

Probability and Representation in Neural Systems: Dynamics, Learning, and Computation

Shirui Chen

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2026

Reading Committee:

Eric Shea-Brown, Co-Chair

Lillian Ratliff, Co-Chair

Bamdad Hosseini

Program Authorized to Offer Degree:

Applied Mathematics

© Copyright 2026

Shirui Chen

University of Washington

Abstract

Probability and Representation in Neural Systems: Dynamics, Learning, and Computation

Shirui Chen

Co-Chairs of the Supervisory Committee:

Eric Shea-Brown and Lillian Ratliff

Department of Applied Mathematics

Department of Electrical and Computer Engineering

Across biological and artificial neural systems, this dissertation examines two linked aspects of inference and action from incomplete, indirect observations. Part I of this dissertation covers how neural systems represent, learn, and use probability distributions. Our study of stochastic recurrent networks relates sampling capacity to the score functions their dynamics can approximate and shows that reservoir-sampler architectures can approximate a broad class of target score fields on compact regions. We also examine pretrained video-language models’ task-completion token probabilities and use them as training-free progress signals that, as robot-learning rewards, improve real-robot behavior cloning. Finally, we jointly train token prediction and unmasking order in a masked diffusion language model with on-policy reinforcement learning, improving parallel-generation accuracy–efficiency tradeoffs. Part II of this dissertation examines how learning shapes the internal representations supporting those computations. We find that sharpness-based quantities bound local representation robustness and compression. Additionally, we study feedforward memory networks, whose plastic input weights correspond to Hopfield memory patterns, and find that heterogeneous synaptic dynamics affect pattern diversity and task performance.

Contents

Acknowledgments	xxviii
------------------------	---------------

1 Introduction	1
1.1 Uncertainty as a computational problem	4
1.2 Probabilistic modeling in artificial networks	7
1.3 How learning shapes representation	9
1.4 Contributions and scientific scope	10
1.5 Roadmap	12

I Probabilistic Computation in Biological and Artificial Neural Systems	13
--	-----------

2 Expressive Probabilistic Sampling in Recurrent Neural Networks	14
2.1 Introduction	14
2.2 Background	17
2.2.1 Fokker-Planck equation and stationary distribution	17
2.2.2 Score-based generative modeling	18
2.3 Methods	19
2.3.1 Do we really need to match the score?	19
2.3.2 Synaptic current dynamics: sampler-only networks with limited capacity	22
2.3.3 Firing-rate dynamics could be a universal sampler	23

2.3.4	An efficient RNN weight learning algorithm	26
2.4	Experimental results	27
2.4.1	Learning bimodal distributions	28
2.4.2	MNIST generation task	29
2.5	Discussion	30
2.6	Conclusion	34
3	Token Probabilities as Progress Rewards for Robot Learning	35
3.1	Introduction	35
3.2	Related Work	38
3.3	TOPReward	40
3.3.1	Token probability as the reward	41
3.3.2	Progress estimation from trajectory prefixes	42
3.4	Experiments	43
3.4.1	Large-scale real-world evaluation	43
3.4.2	Instruction sensitivity	46
3.4.3	Success detection	47
3.4.4	Real-world reward-weighted behavior cloning	48
3.5	Limitations	48
3.6	Conclusion	49
4	Learned Parallel Decoding for Masked Diffusion Language Models	50
4.1	Introduction	50
4.2	Preliminaries	55
4.2.1	Sequence Factorization	55
4.2.2	Training Objective	55
4.2.3	Connection to Any-Order Autoregressive Modeling	56
4.3	Methods	56

4.3.1	Path Planning with Unmasking Probability	56
4.3.2	Exact Likelihood Calculation	58
4.3.3	Initializing On-policy Planners with Heuristic Methods	60
4.3.4	On-policy Distillation Reward	60
4.3.5	GRPO Training Framework	61
4.4	Results	62
4.4.1	Training Dynamics	63
4.4.2	Main Results	63
4.4.3	Speed-Quality Trade-off	65
4.4.4	Analysis of Learned Unmasking Strategies	65
4.5	Conclusion	66

II Dynamics and Geometry of Representation Learning 68

5 Loss Sharpness and Compressed Neural Representations 69

5.1	Introduction	69
5.2	Background and setup	74
5.3	From robustness to inputs to compression of representations	79
5.3.1	Sharpness bounds local volumetric transformation in the feature space	79
5.3.2	Maximum Local Sensitivity as an allied metric to track neural representation geometry	82
5.3.3	Local dimensionality is tied to, but not bounded by, sharpness	85
5.3.4	Relation to reparametrization-invariant sharpness	87
5.3.5	Connection to neural collapse and compressed neural representations	87
5.4	Experiments	89
5.4.1	Sharpness and compression metrics during training: verifying the theory	89
5.4.2	Correlation between compression metrics and their sharpness-related bounds	91

5.4.3	Empirical evidence in Vision Transformers (ViTs)	93
5.4.4	Sharpness and local dimensionality	93
5.5	Discussion: connection to generalization	95
5.6	Summary and Conclusion	97
6	Feedforward Memory Networks and Online Synaptic Representations	98
6.1	Introduction	98
6.2	A Hopfield formulation of feedforward memory networks: Preliminaries	101
6.3	Learning dynamics of stored patterns	102
6.3.1	HebbFF networks are HPNs with uniform decay and plasticity rates	103
6.3.2	MPNs are HPNs with uniform decay rates and structured heterogeneous plasticity rates	103
6.3.3	Temporal capacity of stored patterns	104
6.4	Results	108
6.4.1	Uniform synaptic plasticity gives worse performance	109
6.4.2	Heterogeneous Plasticity Rates Yield Superior Performance Compared to Heterogeneous Decay Rates	109
6.4.3	Visualization of learned patterns	110
6.5	Discussion	110
6.6	Conclusion	113
7	Conclusion	114
7.1	Synthesis	114
7.2	Future directions in probabilistic computation	117
7.3	Future directions in representation learning	118
7.4	A perspective on biological and artificial intelligence	119
7.5	Closing perspective	121

A	Appendix to Chapter 2: Reservoir-Sampler Networks	122
A.1	Proof of proposition 2.1 via the Helmholtz-Hodge decomposition	122
A.2	Neural dynamics not following gradient fields	124
A.2.1	Hilbert projection theorem and proof of proposition 2.2	125
A.3	Proof of theorem 2.3	125
A.4	Pseudocode for training the reservoir-sampler network	130
A.5	Accelerated sampling with reservoir-sampler network	130
A.6	Additional experiments	133
A.6.1	Failure cases for sampler-only networks with hyperbolic tangent non-linearity	133
A.6.2	Sampling using autoencoder	133
A.7	Related work	133
A.8	Details of numerical experiments	139
A.8.1	Training details for section 2.4.1	139
A.8.2	Training details for section 2.4.2	140
A.9	Statement of code availability	140
B	Appendix to Chapter 3: TOPReward	141
B.1	Alternative Reward Formulation	141
B.2	Why the True Token?	142
B.3	Success Detection Score	142
B.4	Dataset-level breakdown	143
B.5	Additional qualitative results and ablations	147
B.5.1	Real-world policy details	147
B.5.2	Chat-template ablation	147
B.6	Additional details of ManiRewardBench	149
B.6.1	Subtask Annotation	151

C	Appendix to Chapter 4: dUltra	158
C.1	Limitations	158
C.2	Related Work	158
C.3	Mode Filtering Illustration	161
C.4	Motivating Case Study	161
C.5	Baseline Supervision Comparison	162
C.6	Planner Initialization	163
C.7	Advantage Clipping Details	164
C.8	Training Dynamics and Stabilization	164
C.9	Additional Main Results with Large Block Size	165
C.10	Wall-Clock Throughput	166
C.11	Pareto Frontiers	166
C.11.1	Ablation Results Without the Planner	167
C.12	Learned Unmasking Strategies	168
C.13	MDLM Background	170
C.13.1	Single-Token Forward and Reverse Process	170
C.13.2	NELBO Derivation	170
C.14	Experimental Setup	171
C.14.1	Implementation Details	171
C.14.2	Reward Function	172
C.15	Training Algorithm	174
D	Appendix to Chapter 5: Sharpness and Compression	179
D.1	Linear stability trick on penultimate-layer and middle-layer features	179
D.2	Adaptation of Inequality 5.6 to Residual Layers	181
D.3	Proof of the sharpness identity	181
D.4	Proofs of the volume bounds	183
D.4.1	Proof of the local volumetric-ratio inequality	183

D.4.2	Proofs of the local and network volumetric-ratio bounds	183
D.5	Proofs of the MLS and NMLS bounds	186
D.6	Reparametrization-invariant sharpness and input-invariant MLS	188
D.6.1	Reparametrization-invariant sharpness of Tsuzuku et al.	188
D.6.2	Reparametrization-invariant sharpness upper-bounds input-invariant MLS	191
D.7	Numerical approximation of normalized MLS and elementwise-adaptive sharpness	193
D.8	Empirical analysis of the bound	195
D.8.1	Tightness of the bound	195
D.8.2	Correlation analysis	196
D.9	Additional experiments	201
D.10	Sharpness and compression on test set data	202
D.11	Computational resources and code availability	203
E	Appendix to Chapter 6: Feedforward Memory Networks	204
E.1	Biological motivation of feedforward memory networks	204
E.2	Modern Hopfield network	205
E.2.1	Notations	206
E.2.2	Neural dynamics and Energy function	207
E.2.3	Relation to attention mechanism	208
E.2.4	Element-wise nonlinearity	210
E.2.5	Feedforward memory networks implement cross-attention	210
E.2.6	Feedforward memory networks and Hopfield networks	212
E.3	Proofs	213
E.3.1	Proof for uniform plasticity	213
E.3.2	Proof for heterogeneous decay rates	215
E.3.3	Proof for heterogeneous decay and plasticity rates	216
E.4	List of tasks	217

E.5	Training details	217
-----	----------------------------	-----

List of Figures

2.1	Reservoir-Sampler Networks versus Traditional Sampler-Only Networks. The sampler neurons which produce samples from the target distribution are present for both the sampler-only (SO) network and the reservoir-sampler (RS) network. The reservoir neurons (shown within the large black circle) are only present in the RS network. We explore the firing rate (FR) dynamics for both networks and the synaptic current (SC) dynamics only for the sampler-only network because the stationary distribution of the output neurons is intractable in the RS-SC case. We evaluate the approximation power of the function set represented by the drift term of the neural dynamics, and list the number of basis functions that span the set, whether these basis functions are fixed, and whether the function set is closed under addition (shown in the table).	17
-----	--	----

