_{vib}(T) \qquad (Eq.\ 1\text{-}4)$$

See Table A.2.1-a in the Appendix for the equations of each partition function term in Eq. 1.4. Although some products consisted of two or more distinct molecules, these had been considered as single structures when energies and hessians were computed in the original dataset.[214] Therefore these single geometries were used to compute the partition functions. A more accurate representation would require the separation of the product geometry into the individual molecular geometries. When computing the vibrational partition functions, zero-point energies were

included. For the rigid rotor rotational partition function, the symmetry number was determined by identifying invariant symmetry operations for each molecule using pymatgen,[215] and by testing which symmetry operations were proper. In particular, the exception of linear molecules for which C_2 rotations are returned as improper reflections by pymatgen was handled.

Partition functions were computed over a set of 50 temperatures randomly sampled for each reaction from a uniform distribution of $1/T$ within the range of $T = (50, 2000]$ K. A histogram of the natural logarithm of the partition functions for all structures at all temperatures is shown in Figure 1-8. The histogram shows a smaller count of low values of the natural logarithm of the partition function. This is due to the dominating contribution of the vibrational partition function to Eq. 1.4. The data has been made open source and is available to download from Ref.[216]

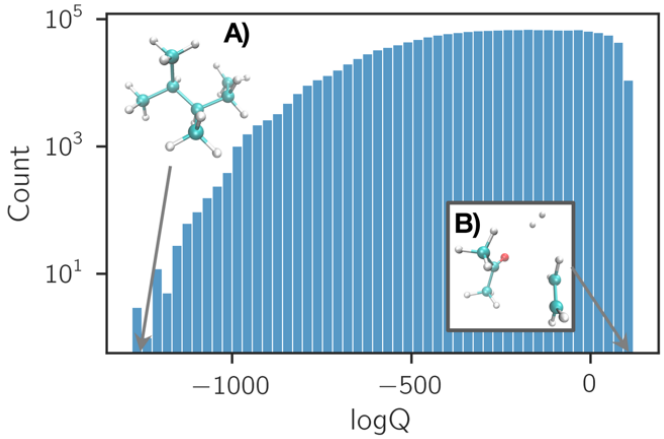

Figure 1-8: Histogram of all partition function values computed in the dataset for 35,883 organic chemistry species. Each entry is either a reactant, product, or transitions state at of one 50 temperatures, sampled inverse-uniformly and independently for each of the 12k reactions. Structures that exhibit the smallest and largest partition functions in the dataset are depicted in subpanels A and B. The shape of the histogram is largely due to the dominating contribution of the vibrational partition function in Eq. 1.4

The entire dataset was split into a hold out test set (10%) and a development set (90%). The hold out set was used to test final ML predictors. The development set was further split into 5 folds to use for cross validation during model optimization. Both the hold-out and the fold splitting were conducted using the scaffold of the reactant molecule.[82] By taking into account molecular structure backbones in scaffold splitting it was ensured that the structural overlap of the different datasets was minimized, producing a better estimate of the models accuracy on unseen data. For Qest, the input consisted of a molecule's featurized structure (reactant, product, or transition state) and the inverse temperature, while the target was the corresponding natural logarithm of the partition function at that temperature. On the other hand, for QesTS the inputs were the difference between product and reactant featurized structures, product and reactant partition functions, and the inverse temperature while the target was the natural logarithm of the transition state partition function.

Optimization and training of ML models

First, the optimal format of the input data was determined. This involved a screening of featurization methods and data standardization. It was found that the EncodedBonds[72] featurizer with min-max scaling and target normalization were optimal for the Qest Model. For the QesTS model the same input data feature representation was employed as Qest, i.e. Encoded Bonds. Screening of data standardization was also carried out, finding that standardization was not beneficial. It is believed that this is because the distribution of values for the difference in product and reactant feature vectors is peaked around zero; also, the distribution of partition function values for transition states was narrower than the overall partition function distribution for all species. For more information see Appendix A.2.2.

With these optimal input features, a search over DNN hyperparameters was conducted to identify the best activation function, regularization, bias and weight initialization, learning rate, and neuron configuration for Qest and QesTS. For more information see Appendix A.2.3.

Partition function predictions

The optimal featurizer, standardization, and hyperparameters (Appendix Table A.2.3-b) were used to train the final DNN models on the development dataset and test its performance on the test set. To limit extrapolation in the final model, some outliers were dropped from the full dataset, see Appendix A.2.5.

In Figure 1-9 parity plots of the final QestTS (panel A) and Qest (panel B) model predictions with respect to the test set are shown. In both cases note a low test set mean absolute error of 4.01 (2.0%) and 4.37 (2.1%) on the logarithm of the chemical species or transition state partition function for Qest and QesTS respectively. Percentages are MAE compared to the test set standard deviation. Also, in both cases the spread of the distribution of predictions around the identity is quite narrow. This indicates that both DNN models accurately predict the partition functions.

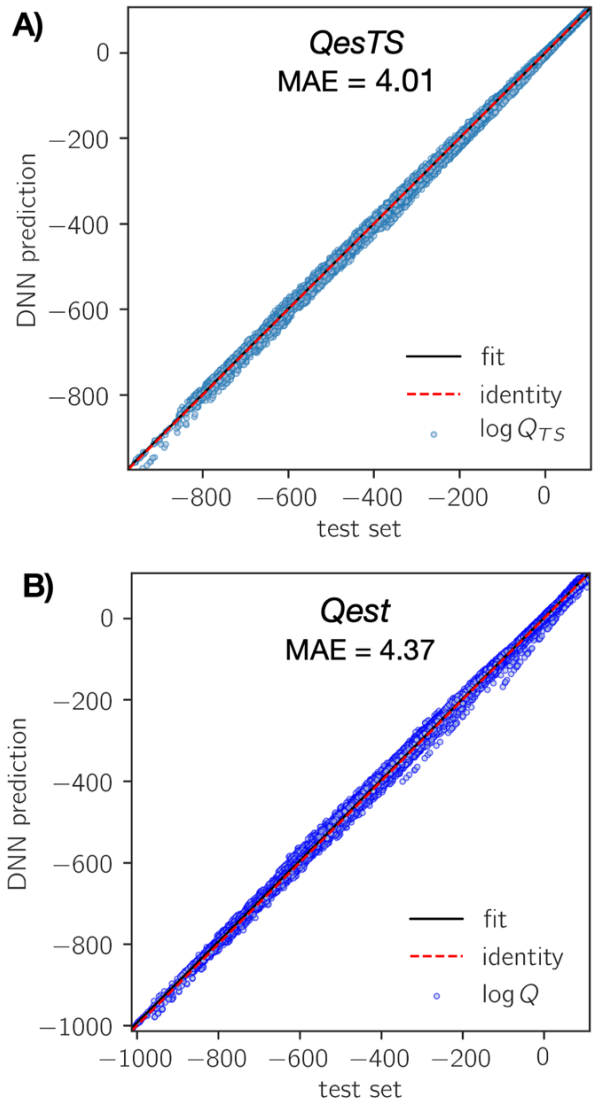

Figure 1-9: Panel A) Parity plot of the final QesTS DNN predicted values of transition state Q (y-axis) with respect to exact unseen test set values (x-axis). A perfect model is represented by the identity line (red dashes). Predicted examples (blue dots) closely match the true value and the overall MAE on the is 4.01 (2.0%). Panel B) Same as panel A but for the Qest prediction of Q. Here the overall MAE on the test set is 4.37 (2.1%). Percent error is compared to the test set standard deviation. Note a strong agreement of the DNN predictions with the test set values.

To verify that the Qest model was learning from the molecular structures in the dataset and not simply from the temperature, a comparison "null" linear model $log\ Q(T) = m \cdot 1/T\ + b$ was

considered and fit to the development set. The null model performed poorly on the test set with an MAE of 34.5%. Qest model error was much lower: 2.1%, confirming that the Qest model learned from the molecular structure and not just temperature averaging. The prediction MAEs on the test set for Qest, QesTS, and the null model are listed in Table 1-3.

Table 1-3: Mean absolute error of the null, Qest and QesTS models in predicting the test set molecular (reactant, product and transition state) or transition state partition functions on the log scale. Percent error is given compared to the test set standard deviation.

	Null MAE	***Qest* MAE**	***QesTS* MAE**
$log\ Q$	72.5 (34.5%)	4.37 (2.1%)	N/A
$log\ Q_{TS}$	72.1 (35.1%)	4.25 (2.1%)	4.01 (2.0%)

Given that both models predicted with a low error on the test set, these were combined to predict transition state partition functions with machine learned reactant and product partition functions. This "Double" model is discussed in the next sub-section. The trained models are available on GitHub.[217]

Reaction rate constants computed with machine learned partition functions

With the Qest and QesTS partition function estimators, and the existing activation energies,[214] transition state theory reaction rate constants were computed for 1,086 test set reactions. To understand how machine learned partition functions influence the accuracy of the reaction rate constant, three approaches were considered. In the first both reactant and transition state partition functions were predicted from their geometries by using Qest. In the second approach, the reactant and product geometries and computed *ab initio* partition functions were used to predict the transition state partition function with QesTS. This approach avoids the time demanding search for a transition state geometry. In the last approach, Qest was used to predict reactant and product partition functions which were used, together with their geometries, as input

to QesTS to predict a transition state partition function. This method, hereafter referred to as the Double predictor, only requires knowledge of the reactant and product geometries and does not need computed partition functions or transition state geometries.

The MAE of the three methods for predicting test set transition state partition functions and TST reaction rate constants is listed in Table 1-4. For the prediction of transition state partition functions (Table 1-4, column 1) all models have a similar MAE. The Double approach is the least accurate with an MAE of 2.7% while the QesTS approach has the lowest MAE of 2%.

Table 1-4: Performance of transition state partition function prediction, and TST reaction rate constant calculation using predicted partition functions for our three ML based methods (Qest, QesTS and Double). Here E_a is the reaction activation energy. Percent error is given compared to the test set standard deviation.

	$log\ \boldsymbol{Q_{TS}(T)}$ **MAE**	$log\ \boldsymbol{k^{TST}(T)}$ **MAE**	**Required inputs for $k(T)$**
Qest	4.25 (2.1%)	4.50 (1.7%)	Reactant, transition state structures, temperature, E_a
QesTS	4.01 (2.0%)	4.01 (1.5%)	Reactant, product structures and partition functions, temperature, E_a
Double	5.58 (2.7%)	4.50 (1.7%)	Reactant, product structures, temperature, E_a

The fact that QesTS is the best predictor is not surprising given that it was trained specifically to predict transition state partition functions while Qest was not. Further, QesTS has knowledge of reactant and product partition functions from its input. In column 2 of Table 1-4 note that when computing reaction rate constants by using the predicted partition functions, the Double and Qest approaches have the same MAE while the QesTS approach remains the most accurate with an MAE of 4.01. The increase in accuracy of the Double approach compared to the other methods is most likely due to a cancellation of errors from Qest predicted reactant and QesTS predicted transition state partition functions. Nonetheless, the same average accuracy of reaction rate

constant calculations is achieved when using only reactant and product structures and no information on the transition state geometry. The only cost remains providing the value of the activation energy. Here it is worth noting that machine learning has successfully been employed to predict activation energies.[218,219]

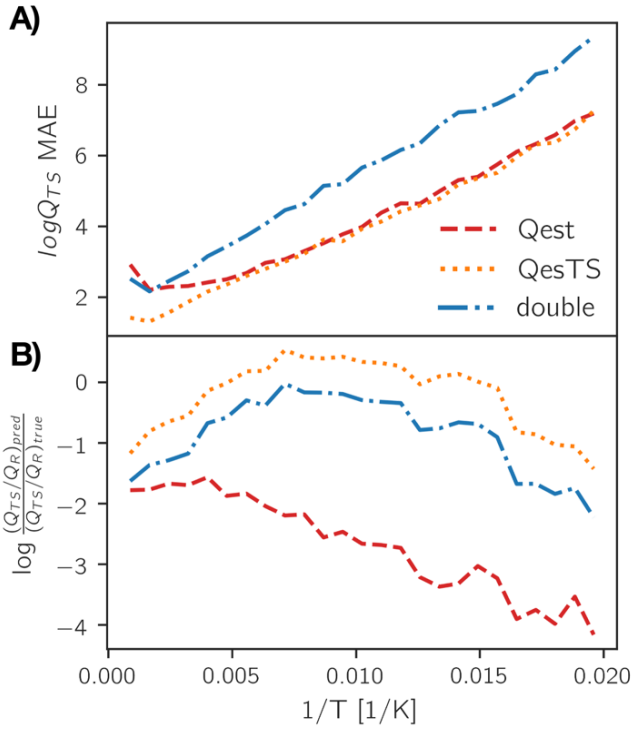

Figure 1-10: Panel A) MAE averaged over temperature bins on the test set for predictions of the logarithm of the transition state partition function using Qest, QesTS, and Double models. While the error of the transition state logged partition function increases as temperature decreases, the ratio of transition state to reactant partition function, which is the prefactor for TST rate constants, has low error. This is shown in the plot of the logged ratio of true to predicted prefactors binned over temperature (Panel B). Here 0.0 indicates a perfect match between predicted and true value.

In Figure 1-10, panel A the error of the predicted transition state partition function with respect to the test set partition functions as a function of temperature is shown for Qest, QesTS,

and Double. In panel B the logarithm of the ratio of predicted transition state theory prefactors, Q_{TS}/Q_R, with respect to computed prefactors is given. Note some error cancelation: the error in the ratio of partition functions (panel B) is smaller than that for the single values (panel A). Errors for all models tend to increase at lower temperatures even though low temperatures were sampled frequently when generating the data. This could come from the fact that the partition function is more sensitive to small changes in temperature at low temperatures, making it more difficult to learn. Regardless, these errors do not significantly impact the predicted value of the reaction rate constant, with average error orders of magnitude lower than the value of the logarithm of the reaction rate constant. This is also seen in Figure 1-11, where the average error compared to the reaction rate constant for the ring breaking of gamma-Butyrolactone is shown.

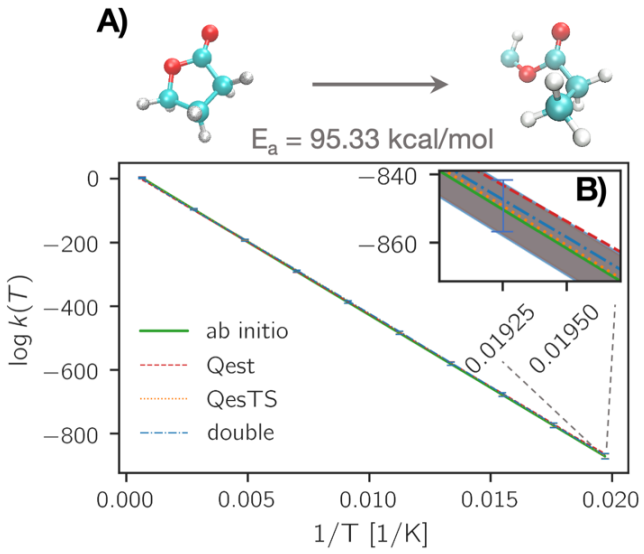

Figure 1-11: Panel A) Natural logarithm of the reaction rate constant for a randomly selected test set reaction: the ring breaking of gamma-Butyrolactone. Here we show the logarithm of the reaction rate constant computed with predicted Qest (red), QesTS (orange), and Double (blue) partition functions as well as the reaction rate constant computed using the ab initio values of the partition functions (green). The error bars correspond to the error on the entire test set averaged over temperature bins. Panel B) A zoom in on the low temperature section of the data in plot A. Here the average error is plotted as dotted margins instead of error bars. One error bar (blue) is depicted. The error on k(T) for all prediction methods increases at low temperatures, however the average error is significantly smaller than the magnitude of the reaction rate constant.

In Figure 1-12 Panel A, a plot of the transition state theory reaction rate constant for another reaction randomly selected from the test set, the ring opening of tetrahydro-2H-pyran-2-imine, is given. Here the partition functions were computed using the original ab initio data (green line) and predicted with the ML models. In Panel B, show the absolute error as the normed difference between true and predicted values is shown. The error of the models is more than an order of magnitude smaller than the value of the logarithm of the reaction rate constant which confirms the accuracy of the trained model.

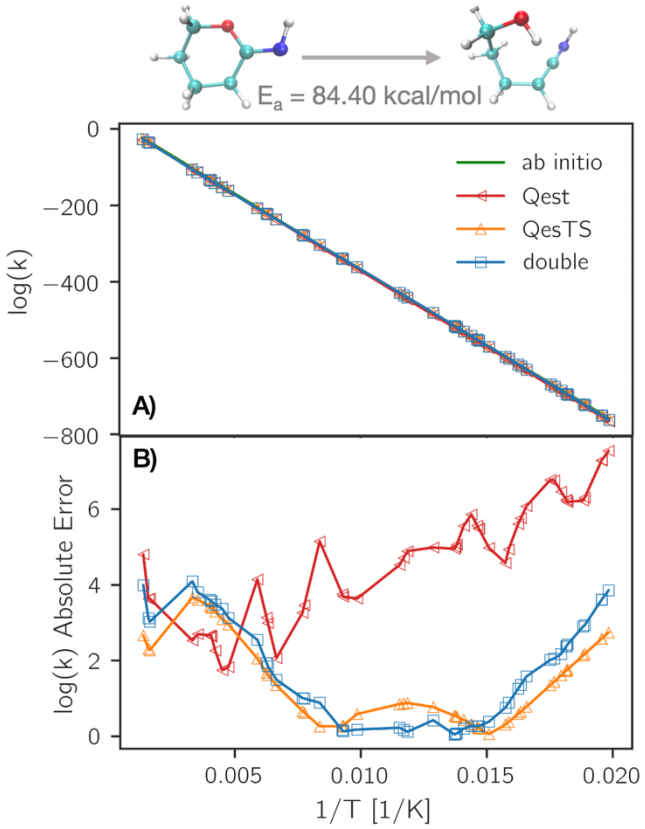

Figure 1-12: Panel A) Natural logarithm of the TST reaction rate constant as a function of inverse temperature, 1/T, using Qest, QesTS, and Double predicted and computed partition functions for the ring opening of tetrahydro-2H-pyran-2-imine. This reaction was randomly selected from the test set. The predicted values are all close to the ab initio values. The absolute error of these models on the logarithm of the rate constant is shown in Panel B. Qest prediction has larger error at low temperatures for this reaction, however all methods predict with error an order of magnitude lower than the logarithm of the rate constant.

Conclusions

In this chapter, I outlined two distinct applications of Dense Neural Networks to help predict thermal reaction rate constants. Both required a novel dataset to be developed. The unique strategies for representing the reaction system and developing a predictive model

were detailed, rigorous hyperparameter optimization searches were conducted. The ability of the models to predict stringently held out test examples as well as blind systems are analyzed. These models help alleviate extreme costs in predicting complex reaction networks involving many reactions, by avoiding resource intensive experiment of computation.

For systems for which quantum accuracy is desired, like atomic diffusion or low temperature reactions, work here shows that non-classical effects on the rate constant can be recovered without expensive downstream time or energy propagation calculations. This reduces the cost of predicting a rate constant to that of Transition State Theory, with some error compared to rigorous dynamics. Of note, the systems for this DNN application were featurized using simple barrier-agnostic features e.g. the width and heigh of potentials. Thus, while the examples in the dataset used the train the model were theoretical barriers, the strategy could be applied to any 1D reaction barrier for which the features can be posited. This was tested by fitting analytical functions to literature energy barriers, which led to predictions that were quantitative. Given that the rate constant product is deterministically dependent on the reaction barrier, this also suggests that future neural network models could further reduce the error compared to calculation. It is worth applying the observed successes of learned representations for molecular systems to those of reaction barriers: convolutional neural networks or transformers have been used in other settings an could be applied directly to the barrier, potentially improving over engineered barrier features detailed here.[220,221]

For systems where TST rate constants are sufficient, there is still significant cost associated with finding a transition state, and work here shows that the partition function

of such an unknown transition state can be predicted. Engineered 3D features were used to train DNNs to predict the partition functions as a function of temperature for arbitrary 3D small molecular structures involving no more than 9 heavy atoms. When these features for a reactant and a product were combined, prediction of partition functions for an unknown transition state followed. When combined with activation energy ML predictors, this completely removes the need to search for MEPs, allowing for reaction rate constant predictions at TST rigor in sub second timescales. This is transformative when considering large reaction networks where hours to days of time is normally required for each reaction.

While these tools are no complete substitute for accurate experiment or calculation, they push the needle forward on rapid qualitative evaluation. Through careful design of features and future models, the gap in accuracy between first principles and data driven methods will be reduced.

CHAPTER 2: TEACHING ML ON CHEMICAL AND BIOLOGICAL SYSTEMS

Data science is becoming increasingly ubiquitous in nearly every industrial field[15,222] and has found traction in Chemical Engineering[2,14,223] (ChE), yet instruction is sluggish in primary education[224] and only recently beginning to diffuse out of computer science coursework in higher education.[225,226] Resources for students to seek out on their own such as online coursework or articles often describe methodologies in the contexts of traditional computer science and do not provide case studies relevant to Chemical Engineers. Data Science is an umbrella of topics and tools that need to be discussed in the context of their application.[227] The disparity between the applications of data science to ChE topics and their joint instruction calls for more opportunities for students to learn and apply data science to ChE problems. In the next section, an optional event for undergraduate students "C-HACK" to learn and apply data science and machine learning to real industrial data is discussed. This event was designed with a team of representatives from faculty, students and industry, and occurred in both 2021 and 2022 across multiple college campuses. An increase in student understanding of python topics was observed, as measured by self-assessment.

A complementary Hack-a-thon

Hackathon objectives

From a pedagogical standpoint, active learning has received a lot of recent discussion.[228–231] The stated primary objective of active learning is to engage the student in the learning process. This goal seems intuitive and straightforward, but it is difficult to prove with evaluated outcomes that active learning methods are always the optimal choice. Improvements in learning outcomes are associated with more subtle design choices, such as cooperative settings, instruction during problem solving, and centering learning around an inquiry or problem.[232] Subsets of active learning methodologies such as Inquiry Based Learning (IBL)[233] and Problem Based Learning (PBL)[234] focus on providing students with an exploratory task, through which they discover key concepts and effective strategies for themselves. The core necessities for these types of methods to be beneficial are to provide students with sufficient motivation, and give them an opportunity to collaborate.[235] We have already seen project based coursework being used to teach data science,[236] the challenge remains integrating it in the contexts already being explored by ChE students. One method that could be used is coopetition,[237–239] where students work in teams on a problem with access to learning resources.

While designing and incorporating active learning of data science into chemical engineering curricula would improve students' skillsets, reforming degree tracks requires a lot of planning and can be slow. Optional learning opportunities,[240,241] such as the chemical engineering hackathon, discussed here, provide students with intermediate agency in their learning of this subject matter. C-HACK is an optional 2-week active

learning event designed to teach students data science in the context of chemical engineering.[27] The event was conducted in 2021 and 2022. The primary objectives of the event were:

Objective 1: Provide an opportunity for undergraduate ChE students to gain data science skills, *regardless of starting comfortability with the material*, specifically targeting groups typically under-supported in computing.

Objective 2: Give students an opportunity to work in teams to solve *a current industrial problem* in a limited amount of time using the skills from Objective 1.

Objective 1 stemmed from the goal to ensure that 1) there was no bias or discrimination in terms of registration and participation in the event such that those who were not yet comfortable with writing code and working with data still signed up, and 2) that all students were able to tackle the problems in Objective 2. To fulfill subgoal 1, no experience was requested and the material was prepared assuming participants had no data science background. Contemporarily, participants needed only an internet connection. To address the second subgoal, the first week of the event consisted of a series of interactive tutorials ranging from the basics of Python coding to some machine learning topics that could be used for Objective 2. Python was chosen as coding language to teach data science as it is extensively used for data science applications and has an active open source community.[242,243]

Objective 2 aimed to provide students an experience in team dynamics so that they could develop soft skills sought after by industry employers. To fulfill this goal the second part of C-HACK consisted of a timed hackathon using real data provided by Dow, Inc. (“Dow”), where students had the opportunity to work with industry representatives. The hackathon portion of the event is a reward style coopetition where participants compete in

teams to tackle a problem. To recognize student achievement, participants received official certificates for their work and top teams won monetary prizes and awards.

An overview of the event including the organizational period, the tutorials, and the hackathon are shown in Figure 2-1. In the following section the observed outcomes of the event are described, and in the Appendix B the detailed format of the event is given.

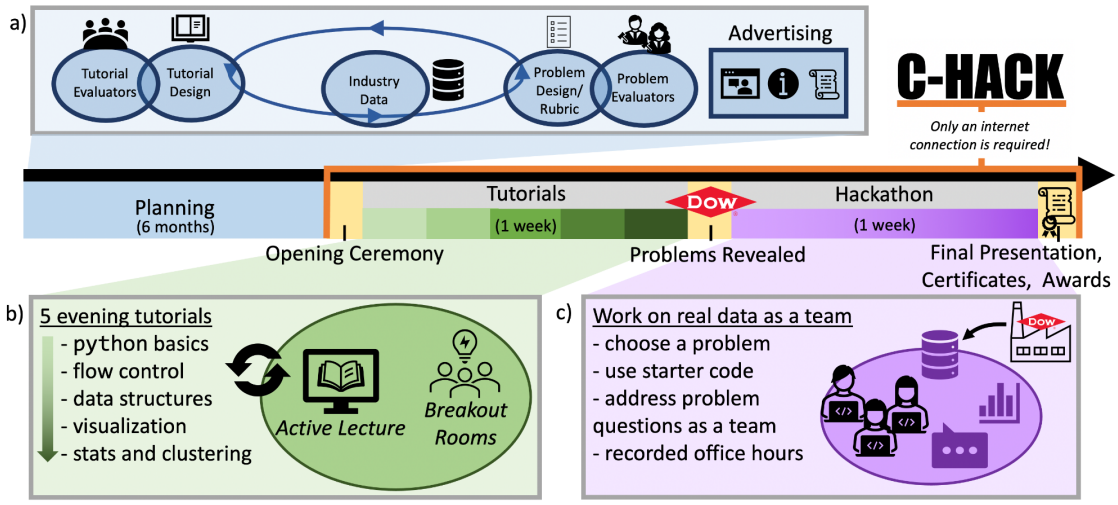

Figure 2-1: Overview of the two-week event. a) Planning period. Committees were created to evaluate tutorial content, assist with tutorial delivery, and evaluate participants' work on their projects. The format and content of the event was designed to adhere to Objectives 1 and 2, including student incentives, software, scheduling, tutorial material, hackathon problems based on industrial data. The event was advertised through fliers, virtual info sessions, and before core undergraduate class lectures. b) Event starts with tutorials. After an opening ceremony where the agenda, resources, and industry representatives were introduced, one week of evening interactive tutorials on Python and data science topics was conducted. c) Hackathon for ChE Undergrads. Two days after the last tutorial, representatives from Dow introduced their dataset, the story behind it, why it is important, and the coopetition problems to choose from. Participants had four days to use data science to work on the problems. Two office hours were open for students to ask organizers and industry representatives about the data and problems, which were recorded and made available to all participants. After their work was turned in, participants gave 10-minute presentations. Rubric assessment by judges was conducted on the students' work and an award ceremony followed. Dow Logo is a trademark of "Dow, Inc." ("Dow") or an affiliate of Dow.

Event outcomes

The results shown below were observed for the 2022 event year. In total, 107 students from two campuses (22 Oregon State University, 85 University of Washington) participated at some level, and 67 in 19 teams completed their team based activities in the

coopetition. Before and after the event, students were asked a series of 8 questions via survey to understand how well we accomplished the event goals, listed in Table 2-1. A pdf of the survey can be found in the event repository.[244] Qualitative questions (all but the first question) were given on a scale from 1 to 5. For the first question ("Gender: How do you identify?") students were given options of "Female", "Male", "Transgender", "Gender variant/non-conforming", "Prefer not to disclose," and "Other." These options were not mutually exclusive. Those that identified as male are referred to as men and those that identified as female as women. Abbreviations for the survey questions are listed in column two and used to refer to the questions hereafter. The objective that the questions aimed to measure is listed in column three.

Table 2-1: Questions asked to participants and the goal they measure.

Question asked	Abbreviation	Primary Objective
"Gender: How do you identify?"		Objective 1
"I understand what variable data types are."	"Data Types"	Objective 1
"I understand the concepts of flow control in Python."	"Flow Control"	Objective 1
"I am comfortable with using Python to visualize and analyze scientific data for problem solving."	"Python Visualization"	Objective 1
"I am comfortable working in teams to solve problems by dividing the work in subtasks and communicating my progress."	"Team Projects"	Objective 2
"I am comfortable working on open-ended problems (where there is no single correct answer) to find a solution by brainstorming with my team."	"Open Ended"	Objective 2
"I am comfortable writing and orally presenting work I have done on open-ended problems."	"Communication"	Objective 2
"'I feel I have opportunities to apply what I am learning in my degree on problems similar to what I will encounter in the future.'"	"Degree Applicability	Objective 2

A total of 79 participants responded to the survey before the event, 30 identified as men, 47 identified as women, and two identified as "Other" or "Prefer not to disclose." Given these responses, no meaningful comment could be made on the effect of the event on nonbinary individuals, but if we compare the ratio of women to men to the overall population in ChE at the University of Washington (46%) we can see that feedback on the event was preferentially provided by women. It is unclear if the distribution of gender identities for those that filled out the surveys is representative of the 107 total participants.

For the first survey, the response to the questions by women and men are shown in Figure 2-2. There was room for improvement by both men and women for all topics, however the students were least comfortable with the three python related topics (Data Types, Flow Control, and Python Visualization).

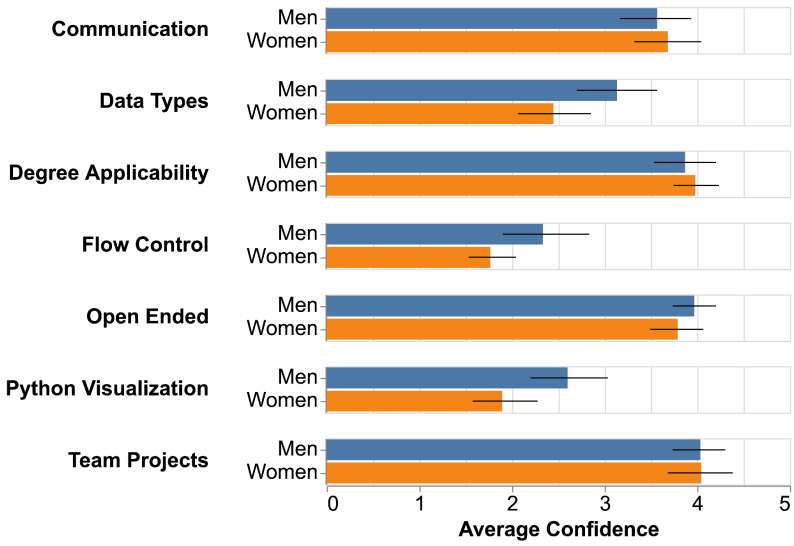

Figure 2-2: Average response to the 7 qualitative questions asked of participants before the event, by self-identified man or woman. Black lines are one standard deviation. Responses for the three python related questions show the most room for improvement.

A total of 39 participants responded to the survey after the event concluded. The responses to the questions among the overall population before and after the event is shown in Figure 2-3. Note that while Objective 2 responses did not substantially change, responses to Objective 1 questions related to Python showed significant improvement due to the event. The magnitude of these improvements in terms of effect size is shown in Table 2-2. The p-value indicated by the two-sided unequal variance t-test is shown in the third column, where statistically significant changes in response are emphasized. While the average change in response was positive for each question, only Objective 1 questions have improvements greater than 1.0 effect sizes.

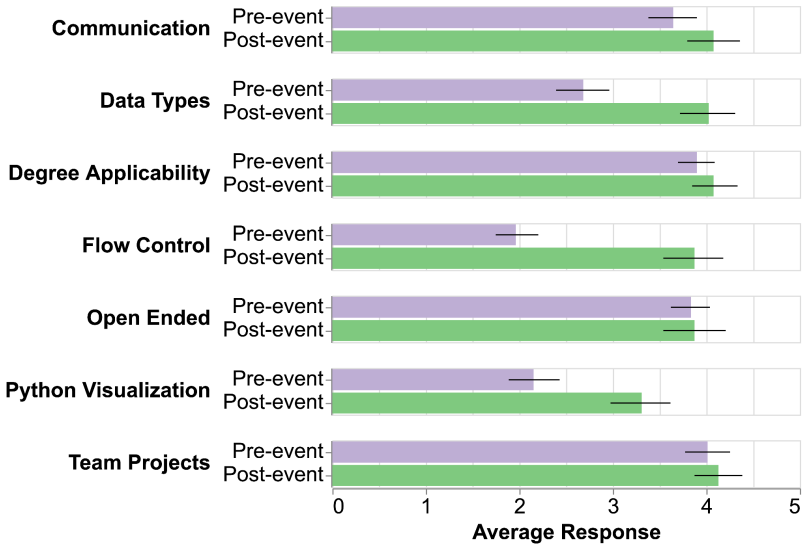

Figure 2-3: Average response of all participants to the qualitative questions before (79 response) and after (39 responses) the event. Black lines are one standard deviation. Among the Objective 2 questions, notice no improvement within confidence. For Objective 1 questions, note significant improvement before and after the event.

Table 2-2: Improvement in confidence for 6 qualitative questions after the event

Question	Improvement[a]	p-value
"Data Types"	1.34	8.52e-9
"Flow Control"	1.91	1.20e-14
"Python Visualization"	1.16	1.02e-6
"Team Projects"	0.11	5.29e-1
"Open Ended"	0.04	8.57e-1
"Communication"	0.43	3.08e-2
"Degree Applicability"	0.21	2.81e-1

[a]Number of effect sizes improved, where effect size is computed as the average of standard deviations of the pre and post populations

Finally, to assess the change in comfortability of individuals, especially among women who are traditionally undersupported in computer science and stem,[245,246] consider the participants (n=34) who responded to both surveys. Of these, 19 self-identified as

women and 15 self-identified as men. Figure 2-4 shows the distribution of the difference in students' responses between the two surveys. The distribution of improvement, represented by positive change, is skewed most positive for Objective 1 questions. On average, the improvement for women is greater than the improvement for men, indicated by vertical colored lines plotted over the histogram. For Objective 2 questions, note little to no improvement or even a reported decrease in comfortability by men for some questions. While they are not statistically significant, it is an interesting observation. This may be due to some teams having to rearrange or drop out due to participants not having time to contribute. Note that teams that completed the projects scored well on their rubric evaluated oral presentations, at 70.0% on average. The subsections of this score (oral presentation, teamwork, and project results) directly correspond to three Objective 2 questions, see the rubric in the event repository for details. [244]

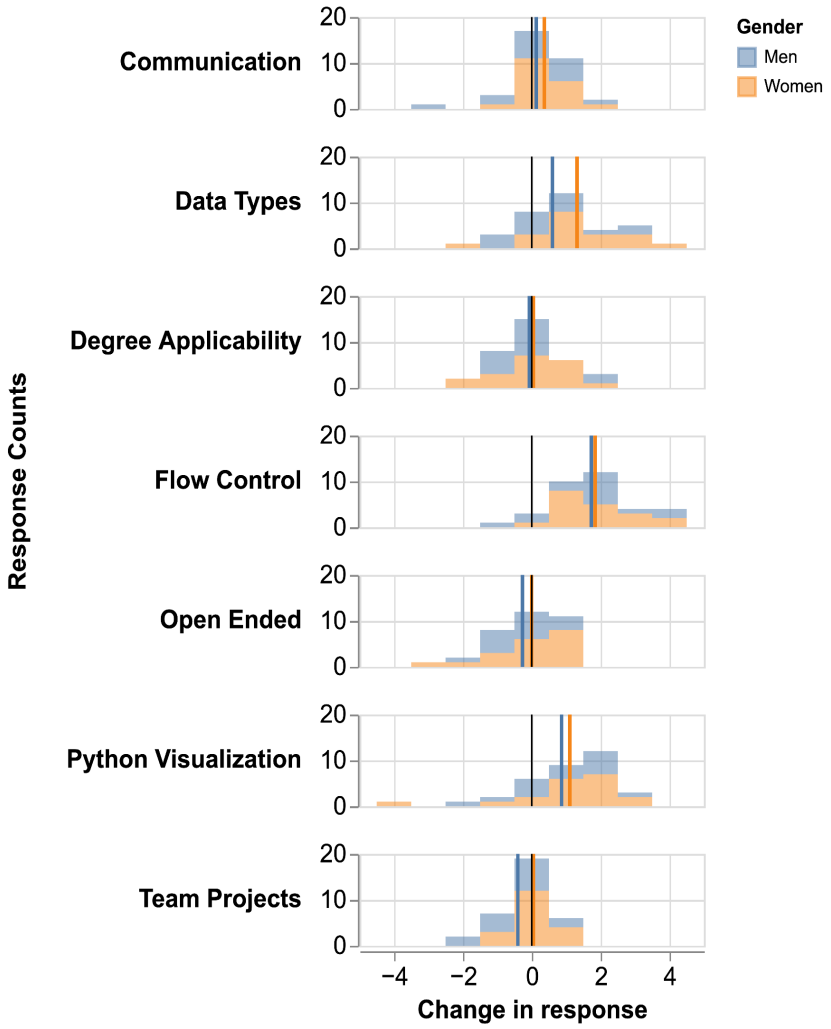

Figure 2-4: Histogram of change in survey response, for each of the 7 qualitative questions by 34 participants that responded to both surveys. Histograms are stacked such that responses by women (blue) and men (orange) together compose the overall distribution. A positive change indicates that the participant reported improved confidence to the question after the event compared to before, while negative indicates that confidence was decreased by the event. A black vertical line is shown at 0, representing no improvement. The average change for men and women is shown as vertical colored lines. For Objective 1 questions, where the average improvement is significantly more than none, women reported a larger improvement than men in all cases. This is most significant for the data type question.

Conclusions

An optional learning opportunity, "C-HACK", aiming to give undergrad ChE students an opportunity to learn and apply python, data science, and team soft skills, was discussed. While an improvement in soft skills was not observed, a significant

improvement in python and data science related knowledge was observed as measured by participant self-evaluation surveys and is even more pronounced for participants identifying as women. This is especially valuable given the minimal time commitment by students of two weeks during the relaxing part of the academic term. Preparing the event was achievable by a small team in a few months. Having an industry partner was invaluable in order to keep the event centered on the skills and real-world problems relevant to ChE industry. Given this, C-HACK can be considered a high efficiency method to provide students with a foray into data science as it relates to ChE in institutions that do not have data science and ML dedicated courses or tracks.