2.2	Bimodal distribution sampling results. The 3 tractable cases shown are Sampler-only (SO) networks with both synaptic current (SC) dynamics and firing rate (FR) dynamics and Reservoir-sampler (RS) networks with FR dynamics, which are named SO-SC, SO-FR and RS-FR respectively. a-c) The score function learned compared to the true score function (orange curve) as we gradually decrease the noise level (the darker the line, the lower the noise level). We see that RS-FR is capable of perfectly fitting the score function, while SO-SC and SO-FR are only able to fit the score function with piecewise linear functions when using the ReLU transfer function. d-f) Histogram of sampled points, and the (scaled) density function of the target distribution. Again the reservoir-sampler network is able to generate samples whose distribution matches the target distribution, while the sampler-only network is not able to do so due to the incorrectly matched score function.	29
2.3	RS-FR model learning a mixture of 2-D Laplace distribution. Left to right: sample density from the true distribution (brighter color denotes higher density); sample density from the learned distribution; marginalized kurtosis of each dimension from the true and learned distribution.	30

2.4	Learning to sample the MNIST image distribution. a) An MNIST image is projected to a 300-D latent space (orange circle) spanned by the first 300 principal components using PCA. The sampler learns to sample from the distribution of this latent space and generates images by applying inverse PCA projection to these samples. The diagram illustrates the RS-FR model. b) The loss curves for 3 different models during training. Every 100 epochs the noise level added to the training samples is reduced (Appendix A.8), and the noise increases to a higher value because the score matching loss magnitude depends on the noise level. As shown in the inset, the loss of the RS-FR model decreases throughout the training process when using the lowest fixed noise level. Meanwhile, the losses of the other two models remain unchanged. c-f) The images generated for the 3 models compared to the digit images generated from latent training samples.	31
2.5	Classic Necker cube ambiguity. The same two-dimensional line drawing supports multiple three-dimensional percepts. This ambiguity provides cognitive evidence for the existence of probabilistic neural dynamics: the stimulus is fixed, but the internal interpretation can require time to stabilize or switch between competing modes.	33
3.1	Result highlights. TOPReward enables effective zero-shot estimation of task progress across diverse and challenging real-world manipulation tasks, without task-specific training. Across several VLM backbones, TOPReward provides a temporally consistent visual reward signal that supports multiple downstream applications, including success detection, policy improvement, and evaluation on our in-house benchmark, ManiRewardBench.	37

3.2	Instruction sensitivity. Instruction-grounding matrices for TOPReward and Robometer-4B. Rows are terminal trajectory videos and columns are candidate task instructions. For TOPReward, each cell is the terminal TOPReward score for that video–instruction pair; for Robometer-4B, each cell is its terminal reward prediction. We apply Sinkhorn normalization for visualization and report mean diagonal mass, so both rows and columns sum to one and bright diagonal cells indicate strong one-to-one matching between trajectories and their intended instructions.	46
4.1	dUltra accuracy-efficiency summary. Across math reasoning and code generation benchmarks, dUltra consistently achieves strong task accuracy while decoding more tokens per model forward pass than Fast-dLLM, dParallel, and d3LLM. We report the main setting with generation length 256 and block size $B = 32$; complete $B = 32$ and $B = 128$ results are provided in Tables 4.1 and C.2. T/F denotes generated tokens per model forward pass, so higher values indicate more parallel decoding.	51
4.2	dUltra overview. Left: Accuracy versus number of function evaluations (NFE; denoising steps) on math and coding benchmarks for Fast-dLLM, dParallel, d3LLM, and dUltra (ours) variants. Here we use a block size of 128 and a generation length of 256. Right: dUltra architecture: a masked diffusion LM (LLaDA blocks) produces last-layer representations that feed (i) a token decoding head to output logits for masked positions and (ii) an unmasking planner head to output per-position unmasking probabilities; the Bernoulli sampled unmasking decisions update the masked sequence ([M] denotes the mask token).	54

4.3	Learned unmasking order on HumanEval. Red bars indicate tokens newly unmasked at each denoising step. The planner mostly follows a left-to-right trend, but it also jumps to predictable code structure such as function signatures, control-flow keywords, and return statements, creating opportunities for parallel decoding while respecting stronger local dependencies.	66
5.1	A schematic illustration of why flatness alone does not reliably predict generalization. (a) Under the common but unjustified assumption that the test loss is a horizontal shift of the training loss, flat minima trivially exhibit a smaller generalization gap than sharp minima. (b) In a hypothetical but possible scenario, the test loss differs from the training loss by an additive perturbation rather than a simple shift. Here the flat minimum can exhibit a <i>larger</i> generalization gap than the sharp minimum, demonstrating that curvature alone is insufficient to predict generalization.	71
5.2	Trends in key variables across SGD training of the VGG-11 network with fixed batch size (equal to 20) and varying learning rates (0.05, 0.1 and 0.2). Higher learning rates lead to lower sharpness and hence stronger compression. From left to right: Test accuracy, NMLS, sharpness (square root of Equation (5.3)), log volumetric ratio (Equation (5.7)), MLS, and local dimensionality of the network output (Equation (5.15)).	90
5.3	Trends in key variables across SGD training of the VGG-11 network with fixed learning rate size (equal to 0.1) and varying batch size (8, 20, and 32). Smaller batch sizes lead to lower sharpness and hence stronger compression. From left to right in row-wise order: test accuracy, NMLS, sharpness (square root of Equation (5.3)), log volumetric ratio (Equation (5.7)), MLS, and local dimensionality of the network output (Equation (5.15)).	91

5.4	We trained 100 different models for each combination of datasets and networks by varying learning rates, batch size, and random initializations. Pairwise scatter plots between MLS (resp. NMLS) and the sharpness-related bound on MLS (resp. NMLS) are shown here. For MLS (resp. NMLS) bound see Proposition 5.8 (resp. Proposition 5.10). The Pearson correlation coefficient ρ is shown in the top-left corner for each scatter plot. See Appendix D.8.2 for the full pairwise scatter matrix.	92
5.5	Adaptive sharpness vs Normalized MLS for 181 ViT models and variants. Different colors represent different model classes. For most models, there is a positive correlation between Sharpness and MLS. However, outlier clusters also exist, for MobileViT [Mehta and Rastegari, 2022] models in the upper left corner, and two FastViT [Vasu et al., 2023] models in the lower right corner.	94
6.1	Unified Hopfield formulation to understand FMNs. A: The equivalence between feedforward memory networks (FMNs), modern Hopfield networks, and cross-attention mechanisms (transformer models). The memory patterns are stored in synaptic weights, attractors, and key-value caches (KV Cache) [Pope et al., 2023] respectively. This unified framework shows how the stored memory patterns in these formulations are essential to understand the learning dynamics of FMNs and make predictions about their performance. B: Illustration of FMNs. The only difference between different FMNs lies in the synaptic dynamics of W_{in} , which determine how memory patterns evolve through time. Different shades of input weights represent different memory patterns.	102
6.2	Performance of HPNs with different parameter settings on NeuroGym tasks.	109
6.3	Performance of HPNs with different parameter settings on NeuroGym tasks.	110

6.4	Learned memory patterns of HPNs with different parameter settings at cue time in the context-dependent decision-making task (different colors indicate different cue times). For the HPN with uniform $\boldsymbol{\eta}$ and $\boldsymbol{\lambda}$, noise is added for visibility.	111
A.1	Sampling speed of accelerated RSN We repeat the experiment described in section 2.4.1 where we try to sample from a 2-D Laplacian mixture distribution. The figure shows the Wasserstein distance between the marginalized distributions of generated samples and true samples for both dimensions. We see that compared to vanilla RSN, accelerated RSN is able to sample from the stationary distribution with faster relaxation time.	132
A.2	Networks with hyperbolic tangent nonlinearity applied to a 1-D Gaussian mixture distribution with a symmetric score function. See Figure 2.2 for an explanation of the abbreviations. Here we use tanh nonlinearity for sampler-only networks (SO-SC and SO-FR). SO-FR and RS-FR can represent the exact score, while SO-SC approximates it over the plotted range.	134
A.3	Sampler-only network with hyperbolic tangent nonlinearity fails to sample from a 1-D Gaussian mixture distribution with a non-symmetric score function. See Figure 2.2 for an explanation of the abbreviations. Here we use tanh nonlinearity for sampler-only networks (SO-SC and SO-FR). Only RSN is able to match the score function and sample from the distribution.	135
A.4	Samples produced by our reservoir sampler network when trained to fit the latent distribution produced by a non-linear convolution autoencoder from MNIST. The distribution was 32-dimensional. The reservoir sampler network had 500 reservoir neurons.	136

A.5	Samples produced by our reservoir sampler network when trained to fit the latent distribution produced by a non-linear convolution autoencoder from CIFAR-10. We show both the original images and the reconstructed images to evaluate the sampling quality. The distribution was 128-dimensional. The reservoir sampler network had 20000 reservoir neurons.	137
B.1	Top 10 tokens by absolute difference in mean final-step token probability between successful and failed trajectories. The affirmative token True shows the largest separation, motivating its use as the completion token in TOPReward . Left: mean token probability by group; right: absolute difference in mean token probability.	142
B.2	VOC comparison across datasets. Mean dataset-level VOC for GVL (0-shot) and TOPReward across two evaluation sets: filtered OXE (33 datasets, 20 episodes each) and the ManiRewardBench successful progress-evaluation subset (4 datasets, 113 tasks, 497 episodes). Error bars denote standard deviation across datasets within each evaluation set.	144
B.3	Per-episode VOC distributions, broken down by evaluation set (ManiRewardBench vs Open X-Embodiment) and model backbone.	144
B.4	Distribution of dataset-level Δ VOC by model backbone.	145
B.5	Progress traces for ManiRewardBench. Example progress traces predicted by TOPReward (orange) compared to stage-aware ground-truth completion (dashed) from ManiRewardBench , computed from annotated subtask boundaries. We also overlay Gemini-GVL (blue) on the same episodes when available.	145
B.6	Qualitative example of “Fold the Towel”: Instruction-conditioned progress estimation on a real trajectory. The curve shows TOPReward ’s predicted completion value over time, with annotated values at selected frames corresponding to semantic subtasks.	147

B.7	The six real-world single-arm SO-100 manipulation tasks used for reward-weighted behavior cloning evaluation.	147
B.8	Qualitative comparison on “Place doll in box.” The pretrained policy and behavior cloning (BC) both fail, while TOP-RWBC, fine-tuned with reward weights from TOPReward, succeeds consistently. Frames are uniformly sampled from evaluation rollouts.	148
B.9	Counts of different example tasks in the single-arm YAM dataset.	152
B.10	Counts of different example tasks in the bimanual YAM dataset.	153
B.11	Counts of different example tasks in the SO-100 dataset.	154
B.12	Counts of different example tasks in the Franka dataset.	155
B.13	Frequency of verbs	156
B.14	Screenshot of the Annotation Tool.	156
B.15	Expert demonstration with annotation for the task "Clean the table". . .	157
C.1	Parallel decoding of dependent ambiguous tokens can lead to unwanted modes.	161
C.2	Given the same prompt and ground truth, we mask 30% of tokens using both random masking and autoregressive (AR) masking. The left column shows the masked input sequence with random masking (<m> represents the mask token), and the corresponding one-step decoding result. The right columns show the masked input sequence with AR masking, and the corresponding one-step decoding result. We use LLaDA-8B-Instruct for both generations.	162
C.3	Training dynamics on GSM8K.	165