CHAPTER 3: THERMAL STABILITY OF PROTEINS

Proteins have made their way into application spaces such as unique materials,[247–249] catalysis,[250,251] therapeutics,[252] among many others. Suffice it to say that utilizing the tunability of protein design is of extremely high importance for creating novel methods in a post-petroleum industry.[30,253] Recent years have seen proliferation of successful application of protein design, whether by modifying natural proteins, or creating new ones from the ground up.[254] One critical task is improving the stability of the protein fold, particularly for high temperatures. A protein retaining functionality with high thermal stability can be leveraged in processes operating at high temperatures, increasing both reaction and transport kinetics, and retaining the desired structure. Perhaps the most blatant example of a thermostable protein being leveraged by humans is the family of cellulases purified from high-temperature prokaryotes (thermophiles) and fungi, which had a global market value of $1.6b in 2022 and is expected to grow to $3.2b by 2032.[255,256] This family of proteins break the glycosidic linkage bonds connecting glucose in cellulose polymers. This necessary step in many textile applications, pulp and paper production, and biofuel synthesis can only be conducted with enough throughput to be leveraged industrially when conducted at high temperatures, thus the enzymes used in practice are thermostable variants.[257–259] The use of proteins at high temperature is not limited to enzymes, for example within the therapeutics domain. Here, in addition to some enzyme applications, structural proteins bind to targets and act to inhibit, increase, replace, or otherwise modulate biological pathways. Examples include insulin, vaccine components, and are being considered as treatment for Alzheimer's, among others.[260–263] One of the major difficulties with protein based therapeutics is the degradation at moderate temperatures,

requiring energy intensive and expensive cold-chain transportation at ~6ºC.[264,265] Recently, a thermostable closed spike protein trimer of SARS-CoV2 as been designed.[266] This protein instigates viral binding to the cell membrane, and the thermostable variant retains its structure up to 60ºC, allowing it to be easily transported if used to develop a vaccine, as well as for serological studies. Beyond cellulase and therapeutic applications, one can think of a wide spectrum of applications where proteins offer promise but must be stable above room temperature, such as degradation of organophosphorus compounds in wastewater remediation plants,[267] and preparation of soy product for food.[268] It is clear that proteins have the potential to be leveraged in many high temperature applications, but while the protein design space is vast, only some functionalities are naturally occurring in thermophiles, leaving many applications still locked.

There has been a host of studies in order to identify methods, rules, and techniques such that proteins can be designed for use at high temperatures, however these are limited to chemical methods with significant disadvantages, broad statistical rules with marginal effect, or strategies limited to one particular family of proteins.[8] It is of high interest to develop a context-dependent understanding of primary protein structure's effect on thermal stability. The protein stability problem, from a thermodynamics perspective, observes Gibbs law as described by Eq. 0-7. If the unfolded state at some temperature T is lower in Gibbs energy, the equilibrium between states will shift to the protein unfolding, and thus the protein is not stable at that temperature. It is clear that stability at a higher temperature is more difficult than stability at a moderate temperature: the entropic term dominates at high temperature, and an unfolded, free moving sequence of amino acids has higher entropy than the packed, folded state. For a given protein, it is necessary to change the

enthalpy and/or entropy delta of the folded/unfolded system in order to increase its stable temperature. The problem of high temperature or thermal stability, which is the focus of this work, is often described by the melting temperature, T_M, after which this equilibrium shifts.

In the remainder of this chapter, I overview the current methods for improving a target protein's thermal stability, the data that have been used to attempt to derive conclusions and rules about nature's modes of thermal stability, and the ways in which ML has been used to accelerate these processes. The gaps in current data and models are discussed, and a new pipeline described and analyzed that produced the current largest dataset of context-dependent, paired protein homologs across temperature regimes. Lastly a novel generalized protein model which aims to introduce thermally stabilizing variation in a context dependent manner using neural machine translation, NOMELT (Neural Optimized Machine Enabling Learned Thermostabilization), is described and its capabilities explored.

The design of functional proteins with high thermal stability

Improving protein thermal stability

Initially, naturally occurring proteins were externally stabilized in order to improve their ability to operate at high temperatures. [269] This can involve covalent modification or crosslinking of surface residues,[270–272] addition of a stabilizing solvent,[273] or binding to a surface.[274] These methods are still being used, but pose a few barriers: external stabilization of proteins requires time and material input in order to purify and stabilize the protein, and

additionally the stabilizations that are compatible with living organisms are limited to natural post translational modifications.[275,276] For some applications this may be acceptable, but an inherently thermally stable protein would be preferable. This must be possible due to the multitude of extremophile organisms whose proteome retains functionality at extreme temperatures.[277] Natively high temperature proteins are additionally desirable because they could be used in the context of bioreactors and other systems involving living organisms. A depiction of the three strategies is shown in Figure 3-1.

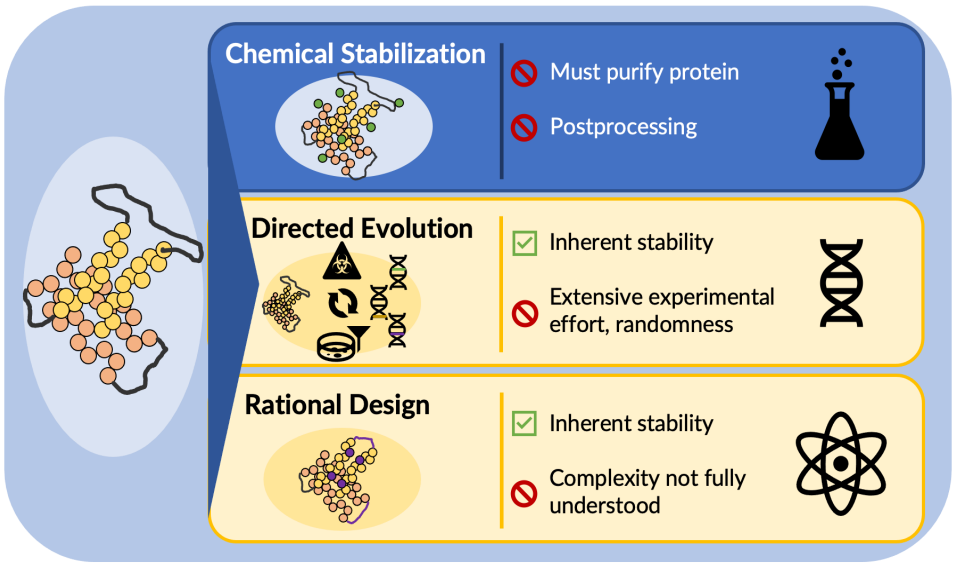

Figure 3-1: Current strategies to stabilize proteins for high temperatures. Chemical modification and stabilization involve post-translation processing of purified proteins with stabilizing agents, either by covalent bonding, immobilization, or stabilizing solvents. Sequence design modifies the amino acid sequence to be translated to infer thermal stability, which can then be purified or left to exist in a cell environment. This includes directed evolution, wherein the sequence is modified randomly or semi-randomly by mutagenesis or recombination, and beneficial variants are selected for. Alternative, first principles and statistical potentials can be used to help design the 3D structure of the protein targeting stability. Both directed evolution and rational design require significant effort to stabilize a protein of interest while retaining function.

Most recent work has approached the problem of thermal stability from the perspective of better sequence design. This tracks back to nature - related organisms living

at high temperature and moderate temperature, thermophiles and mesophiles respectively, have highly analogous proteomes. Same-function proteins between these organisms typically have >40% sequence similarity, have superimposable 3D structures, and have the same binding mechanisms, yet observe enough differences in sequence to account for the temperature delta in stability of up to ~80C.[278] The necessary difference in sequence comes from the effect of temperature on the atom-wise interactions within the native folded state and the unfolded state. For example, we credit the preference of the protein to a native folded state over an unfolded one in large part to packed hydrophobic cores,[279] yet hydrophobic interaction strength does not increase for temperatures between 25C and 110C,[280] so thermophiles cannot rely on hydrophobic packing alone in order to retain its folded proteins at high temperature. Thus, thermophiles have had to adapt their proteomes in other ways to convey stability than what would be sufficient at moderate temperatures. While there are some exceptions where environmental conditions in hyperthermophiles act to stabilize the protein,[281] the modern approach to increasing stability relies on modification of the protein sequence.[8]

Improving thermal stability by determining an alternative primary sequence can be approached head on with directed evolution.[282] Screening/selection is used to iteratively improve a protein's function, activity or property. Here variation is introduced into parts (determined via intuition) or the whole protein sequence experimentally by mutagenesis, recombination, or even targeted oligonucleotide mutations, and downstream variants are evaluated at some throughput.[283] This method does come with challenges: the phenotype of interest must have a causal effect on the measurement. Specific assays must be developed such that either the protein is purified and its activity measured in solution at/as

a function of high temperature, or the protein is a core component of the organism's metabolism as temperature stress is induced.[283] This means that currently only thermostability of enzymes have been improved in this manner to the best of my knowledge.[284–286] Yet unstated: DE can be extremely time consuming and expensive, both to design and conduct. Another approach is the design of hotspots.[287–289] Requiring the 3D structure of the protein, simulations can be used to find areas of the protein that are weakly folded, which can then be targeted for improved stability. Mutations at the hotspot can be directly designed with rational design or optimized for with site specific mutagenesis.[290] This strategy is capable of increasing the melting temperature by 9C, in just one example.[291] One might also consider consensus concept modeling, [292] where multiple sequence alignments are constructed with the protein's homologs, and parts of the primary sequence that are in consensus among the homologs are inserted into the target protein. Unfortunately, many of the suggested mutations using this method are destabilizing, and in practice the sequence alignments are instead used in conjunction with rational design or the homologs used to introduce variation for DE.[293,294]

While these methods have been effective at increasing the stability of a system of interest, they confer a significant experimental or computation cost, and are protein specific - they must be restarted for other proteins with different folds. For rapidly improving thermal stability in the future, it is desirable to complement protein-specific engineering with a generalizable translation over governing factors that nature uses to confer high temperature stability. This is particularly challenging because our current state of understanding suggests that these governing modes are context dependent.[29]

Analysis of the modes used by nature to improve thermal stability

In order to elucidate how and with what modes sequence effects the thermal stability of the protein (in a data driven manner), we require labeled data - e.g. proteins whose stability has been measured. Rigorously, one could measure the thermal stability, or the ability of the protein to avoid unfolding at high temperatures, by its melting temperature T_M, while the kinetic thermal stability or the ability of the protein (if an enzyme) to retain its activity and avoid denaturing can be measured by temperature of half inactivation T_{50}.[295] Unfortunately, the quantity of these data is vastly lower than the total number of proteins for which we have sequences, on the order of thousands. ProThermDB[296] contains ~32k wild type and mutated proteins with thermodynamic measurements, many of the mutations being destabilizing. FireProtDB[151] filters this data with additional data sources, as well as cross referenced sequence annotations with the purpose of allowing users to filter the data for machine learning, yet the total number of labeled mutations is a little over 13k. The largest known dataset of natural proteins with melting temperatures is 48k, derived from only 13 organisms.[150] To increase the amount of available data, many have worked with the organism's optimal growth temperature (OGT) as a proxy measure of stability. These datasets, among other small sources, or in house mutagenesis data, have been used to try to elucidate modes and methods of thermal stability using data driven approaches. A selection of some of the largest repositories of data is given in Table 3-1. Effort here can be separated into two categories: extracting patterns and statistical differences between the pools of known high temperature proteins and moderate temperature ones, and direct comparison of the 3D structure of same-function proteins across temperature.[8] From these studies it has been suggested that hydrophobic residue ratio, hydrogen bonding networks,

AA substitutions such as Glu → Pro, salt bridge formation, Pi-Pi and Cation-Pi interactions, among others, all impact the thermal stability of the protein. A summary of work can be found elsewhere.[29] While some of these suggestions have become generally accepted, such as salt bridge formation[297,298] and a few AA preferences that have marginal effect when applied unilaterally,[299] the only general consensus is that there is a delicate balance of rigidity and entropic flexibility, determined by short and long ranges interaction of all kinds, in stability.[29,300,301] In other words, we can gain stability by decreasing the folded enthalpy, increasing the folded entropy, or even decreasing the unfolded entropy, but a specific set of rules for how to do this has not been discovered. Many of the suggestions of governing modes have contradictory work or are weakly supported. Studies that rely on statistical comparison of pools of high temperature and low temperature proteins (using either T_M or OGT)[297,302–307] do not consider the problem in a sequence-dependent manner, while those that rigorously evaluate individual protein or protein pair 3D structure are too few in data to be generalized.[308–312] To make this problem more complex it is not a guarantee that by, for example, decreasing the folded enthalpy in a certain way, that the function of the protein is not hindered. Indeed, it is clear that the modes of protein thermostability are context dependent: while there may be generalizable methods and rules used by nature, they do not exist independent of each other and cannot be expected to be leveraged in the same way for all protein families.[300,312]

Table 3-1: A selection of the largest publicly available datasets of proteins and their thermal stability.

Name	Size Proteins	Size Prot. Pairs	Diversity	Label
Meltome Atlas [150]	48k	-	Diverse proteins, 13 organisms	T_M
ProThermDB[296]	32k	-	Diverse proteins and single point mutation	T_M
FireProtDB[151]	16k	-	Single point mutations	T_M
Engqvist et al.[84]	5.5 mil	-	Diverse enzymes, multiple examples across evolution	OGT
TemStaPro[85]	2.5 mil	-	Diverse proteins, multiple examples across evolution	OGT
Hait et al.[313]	700	1.6k	Diverse protein pairs	OGT

Largest existing dataset of thermo-meso protein pairs

The largest dataset of mesophilic-thermophilic protein pairs to date was created by Hait et al.,[313] with ~1600 pairs. The authors started with sequences with high resolution 3D structure from the protein data bank (PDB) and considered the organism's OGT. The choice of OGT over T_M as a thermostability label means that most structures found within the PDB are labeled (tens of thousands), far more than those with measured melting temperature. In order to identify proteins with the same or similar function (orthologs), one typically runs tools in an arsenal of sequence similarity searching strategies,[106] and compares the aligned 3D structure using eg. RMSD. The accepted method for sequence ortholog detection of proteins using sequence alone is to use multiple sequence alignment (MSA) tools such as HMMER[114,314] to known families of proteins. This is effective for finding very long range homologs, but relies on a validated set of seed sequences. To overcome this, the authors started with the local alignment tool BLAST[108] to find putative

protein orthologs among mesophilic and thermophilic structures. Here, the statistical strength of homology is rigorously defined as the "expectation value" (E-value) expressed as the probability of finding the given alignment in a sequence of random amino acids, which is normalized by the length of both sequences and the search space. See section "**Homology modeling**". Potential protein pairs were then filtered by labeling domains using validated Pfam families,[315] an external database of MSAs. It is worth noting again that even with these strategies, same-function proteins across species are difficult to rigorously label, and "function" in a protein can be ambiguous.[316] The authors started from a small set of proteins with known 3D structure from PDB, so the dataset produced by this work contains only a few thousand protein pairs, vastly less than the number of known proteins and too few to train a generalizable deep learning model. See Figure 3-4-right that shows that this dataset covers only a small portion of protein space.

Applications of machine learning to protein engineering

Machine learning has been incorporated to many parts of the protein engineering ecosystem and has been an accelerating force. Applications can broadly be categorized as follows: targeted predictors, active learning predictors, generalized predictors, sequence to structure models, structure to sequence designers, and targeted sequence variation modelers. Comprehensive surveys are given elsewhere, but here I summarize these applications and motivate NOMELT, a generalized sequence generator, in the remainder of this sub-section.[76,317–320] Figure 3-2 gives and overview of the different applications of ML to protein Engineering.

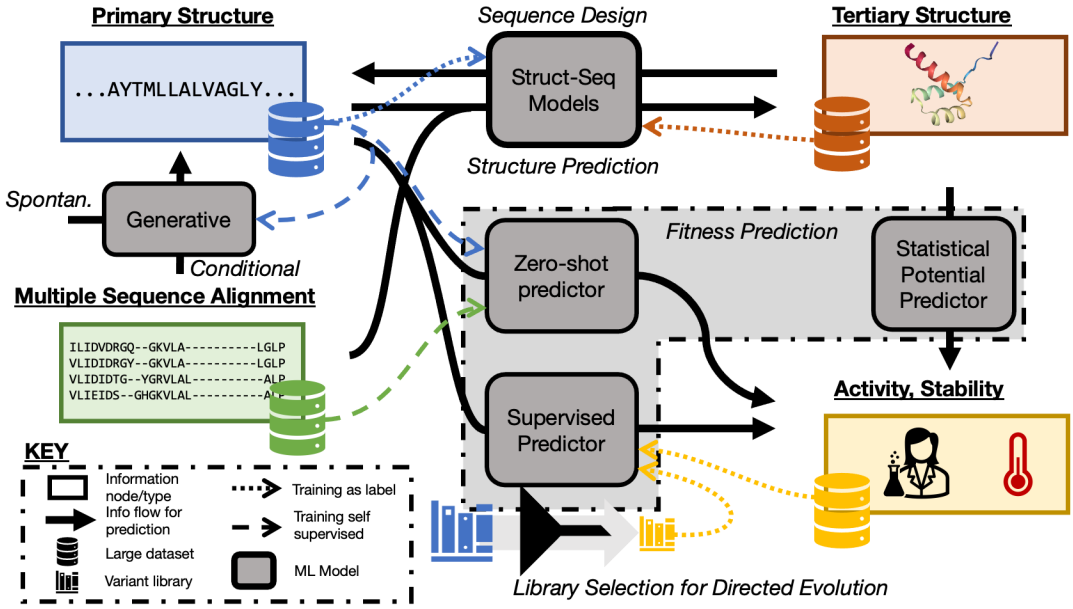

Figure 3-2: Summative flow of protein information through various machine learning applications for protein engineering. All tools ultimately aim to accelerate the arrival at a protein with properties desirable for a particular application. Models allow for prediction of structure from sequence, and of sequence design from desired fold. Generative sequence models can produce novel protein sequences either spontaneously or conditioned on a particular MSA of sequences. Fitness and stability predictors can help rank or prioritize variants in directed evolution campaigns. These predictors are often trained on the assay data for the specific protein in question in order to help find global optimums within a library, but can also be trained on more diverse datasets thus far less qualitative than required for library selection. Zero-shot predictors, either trained on vast sets of unbaled sequences or sequences relevant to the protein of interest, have shown some ability to predict activity without labels, and can help filter libraries to develop better training sets for supervised predictors.

Deep mutational scans or other variant libraries, when combined with an effective assay for a phenotype of interest like enzymatic activity, can be used to create protein specific predictors of the assay target.[87,321,322] Here, mutations upon a parent sequence or whole sequences are encoded for a machine learning model in order to predict the activity label. These models can achieve impressive accuracy and accelerate the exploration of a protein's functional landscape, e.g. during a directed evolution campaign.[323–326] Unfortunately, while these targeted predictors can vastly reduce the effort taken to identify a high performing variant, they rely on labeled data for the specific assay of interest, and

so significant experimental effort is still required for each use case. Another targeted strategy is to train models that learn the distribution of natural variation among the protein in order to rank variant activity.[92,327,328] While their performance is impressive given that they require no labeled data, they do require a wide MSA of homologs for the protein of interest, which is not always available.[329] Further, while their predictions are advantageous compared to random selection, in practice they are not accurate enough to select the global best performer, but instead used to create an effective training set for a supervised model.[330]

When targeted predictors are wrapped around a component of uncertainty or exploration, e.g. in a Bayesian framework, or with advanced sampling methods, these acquisition functions can be used to optimally balance exploration and exploitation, for the same use case as a targeted predictor. Here variants are selected to maximize learning.[331–335] In experimental frameworks that allow for the expression of a specific sequence, one can even design active learning frameworks to generate the next variants instead of filter them.[336] Again, though, this process must be conducted for each protein of interest.

Recently, there have also been attempts to create generalized predictive models, trained on more than one protein target. There are many global targets like solubility, expressibility,[337] and includes the target of this work: thermal stability. For example, the data described in section "Analysis of the modes used by nature to improve thermal stability" have been used to train machine learning models that are quite effective at predicting if a protein came from a thermophile or a mesophile and even somewhat accurate prediction of melting temperature or half activation temperature.[134,146,338–340] While this is encouraging in that it shows that there is some amount of generally learnable signal for thermophilicity and thermal stability, the predictors are not (yet) accurate enough to guide

researchers to a global high performing variant. This is likely due to the lack in diversity in the training sets, where only a few protein families or species are represented.[142] Most impressively, it has been shown that foundational language models, trained on vast sets of proteins in a self-supervised fashion, actually have some predictive power over variant activity.[41,50,88,341] These models can be used to help screen initial libraries of variants like targeted predictors, albeit with an accuracy that necessitates the eventual training or incorporation of a supervised predictor.[88,342] For the prediction of stability, they do not have the accuracy necessary to fully resolve close performers, see "Prediction of protein stability". Additionally, given that these models are trained on huge sets of natural sequences, they are biased towards ambient or average conditions, and no model yet exists that explicitly tries to model high temperature proteins.

It is worth mentioning specifically sequence-to-structure and structure-to-sequence design models. The impressive capability of the transformer attention mechanism along with the growing size of the protein 3D structure repository, PDB,[343] has allowed researchers to create deep learning models for the translation between primary structure and tertiary structure in both directions.[91,120,344–347] Attaining a 3D structure is critically important for developing and intuition as to a proteins mechanism and running downstream dynamic analysis. On the other hand, designing - at the very least a starting point - bespoke proteins *de novo* to e.g. to bind to a specific receptor has been made attainable with ML, with many designs being stable at ambient temperatures. These models have also been used to help predict protein binding.[348,349] While transformative, these strategies do not address high temperature stability.

Lastly, it is becoming possible to directly model variation in protein sequence. For this use case, generative models are trained on homologs of a protein, and are able to sample from the family/group.[336,350,351] This is useful when the experimental framework allows for the expression of full sequences, and can be helpful in introducing non-random variation into the library itself, as opposed to filtering large libraries with random variation. Unfortunately, this strategy requires a robust set of homologs, and is limited to the variation space provided by those input sequences. Again, it does not target high temperature stability, as the variation used to train these models is not specifically thermally stabilized.

The model detailed and evaluated in "**Leveraging neural machine translation for protein design**" (NOMELT) is a sequence-to-sequence model designed to translate from ambient temperature protein regime to high temperature protein regime. Not only is this a novel strategy, but it attempts to fill a niche that none of the above methods fill simultaneously: it is a generative and generalized designer of protein sequence that specifically targets high temperature stability.

Success of transformers on biological sequences

There have been recent attempts to translate the success of natural language processing technologies to the understanding of proteins. The analogy is a natural one: amino acids are words in a vocabulary, sequences of them are sentences, and the protein domain structure/function is the meaning. Much of the recent success in leveraging ML for protein engineering, as discussed in "Applications of machine learning to protein engineering", is owed to the emergence of transformers. See "**Attention based "transformers" for arbitrary sets**" for details on this architecture. While there are some differences in protein and natural language applications, such as a vocabulary that is only

20 tokens (amino acids) that have some functional redundancy across tokens, it is clear that the success of transformer attention-based models can be leveraged for proteins using large unlabeled datasets. See "Representation of protein primary and tertiary structure"

Furthering the protein language analogy, the task of improving the thermal stability of a protein is akin to the classic dialect translation problem in NLP.[352,353] Proteins in thermophiles have modified mechanisms of stability due to temperature's effect on the enthalpic and entropic forces that stabilize the protein, but they still require the same function (meaning). Thus thermophiles have a different dialect of the protein language that provides them the same meaning but using slightly different syntax. With this in mind, the possibility that a transformer based NMT model could learn the difference in dialect between moderate and high temperature proteins – allowing for future translation of arbitrary proteins into thermally stable variants – is explored in the remainder of this chapter. Given that the largest dataset of "translations" (moderate temp and high temperature pairs) contains only ~1600 examples, thermostability as an NMT task has yet to be unlocked.

Leveraging neural machine translation for protein design

Developing a database of low and high temperature paired proteins

To achieve a large dataset of pairs of homologous proteins across low and high temperature, OGT was leveraged as a label and 3D structure was not used as a starting point, unlike Hait et al.[313] Specifically, existing growth temperature data was combined

with large sets of proteomes and 16s rRNA sequences, and finally searched for homologous sequences across temperature.

It is well known that OGT and T_m are not equivalent, however a protein utilized by an organism *must* have stability of at least the temperature it is growing at. Indeed, it has been found that the relationship between OGT and T_m is approximately 24.4+0.93*OGT.[354] Thus, on average, a protein from an organism exhibiting a higher OGT will have higher thermal stability than proteins from a more ambient organism. This is clearly a more noisy metric for protein stability than T_m. An organism's proteins occupy a distribution of melting temperatures, and OGT is a somewhat imprecise measurement.[355] Optimal growth rate can occur over a range of temperatures, and is also media dependent. This noise is acceptable considering that any organism whose growth temperature has been studied can contribute its proteins (N=~4000 for prokaryotes) to a pool of labeled data, while the exact increase in temperature is not strictly necessary to infer patterns of improved stability.[297] The use of proteins labeled with OGT (millions) compared to those available with experimental T_m (thousands) lends itself towards leveraging complex NLP based sequence transformers.

The use of proteins without 3D structure makes it especially challenging to pair same-function proteins, but unlocks millions of known protein sequences instead of tens of thousands of known structures. Even with structure, function is a somewhat ambiguous attribute: do structurally similar proteins with slightly different target ligands count as the same, or activity on the same ligand but with different strengths? Indeed, protein space is diffuse and overlapping, and crisply labeling function is an open question.[89] Despite this, the goal can be more generally rephrased as seeking related proteins that likely occupy a

similar niche. Protein pairs were instead determined using pairwise-sequence alignments and a sample of the results validated using existing protein family models and alignment of AF2 predictions.[91]

The pipeline to produce the dataset – entitled learn2therm – is depicted in Figure 3-3 and detailed in this section.

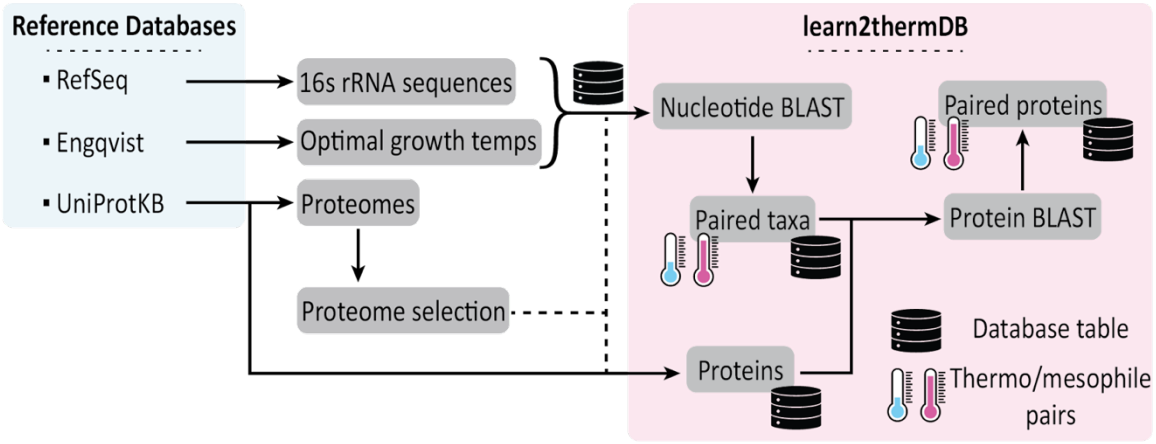

Figure 3-3: Workflow for labeling homologous protein pairs across temperature. Raw data included RefSeq 16s rRNA sequences, OGT labels from Engqvist, UniProtKB proteome metadata and proteins. Proteome metadata was parsed to identify a single proteome for highly represented organisms, while retaining data for weakly studied taxa. Proteins were filtered such that only ones from the chosen proteomes and for which OGT labels were present were kept. Protein pair search space was filtered by first identifying pairs of related organisms via 16s rRNA alignment. Protein pairs were searched for by alignment of sequences. Final database tables are taxa, pairs of meso/thermo taxa, proteins, and pairs of meso/thermo proteins.

To start, Archaeal and Bacterial 16s rRNA sequences, along with their associated NCBI taxonomy identifier (taxid) from NCBI BioProjects 33175 and 33317 were retrieved using the Entrez API.[356,357] Only full sequences were retained, ranging from 1300 to 1600 bases. When multiple sequences mapped to the same taxid, >98% identity was observed and the longest was retained. Curated OGTs, produced by Engqvist, were then downloaded.[358] For cases with multiple OGTs corresponding to the same species taxid, the average was used. The OGTs and 16s rRNA sequences were subsequently linked via taxid, generating a table with both quantities measured. For the next steps, organisms with OGT

>40 °C were labeled as thermophiles. It should be noted that this information is tracked, enabling a user to filter data using a more stringent thermophilic label, such as 60 °C.

Archaeal and Bacterial protein sequences were downloaded from UniProtKB.[359] UniProtKB Proteome metadata was also retrieved to minimize redundancy while preserving taxonomic diversity. For strains with multiple proteomes, if any, a single proteome was selected based on UniProt's priority labels: "Reference and representative proteome", "Reference proteome", "Representative proteome". If these labels were absent, "Redundant proteome" and "Excluded proteome" were discarded, and the most populous remaining proteome was selected. This process was repeated at the species level. Protein data files were parsed for primary sequence, host organism taxid, PDB, and AlphaFold database ID. Proteins not mapping to an organism in the taxa table were discarded. If a protein mapped to a proteome, it was retained if it belonged to the priority proteome identified for the organism, or if the priority proteome for the organism contained fewer than 2000 proteins. This avoided dataset saturation with model organism proteins while still including novel or infrequently studied proteins.

To provide context-dependent information on protein thermal stability, pairs of proteins must be identified from a large pool of thermophilic and mesophilic ones. To do this, homologous sequences were identified using a BLAST-like local alignment.[111] The search space, considering only proteins <250 amino acids, was approximately 4.5 trillion pairs. A full pairwise BLAST-like search given the protein data space was projected to cost an unacceptable amount of carbon at 50,000 kg. To mitigate this, the search space was filtered by identifying evolutionarily related taxa across temperatures. Using BLASTn, 16s rRNA sequences for mesophilic and thermophilic organisms were aligned. This sequence

evolves slowly and is frequently used for inferring taxonomic relationships.[360–362] See Appendix C.1.3 for alignment parameters. Any organism with >81% gap compressed sequence identity and >98.5% coverage on both strands was considered a taxa pair for the subsequent protein homolog searches, resulting in 150k taxa pairs and a search space of 230 billion possible protein pairs. The taxonomic breakdown and protein contributions of these pairs are depicted in Fig. 3-4-left.

Using DIAMOND, pairwise local sequence alignments were executed for each thermophilic-mesophilic taxa pair, considering only proteins of 250 amino acids or less. A resource test indicated that DIAMOND had a twofold speed increase over BLAST, while preserving sensitivity (see Appendix C.1.4 for alignment parameters and a comparison with BLASTp). Several alignment metrics such as percent identity and alignment coverage were monitored (refer to Appendix C.1.2 for a comprehensive list and definitions). This process yielded 69 million putative protein pairs with an E-value less than 1e-4 and over 75% coverage of both sequences. Using a stricter definition of thermophilicity (OGT >60 ^{o}C), the number of protein pairs reduces to 9 million. Despite this reduction, the dataset remains significantly larger than any existing homologous sequence collection across temperature.[313] The protein space occupied by protein pairs is depicted in Figure 3-4-right, covering much of known protein space however with notable gaps, likely due to lack of sensitivity of DIAMOND to find distant homologs across temperature. Summary statistics for the dataset are given in Table 3-2, and Figure 3-5 shows that while the majority of protein pairs have minor temperature differences, millions of protein pairs are still retained when ΔOGT >30 ^{o}C.

Table 3-2: Number of various entities in the dataset all with temperature labels. Organism and protein pairs were identified via local alignment, thus not all taxa/proteins participate in pairing, and those that do may occur multiple times.

Number of organisms (in pairs)	Number of proteins (in pairs)	Number of organism pairs	Number of protein pairs
9,536 (5,028)	24 mil (4 mil)	150 k	69 mil

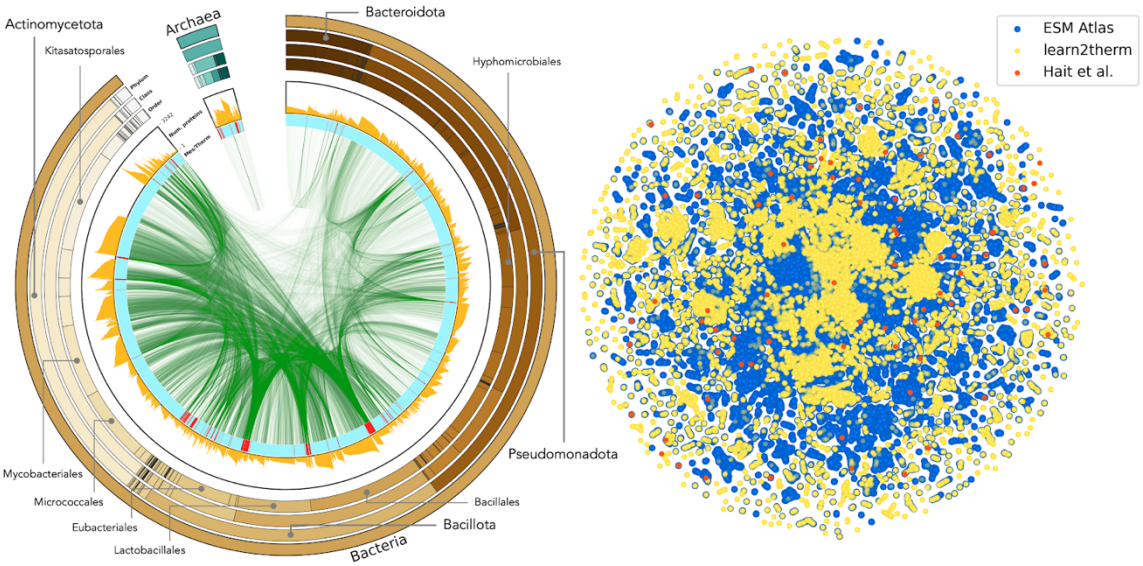

Figure 3-4: The taxonomic and protein space covered by the protein homologous pairs (N=69 mil) within the dataset. Left) NCBI taxonomic breakdown of the dataset.[363] The outer ring depicts super kingdom, Phylum, Class, and Order, where the size of wedges indicates the number of organisms in the classification that contain at least one protein in a learn2therm protein pair. Highly populated Phylum and Class are labelled. The inner ring moving inward is a histogram of the number of proteins participating in pairs per organism followed by a colour mapping labelling organisms as mesophilic in blue and thermophilic in red. Central connections indicate taxa pairs contributing to protein pairs. Right) Two dimensional mapping (using t-SNE) of a sample protein space as determined by Evolutionary Scale Model (ESM) embeddings.[50] In blue, a sample of data from the ESM Atlas with highest structural confidence. Note that this data contains eukaryotic proteins. In yellow, our proteins in pairs, and in orange, the current largest set of protein pairs across temperature. Size of samples conserves relative size of our proteins vs. the Atlas and reference dataset. For details of the mapping procedure, see Appendix C.1.8.

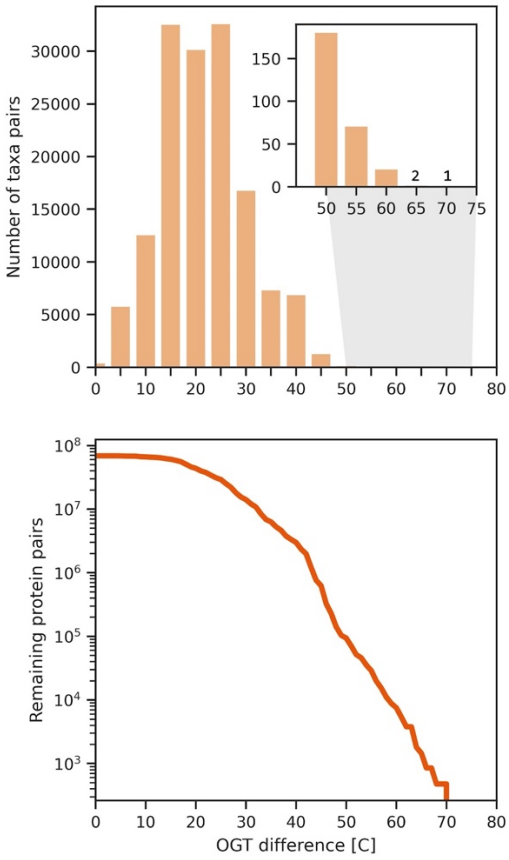

Figure 3-5: The data distribution as a function of difference in growth temperature between items. top) Histogram of difference in OGT between pairs of organisms. The data is skewed towards 0, but there are still many pairs with temperature differences >30 ºC. bottom) Count of protein pairs remaining as minimum OGT difference is increased. Nearly 14 mil protein pairs are retained with OGT difference >=30 ºC.

The entire process of developing the dataset, from extraction of raw data to downstream validation steps (see Validation of protein pairs) is tracked using data version control (DVC).[364] Thus, the complete history of how the data changed as the code was developed and the parameters were changed (e.g. maximum protein length, minimum alignment metrics etc.) is available. Additionally, the pipeline can be rerun with a single command after environment and computing cluster configuration. The data tables created in this process (taxa, proteins, taxa_pairs, and pairs) were collected and linked as a

relational database using DuckDB, allowing for fast access and filtering of the data.[365] A key subset of tunable parameters is shown in Table 3-3 along with the values used for the presented data. The carbon cost of compute consuming steps in the pipeline was estimated using CodeCarbon at 11.5 kg. The total cost of developing the method was estimated at 70 kg. A comprehensive description of the pipeline steps and parameters is given in Appendix C.1.1.

Table 3-3: A small selection of tunable parameters used to produce the final dataset.

Parameter/s	Description	Published value
OGT threshold	Binary threshold to split organisms into mesophiles and thermophiles.	40.0 °C
16s alignment coverage	Minimum alignment coverage (both strands) of 16s rRNA sequence to be considered a taxa pair	98.5%
16s alignment % id	Minimum alignment gap compressed percent identity of 16s rRNA sequence to be considered a taxa pair	81%
protein alignment coverage	Minimum alignment coverage (both strands) of protein sequence to be considered a protein pair	75%
protein alignment E values	Maximum E value of protein alignment to be considered a protein pair	1e-4

The dataset consists of four main tables: ‘taxa’, ‘proteins’, ‘taxa_pairs’ and (protein) ‘pairs’. An abbreviated schema detailing the relationships between these tables is illustrated in Figure 3-6. For a more detailed schema and a comprehensive description of each field, please refer to Appendix C.1.5. The dataset is not only free and openly available, but the code pipeline is open source and parameterized such that users can customize the data.[28]

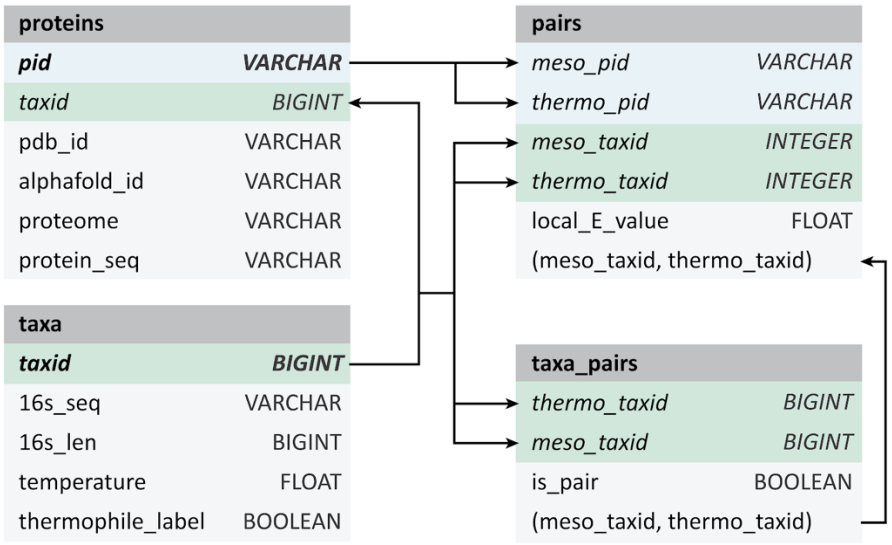

Figure 3-6: Abbreviated schema for the presented database. The taxa table contains NCBI taxonomic information, from superkingdom to species, as well as 16s rRNA sequence and OGT. The protein table contains protein sequence and external database links. The taxa_pairs and pairs table contain metrics for alignment for 16s rRNA sequences and protein sequences, respectively.

Validation of protein pairs

In order to confirm that the protein pairs produced in the database were valid, homologous pairs across temperature, a few additional steps were undertaken. First, proteins were mapped to existing published datasets and the OGT labels confirmed. Additionally, remote homology HMM search was used to compare protein pairs by applying known protein family labels. Structural and sequence alignments were undertaken and the results compared to the results from the largest existing dataset of protein pairs. Lastly, a growth temperature predictor was trained to identify signal in the labeling. These steps are described in the remainder of this subsection.

To ensure a proper mapping of OGT label to protein, learn2therm proteins were joined with enzymes from Engqvist et al.[84] via UniProt ID. Comparing the labels for the records in both datasets (N=1.6 mil) yields an R^2 of 0.995. Some small differences in OGT

labels arise due to the strain aggregation procedure, described in Developing a database of low and high temperature paired proteins.

To validate that the OGT is a suitable substitute for melting temperature as a measure of thermal stability, and to ensure that temperature data has been accurately mapped, proteins within the dataset were queried against existing melting temperature datasets. Wild type proteins from FireProtDB and the Meltome atlas were paired to proteins within the learn2them dataset using >99% coverage and identity, yielding 4,640 proteins with both internal OGT labels and external T_M labels.[150,151] A Spearman's correlation of 0.85 with a p-value of 0.0 was observed between these two quantities, suggesting a strong correlation. Furthermore, a binomial test was performed to determine if the melting temperature has a >99% chance of being greater than OGT, yielding a P-value of $2.68e^{-19}$. Figure 3-7 provides a parity plot of these two values. This analysis indicates that the trend observed in an organism's protein melting temperatures is reflected in its growth temperature.

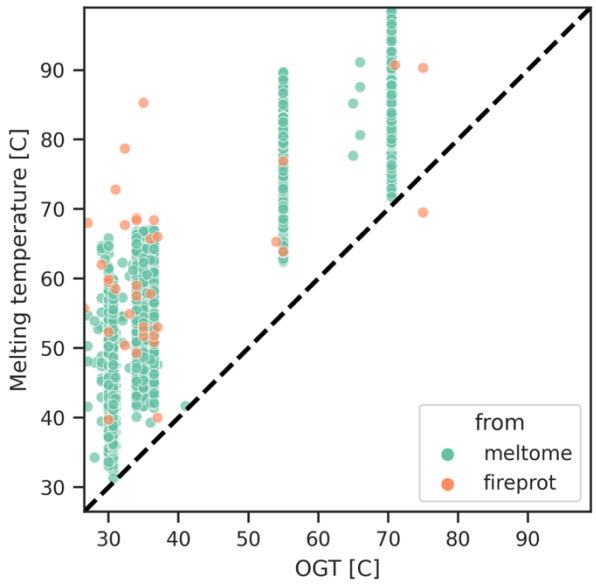

Figure 3-7: Melting temperature vs OGT, using data from two third party databases. The black dashed line is identity, and almost all examples fall on the side of T_M > OGT. The adage that melting temperature is greater than OGT is clearly supported, with a Spearman's of 0.85 (P value 0.0), and a binomial test of >99% chance of passing with alternative P-value $2.68e^{-19}$.

Hait et al. previously produced the current largest dataset of functional protein pairs across temperature at 1.6k pairs by starting with structures of PDB entries.[313] This dataset served as a benchmark, with the quality of their protein pairs used for comparison. To facilitate this comparison, their pairs were aligned using DIAMOND, and the alignment metrics compared to those from learn2therm.[111] Statistically similar or even superior alignment scores to the baseline were observed in the learn2therm dataset, as depicted in Table 3-4. Distributions of alignment percent identity and homology (as indicated by normalized bit score) are presented for both our data and the baseline pairs in Figure 3-8a-b. Further comparisons given in the table and figure are discussed below.

Table 3-4: Statistical comparison of learn2therm protein pairs to Hait et al.'s pairs. The base set of protein pairs (alignment coverage >75%) as well as a subset (N=24 mil, coverage >95%) are shown. Percent identity, bit score (normalized to the average length of both proteins), and Pfam annotation Jaccard score are t-tests, while output P-value of structural alignment was treated as a binary if smaller than 0.001. For all metrics, learn2therm pairs meet the baseline on average for the subset with >90% alignment coverage, and most are still quality for the full set. Statistical significance is bolded.

Score	Statistical Test	Probability (Cov>75)	Probability (Cov>95)
Normalized percent identity	left t	**$1.75e^{-6}$**	**$6.06e^{-155}$**
Normalized bit score	left t	**$1.94e^{-8}$**	**$9.47e^{-117}$**
Pfam Jaccard	left t	0.99	**$3.24e^{-13}$**
Structural P score	binomial P<0.001	**<0.001**	**<0.001**

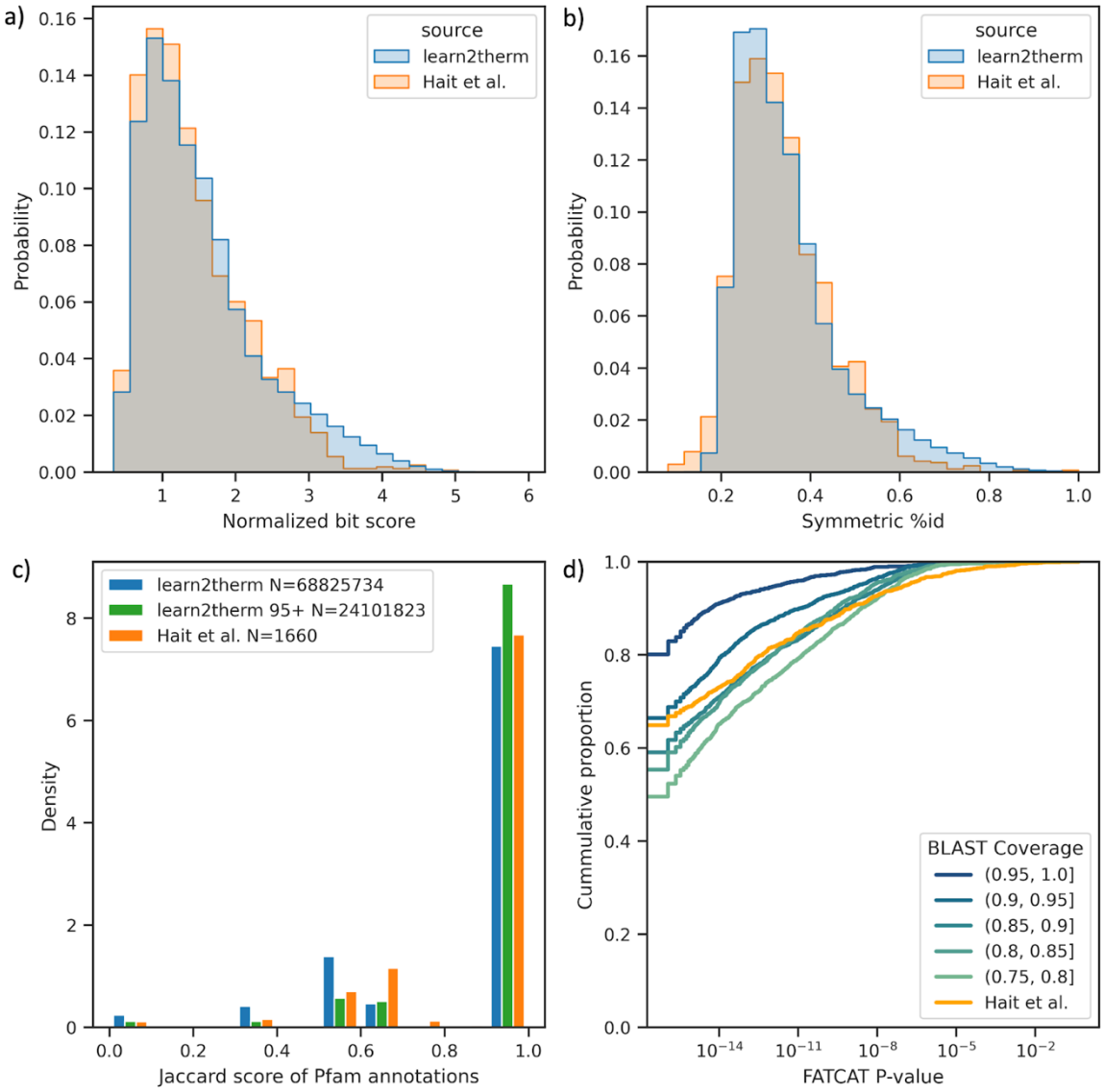

Figure 3-8: Comparison of protein pair quality between learn2therm data and Hait et al.'s 1660 protein pairs.[313] a) Empirical distribution of local alignment homology as bit score normalized to the average length of both protein strands. The data has a right shifted score on average with t-test probability=1.94e^{-8}. b) Same as A, except with percent identity. The data has a right shifted score on average with t-test probability=1.75e^{-6}. c) Empirical distribution of Jaccard score over Pfam annotations for learn2therm data (blue) compared to the baseline data (orange). The full data has more annotation mismatches on average. When only the 25 mil protein pairs with BLAST coverage >95% are considered, the Pfam annotations become indistinguishable from the baseline with t-test probability=3.24e^{-13}. d) Cumulative distribution of FATCAT structural alignment P-value for bins in BLAST coverage uniformly sampled from learn2therm data, compared to the baseline structural alignments. Even low coverage pairs are more likely to have less than one in a thousand P-value with binomial confidence >99% .

Pfam was used to annotate proteins in both the learn2therm dataset and Hait's pairs.[314,315] Retaining matches with an E-value < 1e^{-10} (normalized using the size of Pfam

35.0) resulted in 86.1% of learn2therm proteins and 99.8% of Hait's proteins being labelled with at least one annotation. This suggests that learn2therm data include novel proteins not extensively represented in Pfam. The quality of a protein pair according to Pfam was evaluated by calculating the set Jaccard score of accession annotations (Appendix C.1.6 Eq. C.1.6). Only pairs where at least one member was labelled with at least one annotation was considered, as the remaining are out of scope for Pfam. Although learn2therm data contain slightly more annotation mismatches, if only the N=24 million pairs with >95% sequence alignment coverage are considered, the scores are indistinguishable from the baseline pairs, with a t-test probability of $3.24e^{-13}$. The score distributions for both datasets are compared in Figure 3-8c. A detailed description of this search is available in Appendix C.1.6. Homology searches using HMMs are highly sensitive to evolutionary distance, suggesting that true pairs of proteins with the same or similar functions are likely to share Pfam annotations, if available.[106]

FATCAT was used to align PDB structure of the baseline pairs with flexible alignment.[119,343] Chain A was chosen if present. This flexible protein structure alignment algorithm provides an empirically scaled score "P-value" which indicates the likelihood that the raw alignment score were to occur between two random proteins, thus a P-value close to 0.0 indicates a quality structural overlap. This process was repeated for learn2therm protein pairs using PDB structures where available, otherwise utilizing AF2 predicted structures form their database.[91,120] For learn2therm protein pairs, a subset of 10k pairs was used due to computational cost limitations of structural alignment. This sample came uniformly from 5 bins in sequence alignment coverage between 75% (the dataset minimum) and 100%. The cumulative distribution of probability scores from FATCAT for

each of these subsets is given in Figure 3-8d, where learn2therm protein pairs have alignments even less likely to be observed randomly than the already quality baseline. To compare the alignment results to the baseline statistically, a binary problem was considered where FATCAT probability of occurring randomly less than one in a thousand is labeled a quality protein pair. Conducting a binomial test, even pairs with sequence alignment coverage <80% are indistinguishable from the baseline, and higher alignment coverage yields structural alignment with smaller P-values than Hait's on average with >99.9% confidence.

Approaching protein thermal stability as a translation task

With the goal of creating a translator to introduce thermally stabilizing variation into engineering target proteins in mind, it is critical that the choice to approximate stability as organism level optimal growth temperature contains enough signal to learn. This relationship between protein sequence and OGT has been touched on in other work, where the authors were able to relate or predict organism OGT based solely on genomic and proteomic features. [277,338,355] In the work by Sauer et al., a predictor had a mean error of ~5 Celsius. This makes sense, as OGT is a somewhat imprecise measurement: a reported OGT of 25 ºC is nominally if at all different than 30 ºC. It is necessary to confirm that it is possible to resolve a difference in OGT for the learn2therm dataset before training a translator. To facilitate this test, the data was preprocessed using the following steps: binarization by OGT <30 ºC or >=60 ºC, class balancing, deduplication of similar sequences, and splitting based on NCBI taxonomy of host species, resulting in a training and test set of 290k and 28k proteins respectively, each labeled as mesophilic or thermophilic (see Appendix C.1.7 for details on the rigorous preprocessing conducted).

A recent predictor, TemStaPro, was executed on the test set of 28k proteins.[85] This model is an ensemble of neural networks trained on top of the output embeddings of ProtT5XL, a Large Protein Language Model (LPLM).[41] The TemStaPro ensemble outputs predictions in bins of 5 ºC between '<40' and '65<=', and conducts a self-consistency check between ensembles. For the data, the model was consistent for 95.5% of examples, and of those, predicted the correct class with 91% accuracy compared to a null model of predicting the majority class with 61% accuracy. The distribution of predictions is shown in Figure 3-9 below, where the model is clearly a predictor of >=60 and <30 ºC labels.

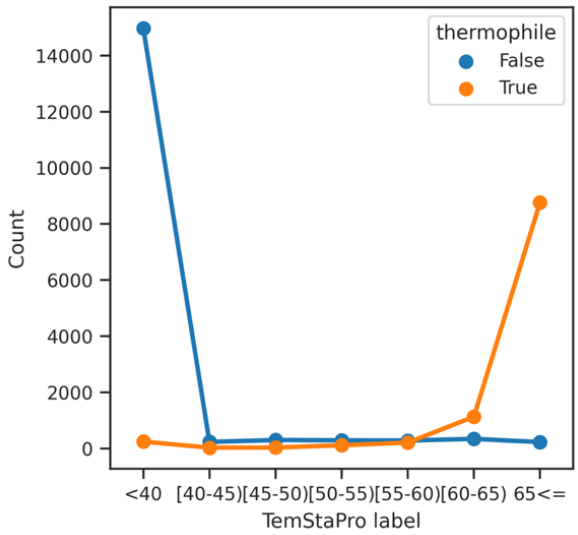

Figure 3-9: Predictions on a learn2therm protein test set by TemStaPro.[85] Thermophilic label is >=60 ºC versus <30 ºC. The model made very few predictions between these two quantities where there is no data, as expected, and was able to recover the actual classes with 91% accuracy.

Given that TemStaPro was trained on UniParc proteins, it is possible that the model has seen data from this test set before. To ensure that high performance was not due to data leakage, a new model was trained on the development set of the data. In order to save on training cost and carbon, ProteinBERT, a LPLM, was finetuned.[366] The training parameters

and architectural details can be found in Appendix C.1.7. The performance on the held out test set is shown in Table 3-5 below. The model was highly predictive, confirming that the dataset of proteins and OGTs retain signal.

Table 3-5: Test set performance of a fine tuned LPLM.

Balanced Accuracy	F1 Score	Matthew's Correlation	Confusion Matrix
92.5%	0.91	0.85	[16069, 1459, 671, 10296]

It was clear that the data can support the foundational machine learning task of thermophilic classification, and has the resolution required for a transformer model to differential low and high temperature proteins. With the addition of homologous pairing, the data can support the training of a seq2seq translator. In order to save on carbon cost, the foundational protein encoder-decoder transformer model ProtT5[41] was used as a starting point. This model was trained in a self-supervised manner to reconstruct protein sequences, and it is one of few existing foundational protein models that is capable of causal language generation out of the box.

The original learn2therm dataset was filtered according to the findings for the OGT predictor. Only pairs with a mesophilic temperature of <=40ºC and a thermophilic one of >=60ºC were kept. The protein pairs themselves were filtered such that only those with >95% sequence alignment coverage of both strands were kept, and the difference in sequence length less than or equal to 10%. This produced 4.7 million protein pairs. Due to redundancy and high sequence similarity among homologs in the dataset and the desire for functionality across a diverse set of proteins, uniform random splitting of the data would have been a blatant leakage and invalidate validation and test set performance. A more

rigorous splitting strategy was necessary because the underlying assumption in canonical ML tasks such as image processing, e.g. that the data are IID, is critically incorrect for protein sequences due to evolutionary relationships. Splitting is not an exact science, see section Data splitting for effective evaluation of trained models. Here, the dataset was clustered withMMSeq2 at 50% identity using the cascading connected component method, which resulted in 85k clusters. See Appendix C.2.3 for MMSeqs2 parameters. After splitting, an average E-value of test sequences against train sequences of $1.6e^{-4}$ according to blast alignments (See Appendix C.2.1 for BLAST parameters) was observed, which is consistent with existing splits in the literature.[93] The test and validation set was further filtered to only include a single random protein from each cluster in its split, in order to eliminate large and populated clusters from dominating evaluation. Only 1000 of these filtered sequences for each set were retained due to the expense of model validation and BEAM searches. This set of sequence pairs is used for scoring of the model in "Thermophilic sequences can be recapitulated from mesophilic homologs" and "Known thermophilic protein attributes are captured by NOMELT", while all of the data was combined to train a final model variant for downstream engineering applications in "NOMELT as an Engineer".

The hyperparameters used to train the model are summarized in Table 3-6 below, which includes both the model used for test evaluation, and a final model trained on all of the data. Note that these are the result of a small amount of hyperparmater optimization, briefly described in the next section. The data preparation and model training was tracked with DVC, and the entire set of hyperparameter knobs able to be modified is given in Appendix C.2.4 along with the T5 model pytorch printout. The model takes approximately

9 hours to train using 8 NVIDIA a40 GPUs with DeepSpeed ZeRO-3 configuration. This corresponds to about 4.8 kg of approximate CO_2 emissions on the Washington State energy grid.

Table 3-6: Major hyperparameters for the meso-2-thermo NMT. The Eval Model was used to compute aggregate performance metrics for a held out test set, while the full model was trained on all data and used for downstream case studies. *Epochs were determined by the early stopping point of the Eval Model.

Hyperparameter	**Value**	
	Eval Model	Full Model
	Data Hyperparameters	
Meso/Thermo OGT Window	40°C/60°C	40°C/60°C
Minumum sequence coverage both strands	95%	95%
Max sequence length difference	10%	10%
MMSeqs2 %id threshold	50%	-
Data size (train/val/test)	4m/1000/1000	4.7m/0/0
	Model Hyperparameters	
Model parameter count	2.8b	2.8b
Frozen layers/trainable layers per encoder and decoder, from bottom	4/20	4/20
Effective batch size	1120	1120
Max learning rate	$1e^{-4}$	$1e^{-4}$
Learning rate schedule	Linear, 10% warmup to 1 epoch	Linear, 10% warmup to 1 epoch
Parameter precision	bf16	bf16
Optimizer	AdamW (0.9, 0.999)	AdamW (0.9, 0.999)
Loss	CrossEntropy	CrossEntropy
Label smoothing factor	0.001	0.001
Dropout rate	10%	10%
Early Stopping Threshold	0.01	-
Early Stopping Patience	4	-
Epochs	0.18	**0.18***

Thermophilic sequences can be recapitulated from mesophilic homologs

It is known that proteins among a family can have as little as 20% identity and still retain same-or-similar function, and that residues on the length of the protein can occupy a distribution of amino acids and can include gaps and insertions between sequences.[367] Thus, we would not expect nor want the model to perfectly recover every thermophilic amino acid in a particular protein pair. This means that the measure of success is not as clear as more canonical DL applications where there is one or a small set of synonymous correct answers. As a baseline performance to target, the current standard for introducing non-random protein variation targeting stability was used: natural variation. Consider the rare situation for which the design target has many known thermotolerant variants. In such a case, optimizing over natural sequence space will quickly saturate to quality stabilization. It is desirable to achieve close to or similar performance to this method, but without the need for constructing an MSA over known thermophilic homologs. To measure this quantitatively, cross entropy over residues according to Eq. 0-3 is considered, which can be compute for an MSA as well as for model predictions. MSAs for thermophilic protein sequences were constructed using jackhammer for the 1,000 test sequences against all learn2therm thermophilic proteins. See Appendix C.2.2 for the alignment parameters. The cross entropy over the distribution of amino acids per position in these MSAs against the ground truth thermophilic sequence at each position serves as the baseline at 0.94 on average. For further reference, the null alternative of uniform random for the data is a cross entropy of 3.0.

The data that the model was trained on is redundant among protein pairs, as seen in Figure 3-11, where the probability density of the number of times a particular sequence is

seen in the training set for mesophilic and thermophilic sequences is shown. Each thermophilic sequence to be constructed has on average 107 mesophilic counterparts, and 7 for the inverse. This was desirable as NOMELT had some redundancy across evolutionary space to learn from, in order to try to meet the natural baseline.

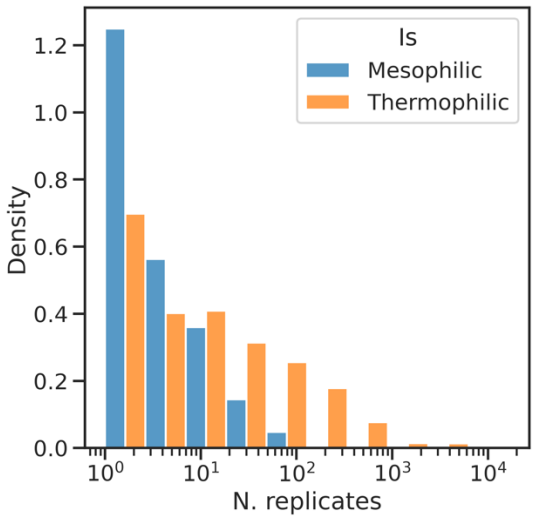

Figure 3-11: Density of replicate counts for mesophilic and thermophilic sequences occurring in the training set. The most common number of replicates for both the input and the target sequences is one, however the majority of proteins occur multiple times (paired to a different homolog) in the training set. Thermophilic replicates are right skewed, as homologous protein pairs were identified from a much larger group of mesophilic sequences to a smaller set of thermophilic ones.

Ideally, many hyperparameter values would be tested and the effect on the validation loss measured until the natural baseline is met. Unfortunately, a rigorous optimization of all the hyperparameters was not possible due to computational cost, but a few different hyperparameters were varied in low throughput single point tests, including label smoothing, reweighing examples for the loss function by inverse MMSeqs cluster size, freezing early layers in the model, and clustering parameters, which resulted in the final

values in Table 3-6. Freezing layers of the model had a minor negative affect on the loss but a linearly positive one on training cost, so it was chosen to freeze the first 20% of layers, for loss change of 1.77 to 1.81 compared to freezing no layers. Most of the model hyperparameters tested had little to no effect, with the exception of the rigor of the MMSeq2 cluster splitting. The cross-entropy validation loss is shown in Figure 3-10 below, which shows that using <50% sequence identity as a cluster threshold resulted in performance worse than the natural (albeit more cumbersome) baseline.

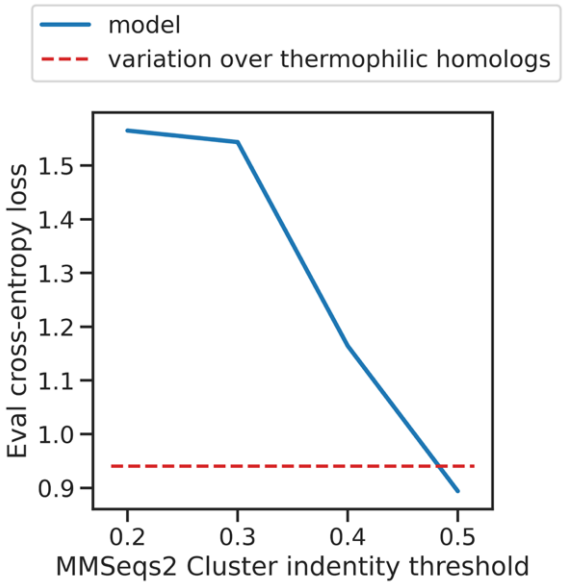

Figure 3-10: Validation set loss of trained models as a function of the MMSeqs2 clustering percent identity threshold used to split data, in blue. Only after the rigor of splitting is reduced to 50% identity did the model perform as well as natural variation. This indicates that, while the model does not require an MSA of high temperature homologs to generate variation like the natural method, it does require that the target protein have similar examples in the training set.

From the perspective of generalizability in large protein models not specific to a particular family, it is desirable to drive the similarity (in this case cluster identity threshold) between data splits as far down as possible. The best base scenario is to produce

a model that still performs well when given a protein that has zero likeness to any protein in the training set, e.g. a novel sequence of amino acids not even distantly homologous to any training set examples. This is unreasonable, however, as deep models are known to be unpredictable when extrapolating.[368,369] Indeed, even hugely complex, state of the art deep learning applications for proteins must have seen some amount of similar morphology during training; CAMEO, the validation set for AlphaFold2, guarantees only <85% sequence identity at 70% coverage to known experimental structures.[86] For existing deep predictors based on protein language models, we see that performance is significantly impacted by the scale over evolution seen during training, and the downstream tasks are either protein specific or have >30% id observed across splits.[41,78,370] Thus, these results are not unexpected.

The test loss of the model after the small hyperparameters exploration is 0.9, which meets the loss over natural variation of 0.94. This indicates that the model is better at predicting the true amino acid at a position than treating each position independently and sampling the most probable residue from natural amino acid distribution over thermophilic homologs. Thus, it is expected that the model could integrate thermophilic variation as well on average as if one were to build an MSA over known thermophilic homologs without needing a single thermophilic sequence, let alone to find a set of them and build that thermophilic MSA, contingent that the engineering target has about 50% identity with at least one training example. In "NOMELT as an Engineer" we see that there is at least some capability for engineering targets even when this requirement is not met. A number of additional metrics were computed on the test set as detailed below and summarized in Table 3-7.

In addition to cross entropy, which is residue-wise, a few metrics were computed by comparing the ground truth thermophilic sequence to a BEAM search over the model's autoregressive output, e.g. a single "translated" sequence. Of note, the model generated sequences with a 2.3% difference in length on average (shorter or longer) compared to the true thermophilic sequence, and had a transcription error rate of 47%, indicating that 53% of the time the model placed the exact correct amino acid in the correct position. Note that the error rate does not consider homology/analogous amino acids, or gaps and deletions, and is normalized with respect to the total number of test residues, as opposed to the number of test sequences. Each generated sequence was also aligned to its true test thermophilic sequence yielding full length alignment identities between 4.5% and 100% with an average of 43%, and bits-per-residue of 2.4 according to the BLOSUM62 scoring matrix. A depiction of the similarity between the generated sequence, its ground truth label, and the mesophilic input is given in Figure 3-12. The majority of model predictions were closer to the label than the mesophilic and thermophilic counterparts were to each other, and when untrue, the model was likely to fall back to a prediction similar to the mesophilic sequence. Only 19% of the time was the generated sequence further from both the mesophilic and thermophilic sequence in identity than they were from each other.

Structural similarity was also compared by running ESMFold prediction on both the ground truth and the BEAM search output in the test set. The secondary structure of sequences was labeled using pyDSSP.[371] The difference in categorical distribution of Helix, Strand, and Loop residues for each pair of generated-ground truth sequence was measured by the Jenson-Shannon divergence. With a value of 0.01 on average, the model was correctly transcribing the expected secondary structures in target proteins. Further, 3D

alignment of the generated and thermophilic predicted structures using FATCAT produces a mean P-value of 0.01 and a max of 0.1, indicating that the generated sequences were close in structure to the ground truth, according to ESMFold and FATCAT.

Table 3-7: Scores of the model for recapitulating the test set. “Residue” scores were computed on a per-amino acid basis, “Sequence” scores were computed by comparing full sequences, and “Structure” were derived from comparison of ESMFold structure prediction. *Does not use the NOMELT model, but instead natural variation over thermophilic homologs.

Test Metric	Value	Type
Cross Entropy Loss	0.90	Residue
MSA Cross Entropy Loss	0.94	Natural*, Residue
Transcription Error Rate	47%	Sequence
Sequence Identity	43%	Sequence
Bits per residue, BLOSUM62	2.4	Sequence
Jenson-Shannon Secondary Structure	0.01	Structure
FATCAT Structural Alignment P-value	0.01	Structure

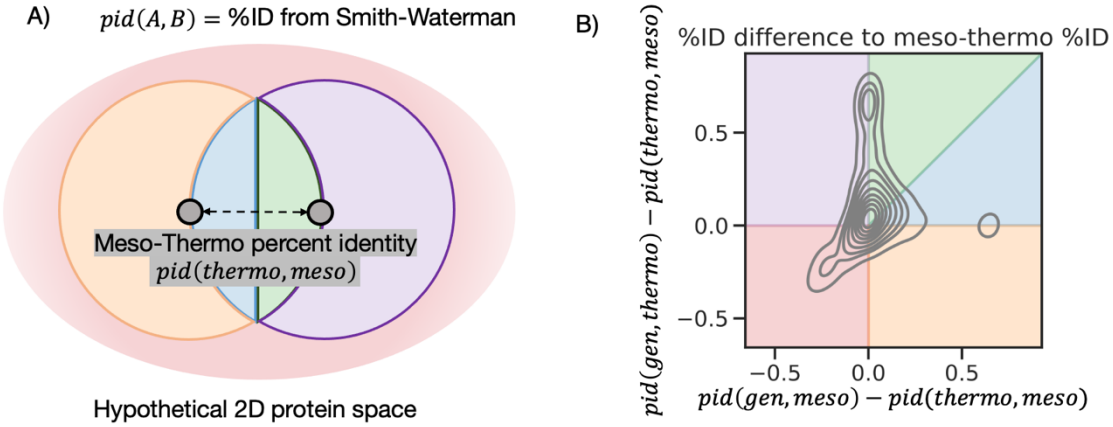

Figure 3-12: Percent identity between test set generated sequences and both the corresponding thermophilic label and mesophilic input. A) the hypothetical 2D protein space depicting the mesophilic and thermophilic proteins in a pair, where traversing the space is dis-identity of the aligned sequence. The meso-thermo percent identity is depicted as dashed line. In green, the translated protein is closer to both the mesophilic and thermophilic protein than they are to each other, and it is closest to the thermophilic sequence. Blue is the same description as green except close to the mesophilic sequence than the thermophilic one. These sectors are similar to the model “interpolating” the sequences. In purple, the generated sequence is similar to the desired thermophilic sequence, but it is more distant from the mesophilic sequence than the protein pair is to each other. This is akin to an extrapolation. Orange is also “extrapolative” but the translated sequence is similar to the input sequence. In red, predictions are further from both the mesophilic and thermophilic sequences than they are from each other. B) The additional percent identity (positive is more identical) of the generated sequence to the mesophilic (x-axis) and thermophilic (y-axis) over the percent identity of the meso-thermo pair. Color corresponds to positions in A).

Known thermophilic protein attributes are captured by NOMELT

The community has been searching through the pool of thermophilic proteins for years in order to determine the rules and modes of high temperature stability.[29,308] While the general consensus is that there is no universal set of rules, with high temperature stability being highly sequence-dependent, a few notable observations are widely accepted. Firstly, it is known that thermophiles leverage a different distribution of amino acids than do mesophiles.[372] The distributions of amino acids for model-generated sequences was measured, e.g. the proportion over all residues in the data of each amino acid. In Figure 3-13, we can see the change in AA prevalence between thermophilic and mesophilic for model-generated sequences followed the previously reported values in direction and magnitude. Note that the literature values are derived from only 16 proteomes, and the data and model observations differed in magnitude for some amino acids.

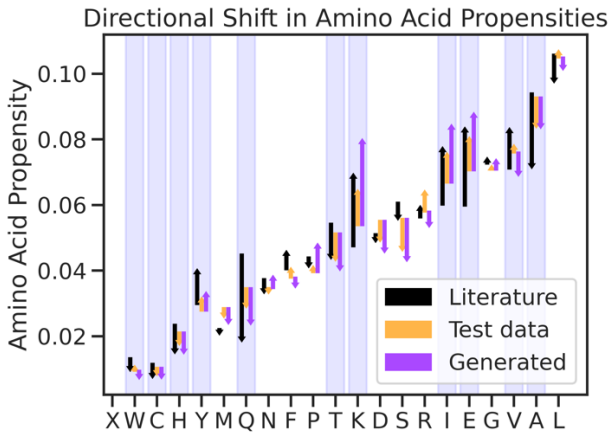

Figure 3-13: Change in Amino Acids frequencies between mesophilic proteins and thermophilic ones. In black, values derived from 16 proteomes in literature.[372] In orange, the test set of the protein pair data, and in purple, generated sequences by the model. Statistically significant shifts determined in literature are highlighted in blue. For almost all significant amino acids, the data has about the same direction in shift as identified from reference proteomes, with most being the correct order of magnitude. The

generated sequences recapitulated the distribution observed in the data.

Next, it is accepted that thermophiles tend to leverage disulfide bonds to a greater extent than do mesophiles.[302] The model outputs were probed for the predicted likelihood (Eq. 0-4) of cysteine to determine if it understands the importance of such bonds for high temperature stability. It was found that the model is significantly more likely to place a cysteine in the sequence to complete a disulfide bond than to place one that does not form a bond (P-value < $8.8e^{-22}$), as seen in Figure 3-14. Bonded and non-bonded cysteine were determined by identifying disulfide bonds in thermophilic and generated sequences using the ESMFold prediction of structure and a heuristic Cα threshold length of 7.5 Å.[373]

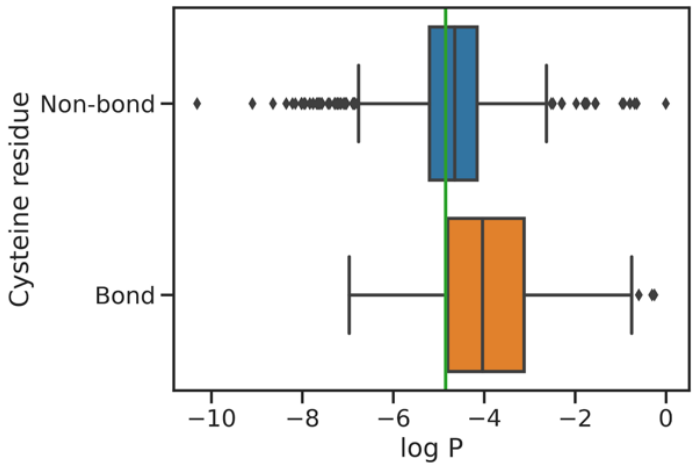

Figure 3-14: Model predicted log likelihood of Cysteine residues for positions that would or wouldn't form a disulfide bond with another cysteine in the predicted structure (C_α distance <7.5Å[373]). In green, the log likelihood of an amino acid assuming random uniform. For residues where the Cysteine would not form a bond with an existing Cysteine, the model has little Cysteine bias, sometimes predicting Cysteine with high probability and other times choosing different amino acids. For residue positions that would form a disulfide bond, the model has a very heavy bias towards predicting Cysteine.

Strictly speaking, the model was trained to convert proteins to look more like thermophilic variants, which is not a direct measure of high temperature stability. To probe the stability, the recently developed mAF-min method was adapted by normalizing by

sequence length. See Section "**Protein Structure and Thermal Stability Estimation**" for details. For the test set examples, the estimated stability of the ground truth thermophilic homolog and the model translation was compared to the prediction for the mesophilic protein. Both were estimated to be more stable than the mesophilic protein on average with statistical significance, and the generated sequence scored even better than ground truth. This indicates that the model was successful at learning aspects of stability from thermophilic proteins and may be combining stabilizing strategies from diverse sequences in ways that were not occurring in individual test examples, contingent on the mAF-min method being qualitative for these proteins. See Figure 3-15, below.

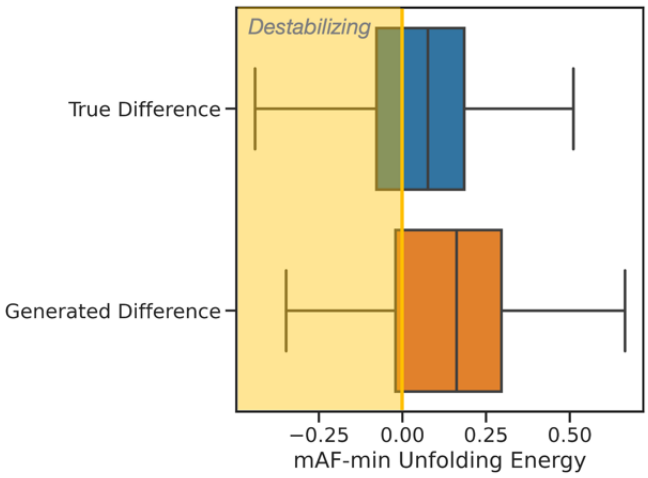

Figure 3-15: Shift in stability from mesophilic protein, according to the mAF-min method. Ground truth thermophilic sequences, were stabilizing on average, with 56% of examples in the dataset having >95% confidence of having a high folding free energy change. The sequences generated by the model on the test also set capture increased stability, with 72% meeting the >95% interval.

NOMELT as an Engineer

The purpose of training NOMELT was to fill a unique niche for accelerating protein engineering: a generative and generalized designer of protein sequence that specifically targets high temperature stability. The process of developing a dataset and training a neural machine translator of mesophilic to thermophilic proteins was strictly an attempt to fill this niche. To evaluate this, Engrailed Homeodomain (En-HD) is considered as a test case.[374] This small transcription factor has been extensively studied in the past: it is extremely fast to simulate in molecular dynamics simulations, which have been used to estimate the thermal stability of this protein, and it has previously been engineered to >98°C melting temperature from the wild type 56 °C.[154] A depiction of the protein is given in Figure 3-16. Note that there are no homologs of this eukaryotic protein in the NOMELT training dataset; a BLASTp search was conducted for En-HD on the dataset set and the best hit had an E-value of 1.6 and a coverage of 40% (See Appendix C.2.1 for BLAST parameters).