C.4	Effect of advantage clipping value C on training stability. Without advantage clipping, the average NFE quickly diverges as the planner collapses to never unmasking any tokens. With advantage clipping, training remains stable and the model learns to reduce NFE while maintaining task performance.	166
C.5	Accuracy-efficiency Pareto frontiers across math reasoning and code generation benchmarks. We compare dUltra against baseline methods on GSM8K (top-left), MATH500 (top-right), HumanEval (bottom-left), and MBPP (bottom-right). The x-axis shows average NFE (lower is faster), while the y-axis shows task accuracy. Each point represents a different threshold for dUltra and a different factor for models without planners. The no-planner dUltra ablations directly use the output logits for confidence-based token selection. dUltra consistently achieves strong accuracy-efficiency trade-offs compared to distillation-based methods, demonstrating the effectiveness of our learned unmasking strategies via on-policy distillation.	176
C.6	Visualization of learned unmasking patterns on GSM8K. Red bars show newly unmasked tokens at each denoising step. The pattern exhibits a predominantly autoregressive (left-to-right) trend with minor deviations. An interesting observation is that sometimes, "Step" and ":" are generated first without the numbering in between, and the numbering is generated in the next step.	177
C.7	Visualization of learned unmasking patterns on HumanEval. While the overall trend remains autoregressive, code generation exhibits more non-autoregressive behavior with visible jumps to unmask function signatures, control flow keywords, and structural elements earlier than strict left-to-right order.	178

- D.8.1 Empirical tightness of the bounds.** We empirically verify that the inequalities in Equation (D.43) hold and test their tightness. The results are shown for a fully connected feedforward network trained on the FashionM-NIST dataset. The quantities A, B, C, and D are defined in Equation (D.43). We see that the gap between C and D is large compared to the gap between A and B or B and C. This indicates that partial sharpness $\|\nabla_{\mathbf{w}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F$ (sensitivity of the loss w.r.t. only the input weights) is more indicative of the change in the maximum local sensitivity (A). Indeed, correlation analysis shows that bound C is positively correlated with MLS while bound D, perhaps surprisingly, is negatively correlated with MLS (Figure D.8.3). 196
- D.8.2 Pairwise correlation among different metrics.** We trained 100 different VGG-11 networks on the CIFAR-10 dataset using vanilla SGD with different learning rates, batch sizes, and random initializations and plot pairwise scatter plots between different quantities: local dimensionality, sharpness (square root of Equation (5.3)), log volume (Equation (5.7)), MLS (Equation (5.11)), NMLS (Equation (5.12)), generalization gap (gen gap), MLS bound (Proposition 5.8), NMLS bound (Proposition 5.10) and relative sharpness ([Petzka et al., 2021]). The Pearson correlation coefficient ρ is shown in the top-left corner for each pair of quantities. See Appendix D.8.2 for a summary of the findings in this figure. 197

D.8.3	Pairwise correlation among different metrics. We trained 100 different 4-layer MLPs on the FashionMNIST dataset using vanilla SGD with different learning rates, batch size, and random initializations and plot pairwise scatter plots between different quantities: local dimensionality, sharpness (square root of Equation (5.3)), log volume (Equation (5.7)), MLS (Equation (5.11)), NMLS (Equation (5.12)), generalization gap (gen gap), MLS bound (Proposition 5.8), NMLS bound (Proposition 5.10) and relative sharpness ([Petzka et al., 2021]). The Pearson correlation coefficient ρ is shown in the top-left corner for each pair of quantities. See Appendix D.8.2 for a summary of the findings in this figure.	199
D.8.4	Pairwise correlation among different metrics. We trained 100 different LeNets on the CIFAR-10 dataset using vanilla SGD with different learning rates, batch size, and random initializations and plot pairwise scatter plots between different quantities: local dimensionality, sharpness (square root of Equation (5.3)), log volume (Equation (5.7)), MLS (Equation (5.11)), NMLS (Equation (5.12)), generalization gap (gen gap), MLS bound (Proposition 5.8), NMLS bound (Proposition 5.10) and relative sharpness ([Petzka et al., 2021]). The Pearson correlation coefficient ρ is shown in the top-left corner for each pair of quantities. See Appendix D.8.2 for a summary of the findings in this figure.	200
D.9.5	Trends in key variables across SGD training of a 4-layer MLP with fixed learning rate (equal to 0.1) and varying batch size (8, 20, and 32). MLS/NMLS closely follows the trend of sharpness during the training. From left to right: test accuracy, NMLS, sharpness (square root of Equation (5.3)), log volumetric ratio (Equation (5.7)), MLS (Equation (5.11)), and local dimensionality of the network output (Equation (5.15)).	201

D.9.6	Trends in key variables across SGD training of a 4-layer MLP with fixed batch size (equal to 20) and varying learning rates (0.05, 0.1 and 0.2). MLS/NMLS closely follows the trend of sharpness during the training. From left to right: test accuracy, NMLS, sharpness (square root of Equation (5.3)), log volumetric ratio (Equation (5.7)), MLS (Equation (5.11)), and local dimensionality of the network output (Equation (5.15)).	201
D.10.7	Trends in key variables across SGD training of the VGG-11 network with fixed learning rate (equal to 0.1) and batch size (equal to 20) for samples of the test set. After the loss is minimized, we compute sharpness and volume on the test set. Moreover, the same quantities are computed separately over the entire test set or only on samples that are misclassified. In order from left to right in row-wise order: test loss, sharpness (Equation (5.2)), log volumetric ratio (Equation (5.7)), MLS, and local dimensionality of the network output (Equation (5.15)).	202
E.2.1	Biological interpretation of modern Hopfield networks. Adapted from Krotov and Hopfield [2021].	206

List of Tables

3.1	Results on the filtered Open X-Embodiment dataset. VOC reports mean dataset-level VOC over the same 33 filtered datasets and 20 episodes per dataset. Higher is better except MAE.	45
3.2	Results on ManiRewardBench. We report metrics averaged over the 497 successful-evaluation episodes. This progress-evaluation subset contains 113 tasks across four robot platforms: LeRobot, Franka, Bimanual YAM, and Single-arm YAM. VOC and Kendall tau-b measure rank agreement, Pearson measures linear agreement, and MAE measures normalized progress error. Higher is better except MAE.	45
3.3	Success detection results. We evaluate 279 cleaned LeRobot episodes from ManiRewardBench (150 successful, 129 failed). TOPReward uses the training-free success score defined in Appendix B.3; GVL uses per-trajectory VOC; Robometer-4B uses its trained success head.	47
3.4	Real-world experiments. Partial success score out of 10 trials for reward-weighted behavior cloning on single-arm SO-100 tasks.	48
4.1	Main results with generation length 256 and block size $B = 32$. dUltra rows use deterministic threshold decoding at inference time. We report accuracy, NFE, and T/F (tokens per model forward pass); bold and <u>underlined</u> values mark the best and second-best result in each column.	64

B.1	Filtered datasets (alphabetical) comparing GVL (0-shot) vs TOPReward (TOPR) and their difference, broken down by model backbone. Bold indicates the higher VOC between the two methods for that dataset/model.	146
B.2	Effect of Chat Template on TOPReward VOC. Chat template degrades Qwen3-VL-8B performance by nearly 50% and Molmo2-8B by 20%, demonstrating that the logit-based formulation is sensitive to prompt formatting.	148
B.3	Summary of ManiRewardBench datasets. 6 tasks appear in both the Lerobot and Lerobot failed splits, giving 130 unique tasks across the full benchmark.	150
C.1	Comparison of supervision and training requirements across acceleration methods. dUltra is the only method that (i) learns an explicit unmasking planner, (ii) trains with on-policy rollouts, and (iii) uses an external AR teacher as a reward signal. This additional supervision is an intentional design choice that enables mode-seeking behavior beyond what self-distillation can provide.	163
C.2	Main results with generation length 256 and block size $B = 128$. dUltra uses deterministic threshold decoding at inference time. We report accuracy, NFE, and T/F (tokens per model forward pass); bold and <u>underlined</u> numeric values mark the best and second-best results for each metric.	167
C.3	Wall-clock throughput on a single H200 GPU (256-token generation, $B = 128$). dUltra achieves 3.0–3.4 $\times$ speedup over the LLaDA baseline and $\sim 1.5\times$ over the best distillation baseline (d3LLM), while simultaneously improving accuracy on both tasks. Bold and <u>underlined</u> values mark the best and second-best results for each metric; tied second-best values are both underlined.	167

C.4	No-planner ablation for dUltra. Each row removes the learned planner and uses confidence-based decoding, reporting the benchmark-matched dUltra result for each task group.	168
E.1	Task Information. Note that the dimension of the input to the network is 10 for all tasks since all original inputs go through a fixed random linear transformation. The action size is the dimension of the output of the network.	217

Acknowledgments

I want to begin by expressing my deepest gratitude to my advisors, Professors Eric Shea-Brown and Lillian J. Ratliff, for their insightful guidance and unwavering support throughout this challenging intellectual journey. Eric’s expertise in mathematical neuroscience inspired me to study neural populations through mathematical models in this modern era of neuroscience, when the activity of thousands of neurons can be recorded simultaneously. Lillian’s expertise in game theory and artificial intelligence gave me a fresh perspective on how insights from biological intelligence can inform artificial intelligence research. Together, their complementary expertise introduced me to a wide range of ideas across biological and artificial intelligence. Their mentorship has been vital in helping me appreciate the nature of intelligence and our community’s progress toward understanding and replicating it.

I also want to thank my committee members, Professors Bamdad Hosseini, Rajesh P.N. Rao, and Adrienne Fairhall. Our discussions helped me formulate my research plans more clearly and strengthen the resulting work. Their mathematical and biological insights encouraged me to consider my research questions more holistically and deepened my understanding of intelligence.

I thank Professors Edgar Walker and Hong Qian for helpful discussions about my reservoir-sampler network (RSN) project. I am also grateful to Professor Banghua Zhu and to the Allen Institute for AI (Ai2) for providing generous computational resources as I completed the dUltra and TOPReward chapters, and to Professor Jiantao Jiao for helpful discussions during the development of the dUltra model.