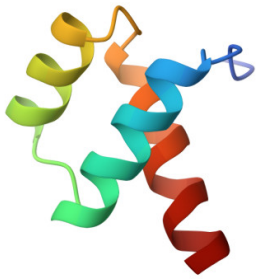

Figure 3-16: Engrailed Homeodomain (En-HD) 3D structure.[374]

NOMELT was given En-HD as a mesophilic input and a BEAM search with temperature 1.0 and 10 beams produced an output sequence with 14 suggested mutations

(insertions and deletions included) when aligned with the wild type protein using a BLOSUM62 Smith-Waterman alignment. The parameters for this alignment can be found in Table C.2.1 in the Appendix. The raw model output was about the same stability as the wild type according to mAF-min (See "**Protein Structure and Thermal Stability Estimation**") with 2.36 and 2.34 respectively (more positive is more stable). For reference, a previously engineered En-HD variant engineered by consensus over many homologs has a score of 2.58, which is much more stable according to the estimator.[375] While the raw model output did not confer a significant estimated increase in stability, NOMELT should be viewed as a generator of variation rather than a single-pass engineer, given that the model may be introducing variation picked up from nature that does not directly impact thermal stability. This yielded a library size of 16,384 considering 14 binary mutations to search over.

Ten rounds of NSGA-ii evolutionary optimization was conducted each with a population size of 10 using the mAF-min method as an objective function. This is an *in silico* analogy to creating a recombination library of the wild type and the NOMELT engineered variants. These rounds explore only 0.6% of the library, but are able to improve the stability score of En-HD to 2.51, or 71% compared to that of the previously engineered variant, using NOMELT suggested mutations. Note that NOMELT requires only a single input sequence, unlike the consensus variant which required a large MSA of homologs. To ground this result, the process was repeated considering 14 random mutations on the protein, which are unable to improve the stability over the wild type with statistical significance. See Figure 3-17.

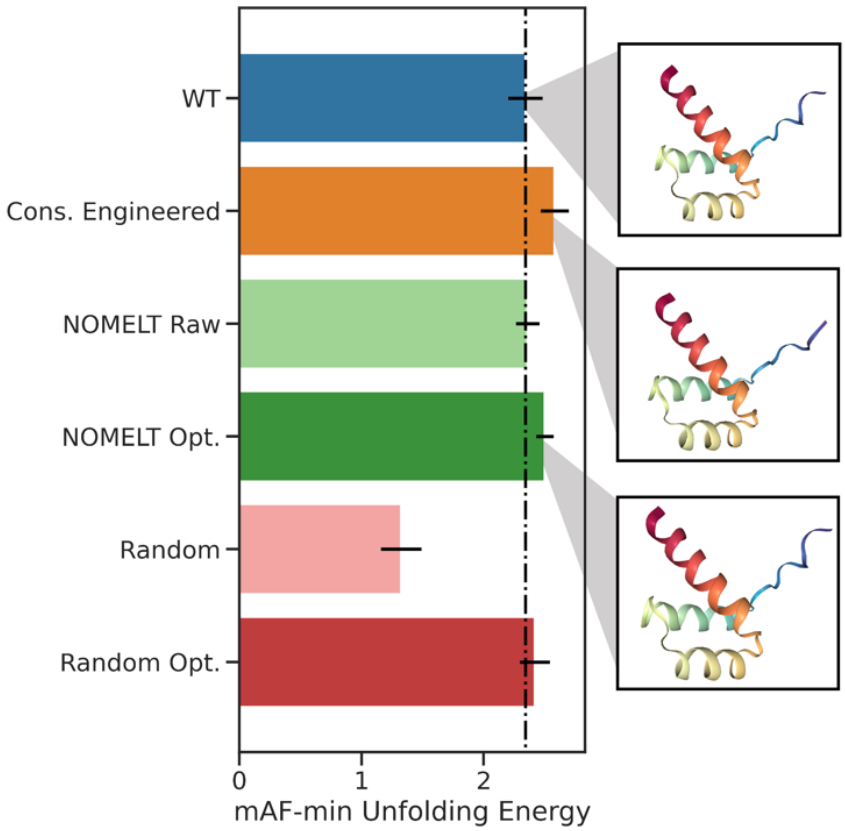

Figure 3-17: Comparison of stability of various designs according to the mAF-min method's unfolding free energy change. Error bars indicate 95% CI over the ensemble of AlphaFold structures. Vertical black line is the wild type's score. The previously engineered variant, by consensus over many homologs (orange) is more stable according to the estimator (P-value $5.9e^{-18}$). The raw output of the NOMELT model given a single input wild type sequence (light green), which includes 14 mutations involving some insertions, is not stabilizing or destabilizing. After only 100 examples from the search space of 2^{14} mutation permutations have been explored using NSGA-ii (dark green), we achieve a statistically more stable variant than WT (P-value 9.5e-14). Introducing the same number of mutations randomly is extremely destabilizing (light red), and running the same number of exploration steps over the random search space cannot improve the protein over wild type.

Of course, this conferred improvement in stability is predicated on the accuracy of the mAF-min method. To further probe these results, molecular dynamics were run on En-HD variants, which has been done extensively to study the stability of this particular protein.[154,376] See section "Prediction of protein stability." Five replicates of 1 microsecond dynamics were run at a number of temperatures for each of the wild type protein, the NOMELT optimized variant, and a previously engineered variant to >98 °C.[154,374] The

RMSD over each simulation was averaged, yielding a distribution of 5 values per protein per temperature. Note that the starting structure (prior to equilibration) for the NOMELT variant is an AF2 predicted structure. In Figure 3-18, the average RMSD relative to 298K is given as a function of temperature. We can see that the literature engineered variant, "UVF," retains a tight distribution of displacement around its starting structure up to 370K, while the wild type protein begins to open up at its melting temperature of 56 ^{o}C, indicated by a tight distribution of RMSD values widening and increase in magnitude discontinuously. The NOMELT variant retains its structure until at least 66 ^{o}C before doing the same. Expression of the En-HD variants and either melting temperature experiments, or ideally a functional at elevated temperature are required to be fully certain that the NOMELT variant is indeed stabilized. One could measure DNA binding of the variants at elevated temperatures using e.g. DNase protection.[377] The method should additionally be tested on other protein, such as enzymes. Chemiluminescent enzymes such as Nano-luciferase would be a case study worth perusing giving the ease of downstream colorimetric assays.[378]

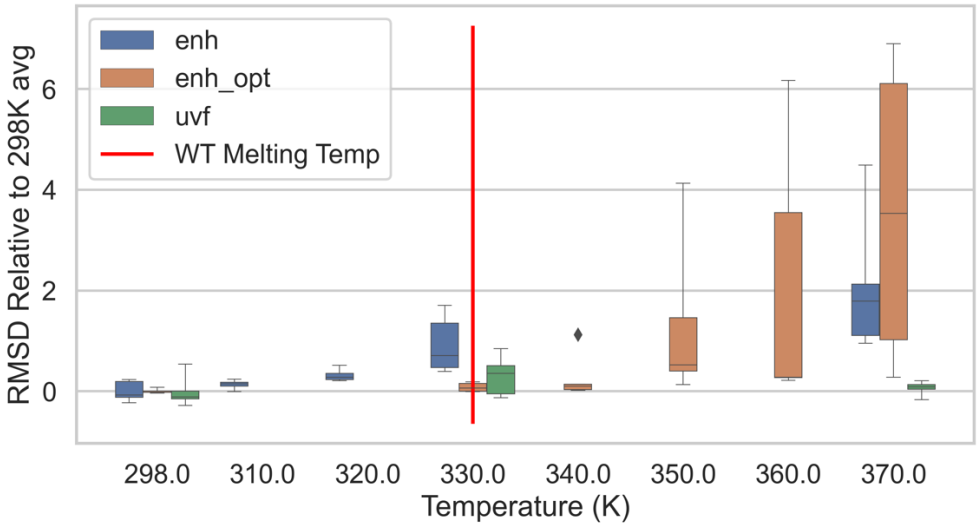

Figure 3-18: RMSD over 1 microsecond dynamics simulations of En-HD variants. RMSD values are relative to the average value of the RMSD of the variant over the 296K simulations. Distributions are over 5 independent simulations. The UVF variant, experimentally determined to have >98°C melting temperature, remains tightly bound to its initial structure at all temperatures. The wild type protein opens up from its starting structure at its melting temperature, while the NOMELT variant retains its structure at least another 10K.

In order to validate against experimental data until engineered variants can be probed *in vitro*, consider an impactful application for machine learning: generalized performance predictors to help filter DE mutagenesis libraries.[318] While this strategy is not generative like the niche NOMELT aims to fill, meaning that the improvement of protein fitness is dependent on the variation present in the library, mutagenesis libraries are more experimentally practical than creating libraries of specifically engineered sequences. For this reason, NOMELT was also tested as a predictor without any further finetuning. Here, the wild type sequence is given to the encoder of the model, and a variant (insertions and deletions are allowed) is given to the decoder. The variant is scored using the decoder output logits according to Eq. 0-5. NOMELT was used to rank LovD and LipA variants, and the model scores compared to experimentally determined melting temperatures.[148,379]

The model was extremely predictive zero-shot for LovD and somewhat for LipA, without any training for melting temperature prediction, and without ever having seen the proteins before (No hits according to BLASTp, parameters in Appendix C.2.1), as seen in Figure 3-19. The model overestimated the wildtype for LipA variants, but is able to rank longer distance variants with temperatures >65 ºC. It is believed that this is due to the high wild type temperature, leading to the decoder ranking it quite high. Additionally, the model is making judgements based on the similarity of the protein to its learned thermophilic distribution, which is conflated with a number of factors other than temperature.

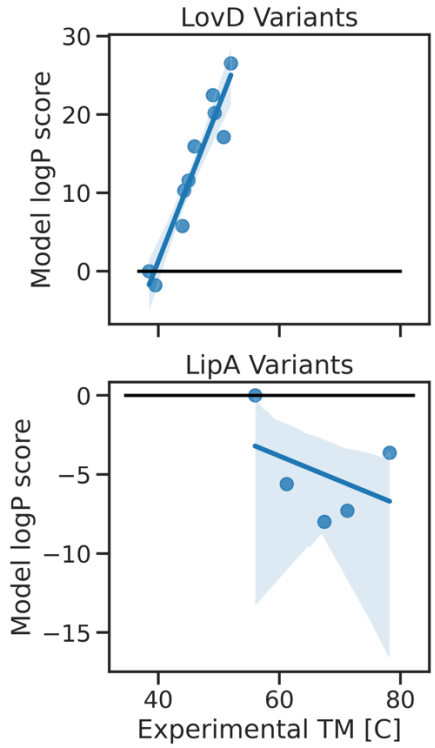

Figure 3-19: Experimental melting temperature of LovD and LipA variants vs NOMELT log prob scores according to Eq. 0-5. The model is extremely qualitative at ranking LovD melting temperatures with a Spearman's R of 0.94 (P value $5.5e^{-5}$). The model is poorer at scoring LipA variants, though qualitatively ranks the three variants with the highest temperature.

Lipase A has also been the subject of a deep mutational scan (DMS) where all single substitutions at a set of positions are explored.[380] This dataset serves as a benchmark dataset for zero-shot estimators in ProteinGym, the only one specifically focused on thermal stability, measuring catalytic half activation temperature.[88] When NOMELT is used to rank the 2,172 variants in the dataset, a Spearman's correlation of 0.32 is achieved. Note that this is equivalent to using ESM zero-shot, which was trained on a significantly larger and more diverse set of proteins self-supervised. This is state of the art discounting models that require MSAs as input (E.g. Tranception, MSATransformer), which can achieve up to ~0.43. A parity of NOMELT's performance is given in Figure 3-20. The high performance of NOMELT compared to other single-sequence models trained on many more data indicates that the training methodology of learning a thermophilic translation is more suited for thermal stability estimation than fully self-supervised learning over all sequences regardless of temperature. Note that the assay in this dataset is somewhat noisy as indicated by measurements over replicates, which when combined with only minor changes in measured value from single point mutations means that many variants are statistically indistinguishable. This is not accounted for by ProteinGym benchmarking. When we consider only the 34% (N=1.7 mil) of protein variant comparisons that are distinguishable from each other according to a T-test with a P-value of <0.05, NOMELT qualitatively distinguishes them 64% of the time. While this is not enough resolution to identify a global optimum in a variant library, it is significantly better than random even for a DMS library with small (~2 ^{o}C) changes in melting temperature.

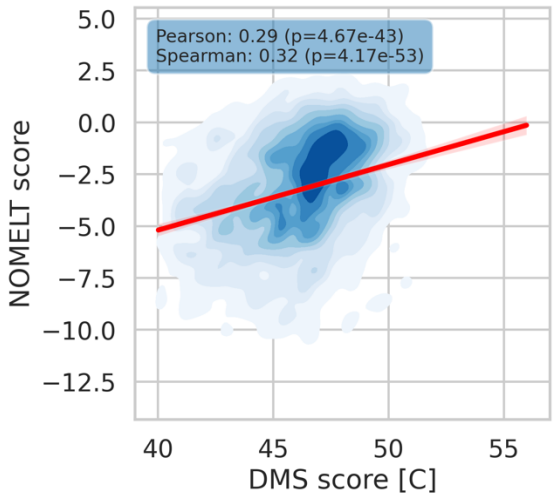

Figure 3-20: Parity of zero-shot prediction on single point DMS variants for catalytic T_{50} in *Bacillus subtilis* LipA. Red line indicates a regression.

Conclusions

A novel generative protein model that specifically targets high temperature stability, trained as a neural machine translator between thermophilicity domains, was detailed. To support the large scale training of the model, a new dataset of homologous protein pairs across mesophilic-thermophilic, orders of magnitude larger than anything in the literature, was developed by joining and searching over proteomic and evolutionary datasets. The dataset covers much of the known protein space, however with notable gaps. The dataset was rigorously preprocessed by incorporating a large window in OGT between organisms, and stringent coverage requirements, and then used to train NOMELT. The model was able to do better than natural variation over thermophilic homologs at generating thermophilic sequences from mesophilic inputs, with the advantage of not requiring multiple sequence alignment over thermophilic sequences. This allows the model to function on novel sequences for which we do not have examples of high temperature variants. Variants

generated by the model closely matched ground truth thermophilic sequences both according to percent identity and according to 3D predicted structure. Further, the model captured known thermophilic attributes, like shifts in amino acid propensities and targeted use of disulfide bonds. According to a recent state of the art qualitative estimator of protein thermal stability, generated variants were more stable on average than the ground truth, indicating that the model is incorporating stabilization techniques from diverse proteins into each example. The model was also evaluated as an engineer of novel protein targets that have very little similarity to anything seen during training. The model is able to improve the stability of En-HD by directly introducing mutations including insertions, according to an estimator of stability and MD simulations of unfolding events. Experimental characterization of the design to confirm increase in stability is necessary, and repeating the NOMELT design process for a protein easier to characterize in lab is recommended, such as small chemiluminescent enzymes. Lastly, the model was evaluated for situations when direct synthesis of a new sequence is not experimentally feasible, and instead a library of variants need to be ranked for targeted experimental evaluation. The output logits of NOMELT specifically target high thermal stability, and has zero shot correlation with protein melting temperature for multiple proteins completely unlike training examples. It is not accurate enough to identify a global maximum among a variant library with high confidence, alike to other existing zero-shot predictors, but it does not require an MSA input nor was it trained at evolutionary scale. This indicates that the strategy of using thermophilicity as a target is advantageous over training language models over natural ambient variation.[88]

The most notable drawback of the model is the observed dependency of model loss on nearby examples seen in training. While the model showed promise for accelerating design even for completely unseen sequences, the model was on average worse than natural thermophilic variation when the protein target was <50% identity with training examples. This suggests a clear next direction for this work: increasing the evolutionary information within the models training data. learn2thermDB is limited in protein space covered compared to ESM Atlas, as was shown in Figure 3-4-right, likely due to the requirement of finding a homolog across temperature using pairwise alignment. As a reminder, only 4 mil out of 24 mil proteins in learn2thermDB contribute to protein pairs. These gaps in protein space are further exacerbated by the stringent filtering used to produce NOMELT's training set (Table 3-6). In Figure 3-21, the fraction of learn2thermDB protein space covered by the training dataset is shown. Considering all of known protein space, the training dataset covers very little.

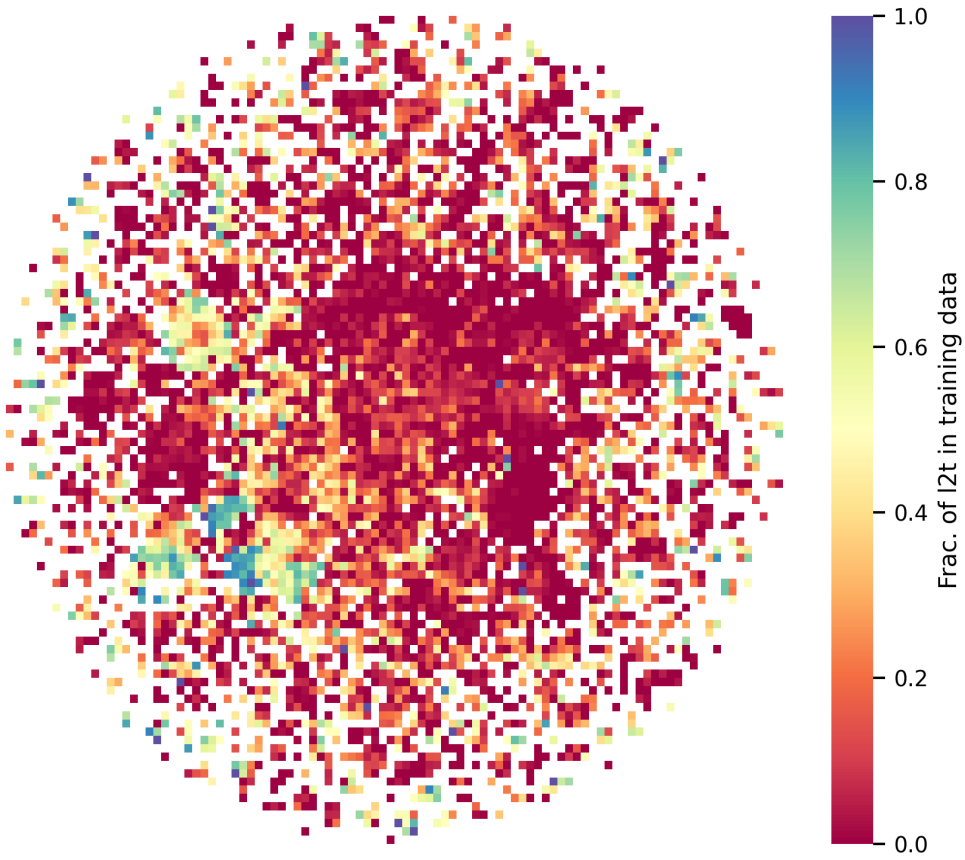

Figure 3-21: Protein space covered by the training set, as compared to all of learn2thermDB. White indicates no data. Very few portions of this space have extensive coverage, with a median coverage of 7%.

An important recent observation within the literature is the effect of evolutionary scale on protein language models.[50,78,381] It is critical that NOMELT's dataset be expanded to this scale, and the gaps in protein space filled. This should be done by utilizing more sensitive homolog searches using HMMs instead of pairwise alignment. As discussed previously, the compute resources necessary to create MSAs for the 24 million proteins is extreme, but researchers are DeepMind have created a set of 64 million MSAs upon 2 bil proteins, effectively the sum of human's knowledge of protein space.[91,382] These HMMs could be used to label segments of proteins within learn2thermDB. This should label most, if not all, of learn2therm, and meso-thermo segment pairs could be identified across much further evolutionary distance.

In addition to using more sensitive pairs for training NOMELT at a larger scale, the need for homologous pairing could be removed altogether by conditioning the generator on organism growth temperature instead of on a low temperature sequence. Specifically, all HMM labeled sequences in learn2thermDB should be cropped to their coverage. Evolutionary information is then given directly to the model *a la* AF2 by using an MSA transformer encoder.[341] OGT is added to the embeddings of the MSA by concatenation and linear projection. The embeddings are given to a traditional or bidirectional decoder which must generate the target sequence belonging to that MSA at that temperature. This extends the framework to the scale of all OGT labeled organisms as opposed to only proteins with close homologs across temperature, and further incorporates metagenomic information through the MSA inputs.

CONCLUSIONS AND FINAL REMARKS

Data driven methods for modeling systems of interest in science and engineering are becoming increasingly necessary to drive progress in an age of massive computation and data flux. While there are best practices and learnings from past applications that should be leveraged to produce an effective model, each disparate use case requires data and model engineering. Even in the subspace of systems involving molecules, there are a breadth of strategies to preprocess and encode data, and a number of deep learning architectures available. Many of these strategies are summarized here. The process of designing deep learning solutions from the ground up for three unique applications are detailed. In each, no existing data was available, and novel datasets were created. Decision making during data acquisition and preprocessing with the goal of creating robust models that address the specific application goals are explained. Model architectures are detailed and the effect of hyperparameters analyzed. The desired impact of the models on the community and next steps are posited.

In Chapter 1, two DNN applications that help predict thermal reaction rate constants while significantly reducing the cost compared to experiment or *ab initio* calculations are detailed. The first, unnamed, successfully recovers some amount of quantum affects to the rate constant associated with a 1D minimum reaction potential energy barrier on a held out test set and additional blind examples, thus reducing the cost of incorporating quantum affects like tunneling down to that of a TST rate constant. The model uses hand engineered features associated with the barrier and a novel training dataset of 6.9 million hypothetical energy barriers. This suggests that the complete relationship between the adiabatic 1D reaction rate constant and the 1D potential energy barrier associated with it could be learned

by neural networks, and modern learned representations based directly on the barrier coordinates e.g. using a CNN might further increase accuracy. Classical rate constants calculated with TST still incur significant expense from MEP searches, which the second DNN application helps to avoid. Partition functions were computed for a dataset of 11k unimolecular bond breaking reactions. This dataset was used to train a DNN on atom distance bins (EncodedBonds) as a 3D molecular embedding. The Qest model was able to predict temperature dependent partition functions given a molecular structure and temperature, and the QesTS model could predict the partition function of an unknown transition state given reactant and product information. These models in conjunction remove the need to search for a transitions state and calculate any partition functions, when combined with activation energy predictors. Again, learned representations may be a method for improving accuracy, and it may be possible to remove the need for predicting partition functions and activation energies altogether by training a predictor of temperature dependent rate constants.

In Chapter 3, a very large transformer-based model was created to treat protein thermal stability engineering as a neural machine translation problem. The model fills a yet unfilled niche within protein engineering deep learning applications: generative protein engineers that specifically targets high temperature stability. Labels for thermostability do not exist in diverse and large quantities, so thermophilicity was used as a proxy. Here, a novel dataset of protein homologs across ambient and high temperature was produced, orders of magnitude larger than anything before it. While the new dataset contains only prokaryotic proteins, large portions of protein space are covered. NOMELT was trained using a subset of this data to convert a mesophilic protein into a thermophilic variant, and

showed better performance at recovering thermophilic sequences than if one were to build an MSA over all known thermophilic homologs and select the consensus residues, yet uses only a single mesophilic sequence as input. Further, sequences generated by the model aligned well at the primary and (predicted) tertiary structure level to ground truth thermophilic variants. According to a 3rd party estimator of thermal stability that was shown to be qualitative, model designs conferred even more stability over the mesophilic input than do the thermophilic ground truths. NOMELT also captured known thermophilic attributes, e.g. a shift in amino acid distribution and a focus on disulfide bonds. The model was also used to engineer a more stable variant of Engrailed Homeodomain, that showed a reduced propensity to unfold at 56 ^{o}C, the wild type's melting temperature. Further, NOMELT can act as a zero-shot predictor of thermal stability. Unlike self-supervised MSA based or evolutionary scale sequence models, which show capability to help predict protein activity, NOMELT ranks proteins by *thermal stability* while never being trained to specifically predict thermal stability. It does so with a Spearman's correlation of 0.94 on 19 LovD variants with experimental melting temperatures, and is qualitative in ranking some variants of LipA. When evaluated on a benchmarking dataset of a deep mutational scan on LipA with a measurement of temperature of half activation, the model scored 0.32 zero-shot, which is state of the art for its model class. While NOMELT can introduce high temperature targeting variation into protein sequences, the work here suggests it is still limited by the scale of the training dataset. To attain evolutionary scale for generative temperature stability modeling, it is suggested to train a new model conditioned on a target sequence's MSA and a temperature, which recovers a variant of that sequence that would occur in an organism at that temperature by causal language modeling.

In Chapter 2, an optional learning event is detailed that aims to give Chemical Engineering students the opportunity to learn and apply coding, data science, and machine learning to problems relevant to their industry, while curriculum catches up. The event, C-HACK, was a coopetition style hackathon consisting of a week of tutorials tailored to provide content useful to address a surprise chemical engineering problem. Students then had one week to work in small teams to address the stated goal using data science. A panel of independent judges ranked team solutions and the top teams received prizes and awards for their professional development. The event was hosted remotely such that multiple Chemical Engineering campuses could participate. Critically, industry partners participated in the event from conception to administration such that it was always centered on skills relevant to students, and gave them an opportunity to interact. While an improvement in team soft skills sought after in industry was not observed, a significant improvement in coding and data science was observed as measured by self-administered survey. The tutorials and the problem give students, regardless of their starting skill level, experience with the choices and techniques used in data science, directly on relevant material. Given the low monetary and time cost with administering the event, it is recommended as an option for Chemical Engineering departments that do not yet have modern data science and machine learning options/curriculum.

Data science and machine learning is not one size fits all, even within just Chemical Engineering and systems involving molecules. Each application requires unique decision making regarding data acquisition, preprocessing, and representation in order to best address a specific goal. Further, this nuance should be tracked through to learning

experience in ChE, where the application space differs from canonical computer science applications.

REFERENCES

1. Dean, J., Patterson, D. & Young, C. A New Golden Age in Computer Architecture: Empowering the Machine-Learning Revolution. *IEEE Micro* **38**, 21–29 (2018).
2. Beck, D. A. C., Carothers, J. M., Subramanian, V. R. & Pfaendtner, J. Data Science: Accelerating Innovation and Discovery in Chemical Engineering. (2016) doi:10.1002/aic.15192.
3. Alshehri, A. S. & You, F. Paradigm Shift: The Promise of Deep Learning in Molecular Systems Engineering and Design. *Front. Chem. Eng.* **3**, (2021).
4. Bender, A. *et al.* Evaluation guidelines for machine learning tools in the chemical sciences. *Nat. Rev. Chem.* **6**, 428–442 (2022).
5. Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O. & Walsh, A. Machine learning for molecular and materials science. *Nat. 2018 5597715* **559**, 547–555 (2018).
6. Karthikeyan, A. & Priyakumar, U. D. Artificial intelligence: machine learning for chemical sciences. *J. Chem. Sci.* **134**, 2 (2021).
7. Komp, E., Janulaitis, N. & Valleau, S. Progress towards machine learning reaction rate constants. *Phys. Chem. Chem. Phys.* **24**, 2692–2705 (2022).
8. Modarres, H. P., Mofrad, M. R. & Sanati-Nezhad, A. Protein thermostability engineering. *RSC Adv.* **6**, 115252–115270 (2016).
9. Yu, Y., Li, M., Liu, L., Li, Y. & Wang, J. Clinical big data and deep learning: Applications, challenges, and future outlooks. *Big Data Min. Anal.* **2**, 288–305 (2019).
10. Jiang, W. Applications of deep learning in stock market prediction: Recent progress. *Expert Syst. Appl.* **184**, 115537 (2021).
11. Sreenu, G. & Saleem Durai, M. A. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. *J. Big Data* **6**, 48 (2019).
12. Gharibshah, Z., Zhu, X., Hainline, A. & Conway, M. Deep Learning for User Interest and Response Prediction in Online Display Advertising. *Data Sci. Eng.* **5**, 12–26 (2020).
13. Danishuddin & Khan, A. U. Descriptors and their selection methods in QSAR analysis: paradigm for drug design. *Drug Discov. Today* **21**, 1291–1302 (2016).
14. Chiang, L. H., Braun, B., Wang, Z. & Castillo, I. Towards artificial intelligence at scale in the chemical industry. *AIChE J.* **68**, e17644.
15. Diez-Olivan, A., Del Ser, J., Galar, D. & Sierra, B. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0. *Inf. Fusion* **50**, 92–111 (2019).
16. Ramprasad, R., Batra, R., Pilania, G., Mannodi-Kanakkithodi, A. & Kim, C. Machine learning in materials informatics: Recent applications and prospects. *Npj Comput. Mater.* **3**, 54–54 (2017).
17. Wei, J. *et al.* Machine learning in materials science. *InfoMat* **1**, 338–358 (2019).
18. Ye, W. *et al.* Harnessing the Materials Project for machine-learning and accelerated discovery. *MRS Bull.* **43**, 664–669 (2018).
19. Cova, T. F. G. G. & Pais, A. A. C. C. Deep Learning for Deep Chemistry: Optimizing the Prediction of Chemical Patterns. *Front. Chem.* **7**, (2019).

20. Haghighatlari, M. *et al.* Learning to Make Chemical Predictions: The Interplay of Feature Representation, Data, and Machine Learning Methods. *Chem* **6**, 1527–1542 (2020).
21. Simm, J. *et al.* Splitting chemical structure data sets for federated privacy-preserving machine learning. *J. Cheminformatics* **13**, 96 (2021).
22. Komp, E. & Valleau, S. Machine Learning Quantum Reaction Rate Constants. *J. Phys. Chem. A* **124**, 8607–8613 (2020).
23. Wakelam, V. *et al.* Reaction networks for interstellar chemical modelling: Improvements and challenges. *Space Sci. Rev.* **156**, 13–72 (2010).
24. Ji, L. *et al.* Community Reaction Network Reduction for Constructing a Coarse-Grained Representation of Combustion Reaction Mechanisms. *J. Chem. Inf. Model.* **62**, 2352–2364 (2022).
25. Lakshmanan, A. & Biegler, L. T. Synthesis of Optimal Chemical Reactor Networks. *Ind. Eng. Chem. Res.* **35**, 1344–1353 (1996).
26. Komp, E. & Valleau, S. Low-cost prediction of molecular and transition state partition functions via machine learning. *Chem. Sci.* **13**, 7900–7906 (2022).
27. Komp, E. A. *et al.* C-HACK: A Data Science Tutorial and Hackathon for Undergraduate Students in Chemical Engineering. *Chem. Eng. Educ.* **57**, 17–26 (2023).
28. Komp, E. *et al.* Homologous Pairs of Low and High Temperature Originating Proteins Spanning the Known Prokaryotic Universe. *Sci. Data* **10**, 682 (2023).
29. Pucci, F. & Rooman, M. Physical and molecular bases of protein thermal stability and cold adaptation. *Curr. Opin. Struct. Biol.* **42**, 117–128 (2017).
30. Pan, X. & Kortemme, T. Recent advances in de novo protein design: Principles, methods, and applications. *J. Biol. Chem.* **296**, (2021).
31. Ramchoun, H., Ghanou, Y., Ettaouil, M. & Janati Idrissi, M. A. Multilayer Perceptron: Architecture Optimization and Training. (2016) doi:10.9781/ijimai.2016.415.
32. DeVore, R., Hanin, B. & Petrova, G. Neural network approximation. *Acta Numer.* **30**, 327–444 (2021).
33. Strubell, E., Ganesh, A. & McCallum, A. Energy and policy considerations for modern deep learning research. in *Proceedings of the AAAI Conference on Artificial Intelligence* vol. 34 13693–13696 (2020).
34. Livni, R., Shalev-Shwartz, S. & Shamir, O. On the Computational Efficiency of Training Neural Networks. in *Advances in Neural Information Processing Systems* vol. 27 (Curran Associates, Inc., 2014).
35. Yang, L. & Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **415**, 295–316 (2020).
36. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.* doi:10.1145/3292500.
37. Vishwakarma, G., Haghighatlari, M. & Hachmann, J. Towards Autonomous Machine Learning in Chemistry via Evolutionary Algorithms. (2019) doi:10.26434/CHEMRXIV.9782387.V1.
38. CodeCarbon. *Estimation of Computation Carbon Cost* https://codecarbon.io/.

39. Selvan, R., Bhagwat, N., Wolff Anthony, L. F., Kanding, B. & Dam, E. B. Carbon Footprint of Selecting and Training Deep Learning Models for Medical Image Analysis. in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022* (eds. Wang, L., Dou, Q., Fletcher, P. T., Speidel, S. & Li, S.) 506–516 (Springer Nature Switzerland, 2022). doi:10.1007/978-3-031-16443-9_49.
40. Budennyy, S. A. *et al.* eco2AI: Carbon Emissions Tracking of Machine Learning Models as the First Step Towards Sustainable AI. *Dokl. Math.* **106**, S118–S128 (2022).
41. Elnaggar, A. *et al.* ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7112–7127 (2022).
42. Alfassy, A. *et al.* FETA: Towards Specializing Foundational Models for Expert Task Applications. *Adv. Neural Inf. Process. Syst.* **35**, 29873–29888 (2022).
43. Perone, C. S., Silveira, R. & Paula, T. S. Evaluation of sentence embeddings in downstream and linguistic probing tasks. Preprint at https://doi.org/10.48550/arXiv.1806.06259 (2018).
44. Wei, J. *et al.* Finetuned Language Models Are Zero-Shot Learners. Preprint at https://doi.org/10.48550/arXiv.2109.01652 (2022).
45. Kojima, T., Gu, S. (Shane), Reid, M., Matsuo, Y. & Iwasawa, Y. Large Language Models are Zero-Shot Reasoners. *Adv. Neural Inf. Process. Syst.* **35**, 22199–22213 (2022).
46. Dollar, O. W., Horawalavithana, S., Vasquez, S., Pfaendtner, W. J. & Volkova, S. MolJET: Multimodal Joint Embedding Transformer for Conditional de novo Molecular Design and Multi-Property Optimization. in (2022).
47. Li, P. *et al.* An effective self-supervised framework for learning expressive molecular global representations to drug discovery. *Brief. Bioinform.* **22**, bbab109 (2021).
48. Chithrananda, S., Grand, G. & Ramsundar, B. ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction. Preprint at http://arxiv.org/abs/2010.09885 (2020).
49. Dollar, O., Joshi, N., Pfaendtner, J. & Beck, D. A. C. Efficient 3D Molecular Design with an E(3) Invariant Transformer VAE. *J. Phys. Chem. A* **127**, 7844–7852 (2023).
50. Lin, Z. *et al.* Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
51. Chen, C. (Sam), Zhou, J., Wang, F., Liu, X. & Dou, D. Structure-aware protein self-supervised learning. *Bioinformatics* **39**, btad189 (2023).
52. Wu, Z. *et al.* MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* **9**, 513–530 (2018).
53. Jeon, J., Kang, S. & Uk Kim, H. Predicting biochemical and physiological effects of natural products from molecular structures using machine learning. *Nat. Prod. Rep.* **38**, 1954–1966 (2021).
54. Gierlich, C. & Palkovits, S. Featurizing chemistry for machine learning — methods and a coded example. *Curr. Opin. Chem. Eng.* **37**, 100840 (2022).
55. Landrum, G. RDKit: Open-source cheminformatics.
56. Wigh, D. S., Gooman, J. M. & Lapkin, A. A review of molecular representation in the age of machine learning. **12**, e1603.

57. Chuang, K. V., Gunsalus, L. M. & Keiser, M. J. Learning Molecular Representations for Medicinal Chemistry. *J. Med. Chem.* **63**, 8705–8722 (2020).
58. Jo, J., Kwak, B., Lee, B. & Yoon, S. Flexible Dual-Branched Message-Passing Neural Network for a Molecular Property Prediction. *ACS Omega* **17**, acsomega.1c05877-acsomega.1c05877 (2022).
59. Withnall, M., Lindelöf, E., Engkvist, O. & Chen, H. Building attention and edge message passing neural networks for bioactivity and physical-chemical property prediction. *J Cheminform* **12**, 1–1 (2020).
60. Heid, E. & Green, W. H. Machine Learning of Reaction Properties via Learned Representations of the Condensed Graph of Reaction. *J. Chem. Inf. Model.* acs.jcim.1c00975-acs.jcim.1c00975 (2021) doi:10.1021/ACS.JCIM.1C00975.
61. Duvenaud, D. *et al.* Convolutional Networks on Graphs for Learning Molecular Fingerprints. *Adv. Neural Inf. Process. Syst.* **2015-Janua**, 2224–2232 (2015).
62. Wu, Z. *et al.* A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **32**, 4–24.
63. Krenn, M., Häse, F., Nigam, A., Friederich, P. & Aspuru-Guzik, A. Self-Referencing Embedded Strings (SELFIES): A 100% robust molecular string representation. *IOP Sci.* **1**, 045024 (2020).
64. Dollar, O., Joshi, N., Beck, D. A. C. & Pfaendtner, J. Attention-based generative models for de novo molecular design. *Chem. Sci.* **12**, 8362–8372 (2021).
65. Polykovskiy, D. *et al.* Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Front. Pharmacol.* **11**, (2020).
66. Walters, W. P. & Barzilay, R. Applications of Deep Learning in Molecule Generation and Molecular Property Prediction. *Acc. Chem. Res.* **54**, 263–270 (2021).
67. Pozdnyakov, S. N. & Ceriotti, M. Incompleteness of graph convolutional neural networks for points clouds in three dimensions. *Mach. Learn.: Sci. Technol.* **3**, 045020 (2022).
68. Batzner, S. *et al.* E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nat. Commun.* **13**, 2453 (2022).
69. Schütt, K. T. *et al.* SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. *Adv. Neural Inf. Process. Syst.* **2017-Decem**, 992–1002 (2017).
70. Rupp, M., Tkatchenko, A., Müller, K.-R. & Lilienfeld, O. A. von. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.* **108**, 058301–058301 (2012).
71. Janet, J. P. & Kulik, H. J. Resolving Transition Metal Chemical Space: Feature Selection for Machine Learning and Structure–Property Relationships. *J. Phys. Chem. A* **121**, 8939–8954 (2017).
72. Collins, C. R., Gordon, G. J., Von Lilienfeld, O. A. & Yaron, D. J. Constant size descriptors for accurate machine learning models of molecular properties. *J. Chem. Phys.* **148**, (2018).
73. Stocker, S., Csányi, G., Reuter, K. & Margraf, J. T. Machine learning in chemical reaction space. *Nat. Commun.* **11**, 1–11 (2020).
74. Meuwly, M. Machine Learning for Chemical Reactions. *Chem. Rev.* **121**, 10218–10239 (2021).

75. Ofer, D., Brandes, N. & Linial, M. The language of proteins: NLP, machine learning & protein sequences. *Comput. Struct. Biotechnol. J.* **19**, 1750–1758 (2021).
76. Qiu, Y. & Wei, G.-W. Artificial intelligence-aided protein engineering: from topological data analysis to deep protein language models. *Brief. Bioinform.* bbad289 (2023) doi:10.1093/bib/bbad289.
77. Nambiar, A. *et al.* Transforming the Language of Life: Transformer Neural Networks for Protein Prediction Tasks. in *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics* 1–8 (Association for Computing Machinery, 2020). doi:10.1145/3388440.3412467.
78. Verkuil, R. *et al.* Language models generalize beyond natural proteins. 2022.12.21.521521 Preprint at https://doi.org/10.1101/2022.12.21.521521 (2022).
79. Alpaydin, E. *Introduction to Machine Learning*. (PHI, 2014).
80. Samala, R. K., Chan, H.-P., Hadjiiski, L. & Koneru, S. Hazards of data leakage in machine learning: a study on classification of breast cancer using deep neural networks. in *SPIE Medical Imaging* vol. 11314.
81. Lo, Y. C., Rensi, S. E., Torng, W. & Altman, R. B. Machine learning in chemoinformatics and drug discovery. *Drug Discov. Today* **23**, 1538–1546 (2018).
82. Bemis, G. W. & Murcko, M. A. The Properties of Known Drugs. 1. Molecular Frameworks. *J. Med. Chem.* **39**, 2887–2893 (1996).
83. AlQuraishi, M. ProteinNet: a standardized data set for machine learning of protein structure. *BMC Bioinformatics* **20**, 311 (2019).
84. Li, G., Rabe, K. S., Nielsen, J. & Engqvist, M. K. M. Machine Learning Applied to Predicting Microorganism Growth Temperatures and Enzyme Catalytic Optima. *ACS Synth. Biol.* **8**, 1411–1420 (2019).
85. Pudžiuvelytė, I. *et al.* TemStaPro: protein thermostability prediction using sequence representations from protein language models. 2023.03.27.534365 Preprint at https://doi.org/10.1101/2023.03.27.534365 (2023).
86. Haas, J. *et al.* Continuous Automated Model EvaluatiOn (CAMEO) complementing the critical assessment of structure prediction in CASP12. *Proteins Struct. Funct. Bioinforma.* **86**, 387–398 (2018).
87. Dallago, C. *et al.* FLIP: Benchmark tasks in fitness landscape inference for proteins. (2021) doi:https://doi.org/10.1101/2021.11.09.467890.
88. Notin, P. *et al.* Tranception: Protein Fitness Prediction with Autoregressive Transformers and Inference-time Retrieval. in *Proceedings of the 39th International Conference on Machine Learning* 16990–17017 (PMLR, 2022).
89. Ramola, R., Friedberg, I. & Radivojac, P. The field of protein function prediction as viewed by different domain scientists. *Bioinforma. Adv.* **2**, vbac057 (2022).
90. Rost, B. Twilight zone of protein sequence alignments. *Protein Eng. Des. Sel.* **12**, 85–94 (1999).
91. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
92. Riesselman, A. J., Ingraham, J. B. & Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods* **15**, 816–822 (2018).
93. Strodthoff, N., Wagner, P., Wenzel, M. & Samek, W. UDSMProt: universal deep sequence models for protein classification. *Bioinformatics* **36**, 2401–2409 (2020).

94. Erickson, E. *et al.* Sourcing thermotolerant poly(ethylene terephthalate) hydrolase scaffolds from natural diversity. *Nat. Commun.* **13**, 7850 (2022).
95. Greener, J. G., Kandathil, S. M., Moffat, L. & Jones, D. T. A guide to machine learning for biologists. *Nat. Rev. Mol. Cell Biol.* **23**, 40–55 (2022).
96. Vaswani, A. *et al.* Attention Is All You Need. Preprint at https://doi.org/10.48550/arXiv.1706.03762 (2017).
97. Lin, T., Wang, Y., Liu, X. & Qiu, X. A survey of transformers. *AI Open* **3**, 111–132 (2022).
98. Min, B. *et al.* Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey. *ACM Comput. Surv.* **56**, 30:1-30:40 (2023).
99. Zhou, M., Duan, N., Liu, S. & Shum, H.-Y. Progress in Neural NLP: Modeling, Learning, and Reasoning. *Engineering* **6**, 275–290 (2020).
100. Freitag, M. & Al-Onaizan, Y. Beam Search Strategies for Neural Machine Translation. in *Proceedings of the First Workshop on Neural Machine Translation* 56–60 (2017). doi:10.18653/v1/W17-3207.
101. Colón-Ruiz, C. & Segura-Bedmar, I. Comparing deep learning architectures for sentiment analysis on drug reviews. *J. Biomed. Inform.* **110**, 103539 (2020).
102. Riesselman, A. *et al.* Accelerating Protein Design Using Autoregressive Generative Models. 757252 Preprint at https://doi.org/10.1101/757252 (2019).
103. Aminabadi, R. Y. *et al.* DeepSpeed Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. Preprint at https://doi.org/10.48550/arXiv.2207.00032 (2022).
104. Luccioni, A., Lacoste, A. & Schmidt, V. Estimating Carbon Emissions of Artificial Intelligence [Opinion]. *IEEE Technol. Soc. Mag.* **39**, 48–51 (2020).
105. Ma, W., Tang, C. & Lai, L. Specificity of Trypsin and Chymotrypsin: Loop-Motion-Controlled Dynamic Correlation as a Determinant. *Biophys. J.* **89**, 1183–1193 (2005).
106. Pearson, W. R. An Introduction to Sequence Similarity ("Homology") Searching. *Curr. Protoc. Bioinforma. Ed. Board Andreas Baxevanis Al* **0 3**, 10.1002/0471250953.bi0301s42 (2013).
107. Altschul, S. F. & Gish, W. [27] Local alignment statistics. in *Methods in Enzymology* vol. 266 460–480 (Academic Press, 1996).
108. Camacho, C. *et al.* BLAST+: architecture and applications. *BMC Bioinformatics* **10**, 421 (2009).
109. Pearson, W. R. Finding Protein and Nucleotide Similarities with FASTA. *Curr. Protoc. Bioinforma.* **53**, 3.9.1-3.925 (2016).
110. Muir, P. *et al.* The real cost of sequencing: scaling computation to keep pace with data generation. *Genome Biol.* **17**, 53 (2016).
111. Buchfink, B., Xie, C. & Huson, D. H. Fast and sensitive protein alignment using DIAMOND. *Nat. Methods* **12**, 59–60 (2015).
112. Zhang, Y., Zhang, Q., Zhou, J. & Zou, Q. A survey on the algorithm and development of multiple sequence alignment. *Brief. Bioinform.* **23**, bbac069 (2022).
113. Altschul, S. F. & Koonin, E. V. Iterated profile searches with PSI-BLAST—a tool for discovery in protein databases. *Trends Biochem. Sci.* **23**, 444–447 (1998).
114. Finn, R. D., Clements, J. & Eddy, S. R. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Res.* **39**, W29–W37 (2011).

115. Marks, D. S. *et al.* Protein 3D Structure Computed from Evolutionary Sequence Variation. *PLOS ONE* **6**, e28766 (2011).
116. Hasegawa, H. & Holm, L. Advances and pitfalls of protein structural alignment. *Curr. Opin. Struct. Biol.* **19**, 341–348 (2009).
117. Holm, L., Kääriäinen, S., Rosenström, P. & Schenkel, A. Searching protein structure databases with DaliLite v.3. *Bioinforma. Oxf. Engl.* **24**, 2780–2781 (2008).
118. Ye, Y. & Godzik, A. Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinformatics* **19**, ii246–ii255 (2003).
119. Li, Z., Jaroszewski, L., Iyer, M., Sedova, M. & Godzik, A. FATCAT 2.0: towards a better understanding of the structural diversity of proteins. *Nucleic Acids Res.* **48**, W60–W64 (2020).
120. Ahdritz, G. *et al.* OpenFold: Retraining AlphaFold2 yields new insights into its learning mechanisms and capacity for generalization. 2022.11.20.517210 Preprint at https://doi.org/10.1101/2022.11.20.517210 (2022).
121. Hornak, V. *et al.* Comparison of multiple Amber force fields and development of improved protein backbone parameters. *Proteins* **65**, 712–725 (2006).
122. Guo, H.-B. *et al.* AlphaFold2 models indicate that protein sequence determines both structure and dynamics. *Sci. Rep.* **12**, 10696 (2022).
123. Choudhuri, S. Protein Folding Problem: A Comparative Analysis of ESM2. SSRN Scholarly Paper at https://doi.org/10.2139/ssrn.4301467 (2022).
124. Kaufmann, K. W., Lemmon, G. H., DeLuca, S. L., Sheehan, J. H. & Meiler, J. Practically Useful: What the Rosetta Protein Modeling Suite Can Do for You. *Biochemistry* **49**, 2987–2998 (2010).
125. Leman, J. K. *et al.* Macromolecular modeling and design in Rosetta: recent methods and frameworks. *Nat. Methods* **17**, 665–680 (2020).
126. Huang, P. *et al.* Evaluating Protein Engineering Thermostability Prediction Tools Using an Independently Generated Dataset. *ACS Omega* **5**, 6487–6493 (2020).
127. Riera, C., Lois, S. & de la Cruz, X. Prediction of pathological mutations in proteins: the challenge of integrating sequence conservation and structure stability principles. *WIREs Comput Mol Sci,* **4**, 249–268.
128. Marabotti, A., Scafuri, B. & Facchiano, A. Predicting the stability of mutant proteins by computational approaches: an overview. *Brief. Bioinform.* **22**, bbaa074 (2021).
129. Pucci, F. & Rooman, M. Towards an accurate prediction of the thermal stability of homologous proteins. *J. Biomol. Struct. Dyn.* **34**, 1132–1142 (2016).
130. Åqvist, J., Isaksen, G. V. & Brandsdal, B. O. Computation of enzyme cold adaptation. *Nat. Rev. Chem.* **1**, 1–14 (2017).
131. Tokuriki, N. & Tawfik, D. S. Stability effects of mutations and protein evolvability. *Curr. Opin. Struct. Biol.* **19**, 596–604 (2009).
132. Peccati, F., Alunno-Rufini, S. & Jiménez-Osés, G. Accurate Prediction of Enzyme Thermostabilization with Rosetta Using AlphaFold Ensembles. *J. Chem. Inf. Model.* **63**, 898–909 (2023).
133. Schymkowitz, J. *et al.* The FoldX web server: an online force field. *Nucleic Acids Res.* **33**, W382–W388 (2005).
134. Parthiban, V., Gromiha, M. M. & Schomburg, D. CUPSAT: prediction of protein stability upon point mutations. *Nucleic Acids Res.* **34**, W239–W242 (2006).

135. Masso, M. & Vaisman, I. I. AUTO-MUTE 2.0: A Portable Framework with Enhanced Capabilities for Predicting Protein Functional Consequences upon Mutation. *Adv. Bioinforma.* **2014**, 278385 (2014).
136. Laimer, J., Hiebl-Flach, J., Lengauer, D. & Lackner, P. MAESTROweb: a web server for structure-based protein stability prediction. *Bioinformatics* **32**, 1414–1416 (2016).
137. Jia, L., Yarlagadda, R. & Reed, C. C. Structure Based Thermostability Prediction Models for Protein Single Point Mutations with Machine Learning Tools. *PLOS ONE* **10**, e0138022 (2015).
138. Cao, H., Wang, J., He, L., Qi, Y. & Zhang, J. Z. DeepDDG: Predicting the Stability Change of Protein Point Mutations Using Neural Networks. *J. Chem. Inf. Model.* **59**, 1508–1514 (2019).
139. Mansoor, S., Baek, M., Juergens, D., Watson, J. L. & Baker, D. Zero-shot Mutation Effect Prediction on Protein Stability and Function using RoseTTAFold. *Protein Sci.* **32**, e4780.
140. Zhou, Y., Pan, Q., Pires, D. E. V., Rodrigues, C. H. M. & Ascher, D. B. DDMut: predicting effects of mutations on protein stability using deep learning. *Nucleic Acids Res.* **51**, W122–W128 (2023).
141. Pucci, F., Bourgeas, R. & Rooman, M. Predicting protein thermal stability changes upon point mutations using statistical potentials: Introducing HoTMuSiC. *Sci. Rep.* **6**, 23257 (2016).
142. Pucci, F., Schwersensky, M. & Rooman, M. Artificial intelligence challenges for predicting the impact of mutations on protein stability. *Curr. Opin. Struct. Biol.* **72**, 161–168 (2022).
143. Louis, B. B. V. & Abriata, L. A. Reviewing Challenges of Predicting Protein Melting Temperature Change Upon Mutation Through the Full Analysis of a Highly Detailed Dataset with High-Resolution Structures. *Mol. Biotechnol.* **63**, 863–884 (2021).
144. Ku, T. *et al.* Predicting melting temperature directly from protein sequences. *Comput. Biol. Chem.* **33**, 445–450 (2009).
145. Gorania, M., Seker, H. & Haris, P. I. Predicting a protein's melting temperature from its amino acid sequence. in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology* 1820–1823 (2010). doi:10.1109/IEMBS.2010.5626421.
146. Yang, Y., Zhao, J., Zeng, L. & Vihinen, M. ProTstab2 for Prediction of Protein Thermal Stabilities. *Int. J. Mol. Sci.* **23**, 10798 (2022).
147. Li, M., Wang, H., Yang, Z., Zhang, L. & Zhu, Y. DeepTM: A deep learning algorithm for prediction of melting temperature of thermophilic proteins directly from sequences. *Comput. Struct. Biotechnol. J.* **21**, 5544–5560 (2023).
148. García-Marquina, G. *et al.* Deconvoluting the Directed Evolution Pathway of Engineered Acyltransferase LovD. *ChemCatChem* **14**, e202101349 (2022).
149. Learning from Directed Evolution: Theoretical Investigations into Cooperative Mutations in Lipase Enantioselectivity - Bocola - 2004 - ChemBioChem - Wiley Online Library. https://chemistry-europe.onlinelibrary.wiley.com/doi/10.1002/cbic.200300731.
150. Jarzab, A. *et al.* Meltome atlas-thermal proteome stability across the tree of life. *Nat. Methods* **17**, 495–503 (2020).

151. Stourac, J. *et al.* FireProtDB: database of manually curated protein stability data. *Nucleic Acids Res.* **49**, D319–D324 (2021).
152. Haselbeck, F. *et al.* Superior protein thermophilicity prediction with protein language model embeddings. *NAR Genomics Bioinforma.* **5**, lqad087 (2023).
153. Pace, C. N. *et al.* Conformational stability and thermodynamics of folding of ribonucleases Sa, Sa2 and Sa3. *J. Mol. Biol.* **279**, 271–286 (1998).
154. McCully, M. E., Beck, D. A. C. & Daggett, V. Promiscuous contacts and heightened dynamics increase thermostability in an engineered variant of the engrailed homeodomain. *Protein Eng. Des. Sel. PEDS* **26**, 35–45 (2013).
155. Beck, D. A. C. & Daggett, V. A One-Dimensional Reaction Coordinate for Identification of Transition States from Explicit Solvent Pfold-Like Calculations. *Biophys. J.* **93**, 3382–3391 (2007).
156. Bharatiy, S. K. *et al.* In Silico Designing of an Industrially Sustainable Carbonic Anhydrase Using Molecular Dynamics Simulation. *ACS Omega* **1**, 1081–1103 (2016).
157. Gonzalez, N. A., Li, B. A. & McCully, M. E. The stability and dynamics of computationally designed proteins. *Protein Eng. Des. Sel.* **35**, gzac001 (2022).
158. Nguyen, C., Yearwood, L. M. & McCully, M. E. Thermostabilization mechanisms in thermophilic versus mesophilic three-helix bundle proteins. *J. Comput. Chem.* **43**, 197–205 (2022).
159. Huang, J. *et al.* CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nat. Methods* **14**, 71–73 (2017).
160. H. Larsen, J. & Chorkendorff, I. From fundamental studies of reactivity on single crystals to the design of catalysts. *Surf. Sci. Rep.* **35**, 163–222 (1999).
161. Schmidbauer, S., Hohenleutner, A. & König, B. Chemical degradation in organic light-emitting devices: Mechanisms and implications for the design of new materials. *Adv. Mater.* **25**, 2114–2129 (2013).
162. Bartholomew, C. H. & Farrauto, R. J. *Fundamentals of Industrial Catalytic Processes: Second Edition*. *Fundamentals of Industrial Catalytic Processes: Second Edition* 966 (John Wiley & Sons, Ltd., 2010). doi:10.1002/9780471730071.
163. Swinney, D. C. Applications of binding kinetics to drug discovery: Translation of binding mechanisms to clinically differentiated therapeutic responses. *Int. J. Pharm. Med.* **22**, 23–34 (2008).
164. Prigogine, I. Chemical kinetics and dynamics. *Ann. N. Y. Acad. Sci.* **988**, 128–132 (2003).
165. Hänggi, P., Talkner, P. & Borkovec, M. Reaction-rate theory: Fifty years after Kramers. *Rev. Mod. Phys.* **62**, 251–341 (1990).
166. Peters, B. *Reaction Rate Theory and Rare Events*. 619 (Elsevier Science, 2017).
167. Zhao, Z.-J. *et al.* Theory-guided design of catalytic materials using scaling relationships and reactivity descriptors. *Nat. Rev. Mater.* **4**, 792–804 (2019).
168. Shen, J. & Nicolaou, C. Molecular property prediction: recent trends in the era of artificial intelligence - ScienceDirect. *Drug Discov. Today Technol.* **32–33**, 29–36 (2019).
169. Rice, O. K. & Ramsperger, H. C. THEORIES OF UNIMOLECULAR GAS REACTIONS AT LOW PRESSURES. *J. Am. Chem. Soc.* **49**, 1617–1629 (1927).

170. Kassel, L. S. Studies in Homogeneous Gas Reactions. I. *J. Phys. Chem.* **32**, 225–242 (1928).
171. Marcus, R. A. Unimolecular Dissociations and Free Radical Recombination Reactions. *J. Chem. Phys.* **20**, 359–364 (1952).
172. Eyring, H. The Activated Complex in Chemical Reactions. *J. Chem. Phys.* **3**, 107–115 (1935).
173. Evans, M. G. & Polanyi, M. Inertia and driving force of chemical reactions. *Trans. Faraday Soc.* **34**, 11–24 (1938).
174. Wigner, E. The transition state method. *Trans. Faraday Soc.* **34**, 29–41 (1938).
175. Truhlar, D. G. & Garrett, B. C. Variational Transition State Theory. *Annu. Rev. Phys. Chem.* **35**, 159–189 (1984).
176. Dellago, C., Bolhuis, P. G., Csajka, F. S. & Chandler, D. Transition path sampling and the calculation of rate constants. *J. Chem. Phys.* **108**, 1964–1977 (1998).
177. Miller, W. H., Schwartz, S. D. & Tromp, J. W. Quantum mechanical rate constants for bimolecular reactions. *J. Chem. Phys.* **79**, 4889–4898 (1983).
178. Miller, J. A. & Klippenstein, S. J. Master equation methods in gas phase chemical kinetics. *J. Phys. Chem. A* **110**, 10528–10544 (2006).
179. Schatz, G. C. Quantum theory of photodetachment spectra of transition states. *J. Phys. Chem.* **94**, 6157–6164 (1990).
180. Bowman, J. M. Reduced dimensionality theory of quantum reactive scattering. *J. Phys. Chem.* **95**, 4960–4968 (1991).
181. Pack, R. T. & Parker, G. A. Quantum reactive scattering in three dimensions using hyperspherical (APH) coordinates. Theory. *J. Chem. Phys.* **87**, 3888–3921 (1987).
182. Helgaker, T., Klopper, W. & Tew, D. P. Quantitative quantum chemistry. *Mol. Phys.* **106**, 2107–2143 (2008).
183. Kohn, W. & Sham, L. J. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys. Rev.* **140**, A1133–A1138 (1965).
184. Collins, M. A. Molecular potential-energy surfaces for chemical reaction dynamics. *Theor. Chem. Acc.* **108**, 313–324 (2002).
185. Henkelman, G., Uberuaga, B. P. & Jónsson, H. Climbing image nudged elastic band method for finding saddle points and minimum energy paths. *J. Chem. Phys.* **113**, 9901–9904 (2000).
186. Peters, B., Heyden, A., Bell, A. T. & Chakraborty, A. A growing string method for determining transition states: Comparison to the nudged elastic band and string methods. *J. Chem. Phys.* **120**, 7877–7886 (2004).
187. Lamberts, T., Kumar Samanta, P., Köhn, A. & Kästner, J. Quantum tunneling during interstellar surface-catalyzed formation of water: the reaction H + H 2 O 2 → H 2 O + OH. *Phys. Chem. Chem. Phys.* **18**, 33021–33030 (2016).
188. Mandrà, S., Valleau, S. & Ceotto, M. Deep nuclear resonant tunneling thermal rate constant calculations. *Int. J. Quantum Chem.* **113**, 1722–1734 (2013).
189. Tromp, J. W. & Miller, W. H. The reactive flux correlation function for collinear reactions H + H2, Cl + HCl and F + H2. *Faraday Discuss. Chem. Soc.* **84**, 441–453 (1987).
190. Eckart, C. The penetration of a potential barrier by electrons. *Phys. Rev.* **35**, 1303–1309 (1930).

191. Berman, L. & Peskin, U. Resonant tunneling probabilities for an N-terminal junction by the flux averaging method. *Int. J. Quantum Chem.* **99**, 752–757 (2004).
192. Ahmed, Z. Tunneling through a one-dimensional potential barrier. *Phys. Rev. A* **47**, 4761–4767 (1993).
193. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
194. Chollot, F. and others. Keras. *GitHub* (2015).
195. Kingma, D. P. & Ba, J. L. Adam: A method for stochastic optimization. in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2015).
196. Yamamoto, H. Resonant tunneling condition and transmission coefficient in a symmetrical one-dimensional rectangular double-barrier system. *Appl. Phys. Solids Surf.* **42**, 245–248 (1987).
197. Truhlar, D. G. & Wyatt, R. E. History of H3 Kinetics. *Annu. Rev. Phys. Chem.* **27**, 1–43 (1976).
198. Miller, W. H., Zhao, Y., Ceotto, M. & Yang, S. Quantum instanton approximation for thermal rate constants of chemical reactions. *J. Chem. Phys.* **119**, 1329–1342 (2003).
199. Truong, T. N., Truhlar, D. G. & Garrett, B. C. Embedded diatomics-in-molecules: A method to include delocalized electronic interactions in the treatment of covalent chemical reactions at metal surfaces. *J. Phys. Chem.* **93**, 8227–8239 (1989).
200. Wonchoba, S. E., Hu, W. P. & Truhlar, D. G. Surface diffusion of H on Ni(100): Interpretation of the transition temperature. *Phys. Rev. B* **51**, 9985–10002 (1995).
201. Dana, A. G. *et al.* Automated Reaction Kinetics and Network Exploration (Arkane): A Statistical Mechanics, Thermodynamics, Transition State Theory, and Master Equation Software. Preprint at https://doi.org/10.26434/chemrxiv-2022-4klsm (2022).
202. Truhlar, D. G. Transition state theory for enzyme kinetics. *Arch. Biochem. Biophys.* **582**, 10–17 (2015).
203. Unsleber, J. P. & Reiher, M. The Exploration of Chemical Reaction Networks. *Annu. Rev. Phys. Chem.* **71**, 121–142 (2020).
204. Häse, F., Valleau, S., Pyzer-Knapp, E. & Aspuru-Guzik, A. Machine learning exciton dynamics. *Chem. Sci.* **7**, 5139–5147 (2016).
205. Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R. & Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nat. Commun.* **8**, 13890–13890 (2017).
206. Ulissi, Z. W., Medford, A. J., Bligaard, T. & Nørskov, J. K. To address surface reaction network complexity using scaling relations machine learning and DFT calculations. *Nat. Commun.* **8**, 1–7 (2017).
207. von Lilienfeld, O. A., Müller, K.-R. & Tkatchenko, A. Exploring chemical compound space with quantum-based machine learning. *Nat. Rev. Chem.* **4**, 347–358 (2020).
208. Faber, F. A. *et al.* Prediction Errors of Molecular Machine Learning Models Lower than Hybrid DFT Error. *J. Chem. Theory Comput.* **13**, 5255–5264 (2017).
209. Chandrasekaran, A. *et al.* Solving the electronic structure problem with machine learning. *Npj Comput. Mater.* **5**, 1–7 (2019).

210. Peterson, A. A. Acceleration of saddle-point searches with machine learning. *J. Chem. Phys.* **145**, 074106 (2016).
211. Koistinen, O.-P., Ásgeirsson, V., Vehtari, A. & Jónsson, H. Minimum Mode Saddle Point Searches Using Gaussian Process Regression with Inverse-Distance Covariance Function. *J. Chem. Theory Comput.* **16**, 499–509 (2020).
212. Koistinen, O.-P., Ásgeirsson, V., Vehtari, A. & Jónsson, H. Nudged Elastic Band Calculations Accelerated with Gaussian Process Regression Based on Inverse Interatomic Distances. *J. Chem. Theory Comput.* **15**, 6738–6751 (2019).
213. Nandi, A., M. Bowman, J. & Houston, P. A Machine Learning Approach for Rate Constants. II. Clustering, Training, and Predictions for the O(3P) + HCl → OH + Cl Reaction. *J. Phys. Chem. A* **124**, 5746–5755 (2020).
214. Grambow, C. A., Pattanaik, L. & Green, W. H. Reactants, products, and transition states of elementary chemical reactions based on quantum chemistry. *Sci. Data* **7**, 1–8 (2020).
215. Ong, S. P. *et al.* Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Comput. Mater. Sci.* **68**, 314–319 (2013).
216. Komp, E. & Valleau, S. Low-cost prediction of molecular and transition state partition functions via machine learning. https://zenodo.org/records/6326560.
217. Komp, E. & Valleau, S. QuickQ. (2023).
218. Grambow, C. A., Pattanaik, L. & Green, W. H. Deep Learning of Activation Energies. *J. Phys. Chem. Lett.* **11**, 2992–2997 (2020).
219. Jorner, K., Brinck, T., Norrby, P. O. & Buttar, D. Machine learning meets mechanistic modelling for accurate prediction of experimental activation energies. *Chem. Sci.* **12**, 1163–1175 (2021).
220. Liu, C., Wang, J., Liu, X. & Liang, Y.-C. Deep CM-CNN for Spectrum Sensing in Cognitive Radio. *IEEE J. Sel. Areas Commun.* **37**, 2306–2321 (2019).
221. Ahmed, S. *et al.* Transformers in Time-Series Analysis: A Tutorial. *Circuits Syst. Signal Process.* **42**, 7433–7466 (2023).
222. National Academies of Sciences, E., and Medicine. Data Science for Undergraduates: Opportunities and Options. *Data Sci. Undergrad.* (2018) doi:10.17226/25104.
223. Venkatasubramanian, V. DROWNING IN DATA: Informatics and modeling challenges in a data-rich networked world. *AIChE J.* **55**, 2–8 (2009).
224. Finzer, W. The Data Science Education Dilemma. *Technol. Innov. Stat. Educ.* **7**, (2013).
225. Kross, S., Peng, R. D., Caffo, B. S., Gooding, I. & Leek, J. T. The Democratization of Data Science Education. *Am. Stat.* **74**, 1–7 (2019).
226. Witt, P., Hickman, D. & Herron, J. Educational Intensification: A Partnership Between Industry and Academia. *Chem. Eng. Educ.* **55**, 211–217 (2021).
227. Irizarry, R. A. The Role of Academia in Data Science Education. *Harv. Data Sci. Rev.* **2**, (2020).
228. Robertson, L. Toward an Epistemology of Active Learning in Higher Education and Its Promise. *Act. Learn. Strateg. High. Educ.* 17–44 (2018) doi:10.1108/978-1-78714-487-320181002.

229. Hartikainen, S., Rintala, H., Pylväs, L. & Nokelainen, P. The Concept of Active Learning and the Measurement of Learning Outcomes: A Review of Research in Engineering Higher Education. *Educ. Sci. 2019 Vol 9 Page 276* **9**, 276–276 (2019).
230. Johnson, R. T. & Johnson, D. W. Active Learning: Cooperation in the Classroom. *Annu. Rep. Educ. Psychol. Jpn.* **47**, 29–30 (2008).
231. Hernández-de-Menéndez, M., Vallejo Guevara, A., Tudón Martínez, J. C., Hernández Alcántara, D. & Morales-Menendez, R. Active learning in engineering education. A review of fundamentals, best practices and experiences. *Int. J. Interact. Des. Manuf. IJIDeM 2019 133* **13**, 909–922 (2019).
232. Prince, M. Does Active Learning Work? A Review of the Research. *J. Eng. Educ.* **93**, 223–231 (2004).
233. Aditomo, A., Goodyear, P., Bliuc, A. M. & Ellis, R. A. Inquiry-based learning in higher education: Principal forms, educational objectives, and disciplinary variations. *Stud. High. Educ.* **38**, 1239–1258 (2013).
234. Barell, John. *Problem-based learning : an inquiry approach*. (Corwin, 2007).
235. Ernst, D. C., Hodge, A. & Yoshinobu, S. What Is Inquiry-Based Learning? *Not. Am. Math. Soc.* **64**, 570–574 (2017).
236. Saltz, J., Saltz, J. & Heckman, R. Big Data science education: A case study of a project-focused introductory... *Themes Sci. Technol. Educ.* **8**, 85–94 (2016).
237. Muijs, D. & Rumyantseva, N. Coopetition in education: Collaborating in a competitive environment. *J. Educ. Change 2013 151* **15**, 1–18 (2013).
238. Tokunaga, S., Martínez, M. & Crusat, X. Coopetition: Industrial interplay to foster innovative entrepreneurship in energy engineering education. *IEEE Glob. Eng. Educ. Conf. EDUCON* **April-2019**, 1063–1068 (2019).
239. Zhong, B. & Xia, L. Effects of new coopetition designs on learning performance in robotics education. *J. Comput. Assist. Learn.* **38**, 223–236 (2022).
240. Szeto, A., Haines, J. & Buchholz, A. C. Impact of an Optional Experiential Learning Opportunity on Student Engagement and Performance in Undergraduate Nutrition Courses. *Can J Diet Pract Res.* **77**, 84–88 (2015).
241. Seifried, E., Eckert, C. & Spinath, B. Optional Learning Opportunities: Who Seizes Them and What Are the Learning Outcomes? *Teach. Psychol.* **45**, 246–250 (2018).
242. Kramer, J. & Srinath, K. R. Python-The Fastest Growing Programming Language. *Int. Res. J. Eng. Technol.* **4**, (2017).
243. Lasser, J., Manik, D., Silbersdorff, A., Säfken, B. & Kneib, T. Introductory data science across disciplines, using Python, case studies, and industry consulting projects. *Teach. Stat.* **43**, S190–S200 (2021).
244. Komp, E. *et al.* CHACK_documents. *GitHub* https://zenodo.org/record/6512221 (2022).
245. Baird, C. L. Male-dominated stem disciplines: How do we make them more attractive to women? *IEEE Instrum. Meas. Mag.* **21**, 4–14 (2018).
246. Niler, A. A., Asencio, R. & DeChurch, L. A. Solidarity in STEM: How Gender Composition Affects Women's Experience in Work Teams. *Sex Roles* **82**, 142–154 (2020).
247. Zhang, L., Lu, J. R. & Waigh, T. A. Electronics of peptide- and protein-based biomaterials. *Adv. Colloid Interface Sci.* **287**, 102319 (2021).

248. Jia, H. *et al.* 3D printed protein-based robotic structures actuated by molecular motor assemblies. *Nat. Mater.* **21**, 703–709 (2022).
249. Davari, N. *et al.* Protein-Based Hydrogels: Promising Materials for Tissue Engineering. *Polymers* **14**, 986 (2022).
250. Victorino da Silva Amatto, I. *et al.* Enzyme engineering and its industrial applications. *Biotechnol. Appl. Biochem.* **69**, 389–409 (2022).
251. Chen, T. *et al.* Identification and structure-guided enzyme engineering of an extremophilic protein phosphatase. *FASEB J.* **34**, 1–1 (2020).
252. Chevalier, A. *et al.* Massively parallel de novo protein design for targeted therapeutics. *Nature* **550**, 74–79 (2017).
253. Huang, P.-S., Boyken, S. E. & Baker, D. The coming of age of de novo protein design. *Nature* **537**, 320–327 (2016).
254. Korendovych, I. V. & DeGrado, W. F. De novo protein design, a retrospective. *Q. Rev. Biophys.* **53**, e3 (2020).
255. Cellulase Market. https://www.futuremarketinsights.com/reports/cellulase-market.
256. Patel, A. K., Singhania, R. R., Sim, S. J. & Pandey, A. Thermostable cellulases: Current status and perspectives. *Bioresour. Technol.* **279**, 385–392 (2019).
257. Akram, F., Haq, I. ul, Aqeel, A., Ahmed, Z. & Shah, F. I. Thermostable cellulases: Structure, catalytic mechanisms, directed evolution and industrial implementations. *Renew. Sustain. Energy Rev.* **151**, 111597 (2021).
258. Singh, A., Bajar, S., Devi, A. & Pant, D. An overview on the recent developments in fungal cellulase production and their industrial applications. *Bioresour. Technol. Rep.* **14**, 100652 (2021).
259. Singhania, R. R. *et al.* Challenges in cellulase bioprocess for biofuel applications. *Renew. Sustain. Energy Rev.* **151**, 111622 (2021).
260. Chen, G. *et al.* Amyloid beta: structure, biology and structure-based therapeutic development. *Acta Pharmacol. Sin.* **38**, 1205–1235 (2017).
261. Wang, M.-Y. *et al.* SARS-CoV-2: Structure, Biology, and Structure-Based Therapeutics Development. *Front. Cell. Infect. Microbiol.* **10**, (2020).
262. Dimitrov, D. S. Therapeutic Proteins. *Methods Mol. Biol. Clifton NJ* **899**, 1–26 (2012).
263. Tsomaia, N. Peptide therapeutics: Targeting the undruggable space. *Eur. J. Med. Chem.* **94**, 459–470 (2015).
264. Chen, D. & Kristensen, D. Opportunities and challenges of developing thermostable vaccines. *Expert Rev. Vaccines* **8**, 547–557 (2009).
265. Lloyd, J. & Cheyne, J. The origins of the vaccine cold chain and a glimpse of the future. *Vaccine* **35**, 2115–2120 (2017).
266. Xiong, X. *et al.* A thermostable, closed SARS-CoV-2 spike protein trimer. *Nat. Struct. Mol. Biol.* **27**, 934–941 (2020).
267. Iyer, R. & Iken, B. Protein engineering of representative hydrolytic enzymes for remediation of organophosphates. *Biochem. Eng. J.* **94**, 134–144 (2015).
268. Synowiecki, J. Thermostable enzymes in food processing. *Enzym. Addit. Process. Aids* **46**, 29–54 (2008).
269. Ó'Fágáin, C. Enzyme stabilization—recent experimental progress. *Enzyme Microb. Technol.* **33**, 137–149 (2003).

270. Gómez, L. & Villalonga, R. Functional stabilization of invertase by covalent modification with pectin. *Biotechnol. Lett.* **22**, 1191–1195 (2000).
271. Bieniarz, C., Cornwell, M. J. & Young, D. F. Alkaline Phosphatase Activatable Polymeric Cross-Linkers and Their Use in the Stabilization of Proteins. *Bioconjug. Chem.* **9**, 390–398 (1998).
272. Agarwal, P., van der Weijden, J., Sletten, E. M., Rabuka, D. & Bertozzi, C. R. A Pictet-Spengler ligation for protein chemical modification. *PNAS* **110**, 46–51 (2012).
273. Kaushik, J. K. & Bhat, R. Thermal Stability of Proteins in Aqueous Polyol Solutions: Role of the Surface Tension of Water in the Stabilizing Effect of Polyols. *J. Phys. Chem. B* **102**, 7058–7066 (1998).
274. Chaudhary, K., Kumar, K., Venkatesu, P. & Masram, D. T. Protein immobilization on graphene oxide or reduced graphene oxide surface and their applications: Influence over activity, structural and thermal stability of protein. *Adv. Colloid Interface Sci.* **289**, 102367 (2021).
275. Wold, F. In Vivo Chemical Modification of Proteins (Post-Translational Modification). *Annu. Rev. Biochem.* **50**, 783–814 (1981).
276. Boutureira, O. & Bernardes, G. J. L. Advances in Chemical Protein Modification. *Chem. Rev.* **115**, 2174–2195 (2015).
277. Zeldovich, K. B., Berezovsky, I. N. & Shakhnovich, E. I. Protein and DNA Sequence Determinants of Thermophilic Adaptation. *PLoS Comput. Biol.* **3**, e5 (2007).
278. Vieille, C. & Zeikus, G. J. Hyperthermophilic enzymes: sources, uses, and molecular mechanisms for thermostability. *Microbiol. Mol. Biol. Rev. MMBR* **65**, 1–43 (2001).
279. Pace, C. N. *et al.* Contribution of Hydrophobic Interactions to Protein Stability. *J. Mol. Biol.* **408**, 514–528 (2011).
280. Baldwin, R. L. Temperature dependence of the hydrophobic interaction in protein folding. *PNAS* **83**, 8069–8072 (1986).
281. Shima, S., Hérault, D. A., Berkessel, A. & Thauer, R. K. Activation and thermostabilization effects of cyclic 2, 3-diphosphoglycerate on enzymes from the hyperthermophilic Methanopyrus kandleri. *Arch. Microbiol.* **170**, 469–472 (1998).
282. Packer, M. S. & Liu, D. R. Methods for the directed evolution of proteins. *Nat. Rev. Genet.* **16**, 379–394 (2015).
283. Wang, Y. *et al.* Directed Evolution: Methodologies and Applications. *Chem. Rev.* **121**, 12384–12444 (2021).
284. Qiu, J. *et al.* Enhancing the activity and thermal stability of a phthalate-degrading hydrolase by random mutagenesis. *Ecotoxicol. Environ. Saf.* **209**, 111795 (2021).
285. Sriprapundh, D., Vieille, C. & Zeikus, J. G. Molecular determinants of xylose isomerase thermal stability and activity: analysis of thermozymes by site-directed mutagenesis. *Protein Eng. Des. Sel.* **13**, 259–265 (2000).
286. Zhao, H. & Arnold, F. H. Directed evolution converts subtilisin E into a functional equivalent of thermitase. *Protein Eng. Des. Sel.* **12**, 47–53 (1999).
287. Pongpamorn, P. *et al.* Identification of a Hotspot Residue for Improving the Thermostability of a Flavin-Dependent Monooxygenase. *ChemBioChem* **20**, 3020–3031 (2019).