I am also grateful for the research communities at Ai2, the Allen Institute for Brain Science, and the University of Washington. In particular, I thank Jiafei Duan, Dieter Fox, Cole Harrison, Linxing Preston Jiang, Ranjay Krishna, Ying-Chun Lee, Zhixin Lu, Stefan Mihalas, Stefano Recanatesi, Zhongzheng Ren, and Angela Jin Yang, listed alphabetically by surname. Their insights and discussions enriched my work on TOPReward, loss sharpness, and the Hetero-Plasticity Network. It has been a great pleasure to work with such talented people.

I also look back fondly on the time I spent with my talented fellow students in Applied Mathematics, especially during our first two years, when we worked through homework problems together. Those experiences gave me a strong foundation in applied mathematics and prepared me for the research that followed. I am grateful to James Hazelden, Helena Liu, Matt Helton, Yusi Chen, Hanson Mo, Ziyu Lu, Daran Xu, Zihan Zhang, Eleanor Devin Byrnes, Sara Ichinaga, Mike Wendels, and Alex Hsu.

I want to thank my parents for nurturing my mathematical abilities from a young age and for their financial support. Without the education and guidance they gave me, none of my achievements would have been possible.

Finally, and most importantly, I want to thank my fiancée, Yajuan Fu, who has been by my side through most of my Ph.D. journey and has shared all of its tears and joys with me. Her invaluable support has given me the resolve to continue whenever my research stalled.

Chapter 1

Introduction

An animal moving through the world receives only indirect and incomplete evidence about its surroundings. A patch of light might be produced by firm ground or by a drop; a sound might come from prey, a predator, or the wind. The sensory input therefore does not uniquely determine the state of the world that produced it. In Marr’s terms, this is the computational problem that a theory of vision must first identify: the organism must infer properties of the world from measurements that are compatible with multiple causes [Marr, 1982]. Perception is thus an inverse problem, and in general an ill-posed one. Probability is needed because the organism must reason from ambiguous observations toward unobserved causes, combining sensory evidence with prior structure rather than simply reading the world state from its input [Knill and Pouget, 2004, Doya et al., 2006].

The same conceptual move underlies probabilistic machine learning: a dataset (i.e., sensory inputs to artificial¹ models) is treated not merely as a finite collection of examples, but as observations drawn from an unknown data-generating distribution p_{data} . Learning uses those finite observations to approximate the distribution itself, a conditional distribution such as $p(y \mid x)$, or quantities needed for prediction and action [Murphy, 2022]. The empirical success of modern generative models, including score-based diffusion models that generate

¹Throughout this dissertation, by *artificial* we mean anything that is not biologically grounded. In particular, neural networks that run on GPUs are artificial, although they are loosely inspired by neurons.

complex, high-dimensional samples, provides evidence for this modeling premise: probability distributions learned from finite observations can capture substantial structure present in observations of the world [Song and Ermon, 2019, Song et al., 2021]. Their success therefore supports treating sensory and other data as samples from a probability distribution. Yet p_{data} is not directly available to the model. The model encounters only samples and must transform them into internal variables from which probabilities, decisions, or score functions can be computed. A *score function*, $s(x) = \nabla_x \log p(x)$, is the vector field giving the direction of steepest increase in log probability density.

These internal variables form a representation of the observations. In the standard language of representation learning, a network maps an input x to a representation vector $h = \phi_{\theta}(x)$ from which later computations can more readily extract useful information [Bengio et al., 2013, LeCun et al., 2015]. For perception and action, h must retain information needed to infer the state of the world and choose an appropriate response. The connection to probability is then direct: a probabilistic head computes a distribution from the representation, for example $p_{\theta}(y | x) = \text{softmax}(g_{\theta}(h))_y$. If two observations produce the same h , that head cannot assign them different conditional distributions. Representation is therefore not a separate capability placed beside probabilistic inference; it is the internal input on which that inference operates.

Chapter 3 makes this dependence visible in an artificial system. A pretrained video-language model maps an instruction and a prefix of a robot trajectory to internal states and then to next-token probabilities. Without task-specific progress supervision, the probability of an affirmative completion token tends to rise as a successful trajectory unfolds and falls when the trajectory is paired with an incorrect instruction [Chen et al., 2026a]. Because this probability is computed from the model’s internal states, its systematic variation provides behavioral evidence that those states preserve progress-relevant information accessible to the output computation.

Representations are themselves products of learning. The distribution of training data, the

objective, the architecture, and the optimization procedure determine what information an artificial network preserves or suppresses; experience-dependent plasticity plays an analogous formative role in biological systems [Bengio et al., 2013, Bittner et al., 2017]. Chapter 5 studies one precise artificial instance of this dependence by asking how the geometry of an optimized neural network constrains the local geometry of its learned representation [Chen et al., 2026b]. Chapter 6 studies the biological side of the same broad issue by asking how heterogeneous synaptic update rules shape the memory patterns stored in plastic weights.

The discussion above leads to two linked questions. Given a representation, which probability distributions can a neural architecture compute, and how can those probabilities be used? Conversely, how do learning and adaptation determine the representation on which probabilistic computation depends? The central view of this dissertation is that these questions must be studied together. Biological and artificial learning processes construct and continually modify representations; those representations are then used for inference, sampling, and action; and probabilistic objectives or feedback can modify them again on subsequent updates [Bengio et al., 2013, Farrell et al., 2022].

Studying biological and artificial neural systems together is not an arbitrary juxtaposition. Artificial neural networks originated as mathematical abstractions of neural computation, and the two fields have continued to exchange architectures, learning principles, and analytic tools [Hassabis et al., 2017, Richards et al., 2019]. Biology supplies organizational hypotheses that are uncommon in current engineering, including stochastic population dynamics, local synaptic plasticity, heterogeneous components, and adaptation across timescales. Artificial networks supply explicit objectives and state variables that can be measured and intervened on, making it possible to test how a proposed mechanism affects representation and behavior.

Against this background, the five projects form two interconnected lines of research. Part I presents three case studies of how biological and artificial neural systems represent, learn, or use probability distributions. Part II presents two complementary studies of how learning shapes representations. The two lines meet at a common constraint: probabilistic

computation inherits the distinctions available in a representation, and those distinctions are themselves products of architecture and learning.

1.1 Uncertainty as a computational problem

Uncertainty is unavoidable whenever the state of the world cannot be read deterministically from the available observations. In perception, the same retinal image can be compatible with multiple three-dimensional scenes. In motor control, an action can have variable consequences because the body and environment are only partially observed. In language and robotics, a prompt or instruction admits many continuations, and the system must select among them using context that may itself be ambiguous. Marr’s analysis of vision made this point especially clear: before specifying an algorithm or a neural implementation, one must identify the computational problem solved by the system [Marr, 1982]. For perception, that problem is often an inverse problem. The organism observes measurements and must infer the latent causes that produced them.

Bayesian inference gives this computational problem a precise form. If z denotes an unobserved state of the world and x an observation, then

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}. \quad (1.1)$$

The likelihood $p(x \mid z)$ describes how possible world states give rise to observations, the prior $p(z)$ expresses structure available before the current observation, and the posterior $p(z \mid x)$ assigns updated probabilities to the alternatives that remain. This framework has been used to explain perception, learning, and action under uncertainty in both psychology and neuroscience [Knill and Pouget, 2004, Doya et al., 2006, Griffiths et al., 2010]. Its significance here is not that every neural computation must literally evaluate Bayes’ rule in this form. Rather, the Bayesian formulation clarifies why probability enters the problem at all: ambiguous evidence must be combined with prior structure to support inference and

action.

This computational-level success leaves open a mechanistic question. A theory may describe what an ideal or approximately ideal observer should infer from uncertain evidence without specifying how neurons perform the inference. If behavior is well described by probabilistic inference, in what form could a biological circuit represent and manipulate the relevant distributions? Knill and Pouget termed this the Bayesian coding hypothesis, while subsequent work emphasized that a computational-level Bayesian account does not by itself identify a neural code or mechanism [Knill and Pouget, 2004, Rao et al., 2002, Ma and Jazayeri, 2014]. The problem therefore shifts from whether probability is a useful description of behavior to how a neural system could make probability available to computation.

Two influential families of neural implementation address this question differently. In probabilistic population codes, a pattern of population activity parametrically represents a distribution, and appropriate operations on population activity implement probabilistic computations [Ma et al., 2006, Beck et al., 2008]. In sampling-based codes, an instantaneous activity pattern is instead interpreted as one sample from a posterior distribution [Hoyer and Hyvärinen, 2002, Orban et al., 2016]. Under an ergodicity assumption, the posterior is represented by the long-run proportion of time that the neural trajectory occupies different regions of state space. The distribution is therefore not encoded by a single fixed activity vector; it is enacted by the circuit over time.

The distinction between the two codes changes the computational interpretation of neural variability. In a parametric account, the activity pattern encodes quantities that specify a distribution, and variability affects how reliably those quantities can be decoded. In a sampling account, variability across activity patterns is itself part of the representation: the frequency of different states carries information about probability. Sampling theories have consequently been used to relate perceptual uncertainty to the stimulus-dependent variability of cortical responses [Orban et al., 2016, Festa et al., 2021]. This evidence motivates sampling as a possible neural code, but it does not yet specify which circuit architectures can realize

the required distributions.

The guiding philosophy of this dissertation is that, given the complexity of the distributions underlying real-world data, we should not assume a priori that neural systems represent probability distributions in fixed, analytically tractable forms. Instead, we treat neural systems as learning flexible internal probabilistic models from observations sampled from the world. This philosophy motivates our emphasis on the sampling view, in which a distribution is realized through the states generated by a neural system over time rather than encoded in a prescribed parametric form.

The sampling view turns probabilistic representation into a mechanistic problem about neural dynamics. To make a target density stationary, the drift of a stochastic circuit must have an appropriate relationship to that density. Langevin dynamics gives the clearest example: its drift is the score $\nabla_x \log p(x)$. Score-based generative models use the same relationship by learning a score field and using it to transform noise into structured samples [Song and Ermon, 2019, Song et al., 2021]. For an abstract sampling algorithm, this score field can be approximated by a flexible function approximator. A biological account cannot stop at this abstraction. It must ask which score fields can be implemented by plausible recurrent dynamics, because the architecture and state variables of the circuit determine the family of stationary distributions the circuit can enact.

Chapter 2 takes up this unresolved architectural question. It compares sampler-only and reservoir-sampler circuits through the score functions their dynamics can approximate, thereby asking which stationary distributions these recurrent models can enact [Chen et al., 2023]. Its role in Part I is to move from a computational claim that a brain may use samples to a mechanistic analysis of what particular neural dynamics can represent.

1.2 Probabilistic modeling in artificial networks

Probabilistic machine learning places prediction and generation within a common language of distributions. A supervised model may learn a conditional distribution such as $p(y \mid x)$, while a generative model may learn a distribution over observations or latent variables. In both cases, the distribution is inferred from finite data rather than specified directly [Murphy, 2022]. This formulation is powerful because it treats a dataset not only as examples to be labeled correctly, but as partial evidence about an underlying stochastic process that could have produced many possible observations. The model’s task is therefore not only to fit individual samples, but to learn a rule that assigns probability across alternatives.