288. Li, G. *et al.* Identification of a hot-spot to enhance Candida rugosa lipase thermostability by rational design methods. *RSC Adv.* **8**, 1948–1957 (2018).
289. Huang, J. X. *et al.* High throughput discovery of functional protein modifications by Hotspot Thermal Profiling. *Nat. Methods* **16**, 894–901 (2019).
290. Carter, P. Site-directed mutagenesis. *Biochem. J.* **237**, 1–7 (1986).
291. Son, H. F. *et al.* Rational Protein Engineering of Thermo-Stable PETase from Ideonella sakaiensis for Highly Efficient PET Degradation. *ACS Catal.* **9**, 3519–3526 (2019).
292. Chaparro-Riggers, J. F., Polizzi, K. M. & Bommarius, A. S. Better library design: data-driven protein engineering. *Biotechnol. J.* **2**, 180–191 (2007).
293. Polizzi, K. M., Chaparro-Riggers, J. F., Vazquez-Figueroa, E. & Bommarius, A. S. Structure-guided consensus approach to create a more thermostable penicillin G acylase. *Biotechnol. J.* **1**, 531–536 (2006).
294. Schmidt-Dannert, C. & Arnold, F. H. Directed evolution of industrial enzymes. *Trends Biotechnol.* **17**, 135–136 (1999).
295. Polizzi, K. M., Bommarius, A. S., Broering, J. M. & Chaparro-Riggers, J. F. Stability of biocatalysts. *Curr. Opin. Chem. Biol.* **11**, 220–225 (2007).
296. Nikam, R., Kulandaisamy, A., Harini, K., Sharma, D. & Gromiha, M. M. ProThermDB: thermodynamic database for proteins and mutants revisited after 15 years. *Nucleic Acids Res.* **49**, D420–D424 (2021).
297. Panja, A. S., Maiti, S. & Bandyopadhyay, B. Protein stability governed by its structural plasticity is inferred by physicochemical factors and salt bridges. *Sci. Rep.* **10**, 1822 (2020).
298. Elcock, A. H. The stability of salt bridges at high temperatures: implications for hyperthermophilic proteins. *J. Mol. Biol.* **284**, 489–502 (1998).
299. Zhou, X.-X., Wang, Y.-B., Pan, Y.-J. & Li, W.-F. Differences in amino acids composition and coupling patterns between mesophilic and thermophilic proteins. *Amino Acids* **34**, 25–33 (2008).
300. Karshikoff, A., Nilsson, L. & Ladenstein, R. Rigidity versus flexibility: the dilemma of understanding protein thermal stability. *FEBS J.* **282**, 3899–3917 (2015).
301. Dominy, B. N., Minoux, H. & Brooks III, C. L. An electrostatic basis for the stability of thermophilic proteins. *Proteins Struct. Funct. Bioinforma.* **57**, 128–141 (2004).
302. Jorda, J. & Yeates, T. O. Widespread Disulfide Bonding in Proteins from Thermophilic Archaea. *Archaea* **2011**, 409156 (2011).
303. Berezovsky, I. N., Zeldovich, K. B. & Shakhnovich, E. I. Positive and Negative Design in Stability and Thermal Adaptation of Natural Proteins. *PLOS Comput. Biol.* **3**, e52 (2007).
304. Folch, B., Dehouck, Y. & Rooman, M. Thermo- and Mesostabilizing Protein Interactions Identified by Temperature-Dependent Statistical Potentials. *Biophys. J.* **98**, 667–677 (2010).
305. Nisthal, A., Wang, C. Y., Ary, M. L. & Mayo, S. L. Protein stability engineering insights revealed by domain-wide comprehensive mutagenesis. *Proc. Natl. Acad. Sci.* **116**, 16367–16377 (2019).
306. Makwana, K. M. & Mahalakshmi, R. Implications of aromatic–aromatic interactions: From protein structures to peptide models - Makwana - 2015 - Protein Science - Wiley Online Library. *Protein Sci.* **24**, (2015).

307. Pucci, F. & Rooman, M. Improved insights into protein thermal stability: from the molecular to the structurome scale. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **374**, 20160141 (2016).
308. Ahmed, Z., Zulfiqar, H., Tang, L. & Lin, H. A Statistical Analysis of the Sequence and Structure of Thermophilic and Non-Thermophilic Proteins. *Int. J. Mol. Sci.* **23**, 10116 (2022).
309. Prajapati, R. S., Sirajuddin, M., Durani, V., Sreeramulu, S. & Varadarajan, R. Contribution of Cation−π Interactions to Protein Stability. *Biochemistry* **45**, 15000–15010 (2006).
310. Kumar, S., Tsai, C.-J. & Nussinov, R. Factors enhancing protein thermostability. *Protein Eng. Des. Sel.* **13**, 179–191 (2000).
311. Gromiha, M. M., Pathak, M. C., Saraboji, K., Ortlund, E. A. & Gaucher, E. A. Hydrophobic environment is a key factor for the stability of thermophilic proteins. *Proteins Struct. Funct. Bioinforma.* **81**, 715–721 (2013).
312. Szilágyi, A. & Závodszky, P. Structural differences between mesophilic, moderately thermophilic and extremely thermophilic protein subunits: results of a comprehensive survey. *Structure* **8**, 493–504 (2000).
313. Hait, S., Mallik, S., Basu, S. & Kundu, S. Finding the generalized molecular principles of protein thermal stability. *Proteins Struct. Funct. Bioinforma.* **88**, 788–808 (2020).
314. Larralde, M. & Zeller, G. PyHMMER: a Python library binding to HMMER for efficient sequence analysis. *Bioinformatics* **39**, btad214 (2023).
315. Mistry, J. *et al.* Pfam: The protein families database in 2021. *Nucleic Acids Res.* **49**, D412–D419 (2021).
316. Tekaia, F. Inferring Orthologs: Open Questions and Perspectives. *Genomics Insights* **9**, GEI.S37925 (2016).
317. Kouba, P. *et al.* Machine Learning-Guided Protein Engineering. *ACS Catal.* **13**, 13863–13895 (2023).
318. Wittmann, B. J., Johnston, K. E., Wu, Z. & Arnold, F. H. Advances in machine learning for directed evolution. *Curr. Opin. Struct. Biol.* **69**, 11–18 (2021).
319. Wu, Z., Johnston, K. E., Arnold, F. H. & Yang, K. K. Protein sequence design with deep generative models. *Curr. Opin. Chem. Biol.* **65**, 18–27 (2021).
320. Ding, W., Nakai, K. & Gong, H. Protein design via deep learning. *Brief. Bioinform.* **23**, bbac102 (2022).
321. Park, Y., Metzger, B. P. H. & Thornton, J. W. The simplicity of protein sequence-function relationships. 2023.09.02.556057 Preprint at https://doi.org/10.1101/2023.09.02.556057 (2023).
322. Livesey, B. J. & Marsh, J. A. Using deep mutational scanning to benchmark variant effect predictors and identify disease mutations. *Mol. Syst. Biol.* **16**, (2020).
323. Saito, Y. *et al.* Machine-Learning-Guided Mutagenesis for Directed Evolution of Fluorescent Proteins. *ACS Synth. Biol.* **7**, 2014–2022 (2018).
324. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16**, 687–694 (2019).
325. Fox, R. J. *et al.* Improving catalytic function by ProSAR-driven enzyme evolution. *Nat. Biotechnol.* **25**, 338–344 (2007).

326. Wu, Z., Kan, S. B. J., Lewis, R. D., Wittmann, B. J. & Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl. Acad. Sci. U. S. A.* **116**, 8852–8858 (2019).
327. Hopf, T. A. *et al.* The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics* **35**, 1582–1584 (2019).
328. Hopf, T. A. *et al.* Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.* **35**, 128–135 (2017).
329. Haldane, A. & Levy, R. M. Influence of multiple-sequence-alignment depth on Potts statistical models of protein covariation. *Phys. Rev. E* **99**, 032405 (2019).
330. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* **12**, 1026-1045.e7 (2021).
331. Qiu, Y., Hu, J. & Wei, G.-W. Cluster learning-assisted directed evolution. *Nat. Comput. Sci.* **1**, 809–818 (2021).
332. Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl. Acad. Sci.* **110**, E193–E201 (2013).
333. Hie, B., Bryson, B. D. & Berger, B. Leveraging Uncertainty in Machine Learning Accelerates Biological Discovery and Design. *Cell Syst.* **11**, 461-477.e9 (2020).
334. Wang, C., Kim, J., Cong, L. & Wang, M. Neural Bandits for Protein Sequence Optimization. in *2022 56th Annual Conference on Information Sciences and Systems (CISS)* 188–193 (2022). doi:10.1109/CISS53076.2022.9751154.
335. Emami, P., Perreault, A., Law, J., Biagioni, D. & St. John, P. Plug & play directed evolution of proteins with gradient-based discrete MCMC. *Mach. Learn. Sci. Technol.* **4**, 025014 (2023).
336. Linder, J., Bogard, N., Rosenberg, A. B. & Seelig, G. A Generative Neural Network for Maximizing Fitness and Diversity of Synthetic DNA and Protein Sequences. *Cell Syst.* **11**, 49-62.e16 (2020).
337. Hon, J. *et al.* SoluProt: prediction of soluble protein expression in Escherichia coli. *Bioinformatics* **37**, 23–28 (2021).
338. Zhao, J., Yan, W. & Yang, Y. DeepTP: A Deep Learning Model for Thermophilic Protein Prediction. *Int. J. Mol. Sci.* **24**, 2217 (2023).
339. Charoenkwan, P., Chotpatiwetchkul, W., Lee, V. S., Nantasenamat, C. & Shoombuatong, W. A novel sequence-based predictor for identifying and characterizing thermophilic proteins using estimated propensity scores of dipeptides. *Sci. Rep.* **11**, 23782 (2021).
340. Lin, H. & Chen, W. Prediction of thermophilic proteins using feature selection technique. *J. Microbiol. Methods* **84**, 67–70 (2011).
341. Rao, R. M. *et al.* MSA Transformer. in *Proceedings of the 38th International Conference on Machine Learning* 8844–8856 (PMLR, 2021).
342. Hsu, C., Nisonoff, H., Fannjiang, C. & Listgarten, J. Learning protein fitness models from evolutionary and assay-labeled data. *Nat. Biotechnol.* **40**, 1114–1122 (2022).
343. Burley, S. K. *et al.* Protein Data Bank (PDB): The Single Global Macromolecular Structure Archive. in *Protein Crystallography: Methods and Protocols* (eds. Wlodawer, A., Dauter, Z. & Jaskolski, M.) 627–641 (Springer, 2017). doi:10.1007/978-1-4939-7000-1_26.

344. Dauparas, J. *et al.* Robust deep learning–based protein sequence design using ProteinMPNN. *Science* **378**, 49–56 (2022).
345. Anishchenko, I. *et al.* De novo protein design by deep network hallucination. *Nature* **600**, 547–552 (2021).
346. Zheng, Z. *et al.* Structure-informed Language Models Are Protein Designers. Preprint at http://arxiv.org/abs/2302.01649 (2023).
347. Melnyk, I., Lozano, A., Das, P. & Vijil, V. AlphaFold Distillation for Protein Design. in (2023).
348. Yamaguchi, S., Nakashima, H., Moriwaki, Y., Terada, T. & Shimizu, K. Prediction of protein mononucleotide binding sites using AlphaFold2 and machine learning. *Comput. Biol. Chem.* **100**, 107744 (2022).
349. Stärk, H., Ganea, O., Pattanaik, L., Barzilay, D. R. & Jaakkola, T. EquiBind: Geometric Deep Learning for Drug Binding Structure Prediction. in *Proceedings of the 39th International Conference on Machine Learning* 20503–20521 (PMLR, 2022).
350. Shin, J.-E. *et al.* Protein design and variant prediction using autoregressive generative models. *Nat. Commun.* **12**, 2403 (2021).
351. Alamdari, S. *et al.* Protein generation with evolutionary diffusion: sequence is all you need. 2023.09.11.556673 Preprint at https://doi.org/10.1101/2023.09.11.556673 (2023).
352. Stahlberg, F. Neural Machine Translation: A Review. *J. Art. Int. Res.* **69**, (2020).
353. Baniata, L. H., Ampomah, I. K. E. & Park, S. A Transformer-Based Neural Machine Translation Model for Arabic Dialects That Utilizes Subword Units. *Sensors* **21**, 6509 (2021).
354. Gromiha, M. M., Oobatake, M. & Sarai, A. Important amino acid properties for enhanced thermostability from mesophilic to thermophilic proteins. *Biophys. Chem.* **82**, 51–67 (1999).
355. Sauer, D. B. & Wang, D.-N. Predicting the optimal growth temperatures of prokaryotes using only genome derived features. *Bioinformatics* **35**, 3224–3231 (2019).
356. O'Leary, N. A. *et al.* Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.* **44**, D733–D745 (2016).
357. Kans, J. Entrez Direct: E-utilities on the Unix Command Line. in *Entrez Programming Utilities Help [Internet]* (National Center for Biotechnology Information (US), 2023).
358. Engqvist, M. K. M. Growth temperatures for 21,498 microorganisms. (2018) doi:10.5281/zenodo.1175609.
359. UniProt: the Universal Protein Knowledgebase in 2023 | Nucleic Acids Research | Oxford Academic. https://academic.oup.com/nar/article/51/D1/D523/6835362.
360. Yang, B., Wang, Y. & Qian, P.-Y. Sensitivity and correlation of hypervariable regions in 16S rRNA genes in phylogenetic analysis. *BMC Bioinformatics* **17**, 135 (2016).
361. Schloss, P. D. The effects of alignment quality, distance calculation method, sequence filtering, and region on the analysis of 16S rRNA gene-based studies. *PLoS Comput. Biol.* **6**, e1000844 (2010).

362. Kim, M., Oh, H.-S., Park, S.-C. & Chun, J. Towards a taxonomic coherence between average nucleotide identity and 16S rRNA gene sequence similarity for species demarcation of prokaryotes. *Int. J. Syst. Evol. Microbiol.* **64**, 346–351 (2014).
363. NCBI Taxonomy: a comprehensive update on curation, resources and tools | Database | Oxford Academic. https://academic.oup.com/database/article/doi/10.1093/database/baaa062/5881509?login=false.
364. Data Version Control · DVC. *Data Version Control · DVC* https://dvc.org/.
365. DuckDB | Proceedings of the 2019 International Conference on Management of Data. https://dl.acm.org/doi/abs/10.1145/3299869.3320212.
366. Brandes, N., Ofer, D., Peleg, Y., Rappoport, N. & Linial, M. ProteinBERT: A universal deep-learning model of protein sequence and function. 2021.05.24.445464 Preprint at https://doi.org/10.1101/2021.05.24.445464 (2021).
367. Whisstock, J. C. & Lesk, A. M. Prediction of protein function from protein sequence and structure. *Q. Rev. Biophys.* **36**, 307–340 (2003).
368. Meredig, B. *et al.* Can machine learning identify the next high-temperature superconductor? Examining extrapolation performance for materials discovery. *Mol. Syst. Des. Eng.* **3**, 819–825 (2018).
369. S. Muckley, E., E. Saal, J., Meredig, B., S. Roper, C. & H. Martin, J. Interpretable models for extrapolation in scientific machine learning. *Digit. Discov.* **2**, 1425–1435 (2023).
370. Rao, R. *et al.* Evaluating Protein Transfer Learning with TAPE. Preprint at https://doi.org/10.48550/arXiv.1906.08230 (2019).
371. Minami, S. PyDSSP. (2023).
372. Singer, G. A. C. & Hickey, D. A. Thermophilic prokaryotes have characteristic patterns of codon usage, amino acid composition and nucleotide content. *Gene* **317**, 39–47 (2003).
373. Gao, X., Dong, X., Li, X., Liu, Z. & Liu, H. Prediction of disulfide bond engineering sites using a machine learning method. *Sci. Rep.* **10**, 10330 (2020).
374. *1ENH*. https://www.rcsb.org/structure/1ENH.
375. Tripp, K. W., Sternke, M., Majumdar, A. & Barrick, D. Creating a Homeodomain with High Stability and DNA Binding Affinity by Sequence Averaging. *J. Am. Chem. Soc.* **139**, 5051–5060 (2017).
376. Beck, D. A. C. & Daggett, V. Methods for molecular dynamics simulations of protein folding/unfolding in solution. *Methods* **34**, 112–120 (2004).
377. Brenowitz, M., Senear, D. F. & Kingston, R. E. DNase I Footprint Analysis of Protein-DNA Binding. *Curr. Protoc. Mol. Biol.* **7**, 12.4.1-12.4.16 (1989).
378. England, C. G., Ehlerding, E. B. & Cai, W. NanoLuc: A Small Luciferase Is Brightening Up the Field of Bioluminescence. *Bioconjug. Chem.* **27**, 1175–1187 (2016).
379. Ahmad, S., Kamal, Md. Z., Sankaranarayanan, R. & Rao, N. M. Thermostable Bacillus subtilis Lipases: In Vitro Evolution and Structural Insight. *J. Mol. Biol.* **381**, 324–340 (2008).
380. Nutschel, C. *et al.* Systematically Scrutinizing the Impact of Substitution Sites on Thermostability and Detergent Tolerance for Bacillus subtilis Lipase A. *J. Chem. Inf. Model.* **60**, 1568–1584 (2020).

381. Ferruz, N., Schmidt, S. & Höcker, B. ProtGPT2 is a deep unsupervised language model for protein design. *Nat. Commun.* **13**, 4348 (2022).
382. Steinegger, M., Mirdita, M. & Söding, J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods* **16**, 603–606 (2019).

APPENDIX A

A.1 Quantum Reaction Rate Constants

A.1.1 Database Generation

The equations for the potentials employed to generate the database of quantum reaction rate constants are defined below in Table A.1.1-a. For the single barriers the exact expressions for the transmission coefficient $T(E)$ were used.

For the double barriers, the transmission coefficient was evaluated as in equation according to Equation 1.2 in the main portion of this thesis, with $x \in [-L, L]$ and $L = 20[au]$. The evaluation of the transmission coefficient was carried out using scipy's integrate.solve_ivp function to solve the initial value problem with the implicit Runge-Kutta method of order 5 as defined by the 'Radau' option. The range of energy values over which the transmission coefficient was evaluated was a set of 300 points in the interval $E_{values} = \left[\frac{E_{max}}{1000}; 2E_{max}\right]$, with $E_{max} = 2 \cdot [V(x)]$. The maximum value of the potential energy was evaluated over 1000 points in the range $-L < x < L$.

Table A.1.1-a: Equations for the one-dimensional potential energy barriers used to build the database. The value of Δx was 0.1 au for symmetric barriers and 0.2 au for asymmetric barriers.

Barrier	Potential energy	Parameters
Single rectangular	$V_R(x) = \{V_1 \ 0.0 < x < w_1 \ 0.0 \ \ x < 0 \ \vee \ x > w_1$	V_1 – barrier height w_1 – barrier width $\alpha = 1.0$

		$s = 0.0$
Single symmetric Eckart	$V_{SE}(x) = \frac{V_1}{(\pi x / w_1)}$	V_1 – barrier height w_1 – barrier width $\alpha = 1.0$ $s = \frac{V(x^*) - V(x^* + \Delta x)}{\Delta x}$ $x^* \mid V(x^*) = max(V(x))$
Single asymmetric Eckart	$V_{AE}(x) = \frac{V_1(1-\alpha)}{1+e^{-2w_1 x}} + \frac{V_1(1+\sqrt{\alpha})^2}{4w_1 x}$	V_1 – barrier height w_1 – barrier width α – asymmetry $s = \frac{\lvert V(x^* + \Delta x) - V(x^*)\rvert + \lvert V(x^*) - V(x^* - \Delta x)\rvert}{2\Delta x}$ $x^* \mid V(x^*) = max(V(x))$
Double rectangular	$V_{DR}(x) = \{V_1\ 0.0 < x < w_1\ V_2\ w_1 + d < x < w_1 + d + w_2\ 0.0\ elsewhere$	V_i – barrier *i* height w_i – barrier *i* width *d* – distance between barriers $\alpha = \frac{1}{2}\left(\frac{\lvert V_1 - V_2\rvert}{\lvert V_1 + V_2\rvert} + \frac{\lvert w_1 - w_2\rvert}{\lvert w_1 + w_2\rvert}\right)$ $s = 0.0$
Double gaussian	$V_{DG}(x) = V_1 e^{\frac{-(x-x_1)^2}{2\sigma_1^2}} + V_2 e^{\frac{-(x-x_2)^2}{2\sigma_2^2}}$ $\sigma_1 = w_1/3; \quad \sigma_2 = w_2/3$ $x_1 = 0.0; \quad x_2 = 3 \cdot (\sigma_1 + \sigma_2) + d$	V_i – barrier *i* height w_i – barrier *i* width *d* – distance between barriers $\alpha = \frac{1}{2}\left(\frac{\lvert V_1 - V_2\rvert}{\lvert V_1 + V_2\rvert} + \frac{\lvert w_1 - w_2\rvert}{\lvert w_1 + w_2\rvert}\right)$ $s = \frac{V(x^*) - V(x^* + \Delta x)}{\Delta x}$ $x^* \mid V(x^*) = max(V(x))$
Peskin barrier	$V_{Pesk}(x) = V_1\left(\frac{1}{(x)} - \frac{1}{(w_1 x)}\right)$	V_1 – barrier height w_1 – barrier width $\alpha = 1.0$ $s = \frac{V(x^*) - V(x^* + \Delta x)}{\Delta x}$ $x^* \mid V(x^*) = max(V(x))$

A.1.2 Dataset

A visualization of the distribution of the training and testing dataset for the work on quantum reaction rate constant products with respect to its input features and output label is shown in Figures A.1.2-a through c. The first plot (a) displays scattering of pairwise feature distributions with kernel density estimate (KDE) diagonals for the entire dataset. The following plots (b-c) show the pairwise distribution of subsets of features and labels for the test and train sets through KDE. Figure A.1.2-d maps the correlation between variables through the Pearson correlation coefficient.

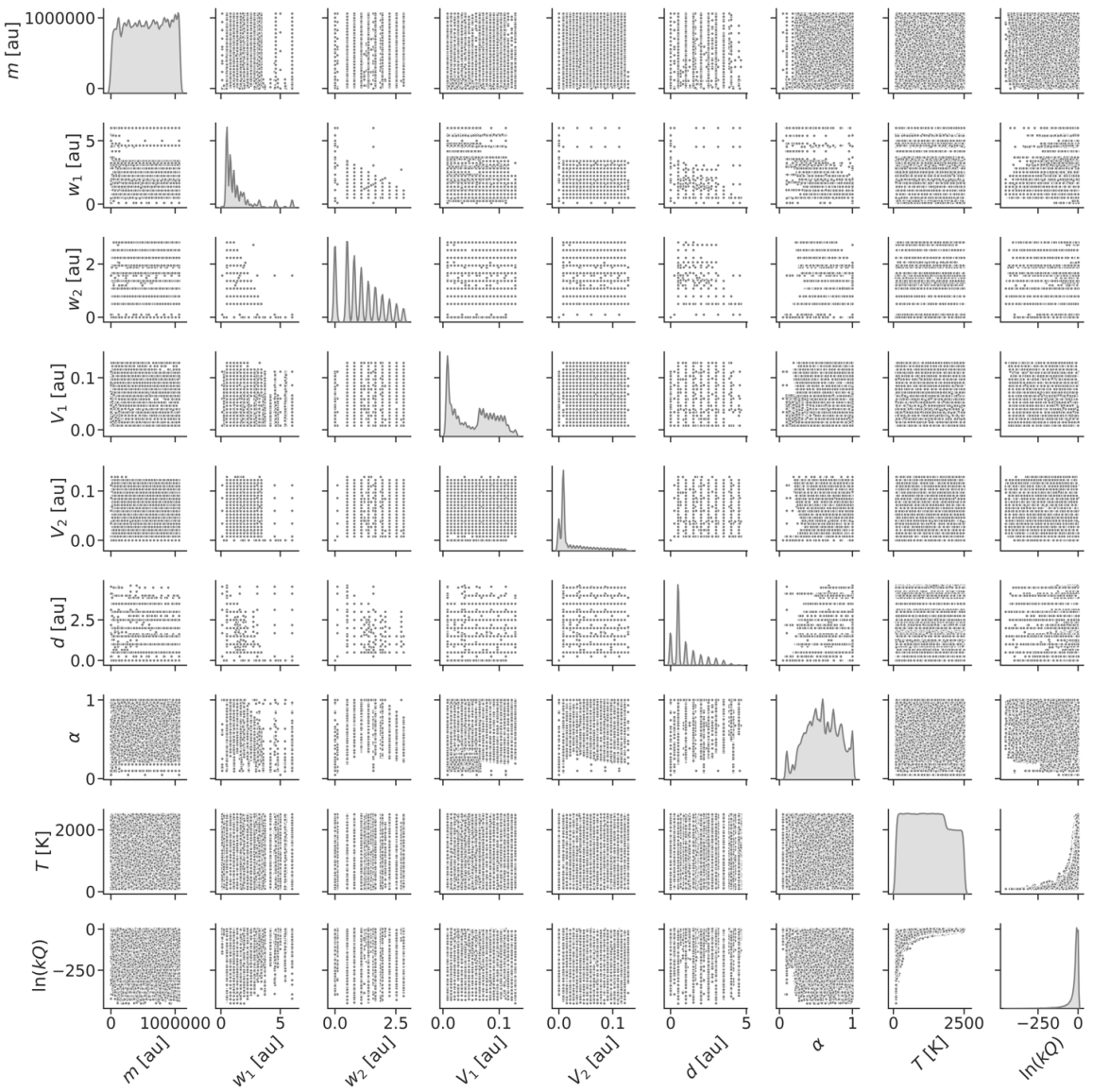

Figure A1.2-a: Database pair-wise distribution for all input features and for the output label. Points are datum. Diagonals are gaussian KDE of single features.

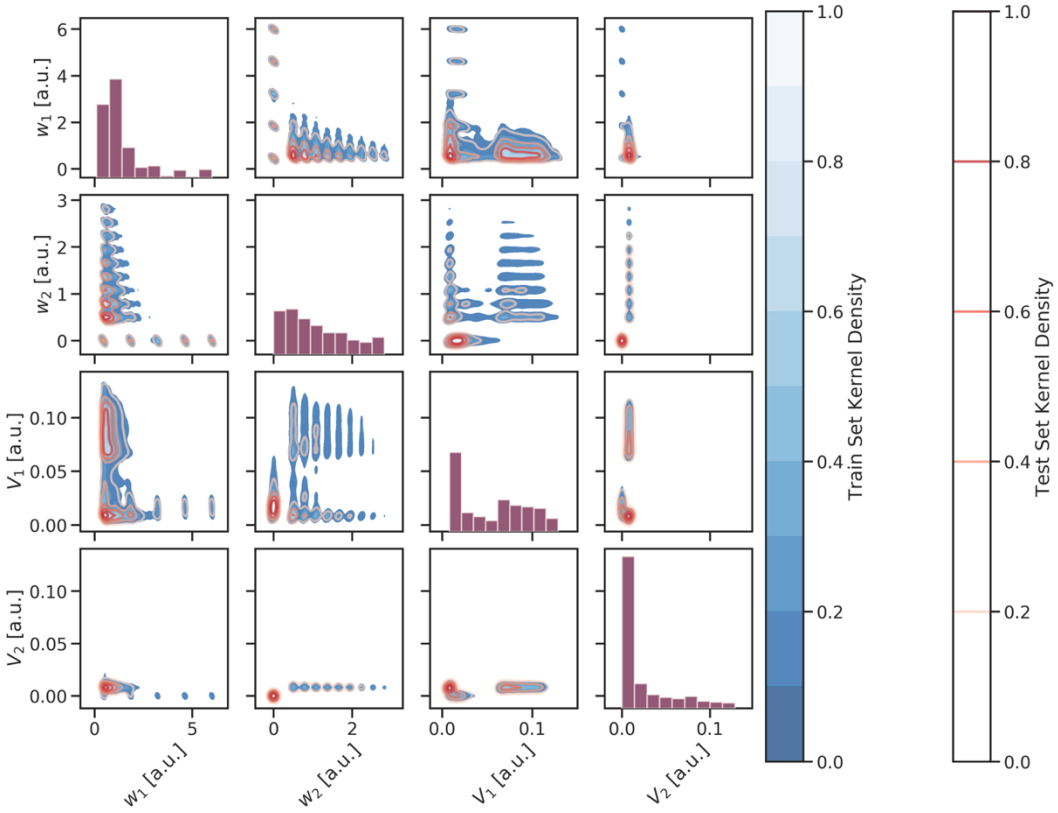

Figure A.1.2-b: Pair-wise distribution of a subset of the database features related to the barrier for the train and test sets. Blue surfaces are train set KDE, while red lines are test set KDE. KDE is normalized to 1.0. A KDE of 1.0 then indicates the region of two-feature space with the highest density of data.

When looking at the diagonal distribution of weights and heights in Figure A.1.2-b, there is a large number of points in the first bin. This comes from the fact that for single barriers the distance between barriers, the second barrier width and height are all equal to zero. Features which included the sum of the heights and the sum of the widths were considered to reduce this bias, however no significant improvement was observed in the trained neural network.

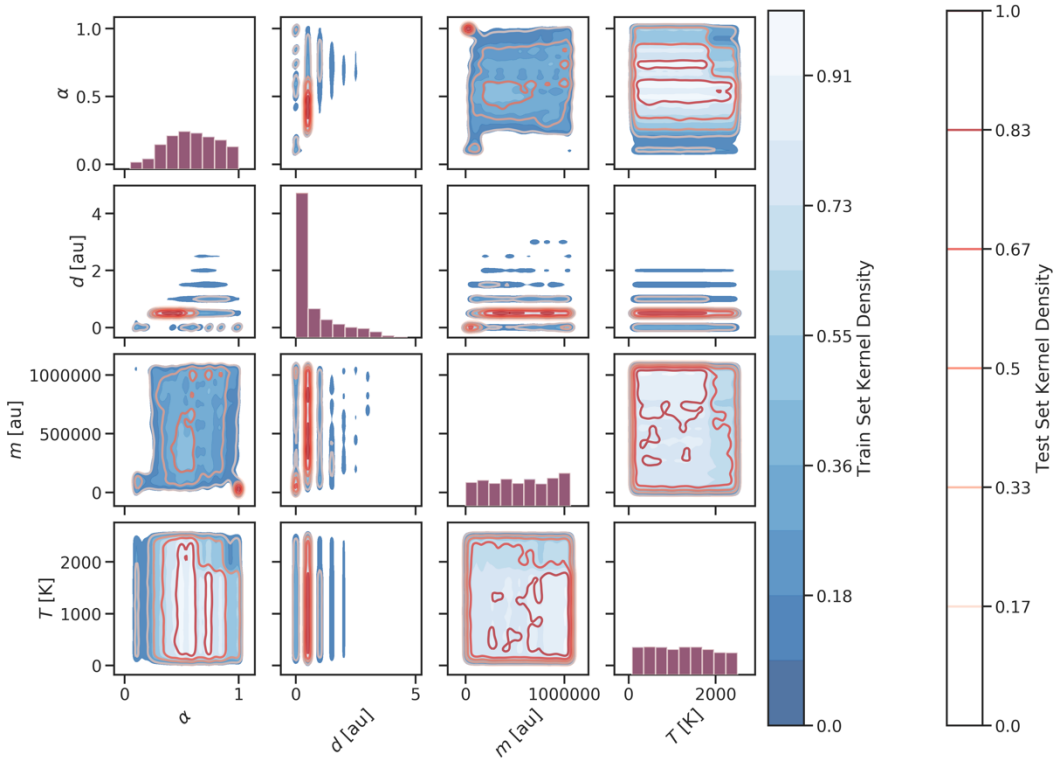

Figure A.1.2-c: Pair-wise distribution of a subset of the database features for the train and test sets. Blue surfaces are train set KDE, while red lines are test set KDE. KDE is normalized to 1.0. A KDE of 1.0 then indicates the region of two-feature space with the highest density of data.

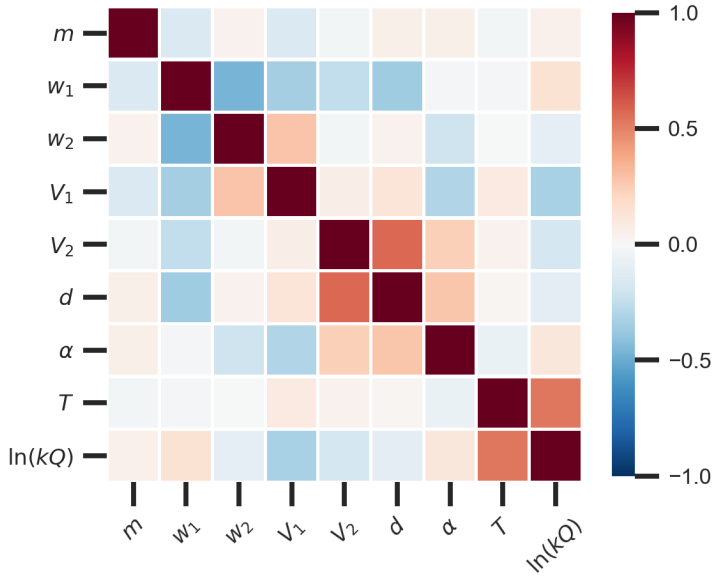

Figure A.1.2-d: Correlation matrix of the Pearson correlation coefficient for input features of the database. Values close to 1.0 indicate a strong positive correlation between variables, while -1.0 indicates a strong negative correlation. Features are perfectly correlated to themselves, as can be seen by the dark red diagonal. Most features have a small correlation, close to zero which indicates little bias.

A.1.3 Grid Search and Hyperparameter Tuning

As described in the first chapter of this thesis for the work regarding quantum reaction rate constant prediction, four subsequent grid searches were carried out. The hyperparameter search spaces for each are shown in Table A.1.3-a through d, and optimal hyperparameters are shown in Table A.1.3-e. Figure A.1.3-a indicates how the logarithm of the mean absolute error, MSE, averaged over all runs, changes with the number of neurons for each single hidden layer. Figure A.1.3-b shows how the loss changes with the learning rate. Figure A.1.3-c shows validation loss and moving average validation loss during training for a range of batch sizes, highlighting the optima.

A. First grid search over neurons, hidden layers, batch size, and epochs

For the first grid search the number of hidden layers, number of neurons per layer, batch size, and number of epochs were considered. The range of parameters is shown in Table A.1.3-a below and the results in the following Figure A.1.3-a. The optimal model was of neuron configuration (128, 24, 12).

Table A.1.3-a: First grid search space values for number of hidden layers, number of neurons per layer, batch size and epochs.

Hyperparameter	**Search space**
number of hidden layers	1, 2, 3
number of neurons in hidden layer	1, 6, 12, 24, 32, 64, 128
epochs	50, 100, 200, 300
batch size	32, 64, 128
Fixed hyperparameters	**Default values**
hidden layer activation function	sigmoid
output activation function	tanh
optimizer	Adam
- learning rate	0.001
- beta 1	0.9

- beta 2	0.9999
weight initialization	random uniform
- range	[0, 1]
loss function	mean squared error (MSE)

Optimal hyperparameters	**Values**
neuron configuration	(128, 24, 12)
epochs	300
batch size	32

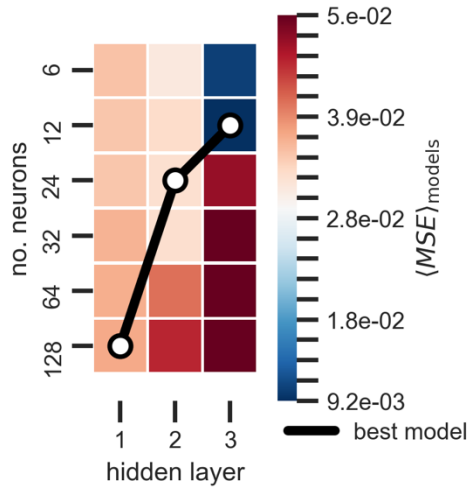

Figure A.1.3-a: Average model validation mean square error (MSE) associated with number of neurons in each of three hidden layers. The color scale corresponds to the average of the MSE over all models with three hidden layers trained for 300 epochs with batch size 32. The black line shows the neuron configuration of the optimal model.

B. Second grid search over activation function

For the second grid search, the choice of activation functions applied to each neuron in the hidden and output layers was explored. The activation functions tested are seen below in Table A.1.3-b. The optimum configuration was found to be sigmoid, linear for hidden layers and output layer, respectively. Other activations such as tanh also performed well in the output layer but only the linear function allows for predictions outside the testing the range.

Table A.1.3-b: Second grid search space values for activation function in both hidden and output layers.

Hyperparameter	**Search space**
activation function	softmax, softplus, softsign, relu, tanh, sigmoid, hard sigmoid linear
Fixed hyperparameters	**Default values**
neuron configurations	(128, 24, 12)
epochs	300
batch size	32
optimizer	Adam
- learning rate	0.001
- beta 1	0.9
- beta 2	0.9999
weight initialization	random uniform
- range	[0, 1]
loss function	mean squared error (MSE)
Optimal hyperparameters	**Values**
hidden layer activation	sigmoid
output layer activation	linear

C. Third grid search over learning rate

The third grid search optimized the value of learning rate for the Adam optimizer. The range of values explored were on the logarithmic scale to best search hyperparameter space and are shown below in Table A.1.3-c. This was conducted on a randomly selected portion of 30% of the training set as well as on 100% of the training set. In both cases, the Adam optimizer's default value of 0.001 was the top performer. The results are also shown in Figure A.1.3-b.

Table A.1.3-c: Third grid search space values for the learning rate.

Hyperparameter	**Search space**
learning rate	1E-5, 1E-4, 1E-3, 1E-2, 1E-1, 2E-1
Fixed Hyperparameters	**Default values**
neuron configuration	(128, 24, 12)
epochs	300

batch size	32
hidden layer activation function	sigmoid
output activation function	linear
optimizer	Adam
- beta 1	0.9
- beta 2	0.9999
weight initialization	random uniform
- range	[0, 1]
loss function	mean squared error (MSE)
Optimal hyperparameters	**Values**
learning rate	0.001

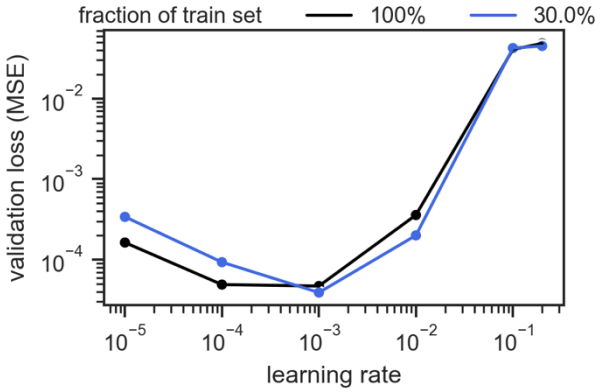

Figure A.1.3-b: Validation loss as a function of learning rate in the third grid search. The black and blue lines correspond to grid search on 100% and 30% of the training dataset. Both searches indicate that 0.001 is the optimal learning rate value.

D. Fourth grid search over batch size

A final grid search on the batch size was conducted to evaluate how fluctuations on the validation loss related to mini batch training batch sizes. Details of the search are described in Table A.1.3-d. A moving average of the validation loss was used as a metric, to average the impact of the fluctuations. It was found that 128, and 256 performed best in terms of reduction in fluctuation and final validation loss magnitude and given that 256

had a lower final rolling average loss it was chosen as the best batch size. This result is shown in Figure S 5.

Table A.1.3-d: Fourth grid search space for batch size.

Hyperparameter	**Search space**
batch size	32, 64, 128, 256, 512, 1024, 2048, 4096, 8192
Fixed Hyperparameters	**Default values**
neuron configuration	(128, 24, 12)
epochs	300
hidden layer activation function	sigmoid
output activation function	linear
optimizer	Adam
- learning rate	0.001
- beta 1	0.9
- beta 2	0.9999
weight initialization	random uniform
- range	[0, 1]
loss function	mean squared error (MSE)
Optimal hyperparameters	**Values**
batch size	256

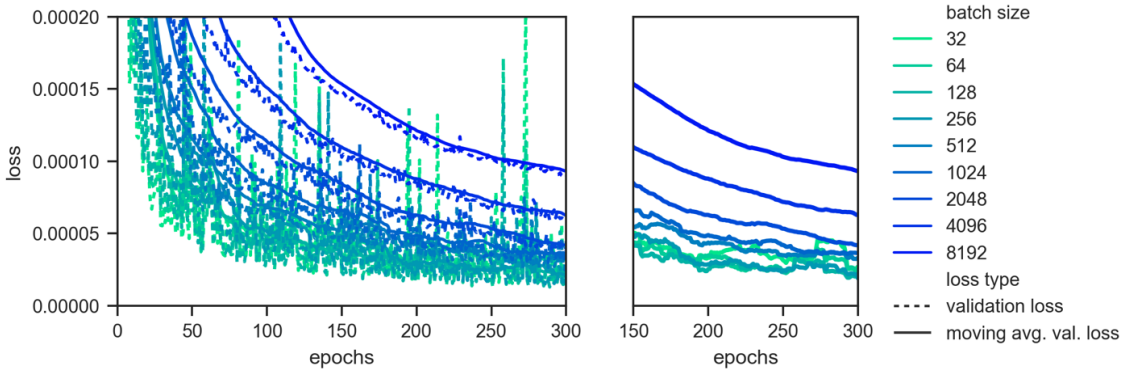

Figure A.1.3-c: Left panel) Validation loss and moving average of validation loss over 20 points for models trained with a range of batch sizes as a function of epochs. Batch size is seen to impact both the fluctuations in the validation loss, as well as the final training and validation loss value. Right panel) Moving average validation loss for the last 100 epochs of training. Batch sizes 128, and 256 perform best with final average losses of 2.0E-5 and 1.9E-5 respectively.

E. Optimal hyperparameters

Table A.1.3-e: Optimal hyperparameters identified from grid searches for the final training.

Hyperparameter	Value
neuron configuration	(128, 24, 12)
epochs	300
batch size	256
hidden layer activation function	sigmoid
output activation function	linear
optimizer	Adam
- learning rate	0.001
- beta 1	0.9
- beta 2	0.9999
weight initialization	random uniform
- range	[0, 1]
loss function	mean squared error (MSE)
Performance	
training loss	1.55E-5
test loss	1.57E-5
test MAE [log(1/ps)]	0.27

A.2 Partition Functions

A.2.1 Dataset

In Figure A.2.1-a the activation energies of the reactions in the dataset used to generate the partition function dataset is shown. Table A.2.1-a the partition function equations employed to compute $Q(T)$ are shown.

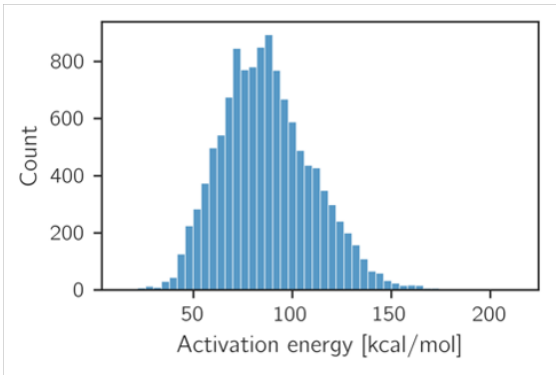

Figure A.2.1-a: Histogram of ground state activation energies of all reactions used to generate the partition functions dataset. The average value of the activation energies 85.3 kcal/mol is much higher than that of most organic chemistry reactions because these reactions involve breaking covalent bonds in unimolecular reactants.

Table A.2.1-a: Equations employed to compute the translational, vibrational, and rotational partition functions with the rigid body, harmonic oscillator, and rigid rotor approximations. Here m is the molecular mass, V the volume, ω are the wavenumbers, $\theta_i = h^2/8\pi^2 I_i$ where I_i is the moment of inertia and σ is the symmetry number. The electronic partition function was approximated as equal to the electronic ground state degeneracy, $g_{el,0}$.

	Partition function	
Translational	$Q_{trans} = V\left(\frac{\sqrt{2\pi m k_B T}}{h}\right)^3$	
Vibrational	$Q_{vib} = \prod_{i=1}^{3N-6} \frac{e^{-\hbar\omega_i/2k_BT}}{1-e^{-\frac{\hbar\omega_i}{k_BT}}}$	
Rotational	$Q_{rot} = \frac{\sqrt{\pi}}{\sigma}\sqrt{\frac{(k_BT)^3}{\theta_x\theta_y\theta_z}}$	3D structure
	$Q_{rot} = \frac{k_BT}{\sigma\theta}$	linear structure

A.2.2 Featurization

For the Qest model, a series of geometry based input features were screened which included Autocorrelation (AC) (J. P. Janet and H. J. Kulik, J. Phys. Chem. A 121, 8939 (2017)), EncodedBonds (EB) (C. R. Collins, G. J. Gordon, O. A. Von Lilienfeld, and D. J.

Yaron, J. Chem. Phys. 148, 241718 (2018), and Coulomb Matrix (CM) (M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, Phys. Rev. Lett. 108, 058301 (2012)) as well as Smooth Overlap of Atomic Positions (SOAP) (A. P. Bartók, R. Kondor, and G. Csányi, Phys. Rev. B 87, 184115 (2013)), and Many Body Tensor Representation (MBTR) (H. Huo and M. Rupp, Unified Representation of Molecules and Crystals for Machine Learning (2017)). To create these, the MolML (C. R. Collins, G. J. Gordon, O. A. Von Lilienfeld, and D. J. Yaron, J. Chem. Phys. 148, 241718 (2018)) and DScribe (L. Himanen, M. O. J. Jäger, E. V. Morooka, F. Federici Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke, and A. S. Foster, Comput. Phys. Commun. 247, 106949 (2020)) software packages were used. Learned featurization techniques were not considered. For each featurization feature and target standardization were tested considering min-max scaling and gaussian normalization. The model hyperparameters used during this screening are shown in Table A.2.2-a.

Overall, over 45 possible combinations of input feature, feature standardization, and target standardization were screened, finding that EncodedBonds with feature min-max scaling and target normalization were optimal. The average performance of the different featurizers is shown in Figure A.2.2-a. The screening of standardization was repeated for QesTS using EncodedBonds, which yielded no standardization as optimal.

Table A.2.2-a: Hyperparameters used during data screening.

Hyperparameter	**Value**
layer sizes	[200, 200]
learning rate	0.001

batch size	100
epochs	20
hidden activation function	relu
l2 regularization	0.0
bias initialization constant	1.0
weight initialization stdev	0.2
loss	mean squared error

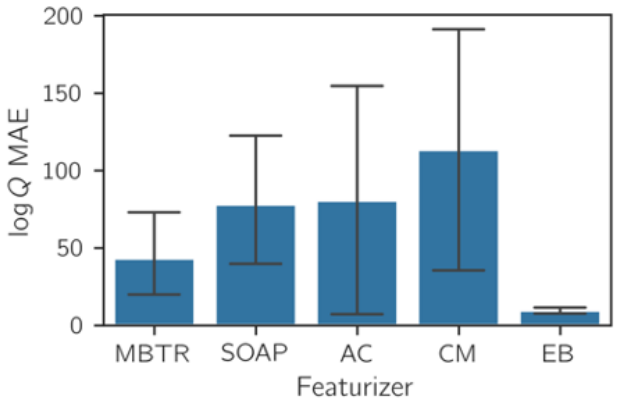

Figure A.2.2-a: Average performance of each featurizer tested during screening. Bars represent the cross validated mean absolute error, while error bars are the standard deviation across the data standardization combinations tested for each featurizer. Models using Encoded Bonds performed best on average. Also, the best performing combination used Encoded Bonds with feature min-max scaling and target normalization.

A. DScribe software

The DScribe software package (L. Himanen, M. O. J. Jäger, E. V. Morooka, F. Federici Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke, and A. S. Foster, Comput. Phys. Commun. 247, 106949 (2020)) was used to generate SOAP and MBTR input features. The

parameters used are listed below. The reader is referred to the software documentation for a description of these parameters.

SOAP – Smooth Overlap of Atomic Positions

The parameters used to produce SOAP features are listed in python dictionary format:

```
{
   'species': ["H", "C", "O", "N"],
   'rcut': 6.0,
   'nmax': 8,
   'lmax': 6,
   'average': 'inner'
}
```

MBTR – Many-body Tensor Representation

The parameters used to produce MBTR features are listed hereafter in python dictionary format:

```
{
      species=["H", "O", 'C', 'N'],
      k1={
      "geometry": {"function": "atomic\_number"},
      "grid": {"min": 0, "max": 8, "n": 100, "sigma": 0.1},
      },
      k2={
      "geometry": {"function": "inverse\_distance"},
      "grid": {"min": 0, "max": 1, "n": 100, "sigma": 0.1},
      "weighting": {"function": "exponential", "scale": 0.5,
"cutoff": 1e-3},
```

```
      },
      k3={
      "geometry": {"function": "cosine"},
      "grid": {"min": -1, "max": 1, "n": 100, "sigma": 0.1},
      "weighting": {"function": "exponential", "scale": 0.5,
"cutoff": 1e-3},
      },
      periodic=False,
      normalization='none',
      flatten=True
}
```

B. MolML software

MolML (C. R. Collins, G. J. Gordon, O. A. Von Lilienfeld, and D. J. Yaron, J. Chem. Phys. 148, 241718 (2018)) was used to define the EncodedBonds, Coulomb Matrix and Autocorrelation features. MolML determines parameters such as the element types present and the maximum number of atoms during featurization. Due to limited RAM, the entire dataset could not be loaded at once, so it was scanned for a set of 3 structures which contained both the full array of chemical elements in the dataset, as well as the largest molecular system. The featurizers were fit to this minimal set, and then used to transform the entire dataset in chunks. The minimum subset of systems is given below in xyz format and angstrom units:

```
14
C       -2.301345240000000      0.245555560000000       -0.146458820000000
O       -1.801117640000000      -1.068638690000000      -0.132535600000000
C       -0.698207700000000      -1.254313550000000      -0.983955260000000
```

C	0.498785910000000	-0.503501480000000	-0.581849450000000
C	1.474872740000000	0.110834400000000	-0.254355130000000
C	2.665120440000000	0.852752900000000	0.143128950000000
H	-2.602697440000000	0.544988140000000	-1.159271040000000
H	-3.175412910000000	0.262006730000000	0.502576870000000
H	-1.561197990000000	0.962926400000000	0.225284290000000
H	-0.479999560000000	-2.323300650000000	-0.969310430000000
H	-0.961290100000000	-0.982075860000000	-2.016315420000000
H	3.153957870000000	0.378701800000000	0.995343110000000
H	3.380779340000000	0.898604960000000	-0.679097780000000
H	2.407752320000000	1.875459330000000	0.422868020000000

10

O	2.288293630000000	0.484583070000000	-0.121712670000000
C	1.127501880000000	0.193607830000000	-0.071145460000000
C	-0.184064230000000	0.924152480000000	0.259412420000000
C	-0.873748740000000	-0.432000910000000	-0.015729600000000
N	0.451962880000000	-0.971798030000000	-0.300035950000000
H	-0.242967130000000	1.286299400000000	1.283947400000000
H	-0.427416880000000	1.726023410000000	-0.434781960000000
H	-1.362208720000000	-0.885687600000000	0.848217390000000
H	-1.550559340000000	-0.441221400000000	-0.871782480000000
H	0.773206630000000	-1.883958250000000	-0.576389100000000

23

C	2.349176060000000	0.084494630000000	-0.905389770000000
C	1.627675400000000	0.053488250000000	0.436342020000000
C	0.236369270000000	-0.590909200000000	0.414770920000000
C	-0.342881300000000	-0.607355670000000	1.828561880000000
C	-0.712898880000000	0.033731390000000	-0.629030880000000

C	-0.976517400000000	1.521383910000000	-0.403857370000000
C	-2.030065150000000	-0.732800180000000	-0.733683520000000
H	2.385265370000000	-0.910486730000000	-1.357281740000000
H	3.377237770000000	0.431331870000000	-0.786465570000000
H	1.861357880000000	0.752318050000000	-1.618183410000000
H	2.239225610000000	-0.501941370000000	1.154397840000000
H	1.554048780000000	1.069878880000000	0.838340710000000
H	0.375865430000000	-1.636038860000000	0.106313200000000
H	0.341081900000000	-1.112555420000000	2.513879630000000
H	-0.495666310000000	0.405190570000000	2.210740010000000
H	-1.298151790000000	-1.131761590000000	1.876033720000000
H	-0.213934640000000	-0.065520660000000	-1.599015070000000
H	-1.615287910000000	1.919106830000000	-1.195418250000000
H	-1.487633380000000	1.698516490000000	0.545714450000000
H	-0.054749190000000	2.106284580000000	-0.401858700000000
H	-2.609548020000000	-0.389412500000000	-1.593433690000000
H	-1.858128490000000	-1.805335340000000	-0.854936010000000
H	-2.651841000000000	-0.591607940000000	0.153459620000000

A.2.3 Optimization of hyperparameters

After screening for input features and standardization (Section A.2.2) a hyperparameter optimization was conducted. The search space for hyperparameter optimization is shown in Table A.2.3-a. The Tree Parzen (J. Bergstra, D. Yamins, and D. D. Cox, 28, (2013); F. Hutter, L. Kotthoff, and J. Vanschoren, Automated Machine Learning (Springer, 2019)) sampler along with the median pruning algorithm was used to identify the optimal hyperparameters (see Table A.2.3-b) with Optuna (T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (ACM, New York, NY, USA, n.d.)) using a 5 trial startup for both models. This corresponds to 1258 and 7685 completed trials for the Qest and QesTS model,

respectively. The number of epochs over which to train the model for the optimal hyperparameters was determined by using early stopping averaged over the 5 folds. From this it was found that 39 and 27 epochs were optimal stopping points for the Qest and QesTS models respectively.

Table A.2.3-a: Parameter space tested for hyperparameter optimization. Layer parameters such as bias initialization are applied to all hidden layers uniformly; the search space is restricted by not considering layers having different parameters. One to three total hidden layers are tested, and in each case all layers have the same number of neurons chosen from the range [2 – 2000]. MSE loss is used.

Hyperparameter	**Search space**	**Distribution**
layer sizes	1-3 layers, 2 - 2000 neurons	categorical choice
learning rate	(1E-5, 1E-1)	loguniform
batch size	(32, 8000)	uniform integer
hidden activation function	relu, tanh, softsign, sigmoid, softmax	categorical choice
l2 regularization	(0.0, 0.1)	uniform
bias initialization constant	(-1.0, 1.0)	uniform
weight initialization stdev	(1E-3, 1E+0)	loguniform

Table A.2.3-b: Optimum hyperparameters for both the Qest and QesTS DNN models found from TPE hyperparameter optimizations using Encoded Bonds as input features.

Hyperparameter	**Qest DNN**	**QesTS DNN**
layer sizes	[816, 816]	[749, 749]
learning rate	1.58E-4	3.13E-4
batch size	34	2609
hidden activation	relu	relu
l2 regularization	8.68E-6	4.20E-5
bias initial value	-0.664	-0.207

weight initialization stdev	0.407	0.059

A.2.4 Model Performance

The distributions of error on the test sets for the Qest and QesTS models are shown below.

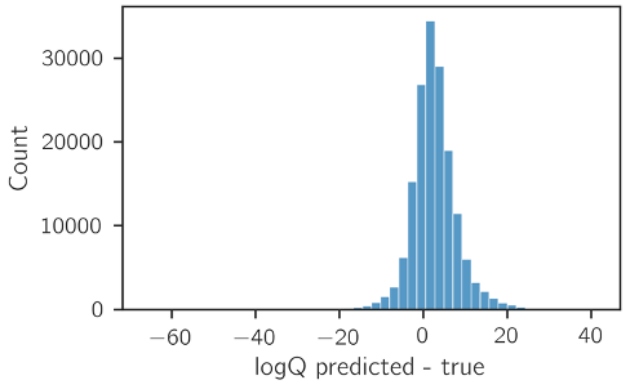

Figure A.2.4-a: Distribution of errors on the natural logarithm of the partition function for the Qest model. Most errors cluster near 0, however there is a small bias towards overestimating the partition function.

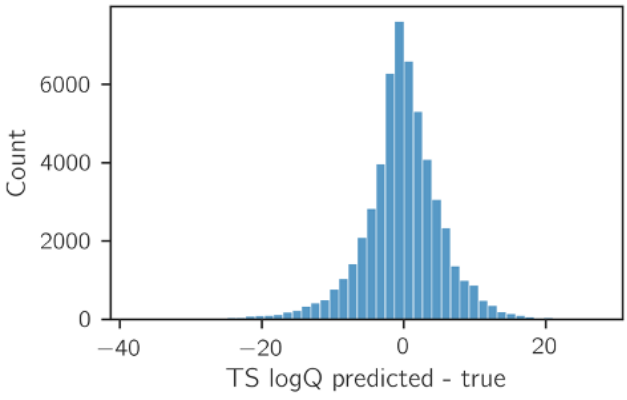

Figure A.2.4-b: Distribution of errors on the natural logarithm of the transition state partition function for the QesTS model. Most

errors cluster near 0. The model is unbiased, and the average error is -0.09.

A.2.5 Outliers

To limit extrapolation in the final model, outliers were dropped from the overall dataset as following. Reactions containing any example with 12 or more feature values 6 standard deviations away from that feature's mean were removed. This resulted in removing 1,108 reactions.

APPENDIX B

B.1 C-HACK event details

The fully virtual event took place over the first two weeks of the student's winter quarter. An overview of the event is shown in Figure 2-1 of the main document. First, we describe the organizational period of the event (Figure 2-1a) such as how committees were formed to create the content, how the schedule was created, and how the event was advertised and incentivized. We then discuss the tools used to facilitate learning and introduce data science. Next, we detail the tutorial content and format (Figure 2-1b), and finally the structure of the hackathon and the problems that were presented to the participants (Figure 2-1c).

B.1.1 Event planning

The event was organized over about six months to address the objectives. The organizational steps are shown in Figure B.1.1. Industrial representatives were contacted to find a partner that could provide data, participate in the organization of the event, and interact with participants. Dow agreed to provide industrial data and organizational time. The team of organizers discussed an emphasis on team dynamic soft skills that industry looks for in potential hires, which led to Objective 2 and the hackathon structure. Here, students work in teams to use data science to solve open ended problems on the Dow data, and present their work to judges (See section B.1.5 Hackathon). To provide motivation in the coopetition setting,[166] we chose to have monetary and social incentives.[291] Participants received a free tee-shirt and a mug, and those that turned in work were given an official certificate of completion and the option to advertise their work on a website. The certificate gave them the option to mention their participation on their resume. Students could also receive monetary prizes and awards for exceptional work. The C-HACK event was conducted with these industry partners in 2021 and in 2022, with both years organized by the same methodology, however the outcomes discussed in the main document and the below event details are the specifics of the 2022 event.

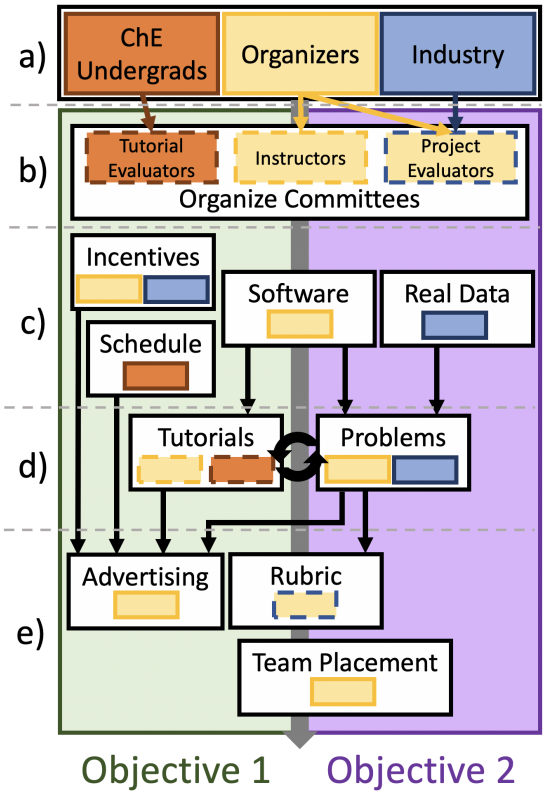

Figure B.1.1: Workflow of the planning of the event over about 6 months to address the objectives. a) Chemical Engineering undergraduates, a group of organizers, and a group of industry representatives contributed to the event. b) Committees were created to help produce the event. A set of ChE undergrads volunteered to give feedback on tutorial material. Tutorial instructors volunteered to create and deliver the material, including a group of graduate students to assist participants during the tutorials. Volunteer faculty, post-docs, and industry representatives were to evaluate hackathon problems. Box colors in a) and b) of the figure are a key for who contributed to the remaining organization. c) Administrative organization such as how students would be incentivized in a problem based learning setting, what times worked best for students to maximize participation, what software would be easiest for participants to quickly be able to access and learn data science, and what industrial data could be used to create problems. d) Tutorials general content and format was determined, problems were crafted based on the data and the material that could be covered, and tutorials were updated to best prepare students for the problems. e) The event was advertised to students, and final details such as a rubric to judge the final projects and how best to organize student teams was determined.

We did not want the event to appeal only to those already familiar with data science in accordance with Objective 1, so the first part of the event was a series of tutorials to introduce and explore data science tools. To maximize participation, polls were given to undergraduate students to determine what logistic choices would be most appealing. These included determining what time during the academic quarter students would be most free, what times of day during which tutorials could be conducted given class schedules and

student desires, rest periods between tutorials and the hackathon portion of the event, and how much time they would be willing to commit. Poll results contributed to the hackathon structure with one week of tutorials and one week of coopetition. We wanted these tutorials to be encouraged but not mandatory, so that participants could join for any content they were unfamiliar with.

Volunteer committees were organized to help conduct the event (Figure B.1.1-b). To align with Objective 1, a committee of instructors was organized to produce and deliver the tutorial content, and a set of undergraduate tutorial evaluators volunteered to give feedback on quantity and clarity of tutorial content before it was delivered. A group of graduate students familiar with the tutorial content volunteered to assist with the tutorials. A committee of post-doc, faculty, and industry judges worked together to design and use a rubric to evaluate the final deliverables of the team projects.

The event was held entirely virtually, and the software tools used to conduct the event and facilitate learning were chosen to make it as easy as possible to participate (see section B.1.2 Choice of Software). Congruently, registration was free, easy, and took place through an online registration tool. Registration was open for any undergraduate student in the department of Chemical Engineering at the University of Washington, and the Chemical, Biological, and Environmental Engineering department at Oregon State University. The event was advertised live and asynchronously on both campuses. This included fliers on campus, flash presentations before core classes, and open info sessions. We focused on motivating the students with what they would receive for participating, and the emphasis that no experience was required to align with Objective 1. A website was created as a hub for information on the event including the timeline, how to register, a code of conduct, and

the incentives for participating. Those who registered had access to a group messaging platform to discuss the event and ask questions.

The basic content structure of the tutorials was determined to introduce data science tools on the 1-week timeline using the chosen software. To make tutorials as active as possible, we chose to have each be an interactive learning engagement followed by studio problems. See section B.1.3 Tutorials for details on the content and structure. Two problems to choose from were designed for the hackathon. One was more straightforward and the other potentially more challenging to give teams of all levels opportunities to apply their knowledge by Objective 1. Both were open-ended questions about the data provided by Dow and aligned with their original industrial objective for collecting the data (See section B.1.6 Open-ended problems). A rubric was created to allow for uniform scoring of student's work for top performing prizes and awards. Tutorials were updated in order to introduce concepts that might be helpful for working with these data and problems. See section B.1.7 Connecting the problems to the tutorials for details. We determined a strategy of team formation based on best practices (See section B.1.4 Team placement).

We chose to conduct an opening ceremony before the tutorials where the timeline, format, and resources were discussed. During the ceremony students could interact with the industry sponsors for the first time. The data and hackathon problems would be revealed two days after the tutorials ended, so that competition played no hindrance in learning and was only introduced once teams had been formed for students to work together within a coopetition setting. Documents provided to the participants such as the agenda, instructions, and code of conduct can be found in the event repository.[173]

B.1.2 Choice of Software

Jupyter notebooks[292] using Google Colab were chosen to give participants access to data science tools through Python for both the tutorials and the hackathon. This means that students could run code without having to install any software: Colab requires only an internet connection and can even be used on a smartphone. Because this was done as an enrichment activity from self-selecting participants, FERPA is not an issue in using Colab. This ease of use of this toolset makes it ideal for increasing participation among under-supported groups. Jupyter notebooks are also a highly interactive format of coding, where the user can execute code snippets and view the results in real time.[293,294] This includes numerical results and visualizations which participants can use to guide their understanding and make meaningful conclusions. Colab is also amenable to collaboration, allowing students to remotely work together in real time or asynchronously. Google Drive was chosen to share all documents and data as it is easily accessible with the participants' institutional emails and is connected to Colab. Zoom was chosen to host all meetings as it was provided for free to students at both institutions and allowed for the recording of meetings for students to refer to later. A Slack organization (group messaging service) was created for participants to communicate with each other and with organizers. With Slack they could discuss the format of the event or ask coding related questions. All of these tools can be accessed using just an internet connection.

B.1.3 Tutorials

Five two-hour tutorials were given over five weekdays in the evening on Zoom. These each consisted of 50 minutes of discussion-like active learning experiences, then a 20 minute break, followed by a set of studio problems that students could work on together in small groups. Attendance was not mandatory but encouraged, so that individuals of all

proficiency levels could develop their coding skills. The topics covered in each tutorial are outlined below:

- Introduction to Python – how to use Python in Jupyter notebooks on Colab, Python syntax, variables, data types, and code commenting
- Flow control – conditional statements, loops, functions and basics of functional programming
- Data structures – organizing, storing, and working with data, NumPy[295] and Pandas[296] libraries
- Visualization – producing informative visual communications of data or results, Matplotlib[297] library
- Project dependent topic – statistics, regression, unsupervised clustering algorithms

Each tutorial was given in the form of a Colab notebook that was reviewed by the undergraduate tutorial evaluation committee. The specific content was chosen to give participants a pure introduction to coding and build up to using Python to conduct data science tasks, so that participants of any level could benefit. No 10-hour course can extensively explore data science using Python, so we focused on introducing specific tools and applications to enable students to work on the hackathon problems at a basic level and apply elsewhere. Links to additional resources such as explanations and useful Python libraries were included to give participants more opportunity to explore. The tutorials were designed to be continuous such that each day built on previous days. To maximize continuity, we used the same dataset to explore different Python tools throughout the tutorial. The lecture notebooks contain pre-filled instructional content described by an instructor, as well as active exercises the instructor completed with students. Questions, pauses, and tangents were highly encouraged to maximize inquiry and exploration by the students. The lectures were all recorded for participants that could not attend the tutorials. After each active learning portion and a break, students were split into small groups of 3-5

and given topic related coding problems to solve for 50 minutes. The small group format was chosen as it has been shown to provide the benefits of cooperation on learning.[298] Assistant instructors familiar with the material were available to give guidance and facilitate cooperation. Given that Zoom allows for screen sharing, and the work is done entirely on a virtual notebook, students and instructors could display their work and progress. The tutorial notebooks are openly available in the event repository.[173]

B.1.4 Team placement

We wanted the format of the teams for the hackathon to provide the best opportunity for participants to develop team dynamic soft skills. We chose to have a maximum team size of four participants. Participants were allowed to sign up as a team, as self-selection has been linked to positive performance.[298] In accordance with Object 1, we also enabled individual sign ups to encourage those without previous coding experience or interested friends to participate. We took special care to balance the solo participants that did not have teammates. Each participant filled out a survey asking for their class level, gender identity, and comfortability with Python on a scale from one to five. An in-house Python tool was created to sort through participants and place them on open teams to minimize the standard deviation of all team's "scores" which were computed as a weighted sum of participants' class level and Python skills. This is especially important because undergraduate students of any age were encouraged to participate, and we wanted to ensure that younger students had opportunities to learn from the soft skills of their older peers, and new coders from experienced coders. Finally, the tool enforces that no self-identified woman or non-binary individual was on a team alone, as these voices have shown to be overshadowed by men

when singled out in STEM settings.[29] The Python tool is freely available in the event repository.[173]

B.1.5 Hackathon

Two days after the tutorials had concluded the representatives from Dow announced a real dataset and two open ended problems related to the data to choose from. Documents describing the problems and the data was shared with the participants via Google Drive. A folder was shared uniquely with each team for them to work together in and turn in their work. Teams had four days to address the problem statements using Python and provide the work they completed in the form of a Colab notebook and a short 1-page description of what they did. Finally, two days after the work was turned in, the students were asked to give a 10-minute presentation on their work to a collection of evaluators. The evaluators gave scores to teams based on their coding work and final presentation according to a rubric. Scores were normalized and averaged to award first through third place teams cash prizes for each of the two problems. Teams also received certificates for their project description and presentation. Detailed instructions, and templates were provided to the participants. These and the rubric can be seen in the event repository.[173]

B.1.6 Open ended problems

As opposed to highly structured problems, open ended problems give students an opportunity to engage and develop their own problem solving skills. Problem solving is touted as the central theme of engineering, but it is unclear how to best design an open ended problem to develop problem solving skills.[299] Not all engineering problems are the same, and so not all problems draw on the same set of proficiencies. Instead, they tend to be ill-structured, where not all elements of the problem are known with confidence, criteria

for success may be variable or unknown, and do not possess a set solution path.[300] Jonassen[300] identifies problem types such as design, selection, and troubleshooting, among others. To align with Objective 2, the hackathon problems should mimic the open-ended qualities listed above, with the only difference being that participants have access to professionals who have worked on the data previously.

The dataset provided to the students is a set of measurements taken from 33 positions spatially distributed within a rectangular basin in Dow's East Waste Treatment Plant (EWTP). EWTP is a typical activated sludge aeration basin[301] used to treat organic pollutants before discharging effluent into receiving waters. Outfall at EWTP would intermittently experience elevated solids and total organic carbon, regulatory compliance parameters, which could result in production curtailment. The samples were collected via a drone and measurements include temperature, suspended solids, metal concentrations, pH, among others, as well as bacterial abundance at taxonomic levels from kingdom down to species. Some classes of bacteria could not be identified so remained as "Unknown". The true identities of the bacteria and metals were replaced with encodings for proprietary reasons. Code and a starter notebook were provided to the participants to expedite data loading from file and aggregation to a desired taxonomic level.

The students were asked to use this information to explore relationships within the basin and comment on potential causes. In Problem 1, participants were tasked with producing a metric for basin performance and using this to visualize the operation of the basin over space. This provides a means for participants to make qualitative assessments based on their own metrics. In Problem 2, participants were asked to explore the distribution of bacterial abundance across the basin, identify any heterogeneity or clusters

in the bacterial population, and identify any association between bacterial abundance and metal concentrations. Data science methods discussed in the tutorials could give insight on these tasks, but there was no "known" final answer that the students had to reach, instead they had to make and justify their own conclusions. The detailed description of the basin, problem statements, and provided code are available in the Supporting Documents.[173]

These two problems fall in the category of troubleshooting: the waste basin is experiencing issues and the participants are asked to investigate the cause. They are mostly unstructured in that there is no fixed solution path, and there is no strictly defined definition of success. Participants must explore the data and come up with their own strategy to understand the basin's function. We would like to note that the focus of this communication is not on the specific problem. The most important aspects of the problems are the use of industrial data to explore a real problem, and the open-ended nature of the tasks. This gives teams the opportunity to use the tools available to them and their intuition to extract relevant information from the real-world in IBL fashion, as they would if working on a team in industry. Readers looking to conduct a similar event should focus on finding an industry partner that can provide a dataset with these qualities, not the particular problem presented here.

B.1.7 Connecting the problems to the tutorials

To provide the participants with knowledge of potentially useful tools related to these problems, the tutorial content was created with the problems in mind. Specifically, Tutorial 4 gave particular emphasis to visualizing data in space and in creating three dimensional plots. Tutorial 5 covered some basic statistical topics, discussed unsupervised machine learning including basic clustering algorithms. Lastly, the dataset used throughout the

tutorials was a record of storm events in the United States from the NOAA archive,[302] which was formatted in latitude longitude space similar to how the problem dataset was formatted in depth, width, length of the basin. These choices ensured that the participants were introduced to tools that might be helpful when addressing the problems.

APPENDIX C

C.1 learn2therm Database

C.1.1 Pipeline steps and parameters

The pipeline for downloading raw data, to validation of learn2therm protein pairs is runnable with a single command: ‘dvc repro’. Alternatively one can run individual stages of the pipeline with ‘dvc repro -s <name_of_stage>’. Many of the stages in the pipeline have parameters that affect the runtime behavior and stage outputs, thus the pipeline is tunable by modifying ‘params.yaml’. Each stage in the pipeline is outlined below, and the parameters are given in Table C.1.1-a.

Data ingestion

1. `**s0.0_get_raw_data_taxa.py**`
   Pull most recent NCBI 16s r RNA sequences, and OGT records from Enqvist
   - Params: `min_16s_len`, `max_16s_len` number of nucleotides required to keep and organism
   - Outputs: `data/taxa.parquet`, columns include OGT, 16s sequence, taxid, other taxonomy
   - Metrics: `n_taxa` total number of labelled organisms, `taxa_pulled_date` when the data was retrieved

2. `**s0.1_get_raw_data_proteins.py**`
   Retrieve single cell uniprot. Uses FTP to download very large uniprot files
   - Inputs: `data/taxa.parquet`
   - Outputs: `data/taxa/uniprot/uniprot_pulled_timestamp` indicates when files were pulled
   - Metrics: `taxa_pulled_date` when data was retrieved
   - Untracked Outputs: The script produces `*.xml.gz` files that are untracked because they take so long to download. DVC ignores them, subsequent calls to the script skip downloading files already present.
3. `**s0.2_get_proteome_mdata.py**`
   Get metadata for UniProt proteomes. Selects one "best" proteome per organism.
   - Outputs: `data/uniprot/proteome_metadata.csv`, columns include taxa pair file, number of hits, emissions, total searchable space
4. `**s0.3_parse_proteins.py**`
   Extract minimal protein data and store in an efficient file format. Skip proteins that we don't have OGT for or are from redundant proteomes
   - Params: `max_prot_per_file` size of parquet files
   - Inputs: `data/taxa/uniprot/uniprot_pulled_timestamp`, `data/taxa.parquet`
   - Outputs: `data/proteins`, contains proteins in chunked files of the form `*.parquet`. Columns include protein sequence, database identifiers, and associated taxa IDs, `./data/metrics/s0.3_protein_per_data_distr.csv` table of number of proteins per taxa
   - Metrics: `n_proteins` total protein count, `percent_prot_w_struc` fraction of proteins with PDB or AlphaFold id

Data pairing

5. `**s1.0_label_taxa.py**`
   Assign booleans for taxa as thermophile
   - Params: `ogt_threshold` binary thermophile threshold
   - Inputs: `data/taxa.parquet`
   - Outputs: `data/taxa_thermophile_labels.parquet`
   - Metrics: `n_meso`, `n_thermo`
6. `**s1.1_get_16s_blast_scores.py**`
   Compute pairwise BLAST pairings of meso vs thermo 16s rRNA sequences.
   - Params: `16s_blast_parameters` (there are a number), `blast_metrics`
   - Inputs: `data/taxa*.parquet`, `./data/metrics/s0.3_protein_per_data_distr.csv` (used to skip alignment if taxa has no proteins)
   - Outputs: `data/taxa_pairs/alignment/*.parquet` table of taxids and BLAST scores.
7. `**s1.2_label_all_pairs.py**`
   Create a list of taxa pairs that meet a minimum 16s rRNA BLAST score.
   - Params: `blast_metric_thresholds` defines thresholds on 16s blast metrics to consider a pair
   - Inputs: `data/taxa_pairs/alignment/*.parquet`

- Outputs: `data/taxa_pairs/pair_labels/*.parquet` 1:1 index mapping to data/taxa_pairs/alignment/*.parquet of boolean labels of whether that taxa are a pair
- Metrics: `num_taxa_pairs_conservative` number of pairs that passed thresholds. Only this number will we blastp, `taxa_pair_found_ratio` fraction of pairs with metrics eg n_taxa_pairs/(n_therm*n_meso) that will be searched for protein pairs

8. `**s1.3_protein_alignment.py**`
   Runs a massive parallel cluster to align protein pairs among taxa pairs using DIAMOND.
   - Params: `dask_cluster_class` class for cluster in dask_jobqueue, `max_protein_length`, `method` local aligner type, `n_jobs` parallel workers, each doing a taxa pair at a time, `method_X_params` where X is eg. "blast" params given to aligner, `blast_metrics` alignment metrics to record.
   - Inputs: `data/taxa_pairs/alignment/*.parquet`, `data/taxa_pairs/pair_labels/*.parquet`, `./data/proteins`
   - Outputs: `data/protein_pairs/*.parquet`, each file is an aggregation of alignments for a number of taxa pairs with protein pids, source organism taxids, and alignment metrics
   - Metrics: `protein_align_X`, where X is "emissions", "hits", "time", "return". The resources and used and return on investment of protein alignment.
9. `**s1.4_make_database.py**`
   Collect processed data files into a relational duckdb database.
   - Inputs: `data/taxa.parquet`, `data/taxa_thermophile_labels.parquet`, `data/protein_pairs/*.parquet`, `data/proteins/*.parquet`, `data/uniprot/proteome_metadata.csv`
   - Outputs: `data/database.ddb` relational duckdb database of taxa, proteins, taxa pairs, and protein pairs

Data validation

10. `**s2.1_get_hait_pairs.py**`
    Parse the protein pairs from Hait et al's excel files, query the PDB ids to get sequences.
    - Outputs: `data/validation/hait_pairs.csv`, columns include meso PDB id, thermo PDB id, and sequences
11. `**s2.2_compare_to_Tm.py**`
    Compare melting temperatures from FireProtDB and Meltome Atlas to OGTs in the dataset.
    - Inputs: `data/database.ddb`
    - Metrics: `Tm_OGT_spearman_p`, `Tm_OGT_spearman_r` Spearman R and null probability of relationship between OGT from our data and melting temperature from 3rd party dataset
12. `**s2.3_run_hait_alignment.py**`
    Compute the alignment metrics for Hait et al. pairs using identical parameters as the alignment metrics for the full dataset.
    - Params: `method` local aligner type, `method_X_params` where X is eg. "blast" params given to aligner, `blast_metrics` alignment metrics to record.