Neural networks changed the scale and flexibility of this program by learning the internal features used to parameterize these distributions jointly with the probabilistic prediction itself [Bengio et al., 2013, LeCun et al., 2015]. The probability assigned at the output is therefore not independent of the learned representation. If a representation does not preserve a distinction, a downstream probabilistic head cannot condition on that distinction; if learning makes the distinction accessible, the same probabilistic form can express a different judgment. This is the artificial counterpart of the neural-coding problem above, although the implementation and scientific aims differ [Murphy, 2022, Bengio et al., 2013].

Modern generative modeling makes the distribution over data the primary object of learning. Score-based models learn gradients of log density and use them to transform samples from a simple source distribution into samples with the structure of the observed data [Song and Ermon, 2019, Song et al., 2021]. Language models represent a distribution over token sequences through conditional token predictions. Autoregressive models impose a left-to-right factorization. Masked diffusion language models instead define a denoising process in which masked positions are predicted from a partially observed sequence, allowing the generation order to vary [Sahoo et al., 2024, Shi et al., 2024]. This creates a point of contact with biological intelligence at the computational level: can an artificial system also use incomplete sensory evidence to form a graded judgment about a latent state of the world

and use that judgment to guide behavior?

Chapter 3 addresses the question for artificial Vision-Language Models (VLMs). Given an instruction and a prefix of a robot trajectory, TOPReward reads the output probability of pretrained VLMs as a surrogate reward measuring how much progress the robot has made toward the goal. The model receives no task-specific progress supervision, so the resulting progress signal tests how the pretrained conditional distribution generalizes to progress estimation. At the computational level, the model is given incomplete visual evidence and asked for a judgment about a latent task state: whether the instruction has been completed [Chen et al., 2026a].

Chapter 4 addresses the problem of probabilistic language modeling. Unlike an autoregressive model, which decomposes the likelihood of language sequences by conditioning each token on all previous tokens, a masked diffusion language model (DLLM) represents language sequences as samples of a distribution over all possible text token sequences. The upshot of this difference is that a DLLM is capable of generating text sequences not only autoregressively, but also in any order. Human reading provides a useful cognitive analogy: reading proceeds in the direction imposed by the writing system, but cognitive processing is not strictly serial: readers use parafoveal preview, skip predictable words, and make regressions when later evidence changes interpretation [Schotter et al., 2012, Rayner et al., 2006b, 2011, 2006a]. This suggests a broader cognitive principle: probabilistic language sequence modeling can follow a bidirectional pattern, where the entire sequence is considered at each denoising step, rather than a fixed left-to-right order alone. dUltra therefore asks whether the generation order can be modeled as a Markov process: a planner maps the model’s hidden representation to per-token unmasking probabilities, and on-policy reinforcement learning jointly adapts the planner and the token predictor using verifiable and distillation rewards [Chen et al., 2025b].

1.3 How learning shapes representation

In systems built from hand-designed features, the choice of representation is made largely by the designer: the learning algorithm operates on variables that have already been selected, transformed, or engineered. Representation learning changes where this work happens. It makes the transformation from raw input to useful internal variables part of the learning problem itself. Contemporary representation learning brings together several traditions, including distributed connectionist representations, unsupervised feature learning, probabilistic latent-variable models, and multilayer neural networks [Bengio et al., 2013]. Deep learning extends this idea by learning multiple levels of representation, with each layer transforming the representation produced by the preceding layer [LeCun et al., 2015].

Once the representation is learned rather than fixed, its properties become part of the scientific problem. Different learning objectives and inductive biases may make some information readily accessible while leaving other information entangled, and a representation useful for one downstream computation need not preserve the distinctions required by another [Bengio et al., 2013]. Representation is central to the dissertation because probability is computed from representation. A learned distribution over labels, tokens, or actions can only depend on distinctions that the internal variables make available.

Representation learning therefore raises questions beyond predictive accuracy: how does the learning process shape the geometry of internal activity, and how is information retained when the neural system continues to receive new sensory inputs? The two chapters in Part II address these questions in different models of biological and artificial neural systems.

The variables that carry a representation are easiest to identify in an artificial model, where intermediate activations, parameters, logits, and latent trajectories can be measured and intervened on directly. Their location is less settled in biological systems. Depending on the computation and timescale, information may be expressed in instantaneous population activity, trajectories of neural dynamics, synaptic or cellular states, or interactions among them [Vyas et al., 2020, Bittner et al., 2017]. In this dissertation, *representation* therefore names

a computational role—making information available to subsequent computation—without assuming a unique biological substrate or code.

Chapter 5 considers representations produced by stochastic gradient descent. The model architecture, training dataset, and loss function together define the loss landscape, while optimization selects a parameter vector within it. The same parameter vector determines how perturbations of an input propagate through the network’s activations. Because sharpness does not reliably characterize generalization across parametrizations and experimental settings [Dinh et al., 2017, Andriushchenko et al., 2023], the chapter answers a different question: what does sharpness reveal about the local geometry of the learned representation? It turns out that sharpness characterizes the robustness of internal representations of common deep neural networks [Chen et al., 2026b].

Chapter 6 considers representations formed by online synaptic plasticity. Feedforward memory networks modify their synaptic weights as inputs arrive, so those weights serve both as parameters of the current computation and as a changing record of recent experience [Tyulmankov et al., 2022, Aitken and Mihalas, 2023]. By interpreting their outputs through modern Hopfield retrieval and cross-attention, the chapter identifies stored memory patterns and investigates how heterogeneous decay and plasticity rates shape the patterns learned over time.

The two chapters in Part II are therefore complementary studies of how learning shapes representation. Chapter 5 relates the geometry in the parameter space to the local geometry of its neural representation space. Chapter 6 relates online synaptic update rules to the memory patterns formed over time.

1.4 Contributions and scientific scope

The five chapters make the following contributions:

1. Chapter 2 relates the sampling capacity of stochastic recurrent networks to their ability

to approximate a target score function. It identifies restrictions of sampler-only networks and gives conditions under which a reservoir-sampler network with separate output units can approximate arbitrary target distributions. It also develops a denoising-score-matching procedure for training the sampler and demonstrates it on complex non-Gaussian distributions [Chen et al., 2023].

2. Chapter 3 introduces TOPReward, a zero-shot method that extracts task-completion probabilities from pretrained video-language models and uses them as instruction-conditioned progress signals. It evaluates this formulation on 130 real-world manipulation tasks, uses the raw completion probability for success detection and normalized prefix scores for progress estimation. It is also shown that TOPReward-weighted behavior cloning improves over standard behavior cloning on six real-robot tasks [Chen et al., 2026a].
3. Chapter 4 introduces dUltra, an on-policy reinforcement-learning framework for accelerating masked diffusion language models. An unmasking planner predicts per-token Bernoulli probabilities, and the planner and token model are jointly optimized using verifiable and distillation rewards together with a reward based on the number of unmasking steps. Experiments on mathematical reasoning and code generation show improved accuracy–efficiency tradeoffs over heuristic and offline-distillation baselines [Chen et al., 2025b].
4. Chapter 5 defines local geometric measures of representation compression and derives sharpness-related upper bounds on Local Volumetric Ratio and Maximum Local Sensitivity, including network-wide and reparametrization-invariant variants. It evaluates the predicted relationships across feedforward, convolutional, and transformer architectures [Chen et al., 2026b].
5. Chapter 6 gives modern Hopfield and cross-attention interpretations of feedforward memory networks and introduces HPN to study heterogeneous synaptic decay and

plasticity. Its linearized analysis predicts how this heterogeneity affects the diversity of stored patterns, and experiments on NeuroGym verify the predicted effects on pattern formation and task performance.

Each result has a specific scope. The recurrent-sampling analysis establishes representational capacity under its architectural and regularity assumptions, not a complete account of biological learning. TOPReward depends on the perceptual and linguistic distinctions available to the pretrained model. dUltra improves parallel decoding within the conditional-independence constraints of masked diffusion language models. The sharpness chapter establishes bounds on local geometric quantities, not a universal equivalence between flatness, compression, and generalization. The HPN theory analyzes linearized pattern dynamics in an abstracted synaptic-plasticity model. Together, the chapters develop two interconnected lines of research without proposing a single mechanism that subsumes them all.

1.5 Roadmap

Part I, *Probabilistic Computation in Biological and Artificial Neural Systems*, contains three case studies in probabilistic modeling. Chapter 2 studies the distributions that stochastic recurrent neural dynamics can sample. Chapter 3 studies task-completion probabilities in pretrained video-language models and their use as progress signals. Chapter 4 studies learned sampling strategies for parallel generation in masked diffusion language models.

Part II, *Dynamics and Geometry of Representation Learning*, contains two complementary studies of how representations are formed. Chapter 5 connects loss-landscape geometry to the local geometry of neural representations, while Chapter 6 connects heterogeneous synaptic plasticity to the memory patterns formed from continuous input streams.

Part I

Probabilistic Computation in
Biological and Artificial Neural
Systems

Chapter 2

Expressive Probabilistic Sampling in Recurrent Neural Networks¹

2.1 Introduction

There is growing evidence that humans and other animals make decisions by representing uncertainty internally and carrying out probabilistic computations that approximate Bayesian inference [Rao et al., 2002, Knill and Pouget, 2004, Doya et al., 2006, Griffiths et al., 2010, Ma and Jazayeri, 2014]. How networks of neurons in the brain represent probability distributions for Bayesian inference has remained a major open question. There exist two major theories: one assumes that the neural activities encode the parameters of the underlying posterior distributions over sensory stimuli [Beck et al., 2008, Ma et al., 2006, Vertes and Sahani, 2018]. The other is the sampling-based hypothesis, which assumes that the neural responses can be interpreted as samples from a posterior distribution [Hoyer and Hyvarinen, 2002]. Under this hypothesis, recurrent neural circuits make use of their inherent stochasticity to produce samples from posterior distributions. The sampling-based theory has explained various

¹This chapter is based on Shirui Chen, Linxing Preston Jiang, Rajesh P. N. Rao, and Eric Shea-Brown, “Expressive Probabilistic Sampling in Recurrent Neural Networks,” published in *Advances in Neural Information Processing Systems 36* (2023) [Chen et al., 2023].

experimental observations regarding neural variability [Echeveste et al., 2020, Festa et al., 2021, Orban et al., 2016], perceptual decision-making [Haefner et al., 2016] and spontaneous cortical activity [Berkes et al., 2011].