- Inputs: `data/validation/hait_pairs.csv`
- Outputs: `data/validation/hait_aligned_scores.csv`, columns include protein ids and alignment metrics

13. `**s2.4_compare_hait_alignment.py**`
Compare metrics for BLAST alignments of Hait et al. pairs to our dataset, and make some plots.
    - Inputs: `data/validation/hait_aligned_scores.csv`, `data/database.ddb`
    - Outputs: `data/validation/hait_alignment/*.png` Distribution comparison of alignment metrics, and size of your dataset if we were to use Hait scores as a filter.
14. `**s2.5_get_HMM_profiles.py**`
Download Pfam HMMs.
    - Outputs: `./data/validation/hmmer/Pfam-A.hmm`
    - Metrics: `HMM_pulled_date` date of Pfam aquisition
15. `**s2.6_hmmer_hait.py**`
Run Pfam against proteins in Hait et al. pairs, compute Jaccard of annotations.
    - Params: `e_value` Maximum e-value for hmmer to report an annotation, `njobs` cores for parallel
    - Inputs: `data/validation/hait_pairs.csv`, `./data/validation/hmmer/Pfam-A.hmm`
    - Outputs: `data/validation/hait_scores.csv`, columns include jaccard score of Pfam annotations
    - Metrics: `mean_jaccard` Mean jaccard score of annotations over Hait pairs, `fraction_found` Fraction of proteins with at least one Pfam annotation
16. `**s2.7_run_hmmer.p**y`
Scan Pfam against all proteins on our database that are in protein pairs.
    - Params: `e_value` Maximum e-value for hmmer to report an annotation, `njobs` cores for parallel, `scan` boolean to hmmscan or hmmsearch, `prefetch` whether to prefetch HMMs into memory or leave as file iterator, `chunk_size` size of protein chunks to run and save to file at one time.
    - Inputs: `data/database.ddb`, `./data/validation/hmmer/Pfam-A.hmm`
    - Outputs: `data/validation/hmmer_outputs/*.parquet` Pfam annotations for each protein, columns include the PID and a ‘accessions’, a semicolon separated string of Pfam accessions annotated for the PID
    - Metrics: `n_proteins_in_pairs` number of proteins in pairs, `n_proteins_labeled` number of proteins with at least one Pfam annotation.
17. `**s2.8_parse_hmmer_result.py**`
Parse hmmer results into a table of protein pairs and their Jaccard scores of annotations
    - Inputs: `data/validation/hmmer_outputs/*.parquet`
    - Outputs: `data/validation/hmmer_labels/*.parquet`, columns include meso PID, thermo PID, and Jaccard overlap of Pfam annotations
    - Metrics: `mean_jaccard` Mean jaccard score of annotations over protein pairs, `fraction_found` Fraction of proteins with at least one Pfam annotation
18. `**s2.9_compare_hait_hmmer.py**`
Compare Pfam annotations of Hait et al. pairs to our dataset, and make some plots.
    - Inputs: `data/validation/hmmer_labels/*.parquet`, `data/database.ddb`

- Outputs: `data/validation/hmmer/compare_jaccard_hist.png` Distribution comparison of alignment Jaccard scores for Hait et al. pairs and our pairs.
- Metrics: `t_pvalue_base` p-value of t-test of Jaccard scores between Hait et al. and our pairs, `t_pvalue_95` p-value of t-test of Jaccard scores between Hait et al. and our pairs with > 95% blast coverage

19. `**s2.10_sample_data_for_structure.py**`
Sample some protein pairs to conduct structural alignment on, uniform over BLAST coverage.
    - Params: `sample_size` number of protein pairs to sample, `metrics` list of queries to make from pairs table to sample pairs uniformly over
    - Inputs: `data/database.ddb`
    - Outputs: `data/validation/structure/sample_l2t_data.csv`, a sample of pairs from the pairs table, columns include meso and thermo PID, and metrics specified in params
20. `**s2.11_structure_hait.py**`
Run FATCAT structural alignment for Hait et al. pairs by getting PDB structures.
    - Inputs: `data/validation/hait_pairs.csv`
    - Outputs: `data/validation/structure/hait_fatcat.csv`, column include the P-value from FATCAT for the alignment
21. `**s2.12_structure_l2t.py**`
Run FATCAT structural alignment for L2T pairs by getting PDB or AlphaFold structures.
    - Inputs: `data/validation/structure/sample_l2t_data.csv`
    - Outputs: `data/validation/structure/l2t_sample_fatcat.csv`, column includes meso and thermo PIDs, metrics originally specified to sample uniformly from in s2.10

Parameters

The full list of parameters and their current values for this work is given in Table C.1.1-a below:

Table C.1.1-a: All parameters in the pipeline to modulate the final result.

Stage	**Parameter name**	**Description**	**Published Value**
get_raw_data_taxa	min_16s_len	Minimum number of bases in 16s rRNA sequence to keep the sequence	1300
	max_16s_len	Maximum number of bases in 16s	1600

		rRNA sequence to keep the sequence	
parse_proteins	max_prot_per_file	Number of proteins per chunk for saving UniProtKB as minimal table	100000
label_taxa	ogt_threshold	Binary threshold to consider a taxa thermophilic, not inclusive	40.0
get_16s_blast_scores	num_threads	BLAST cpus for computation	20
	word_size	Size of initial exact nucleotide matching in BLAST	28
	gapopen_penalty	Penalty for bit score incurred by starting a gap in the alignment	2
	gapextend_penalty	Penalty for bit score incurred by extending a gap in the alignment	1
	reward	Score increase for a match in alignment	1
	penalty	Penalty for mismatch in alignment	-2
	ungapped	Whether to do an alignment without allowing gaps	false
	blast_metrics	List of metrics defined in learn2therm.blast.BlastMetrics to compute for 16s rRNA alignment	* See note 1 below

label_all_pairs	blast_metric_thresholds	Nested dictionary of name of alignment metric and two fields "thresh" which is the threshold for considering a pair, non inclusive, and "greater", a boolean of whether we want bigger or smaller than the threshold	* See note 2 below
get_protein_blast_scores	dask_cluster_class	Class from Dask Jobqueue[1] to use for parallel workers	SLURMCluster
	max_protein_length	Maximum number of amino acids, inclusive, to keep for protein pair search	
	method	Which aligner to use, one of 'blast' or 'diamond'	diamond
	n_jobs	Number of independent parallel workers to keep running	80
	save_frequency	Number of taxa pairs between DVC checkpointing	20000
	method_blast_params	Parameters for BLASTp algorithm	See Table S2 below.
	method_diamond_params	Parameters for DIAMOND algorithm	See Table S2 below.
	blast_metrics	List of metrics defined in learn2therm.blast.	* See note 1 below

		BlastMetrics to compute for protein alignment	
run_hmmer	e_value	Maximum E value for labelling Pfam annotations with HMMER	1.e-10
	chunk_size	Number of vector chunks to run at a time	2000
	prefetch	Boolean of whether to load HMMs into memory, or leave as disk iterator	true
	njobs	Number of CPUs to use for parallel pyhmmer	32
	scan	Whether to use HMMScan, alternative HMMSearch	false
sample_data_for_structure	sample_size	Number of protein pairs to keep for structural alignment	10,000
	metrics	Queries of protein pairs table to compute and use for sampling, the columns is binned into 5 bins and samples uniformly from those bins	[“(query_align_cov+subject_align_cov)/2.0"]

Table C.1.1-b: Parameters for BLASTp, if using it for protein alignment

Parameter name	**Description**	**Published Value**

num_threads	CPUs for each worker to run BLASTp	6
word_size	Size of initial Amino Acid matching in BLASTp	3
gapopen	Penalty to bit score for opening a gap in alignment	11
gapextend	Penalty to bit score for extending a gap	1
matrix	Substitution matrix for comparing amino acids in alignment	BLOSUM62
threshold	Minimum score for word to be added to lookup table	11
ungapped	Whether to run alignment without gaps	false
evalue	Maximum E value to keep scoring alignment	0.00001
qcov_hsp_perc	Minimum percent coverage of query strand to keep alignment	75

Table C.1.1-c: Parameters for DIAMOND, if using it for protein alignment

Parameter name	**Description**	**Published Value**
num_threads	CPUs for each worker to run DIAMOND	6
sensitivity	How sensitive the search should be for initial matching	ultra-sensitive
gapopen	Penalty to bit score for opening a gap in alignment	11
gapextend	Penalty to bit score for extending a gap	1
matrix	Substitution matrix for comparing amino acids in alignment	BLOSUM62
iterate	Whether to start with lower sensitivity alignment	false
global_ranking	Limit on the number of Smith Waterman extensions per query, with the target sequences ranked by their ungapped extension scores.	null
evalue	Maximum E value to keep scoring alignment	0.00001
hsp_cov	Minimum percent coverage of query and subject strand to keep alignment	75

* *Note 1:* Any metric that is a method name of the class learn2therm.blast.BlastMetrics can be specified here. See Section C.1.2 for definitions of these metrics. The values used in this work are: [local_gap_compressed_percent_id, scaled_local_query_percent_id, scaled_local_symmetric_percent_id, local_E_value, query_align_start, query_align_end, subject_align_end, subject_align_start, query_align_len, query_align_cov, subject_align_len, subject_align_cov, bit_score]

* *Note 2:* To threshold by E value less than $1e^{-10}$, we would have a nested specification like the following:

```
blast_metric_thresholds:
  local_E_value:
    thresh: 1e-10
    greater: false
```

C.1.2 Alignment metrics

A number of metrics were computed for pairwise alignments of 16s sequences and proteins, and are given in the database. Their definition, description, and database tag is provided below.

Given an alignment of length L including gaps upon query sequence and subject sequence with lengths A and B, a column at position $i \in [0,L]$ has:

- $I_i = 1$ if the column is a match between the two strands, 0 otherwise
- $G_i = 1$ if the column has a gap in the alignment, 0 otherwise
- $G_i^X = 1$ if the column has a gap in the X strand, 0 otherwise

- $GF_i = 1$ if the column has a gap in the alignment and has no gap column to its left, 0 otherwise
- S_i, the bit score of the alignment according to a substitution matrix (BLOSUM62 was used, however this is a free parameter)

$N(X)$ is used to notate $\sum_{i=0}^{L} X_i$.
M is the number of sequences scanned for alignments.

Table C.1.2: Names and definitions of alignment metrics computed.

Tag	**Description**	**Definition**
local_gap_compressed_percent_id	Percent identity of the alignment, with contiguous gaps counted as only one.	$\frac{N(I)}{L - N(G) + N(GF)}$
scaled_local_query_percent_id	Percent identity normalised to the query strand	$\frac{N(I)}{A}$
scaled_local_symmetric_percent_id	Percent identity normalised to the average strand length	$\frac{2\,N(I)}{A + B}$
bit_score	BLAST bit score	$N(S)$
local_E_value	BLAST E value	$\frac{MA}{2^{N(S)}}$
query/subject_align_cov	Coverage of alignment on strand	$\frac{L - N(G^{query})}{A}$, $\frac{L - N(G^{subject})}{B}$

Note: Start and end positions of the alignment on a strand is also tracked as query/subject_align_start/end. The difference between start and end, e.g., the length of the alignment per strand is tracked as query/subject_align_len.

C.1.3 16s rRNA alignment

Taxa Pairs were determined via alignment of 16s rRNA strands from Refseq using BLASTn 2.14.0+. The parameters for the search are shown in Table C.1.3 below, default if not listed:

Table C.1.3: BLASTn parameters used.

Parameter	**Value**
word_size	28
gapopen	2
gapextend	1
reward	1
penalty	-2
ungapped	false
evalue	1.0
perc_identity	0.5

The best HSP as determined as the one with the most coverage of both strands (averaged) was then retained for each putative pair. Taxa were labelled as pairs if the resulting coverage for both strands was >=98.5% and the gap compressed percent identity >=81%.

C.1.4 Protein alignment

Reported alignment

Protein Pairs were determined via alignment of UniProtKB sequence using DIAMOND version 2.1.6, a BLAST-like software. The parameters for the search are shown in Table C.1.4 below, default if not listed:

Table C.1.4: DIAMOND parameters used.

Parameter	**Value**
word_size	3
sensitivity	ultra-sensitive
iterate	false
matrix	BLOSUM62
global_ranking	null
gapopen	11
gapextend	1
evalue	0.00001
max_hsps	100
id	0
query-cover	75
subject-cover	75

The best HSP as determined as the one with the most coverage of both strands (averaged) was then retained for the pair and is reported in the dataset with a number of alignment metrics (see Section C.1.2).

Cost of alignment

Before running protein alignment on many taxa pairs and potentially wasting compute, the use of BLASTp to DIAMOND was compared using identical parameters wherever possible, eg. word size, matrix, penalties, etc. This was conducted for 10k random proteins vs 10k random proteins. The compute time, carbon, and return on investment in terms of number of hits is shown below if Figure C.1.3. DIAMOND was chosen for alignment given the small decrease in sensitivity compared to near order of magnitude cost reduction.

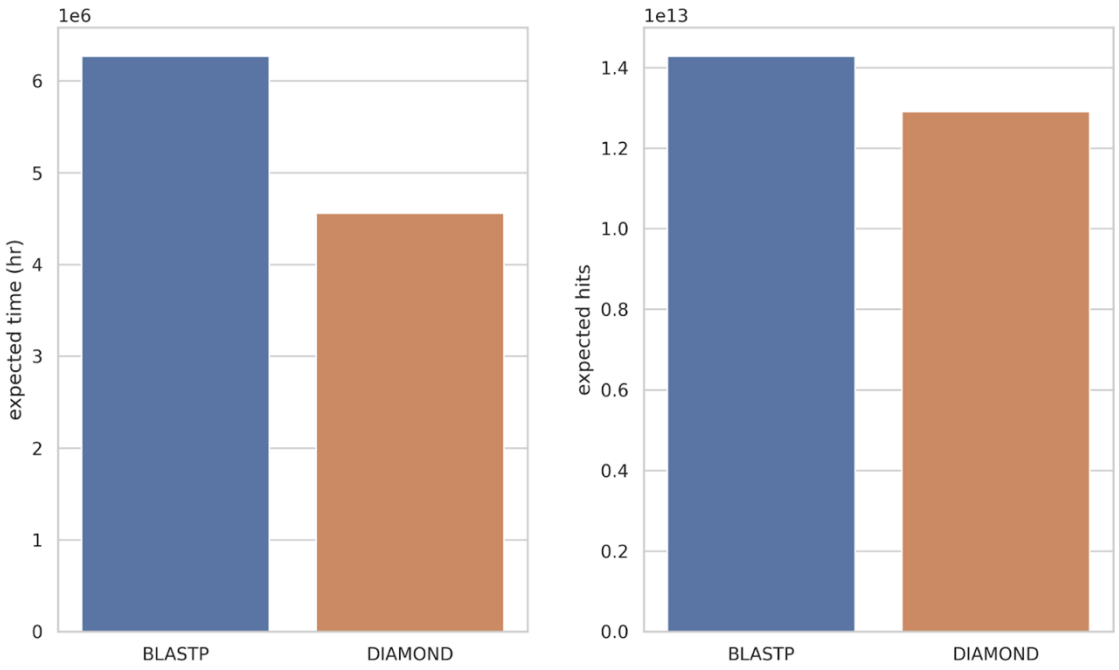

Figure C.1.3: BLASTp vs DIAMOND resource test of protein alignment. DIAMOND significantly reduces cost while retaining most of the sensitivity.

C.1.5 Database Schema

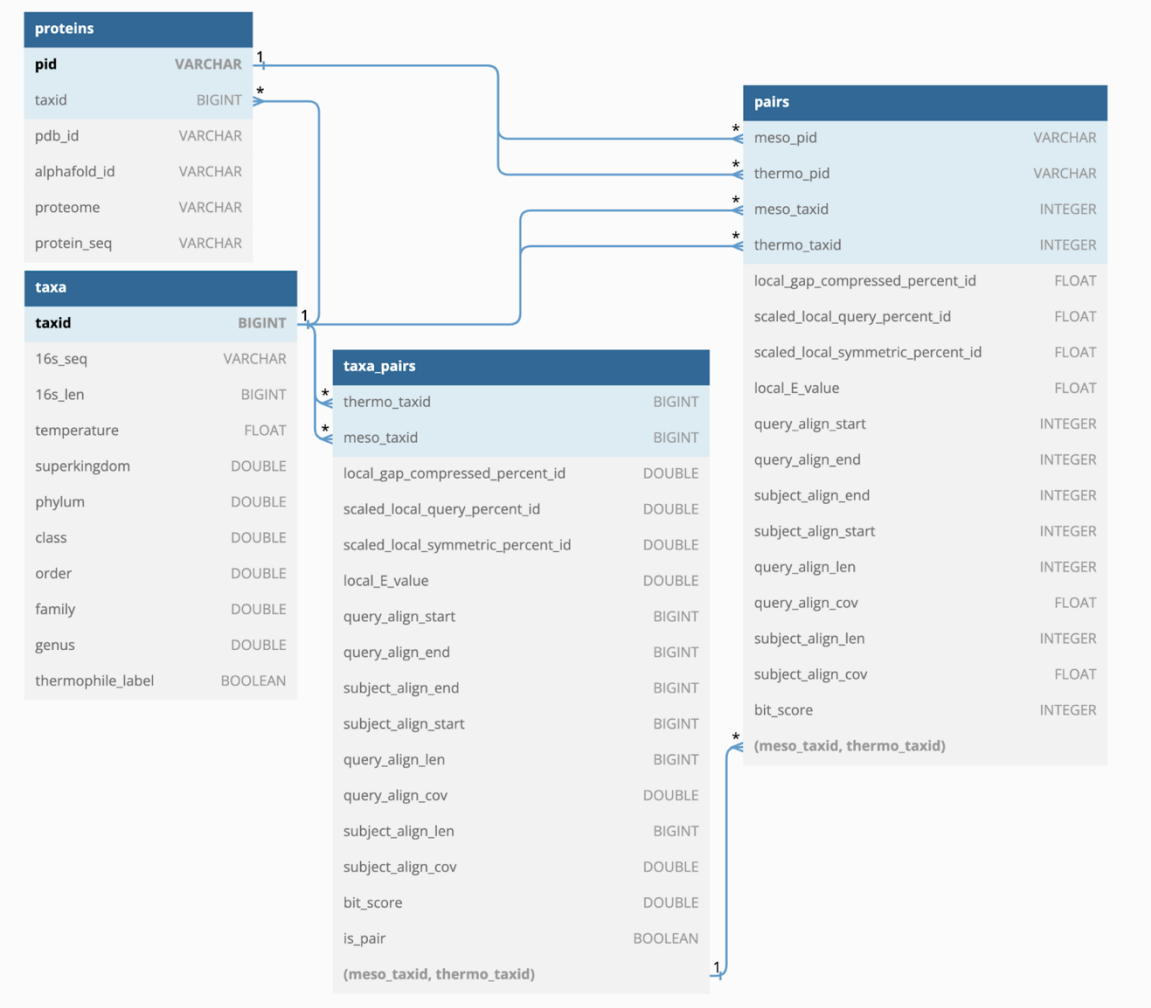

Figure C.1.5: Schema for presented database. The ‘taxa’ and ‘proteins’ tables contain raw data from UniProtKB, RefSeq, and Engqvist. The ‘taxa_pairs’ and ‘pairs’ tables contain information on alignment of 16s rRNA and protein sequences. See Table C.1.5 below for a description of each field in the database.

Table C.1.5: Data fields within the presented relational database.

Table	**Field**	**Description**
taxa	taxid	Primary key, NCBI taxid
taxa	16s_seq	Nucleotides of 16s rRNA sequence

taxa	16s_len	Length of 16s rRNA strand
taxa	temperature	Optimal Growth Temperature
taxa	superkingdom	NCBI superkingdom integer identifier
taxa	phylum, …, genus	NCBI integer identifier for taxonomy
taxa	thermophile_label	Indicator if organism is a thermophile as determined by pipeline parameters
proteins	pid	Primary key, UniProtKB identifier
proteins	taxid	NCBI taxid of host organism, foreign key to taxa
proteins	pdb_id	PDB cross reference if present
proteins	alphafold_id	AlphaFold database cross reference
proteins	proteome	Identifier of UniProtKB proteome that protein is a part of, if present
proteins	protein_seq	Amino acids of the sequence
taxa_pairs	thermo_taxid	NCBI taxid of thermophile, foreign key to taxa. Serves as compound primary key with meso_taxid
taxa_pairs	meso_taxid	NCBI taxid of mesophile, foreign key to taxa. Serves as compound primary key with thermo_taxid
taxa_pairs	is_pair	Indicator if pair of taxa is considered a pair by pipeline parameters for downstream protein alignments.
taxa_pairs	*	A number of alignment metrics, see section C.1.2
pairs	meso_pid	UniProtKB identifier of mesophilic protein, foreign key to proteins. Serves as compound primary key with thermo_pid

pairs	thermo_pid	UniProtKB identifier of thermophilic protein, foreign key to proteins. Serves as compound primary key with meso_pid
pairs	meso_taxid	NCBI taxid of mesophile, foreign key to taxa. Serves as compound foreign key with thermo_taxid to taxa_pairs
pairs	thermo_taxid	NCBI taxid of thermophile, foreign key to taxa. Serves as compound foreign key with meso_taxid to taxa_pairs
pairs	*	A number of alignment metrics, see section C.1.2

C.1.6 Pfam annotations

Pfam was used to annotate proteins in protein pairs. For a given protein pair, both proteins were run against 19,632 HMMs from Pfam version 35.0 using pyhmmer. (Larralde, M.; Zeller, G. PyHMMER: A Python Library Binding to HMMER for Efficient Sequence Analysis. Bioinformatics 2023, 39 (5), btad214. https://doi.org/10.1093/bioinformatics/btad214) All annotations with sequence hit E-value < $1e^{-10}$ were retained. The set Jaccard score was then computed for annotations between two proteins, as defined below:

$$J = \frac{\cap(A,B)}{\cup(A,B)} \quad \text{Eq. C.1.6}$$

Where A, and B are the set of Pfam annotations from Protein A and B, respectively. Thus, if protein A is labelled with families “pfam_0” and “pfam_1”, while protein B is labelled with only “pfam_0”, the Jaccard score is 0.5. If Pfam was unable to find annotations for one of two proteins within a pair meeting the minimum HMMER requirements, the Jaccard

score is 0.0. Note that protein pairs where neither protein was annotated were ignored for analysis, likely falling into an unknown protein family.

C.1.7 OGT Predictor Training

ProtBERT was fine tuned on proteins from the learn2therm dataset to act as a classifier of thermophilic or mesophilic from sequence alone. (Elnaggar, A.; Heinzinger, M.; Dallago, C.; Rehawi, G.; Wang, Y.; Jones, L.; Gibbs, T.; Feher, T.; Angerer, C.; Steinegger, M.; Bhowmik, D.; Rost, B. ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 2022, 44 (10), 7112–7127. https://doi.org/10.1109/TPAMI.2021.3095381) To prepare for this fine-tuning, the data was preprocessed as follows.

The N=10 mil proteins with sequence length less than or equal to 250 amino acids from the database was considered. Data labelled with OGT in [30°C, 60°C) were excluded. Proteins with higher temperature were labelled thermophilic while the others were labelled mesophilic. The resulting data was randomly downsampled in order to produce a balanced dataset of 175k each of thermophilic and mesophilic proteins. In order to minimize redundancy in the dataset, MinHash distance removal of sequence tokens was conducted, a common step for reducing leakage in traditional natural language datasets.(Team, T. A. Text Similarity using K-Shingling, Minhashing and LSH Locality… – Towards AI. https://towardsai.net/p/l/text-similarity-using-k-shingling-minhashing-and-lshlocality-sensitive-hashing, (accessed 2023-03-30)) Here, compute the set of all 3-mers for each amino acid sequence was computed, then the MinHash of the resulting set determined

using 100 permutations. Proteins were clustered using distance of the MinHash Jaccard score > 0.6, and only a single protein was retained from each group. The dataset was finally split into development and test set with a fraction of 10% test using random splitting on the protein's taxid, thus no two proteins from the same organism occur across datasets.

The model itself was downloaded with pre-trained weights from Huggingface, and a new MLP layer was created to act as a binary predictor. Specifically, the pretrained embeddings of size 1024 from the output of ProtBERT were pooled using a linear projection of size 1024 and then mean over amino acids in the sequence, followed by a two-neuron binary predictor. The pytorch readout for the model is given below, note that neural modules are from Huggingface's ecosystem. See their model classes here. (Huggingface. BERT. https://huggingface.co/docs/transformers/v4.30.0/en/model_doc/bert (accessed 2023-06-26))

```
BertForSequenceClassification(
 (bert): BertModel(
   (embeddings): BertEmbeddings(
     (word_embeddings): Embedding(30, 1024, padding_idx=0)
     (position_embeddings): Embedding(40000, 1024)
     (token_type_embeddings): Embedding(2, 1024)
     (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
     (dropout): Dropout(p=0.0, inplace=False)
   )
   (encoder): BertEncoder(
     (layer): ModuleList(
       (0-29): 30 x BertLayer(
         (attention): BertAttention(
           (self): BertSelfAttention(
             (query): Linear(in_features=1024, out_features=1024, bias=True)
             (key): Linear(in_features=1024, out_features=1024, bias=True)
```

```
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.0, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=1024, out_features=1024, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.0, inplace=False)
  (classifier): Linear(in_features=1024, out_features=2, bias=True)
)
```

The model was then fine tuned for two epochs on the 280k training proteins. The training parameters are given below:

```
TrainingArguments(
_n_gpu=1,
adafactor=False,
adam_beta1=0.9,
```

```
adam_beta2=0.999,
adam_epsilon=1e-08,
auto_find_batch_size=False,
bf16=False,
bf16_full_eval=False,
data_seed=None,
dataloader_drop_last=False,
dataloader_num_workers=0,
dataloader_pin_memory=True,
ddp_bucket_cap_mb=None,
ddp_find_unused_parameters=None,
ddp_timeout=1800,
debug=[],
deepspeed=None,
disable_tqdm=False,
do_eval=True,
do_predict=False,
do_train=True,
eval_accumulation_steps=25,
eval_delay=0,
eval_steps=6,
evaluation_strategy=steps,
fp16=True,
fp16_backend=auto,
fp16_full_eval=False,
fp16_opt_level=O1,
fsdp=[],
fsdp_min_num_params=0,
fsdp_transformer_layer_cls_to_wrap=None,
full_determinism=False,
gradient_accumulation_steps=25,
gradient_checkpointing=True,
greater_is_better=False,
group_by_length=False,
half_precision_backend=cuda_amp,
hub_model_id=None,
hub_private_repo=False,
hub_strategy=every_save,
hub_token=<HUB_TOKEN>,
ignore_data_skip=False,
```

```
include_inputs_for_metrics=False,
jit_mode_eval=False,
label_names=None,
label_smoothing_factor=0.0,
learning_rate=5e-05,
length_column_name=length,
load_best_model_at_end=True,
local_rank=0,
log_level=info,
log_level_replica=passive,
log_on_each_node=True,
logging_dir=./data/ogt_protein_classifier/model/runs/Jun19_12-16-35_g3070,
logging_first_step=False,
logging_nan_inf_filter=True,
logging_steps=1,
logging_strategy=steps,
lr_scheduler_type=linear,
max_grad_norm=1.0,
max_steps=-1,
metric_for_best_model=loss,
mp_parameters=,
no_cuda=False,
num_train_epochs=2,
optim=adamw_hf,
optim_args=None,
output_dir=./data/ogt_protein_classifier/model,
overwrite_output_dir=False,
past_index=-1,
per_device_eval_batch_size=32,
per_device_train_batch_size=32,
prediction_loss_only=False,
push_to_hub=False,
push_to_hub_model_id=None,
push_to_hub_organization=None,
push_to_hub_token=<PUSH_TO_HUB_TOKEN>,
ray_scope=last,
remove_unused_columns=True,
report_to=['tensorboard', 'codecarbon'],
resume_from_checkpoint=None,
run_name=./data/ogt_protein_classifier/model,
```

```
save_on_each_node=False,
save_steps=6,
save_strategy=steps,
save_total_limit=None,
seed=42,
sharded_ddp=[],
skip_memory_metrics=True,
tf32=None,
torch_compile=False,
torch_compile_backend=None,
torch_compile_mode=None,
torchdynamo=None,
tpu_metrics_debug=False,
tpu_num_cores=None,
use_ipex=False,
use_legacy_prediction_loop=False,
use_mps_device=False,
warmup_ratio=0.0,
warmup_steps=0,
weight_decay=0.0,
xpu_backend=None,
)
```

The final model model was saved and evaluated against the test set. It can be found on the Huggingface Hub, here. (Komp, Evan; Beck, David. Learn2therm_model, 2023. https://doi.org/10.57967/hf/0815)

C.1.8 ESM Mapping

To produce a 2D mapping of proteins (Figure 3-4-right), ESM2 embeddings were used.(Verkuil, R.; Kabeli, O.; Du, Y.; Wicky, B. I. M.; Milles, L. F.; Dauparas, J.; Baker, D.; Ovchinnikov, S.; Sercu, T.; Rives, A. Language Models Generalize beyond Natural Proteins. bioRxiv December 22, 2022, p 2022.12.21.521521. https://doi.org/10.1101/2022.12.21.521521) First, embeddings from ESM Atlas were

retrieved for all proteins with Tm>0.9 and pLDDT>0.9. These are the proteins that the ESM language model's structural prediction module is most confident in, eg. those that are not likely extrapolative given the training sequences/structures. The embeddings represent the mean over sequence tokens of the pretrained model's output latent space, producing a single size 2,560 vector for each protein sequence.

Due to computational cost, data was sampled in order to make dimensionality reduction achievable. Only 500k proteins from the ESM Atlas were retained by random sampling. Proteins from pairs from learn2therm and Hait et al.'s dataset were then randomly sampled, retaining the ratio of dataset sizes to each other and to the ESM Atlas. This resulted in 240k proteins from learn2therm dataset, and 82 from Hait et al.'s. The same pretrained model version ('`esm2_t36_3B_UR50D`') as was used for ESM Atlas was again used to embed learnt2therm and Hait et al.'s proteins.

Finally, multicore T-SNE was used on the embedded vectors down to two dimensions with the following parameters, the rest default:

Table C.1.8: Parameters used for T-SNE dimensionality reduction of ESM embeddings.

Parameter	**Value**
n_iter_early_exag	500
n_iter	2000
perplexity	30.0
theta	0.5

The T-SNE process was repeated for Figure 3-21, where learn2therm was reduced alone, and those examples that occurred in NOMELT's training dataset were identified, retaining the sampling ratios used to produce the proteins used for embedding.

C.2 NOMELT training and evaluation

C.2.1 Pairwise alignment

Case studies discussed in the main text were BLASTed against the training set of NOMELT to ensure that no close homologs were seen during training. This includes En-HD, LovD, and LipA. The parameters used were:

```
-evalue 2.0 -outfmt 5 -num_threads 32 -word_size 3 -matrix BLOSUM62 -
qcov_hsp_perc 80
```

Full Smith-Waterman alignment in Biopython is also leveraged in order to posit mutations upon a sequence leading to a variant. The parameters are:

```
matrix: BLOSUM62
match_score: 1
mismatch_score: -1
gapopen: -4
gapextend: -1
penalize_end_gaps: false
```

C.2.2 jackhmmer parameters

Jackhammer was used to produce MSAs for each test set example against other thermophilic sequences, in order to compute positional cross entropy over natural variation. The parameters are the same as used by AF2 searches:

```
--noali --F1 0.0005 --F2 0.00005 --F3 0.0000005 --incE 0.0001 -E 0.0001 --cpu 32
-N 1
```

C.2.3 MMSeqs2 parameters

```
--min-seq-id 0.5 --cluster-reassign 1 --cluster-steps 5 -s 7 --max-seqs 1000 -c
0.95 --cov-mode 0 --similarity-type 2 -e 1e-3 --cluster-mode 1 --threads 32
```

C.2.4 NOMELT hyperparameters

NOMELT is a finetuned from ProtT5-XL, the Pytorch readout is given below:


```
{
  "_name_or_path": "Rostlab/prot_t5_xl_uniref50",
  "architectures": [
    "T5ForConditionalGeneration"
  ],
  "classifier_dropout": 0.0,
  "d_ff": 16384,
  "d_kv": 128,
  "d_model": 1024,
  "decoder_start_token_id": 0,
  "dense_act_fn": "relu",
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "relu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": true,
  "is_gated_act": false,
  "layer_norm_epsilon": 1e-06,
  "model_type": "t5",
  "n_positions": 512,
  "num_decoder_layers": 24,
  "num_heads": 32,
  "num_layers": 24,
  "output_past": true,
  "pad_token_id": 0,
  "relative_attention_max_distance": 128,
  "relative_attention_num_buckets": 32,
  "transformers_version": "4.34.0.dev0",
  "use_cache": true,
```

```
  "vocab_size": 128
}
```

The parameters and their final versions used to prepare the data and train the model are given in Table C.2.1.

Table C.2.1: Tunable parameters used to train the final model.

Parameter	Final Value	Description
	Data preparation	
min_thermo_temp	60.0	Temperature to consider a thermophile
min_temp_diff	0.0	Minimum difference in temps to consider pair
max_meso_temp	40.0	Maximum temperature to consider a mesophile
min_align_cov	0.95	Minimum protein pair alignment coverage (avg)
mmseq_params	coverage: 0.95 min-seq-id: 0.5 cluster-mode: 1 similarity-type: 2 sensitivity: 7 max-seqs: 1000 cluster-steps: 5 cluster-reassign: 1 e: 1e-3	Parameters passed to MMSeqs to cluster dataset for splitting.
test_size	0.1	Fraction used to split dev and test set, and val set from dev set. Approximate, based on MMSeqs' cluster sizes.
additional_filters	'abs({seq_len_diff})/{meso_seq_len}<0.1'	Statement that can be evaluated on columns in the protein pairs learn2therm dataset to filter it.

	Model Architecture	
pretrained_model	Rostlab/prot_t5_xl_uniref50	Foundational model to start with
model_hyperparams	dropout_rate: 0.1 relative_attentin_max_distance: 250	Hyperparameters directly passed to Huggingface T5Params class.
generation_max_length	250	Max size of stochastic generations
generation_num_beams	10	Number of beams for BEAM search
	Training	
reweight	false	Whether to weigh examples in training set by inverse cluster size for loss computation
freeze_early_layers	0.2	Fraction of T5 blocks to be frozen, bottom up.
epochs	1	Number of training epochs for the evalution model. Ignored for the final model which trains the same fraction of steps as the eval model did.
early_stopping	True	Whether to stop training early for the evalutation model
early_stopping_patience	4	Steps to wait for improvement before stopping
early_stopping_threshold	0.01	Evaluation loss required to count as improvement
per_device_batch_size	20	Size of training examples per GPU per step.
gradient_accumulation	8	Batches of gradients to accumulate before taking a backward step.
learning_rate	1.e-4	Optimizer learning rate
gradient_checkpointing	True	Whether to use gradient checkpointing

evals_per_save	3	Number of evaluation steps between model evaluation, for checkpointing and early stopping
evals_per_epoch	500	Number of evaluation steps per epoch during training
lr_scheduler_type	'linear'	Type of learning rate schedule over training epochs
warmup_ratio	0.1	Fraction of total training steps spent warming up to max from 0.0 learning rate
label_smoothing_factor	0.001	Smoothing for one-hot encoded labels
optim	'adamw_hf'	Optimizer type used
optim_args	Null	Additional hyperparameters for optimizer
bf16	true	Use bf16 parameter precision
fp16	false	Use fp16 parameter precision
eval_single_example_per_cluster	true	Use one random example from each validation set cluster to run evaluation, otherwise whole set
	Testing parameters	
only_one_from_cluster	true	Use one random example from each test set cluster for held out testing, otherwise whole set
	Mutation optimization	
use_relaxed	false	Whether to run AMBER relaxation for AF2 predictions
fix_msas	true	Whether to skip MSA generation for AF2 inputs that have already been run
num_replicates	25	Number of stochastic AF2 predictions per protein
residue_length_norm	true	Whether to normalize average Rosetta free energy of AF2 ensemble by the number of amino acids
n_trials	100	Number of trials to run, each with a sample of the set of mutations

sampler	NSGAIISampler	Method of sampling next mutations from a set of mutations
population_size	10	Size of generation for each round of evolutionary optimization
cut_tails	null	Number of residues to keep off the end of a mutant when aligned with the wild type when determining mutations suggested by the model
gap_compressed_mutations	true	Compress gaps suggested by model into a single mutation
matrix	'BLOSUM62'	Matrix used for alignment when determining mutations suggested by the model
match_score	1	Score for matches when aligning wild type to model translation
mistmatch_score	-1	Score for mismatches when aligning wild type to model translation
gapopen	-4	Penalty for opening a gap when aligning wild type to model translation
gapextend	-1	Penalty for extending a gap when aligning wild type to model translation
penalize_end_gaps	false	Whether to penalize end gaps when aligning wild type to model translation

C.2.5 Molecular Dynamics Runs

En-HD variants were simulated using GROMACS starting from PDB files as follows, after conversion to .gro format.

The protein was placed in a box, solvated, and neutralized:

```
gmx editconf -f chd.gro -o chd_box.gro -c -d 1.0 -bt cubic
gmx solvate -cp chd_box.gro -o chd_solv.gro -p topol.top
```

```
gmx grompp -f ions.mdp -c chd_solv.gro -p topol.top -o ions.tpr  -maxwarn 1
gmx genion -s ions.tpr -o chd_ions.gro -p topol.top -neutral
```

NPT Equilibration:

```
;
;   GROMACS
;   Input for NPT
;
;
;define         =  -DPOSRES
integrator      =  md
nsteps          =  100000
dt          =  0.002
;
; Removing CM Translation and Rotation
comm_mode       =  Linear
nstcomm         =  1000
;
; Output Control
nstlog          = 1000
nstenergy       = 100
nstxout         = 0
nstvout         = 0
nstxtcout               = 1000
nstfout         = 0
;
; Neighbour Searching
nstlist         = 10
ns_type         = grid
pbc         = xyz
rlist           = 1.0
;
; Electrostatic
rcoulomb        = 1.0
coulombtype     = pme
fourierspacing      = 0.12
;periodic_molecules     = yes
;
; VdW
vdw-type                = shift
rvdw                    = 1.0
;
; Constraints
constraints     = h-bonds
constraint-algorithm    = lincs
lincs_iter      = 4
;
; Temperature
Tcoupl          = v-rescale
tc_grps         = system
tau_t           = 0.1
```

```
ref_t           = 298.15
;
; Pressure
Pcoupl          =  berendsen
Pcoupltype      =  isotropic
tau_p           =  1.0
compressibility     =  4.5e-5
ref_p           =  1.0
;
; Initial Velocities
gen_vel         = yes
gen_temp        = 298.15
gen_seed        = -1
```

Production Run:

```
integrator              = md
dt                      = 0.002
nsteps                  = 500000000     ;1us
nstxtcout               = 40000
nstvout                 = 0
nstfout                 = 0
nstcalcenergy           = 100
nstenergy               = 10000
nstlog                  = 10000
;
cutoff-scheme           = Verlet
nstlist                 = 10
vdwtype                 = Cut-off
vdw-modifier            = Force-switch
rvdw_switch             = 1.0
rvdw                    = 1.2
rcoulomb                = 1.2
rlist                   = 1.2
coulombtype             = PME
;
tcoupl                  = V-rescale
tc_grps                 = System
tau_t                   = 1.0
ref_t                   = 298
;
;
constraints             = h-bonds
constraint_algorithm    = LINCS
continuation            = yes
;
nstcomm                 = 100
comm_mode               = linear
;
```