Many studies have proposed biologically plausible spiking rules and membrane dynamics models to implement sampling-based probabilistic inference. However, most of these studies mainly consider sampling from discrete Boltzmann distributions [Buesing et al., 2011] and multivariate Gaussian distributions [Dong et al., 2022, Aitchison and Lengyel, 2016, Masset et al., 2022, Hennequin et al., 2014], only match the first two moments of the distribution [Echeveste et al., 2020], or employ a Monte-Carlo approximation [Huang and Rao, 2014]. Although these studies use algorithmic substrates that can sample from any distribution with a density function *in theory*, it is not clear whether the underlying neural dynamics are capable of implementing a sufficiently expressive version of these sampling methods. Furthermore, natural image statistics are strongly non-Gaussian [Olshausen and Field, 1996], and experimental evidence shows that humans use non-Gaussian prior representations to support cognitive judgments [Houlsby et al., 2013, Griffiths and Tenenbaum, 2006]. It is known that deep artificial neural networks can be used to generate samples from complex data distributions [Song and Ermon, 2019, Song et al., 2021] using a “U-net” [Ronneberger et al., 2015] backbone. However, the neural circuits in the cortex are highly coupled with an abundance of recurrent synapses. Therefore, an outstanding question for probabilistic computation in the brain is: what kind of recurrent neural network is capable of efficiently learning to produce samples from an arbitrarily complex posterior distribution?

In this chapter, we study this question under the basic assumption that the dynamics of recurrent neural circuits can be described by stochastic differential equations (SDEs). Note that this assumption is common to a broad range of past research that uses rate-based neural dynamics to implement sampling-based coding [Aitchison and Lengyel, 2016, Echeveste et al., 2020, Dong et al., 2022]. Moreover, spike-based models [Savin and Denève, 2014, Rullán Buxó and Savin, 2021, Masset et al., 2022] that implement balanced spiking networks

(BSNs) [Boerlin et al., 2013, Buxó and Pillow, 2020] essentially train spiking networks to follow underlying continuous-time SDEs, so our work applies to this line of research as well (details in Appendix A.7).

The contributions of this chapter are as follows (figure 2.1):

1. We establish the relationship between the sampling power of the neural dynamics and the ability of the dynamics to approximate the score function, which is the gradient of the log probability density function. (section 2.3.1)
2. We show that the synaptic current dynamics of a network of neurons whose outputs directly represent the samples (traditional sampler-only network) is only able to approximate score functions that are in a finite-dimensional function space. (section 2.3.2, proposition 2.2)
3. We prove that the firing rate dynamics of our proposed reservoir-sampler network can sample from a distribution whose score function can approximate that of arbitrary target distributions (with mild restrictions) to arbitrary precision (section 2.3.3, theorem 2.3).
4. We derive a computationally efficient and biologically-plausible learning rule for our proposed model (section 2.3.4) that sidesteps the demands of backpropagation through time, and we empirically demonstrate how our model can sample from several complex data distributions (section 2.4).

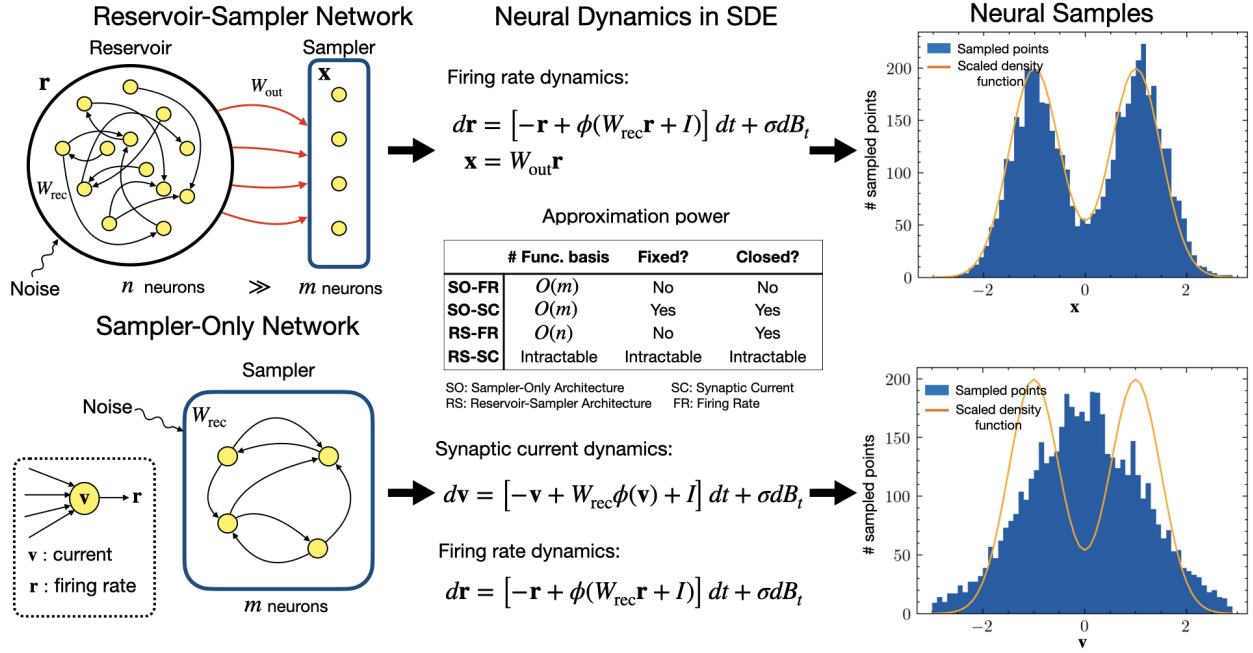


Figure 2.1: **Reservoir-Sampler Networks versus Traditional Sampler-Only Networks.** The sampler neurons which produce samples from the target distribution are present for both the sampler-only (SO) network and the reservoir-sampler (RS) network. The reservoir neurons (shown within the large black circle) are only present in the RS network. We explore the firing rate (FR) dynamics for both networks and the synaptic current (SC) dynamics only for the sampler-only network because the stationary distribution of the output neurons is intractable in the RS-SC case. We evaluate the approximation power of the function set represented by the drift term of the neural dynamics, and list the number of basis functions that span the set, whether these basis functions are fixed, and whether the function set is closed under addition (shown in the table).

2.2 Background

2.2.1 Fokker-Planck equation and stationary distribution

We consider a general time-homogeneous SDE with drift vector $\mathbf{b}(X_t)$ and diffusion matrix $\Sigma = \frac{1}{2}\sigma\sigma^T$:

$$dX_t = \mathbf{b}(X_t)dt + \sigma d\mathbf{B}_t \quad (2.1)$$

where $\sigma \in \mathbb{R}^{n \times m}$ is the diffusion coefficient, and $\mathbf{B}_t$ is an m -dimensional standard Wiener process. The Fokker-Planck equation of this SDE describes the time evolution of the

probability density $p(x, t)$ for a given SDE, assuming the initial density $p(x, 0)$ is known:

$$\partial_t p = \nabla \cdot (\Sigma \nabla p - \mathbf{b}p) \quad (2.2)$$

where $\partial_t = \partial/\partial t$, $\nabla \cdot$ is the divergence operator, and ∇ is the gradient operator. A stationary probability density function is one for which the right-hand-side of equation (2.2) is 0. Mild regularity conditions guaranteeing the existence and uniqueness of a stationary density function are discussed in, for example, Cerrai [2001]. We assume these are satisfied by the SDEs considered in this chapter. Moreover, to ensure ergodicity and a well-defined score function, we assume that p is supported on $\mathbb{R}^n$, i.e. $p(x) > 0$ for all $x \in \mathbb{R}^n$.

As a special case, consider the following Langevin dynamics, in which the drift term is given by the gradient of the log stationary probability density $\nabla p(x)$,

$$dX_t = \nabla \log p(X_t)dt + \sqrt{2}dB_t. \quad (2.3)$$

It can be verified through the Fokker-Planck equation that $p(x)$ in the dynamics above is indeed a stationary probability density function of the dynamics. Therefore the Langevin dynamics can sample from the distribution $p(x)$ as $t \rightarrow \infty$. While the drift term in the Langevin dynamics is a gradient vector field, this is not true in general and is often not the case for recurrent neural dynamics (proposition A.2 in Appendix A.2).

2.2.2 Score-based generative modeling

If we would like a particular dynamical system to implement Langevin dynamics, we need to fit the drift term of an SDE to the score $s_\theta(\mathbf{x}) = \nabla \log p(\mathbf{x})$ of the probability distribution that we are trying to sample from. This procedure of fitting the score function $s_\theta(\mathbf{x})$ is called score matching. In this section, we give a brief summary of one of the major methods of score matching that we will use in this chapter, denoising score matching (DSM) [Vincent, 2011]. The general idea of DSM is to match the score of a noise-perturbed version of the

target distribution. More specifically, the explicit score matching loss (left-hand side) and the denoising score matching loss (right-hand side) are related to each other as follows [Vincent, 2011]:

$$\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})} \left[\frac{1}{2} \|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})\|^2 \right] = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})} \left[\frac{1}{2} \|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\|^2 \right] + C_\sigma \quad (2.4)$$

where $\tilde{\mathbf{x}}$ is a noise-perturbed version of $\mathbf{x}$, so $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma)$, and C is a constant depending on σ but *not* on θ . When the noise is 0, i.e. $\tilde{\mathbf{x}} = \mathbf{x}$, the left-hand side of the equation above is the explicit score matching loss. Although in theory we can start from extremely small Gaussian noise, and directly optimize the right hand of equation (2.4), empirically it is beneficial to start with large Gaussian noise and gradually decrease the noise magnitude until $q_\sigma(\mathbf{x}) \approx p(\mathbf{x})$ and $s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$. Specifically, this has been shown to stabilize the training and improve score estimation [Song and Ermon, 2019].

2.3 Methods

2.3.1 Do we really need to match the score?

The Langevin dynamics provides an elegant way to construct an SDE given a specific stationary distribution. However, as noted above, the neural network dynamics seldom have a drift term that is a gradient field (Appendix A.2). A natural question is therefore whether an SDE with a drift term that is *not* a gradient field (also known as irreversibility) also gives rise to a specific stationary distribution. The answer requires us to look at the Fokker-Planck equation.

$$\partial_t p(\mathbf{v}, t) = \nabla \cdot (\Sigma \nabla p - p F_\theta(\mathbf{v})) \quad (2.5)$$

where $p(\mathbf{v}, t)$ is the probability density function of the variable of interest $\mathbf{v}$ at time t , $F_\theta(\mathbf{v})$ is the drift term of the neural dynamics parametrized by θ , and $\Sigma = \frac{1}{2}\sigma\sigma^T$ is the diffusion matrix. Since the right-hand side of (2.5) needs to be 0 for a given stationary distribution

$p(\mathbf{v})$, we have

$$\nabla \cdot (\Sigma \nabla p - p F_\theta(\mathbf{v})) = 0. \quad (2.6)$$

Therefore $G := -\Sigma \nabla p + p F_\theta(\mathbf{v})$ needs to be a divergence-free (DF) vector field. In other words, $p F_\theta(\mathbf{v})$ is unique up to a DF field given a fixed stationary distribution p . Ma et al. [2015] shows that there exists a skew-symmetric matrix Q such that the DF field can be written as $G = Q \nabla p + p \sum_j \frac{\partial}{\partial \mathbf{v}_j} Q_{ij}$, however, this does not shed more light on how expressive $\{F_\theta\}_\theta$ needs to be without more knowledge of Q and its derivative. We show below that under certain conditions, the DF field G can be altogether regarded as a component that is orthogonal to the score function. Therefore the function space $\{F_\theta\}_\theta$ needs to have enough basis functions that (when projected) are able to approximate the score function of the target distribution sufficiently.

We first note that it would be convenient if $F_\theta(\mathbf{v})$ could approximate the gradient fields $\Sigma^{-1} \nabla \log p$ for any p , in which case $G = 0$. To find out if this is a necessary condition for the dynamics to sample from an arbitrary target distribution p , we let $\Sigma = \mathbf{I}$ and invoke the Helmholtz-Hodge decomposition (HHD) [Bhatia et al., 2013, Majda and Bertozzi, 2002]. The decomposition theorem states that any sufficiently smooth vector field in $L^2(\mathbb{R}^n; \mathbb{R}^n)$ can be uniquely decomposed into a DF vector field and a pure gradient field. In other words, the function space of all DF fields G and the function space of all gradient fields ∇p are the orthogonal complement of each other. Therefore the projection of $p F_\theta(\mathbf{v})$ onto the subspace of smooth gradient fields still needs to be able to approximate ∇p despite the freedom to choose arbitrary DF field G .

For example, in the 1-D case, if we assume that both $p F_\theta(\mathbf{v})$ and ∇p are square-integrable (so they vanish at infinity), then the divergence-free vector field G (which must be constant in 1-D) is 0, therefore $F_\theta(\mathbf{v}) = \nabla \log p$. As a result, $\{F_\theta\}_\theta$ indeed needs to be able to approximate $\nabla \log p$ for every p . In higher dimensions, the same conclusion holds under the assumption of a strict orthogonality constraint:

Proposition 2.1. *Let p be the stationary distribution of the neural dynamics, and the diffusion matrix be the identity matrix. If the DF field G is strictly orthogonal to the gradient field ∇p , meaning that $G(\mathbf{v}) \cdot \nabla p(\mathbf{v}) = 0$ for all $\mathbf{v}$, then the drift term $F_\theta(\mathbf{v})$ can be written as the sum of a divergence-free field $p^{-1}G$ and a gradient field $\nabla \log p$.*

The proof is in Appendix A.1. The form of the above decomposition coincides with that of the HHD [Bhatia et al., 2013, 2014]. Therefore if we enforce the *normal-parallel* boundary condition [Chorin and Marsden, 1993] for the gradient component, the HHD theorem [Bhatia et al., 2013] says that the orthogonal projection of the function space $\{F_\theta\}_\theta$ onto the space of gradient fields is the function space of gradient fields $\{\nabla \log p\}_p$ satisfying the boundary condition given the strict orthogonality constraint on G .

The upshot is that $\{F_\theta\}_\theta$ needs to admit enough basis functions (Appendix A.1). Note that the boundary condition will not be restrictive if we take a sufficiently large bounded region. Therefore it is essential for the neural dynamics to have an expressive functional form that is able to approximate complex score functions, even if their dynamics are not gradient fields.

Previous work has rigorously established that the strict orthogonality constraint holds, in particular, when the nonlinear dynamics is linearized around fixed points of $\mathbf{v}$ (where the drift term is zero [Ao, 2004, Kwon et al., 2005, Qian, 2014]). As a consequence, the conditions of proposition 2.1 are true locally around fixed points.

In what follows, we assume that the conditions of the proposition 2.1 hold. Under this assumption, without loss of generality, we set G to be $\mathbf{0}$ in the following text and explore whether $\Sigma^{-1}F_\theta$, which is determined by the specific neural dynamics (figure 2.1), is able to approximate complex score function $\nabla \log p$.

2.3.2 Synaptic current dynamics: sampler-only networks with limited capacity

We consider the following stochastic synaptic current dynamics [Dayan and Abbott, 2005] (cf. eq. 7.39) that describe a recurrent neural network in terms of the synaptic current that each neuron receives:

$$d\mathbf{v} = D(-\mathbf{v} + W\phi(\mathbf{v}) + I)dt + \sigma d\mathbf{B}_t := \mathbf{m}_{\text{sc}}(\mathbf{v})dt + \sigma d\mathbf{B}_t \quad (2.7)$$

where $\mathbf{m}_{\text{sc}}(\mathbf{v}) := D(-\mathbf{v} + W\phi(\mathbf{v}) + I)$, $\mathbf{v} = [v_1, \dots, v_m]^T \in \mathbb{R}^m$ is the synaptic current of the m neurons in the recurrent network, $D \in \mathbb{R}^{m \times m}$ is a diagonal matrix where diagonal elements are the decay constants, $d_i = \tau_i^{-1}$, $W \in \mathbb{R}^{m \times m}$ is the connection matrix, $\phi(\cdot)$ is a nonlinear transfer function², $I \in \mathbb{R}^m$ is the external input, $\sigma \in \mathbb{R}^{m \times l}$ is the diffusion coefficient and $\mathbf{B}_t$ is an l -dimensional standard Wiener process. The diffusion term can be interpreted as input from other brain areas (due to the large number of incoming connections, the assumption of Gaussianity can be justified by the central limit theorem [Gerstner et al., 2014]). We assume that $\theta = \{D, W, I\}$ are tunable parameters through biological learning.

To show the limited expressivity of $\mathbf{m}_{\text{sc}}$, we have the following corollary from the Hilbert projection theorem showing that $\mathbf{m}_{\text{sc}}$ is only able to approximate functions in a finite-dimensional function space.

Proposition 2.2. *Let $H : \mathbb{R}^m \rightarrow \mathbb{R}^m$ be in the Hilbert space $L_2(\mathbb{R}^m, \mathbb{R}^m; p)$, and assume that $v_j, \phi(v_j) \in L_2(p)$ for every $j = 1, \dots, m$, so that the space E defined below is contained in $L_2(\mathbb{R}^m, \mathbb{R}^m; p)$. Let Π be the orthogonal projection operator onto the vector subspace*

$$E = \left\{ A\mathbf{v} + B\phi(\mathbf{v}) + I \mid A, B \in \mathbb{R}^{m \times m}, I \in \mathbb{R}^m \right\}$$

If $\|H - \Pi H\| > 0$, then $\inf_{\theta} \|H(\mathbf{v}) - \mathbf{m}_{\text{sc}}(\mathbf{v})\| \geq \|(1 - \Pi)H\| > 0$.

²We later need it to be non-polynomial in theorem 2.3

Proof of the proposition is given in Appendix A.2.1. The proposition says that no matter how the parameter of $\mathbf{m}_{\text{sc}}$ is tuned, the difference between $\mathbf{m}_{\text{sc}}$ and the target function cannot approach 0, and the lower bound of the error is given by the norm of the component in the target function that is orthogonal to the finite-dimensional function space E . Therefore, synaptic current dynamics have a limited ability to match the score function and hence limited ability to sample from complex probability distributions under the strict orthogonality constraint in section 2.3.1. The conclusion holds even if we let the diffusion coefficient σ be tunable. Since Σ^{-1} is linear, $\{\mathbf{m}_{\text{sc}}\}_\theta$ share the same set of basis functions as $\{\Sigma^{-1}\mathbf{m}_{\text{sc}}\}_{\theta,\sigma}$. As we will see in the next section, the firing rate dynamics of a recurrent neural circuit with a separate output layer (a reservoir-sampler network) and a learnable diffusion coefficient σ can sample from arbitrary stationary distributions.

2.3.3 Firing-rate dynamics could be a universal sampler

Sampler-only networks

In this section, we consider the firing rate dynamics [Dayan and Abbott, 2005] (cf. eq. 7.11) that describe a recurrent neural circuit in terms of firing rates of the neurons. We first consider the sampler-only network:

$$d\mathbf{r} = D(-\mathbf{r} + \phi(W_{\text{rec}}\mathbf{r} + I))dt + \sigma dB_t := \mathbf{m}_{\text{fr}}(\mathbf{v})dt + \sigma d\mathbf{B}_t \quad (2.8)$$

Here $\mathbf{m}_{\text{fr}}(\mathbf{v}) = D(-\mathbf{r} + \phi(W_{\text{rec}}\mathbf{r} + I))$ and D is a diagonal matrix with decay constants as its diagonal elements. The stationary solution of the corresponding Fokker-Planck equation satisfies

$$\nabla \cdot (\Sigma \nabla p - p \mathbf{m}_{\text{fr}}) = 0 \quad (2.9)$$

where $\Sigma := \frac{1}{2}\sigma\sigma^T$ is symmetric positive definite (SPD) and $\mathbf{m}_{\text{fr}} = D(-\mathbf{r} + \phi(W_{\text{rec}}\mathbf{r} + I))$. Here $\theta = \{D, W_{\text{rec}}, I\}$ is the set of tunable parameters. If Σ is invertible, equation (2.9)

becomes $\nabla \cdot (\Sigma(\nabla p - p\Sigma^{-1}\mathbf{m}_{\text{fr}}))$. Therefore, if $\Sigma^{-1}\mathbf{m}_{\text{fr}} = \nabla \log p$ is a gradient field, then the stochastic dynamics of the recurrent neural network described by equation (2.8) have a stationary distribution p^* such that the score of this distribution is $\nabla \log p^* = \Sigma^{-1}\mathbf{m}_{\text{fr}}$.

Compared with the synaptic current dynamics, for which we could use only the basis functions $\mathbf{v}_i$ and $\phi(\mathbf{v})_i$, we can now freely choose the basis functions spanning $\{\mathbf{m}_{\text{fr}}\}_\theta$ through W_{rec} and I . However, because there is no linear term before the nonlinear transformation ϕ , the function set $\{\mathbf{m}_{\text{fr}}\}_\theta$ is not closed under addition. If we view $\Sigma^{-1}\mathbf{m}_{\text{fr}}$ as a neural network with one hidden layer, the number of hidden neurons must equal the input dimension, and the diffusion matrix Σ (hence Σ^{-1}) is restricted to be an SPD matrix. Therefore, we do not obtain universal approximation power from $\Sigma^{-1}\mathbf{m}_{\text{fr}}$. Combined with the results in Section 2.3.2, this shows that an RNN by itself does not intrinsically produce samples from arbitrary distributions. As we will see below, if a population of output neurons receives inputs from a large reservoir of recurrently connected neurons (a reservoir-sampler network), the output neurons can produce samples from complex distributions.

This limitation is a useful point of comparison with JEPA-style world models [Dawid and LeCun, 2024, Assran et al., 2023, Bardes et al., 2024, Assran et al., 2025, Maes et al., 2026, Wu et al., 2026]. In those models, latent representations are not arbitrary encodings of sensory input: the prediction target and latent regularizer restrict which information is preserved while avoiding representational collapse [Maes et al., 2026, Wu et al., 2026]. While regularization is achieved in JEPA-style models through a specific regularization loss, the restriction studied in sampler-only networks can be treated as a form of regularization with a different origin. It is not imposed by an auxiliary representation-learning regularizer, but is a structural consequence of the neuronal mechanism itself. The choice of state variables, recurrent architecture, readout map, and stochastic dynamics determines the function class available for representing score fields, and therefore restricts the family of probability distributions that a standalone recurrent sampler can realize. In this sense, the dynamics provide a possible biological implementation of representation regularization: the admissible representations

are constrained intrinsically by the RNN dynamics, rather than by an externally engineered objective. If the readout layer in the reservoir-sampler construction is interpreted as the analogue of the encoder in JEPA-style world models, then the reservoir can be viewed as representing a regularized latent distribution of representations (see also Appendix A.6.2).

Reservoir-sampler networks

Now we consider the reservoir-sampler network, in which a linear readout layer $W_{\text{out}} \in \mathbb{R}^{m \times n}$ receives input from the reservoir whose dynamics are given by equation (2.8) (see also the upper row of Figure 2.1). As a special case of Itô’s lemma, we have

$$\begin{aligned} W_{\text{out}} d\mathbf{r} &= dW_{\text{out}}\mathbf{r} = W_{\text{out}}\mathbf{m}_{\text{fr}}dt + W_{\text{out}}\sigma dB_t \\ &= (-W_{\text{out}}D\mathbf{r} + W_{\text{out}}D\phi(W_{\text{rec}}\mathbf{r} + I))dt + W_{\text{out}}\sigma dB_t \end{aligned} \quad (2.10)$$

Now we assume that W_{rec} is the product of $\widetilde{W}_{\text{rec}}$ and W_{out} , i.e. $W_{\text{rec}} = \widetilde{W}_{\text{rec}}W_{\text{out}}$ and $D = \alpha\mathbf{I}$ is a scaled identity matrix. If we denote the output of the recurrent neural network as $\mathbf{x} := W_{\text{out}}\mathbf{r} \in \mathbb{R}^m$, we derive the following stochastic dynamics for output neurons:

$$d\mathbf{x} = (-\alpha\mathbf{x} + \alpha W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}\mathbf{x} + I))dt + W_{\text{out}}\sigma dB_t := \widetilde{\mathbf{m}}_{\text{fr}}(\mathbf{x})dt + \tilde{\sigma}dB_t \quad (2.11)$$

where $\widetilde{\mathbf{m}}_{\text{fr}}(\mathbf{x}) = (-\alpha\mathbf{x} + \alpha W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}\mathbf{x} + I))$ and $\tilde{\sigma} = W_{\text{out}}\sigma$. Therefore in order for the output neurons to sample from a stationary distribution p , we need $s_{\beta}(\mathbf{x}) = (\frac{1}{2}\tilde{\sigma}\tilde{\sigma}^T)^{-1}\widetilde{\mathbf{m}}_{\text{fr}}(\mathbf{x}) := \widetilde{\Sigma}^{-1}\widetilde{\mathbf{m}}_{\text{fr}}(\mathbf{x})$ to match the score $\nabla \log p(\mathbf{x})$. Here $\beta = \{\widetilde{W}_{\text{rec}}, W_{\text{out}}, I, \sigma\}$ are tunable parameters. Additionally, we assume that $\widetilde{\mathbf{m}}_{\text{fr}}$ is $\mathbf{0}$ outside a reasonable range of $\mathbf{x}$. This assumption is used to prevent $s_{\beta}(\mathbf{x})$ from behaving wildly outside the bounded region on which $s_{\beta}(\mathbf{x})$ has the expressivity to match the score. The following theorem proves that with a large enough number of reservoir neurons, the score-matching loss can be arbitrarily small. The proof is given in Appendix A.3.

Theorem 2.3. *Suppose that we are given a probability distribution with continuously differ-*

entiable density function $p(\mathbf{x}) : \mathbb{R}^m \rightarrow \mathbb{R}^+$ and score function $\nabla \log p(\mathbf{x})$ for which there exist constants $M_1, M_2, a, C, k > 0$ such that

$$p(\mathbf{x}) < M_1 e^{-a\|\mathbf{x}\|} \quad (2.12)$$

$$\|\nabla \log p(\mathbf{x})\|^2 < M_2 \|\mathbf{x}\|^k \quad (2.13)$$

when $\|\mathbf{x}\| > L$ for large enough L . Then for any $\varepsilon > 0$, there exists a recurrent neural network whose firing-rate dynamics are given by (2.11), whose recurrent weights, output weights and the diffusion coefficient are given by $W_{\text{rec}} \in \mathbb{R}^{n \times n}$ of rank at most m , $W_{\text{out}} \in \mathbb{R}^{m \times n}$, and $\sigma \in \mathbb{R}^{n \times m}$ respectively, such that, for a large enough n , the score of the stationary distribution of the output units $s_\theta(\mathbf{x})$ satisfies $\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[\|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2] < \varepsilon$.

This theorem says that for any realistic data distribution with a smooth positive density function, there always exists a reservoir of recurrently-connected neurons whose output units give samples from a distribution whose score function approximates that of the data distribution to arbitrary precision³. It is also worth noting that the recurrent weight matrix of the neural circuit in the theorem is of low-rank, so regardless of how many neurons there are in the reservoir, we can always find a low-rank recurrent weight matrix such that the output neurons sample from a correspondingly low-dimensional distribution.

2.3.4 An efficient RNN weight learning algorithm

The training procedure is derived from the proof of theorem 2.3. The main idea is to first train an auxiliary neural network with one hidden layer, then transfer the weights of the auxiliary neural network to the weights of the recurrent network that we are considering. More specifically, we first optimize an auxiliary feedforward neural network with one hidden layer $W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}\mathbf{x} + I)$ using backpropagation with the denoising score matching loss (2.4)

³Under additional Lipschitz, dissipativity, initialization, and discretization assumptions, Block et al. [2022] gives finite-time Wasserstein-2 bounds for a Langevin sampler driven by such an L_2 -accurate score estimate.

such that

$$2\alpha(W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}\mathbf{x} + I) - \mathbf{x}) \approx \nabla \log p(\mathbf{x}) \quad (2.14)$$

Then we can calculate the diffusion coefficient $\sigma = W_{\text{out}}^T(W_{\text{out}}W_{\text{out}}^T)^{-1}$ and the real recurrent weights $W_{\text{rec}} = \widetilde{W}_{\text{rec}}W_{\text{out}}$ accordingly. The noise magnitude added to the data samples is decreased exponentially over the entire training period. Figure 2.2 illustrates how the network can gradually learn the score function during training. We refer the readers to Appendix A.4 for more details.

Note that this method of using an auxiliary neural network is much more computationally efficient than directly matching the score of the stationary distribution of the dynamics (2.11), for which the score function, which involves the matrix inverse $(\tilde{\sigma}\tilde{\sigma}^T)^{-1}$, needs to be recomputed at each optimization step. Furthermore, since the entire training procedure is equivalent to training a feedforward network with one hidden layer, it sidesteps the often challenging temporal computations associated with the Backpropagation Through Time (BPTT) algorithm used to train deterministic RNNs.

Although we assumed that the divergence free field $G = 0$ for the purpose of theoretical analysis, in practice, fast sampling is a major concern for sampling-based inference model of the brain [Hennequin et al., 2014, Aitchison and Lengyel, 2016, Echeveste et al., 2020, Masset et al., 2022], and reversible stochastic dynamics are known for their slow sampling speed. Fortunately, there is a straightforward way to extend our framework and train RSN to implement an irreversible dynamics with a non-zero divergence-free field G , and the sampling speed can be further improved (see Appendix A.5 for details).

2.4 Experimental results

In this section, we present the results from two tasks. First, we try to let the recurrent neural network learn and generate samples from a 1-D double-mode Gaussian mixture distribution and a 2-D mixture of heavy-tailed Laplace distributions. For the second task,

we explore whether a reservoir-sample network with firing rate dynamics is able to sample from the distribution of internal representations computed from PCA filtering of image inputs. All dynamics are simulated with the Euler-Maruyama method. See Appendix A.8 for hyperparameters used and other training details.

2.4.1 Learning bimodal distributions

We consider a 1-D Gaussian mixture distribution whose density function is the average of two Gaussian distributions centered at ± 1 , i.e. $p_{\text{data}}(x) = \frac{1}{2}(\mathcal{N}(-1, 0.25) + \mathcal{N}(1, 0.25))$. We artificially generate 10000 data points from this distribution and minimize the denoising score-matching loss,

$$\mathcal{L}(W_{\text{out}}, \widetilde{W}_{\text{rec}}, I) = \mathbb{E}_{p_{\text{data}}(x)} \mathbb{E}_{\tilde{x} \sim \mathcal{N}(x, \sigma^2)} \left[\left\| 2\alpha(W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}x + I) - x) + \frac{\tilde{x} - x}{\sigma^2} \right\|^2 \right] \quad (2.15)$$

We use $\phi(\cdot) = \text{ReLU}(\cdot)$ and set $\alpha = 1/2$. See figure 2.2 for the numerical results. With $\tanh$ as the transfer function, the exact score is $-4x + 4 \tanh(4x)$, which SO-FR can represent because its inner weight is tunable, while SO-SC only approximates it over the plotted range using its fixed $\{x, \tanh(x), 1\}$ basis. More generally, our theory predicts that each sampler-only network cannot approximate score functions outside its respective finite-dimensional function space. See Appendix A.6.1 for an example where sampler-only networks fail to learn the score function even if hyperbolic tangent nonlinearity is used.

Next, we show that the model can learn mixtures of heavy-tailed distributions that are evident in natural image statistics and the neural representations in the primary visual cortex [Simoncelli and Olshausen, 2001, Olshausen and Field, 1996]. We trained the Reservoir-Sampler network with FR dynamics (RS-FR) model on 20000 sampled data points from a 2-D Laplace mixture distribution, whose density is given by $p_{\text{data}}(\mathbf{x}) = \frac{1}{2} \left(\text{Lap} \left(\mathbf{0}, \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix} \right) + \text{Lap} \left(\mathbf{0}, \begin{bmatrix} 1 & -0.9 \\ -0.9 & 1 \end{bmatrix} \right) \right)$, where Lap denotes the multivariate Laplace distribution. The model successfully learned the probability density of the mixture

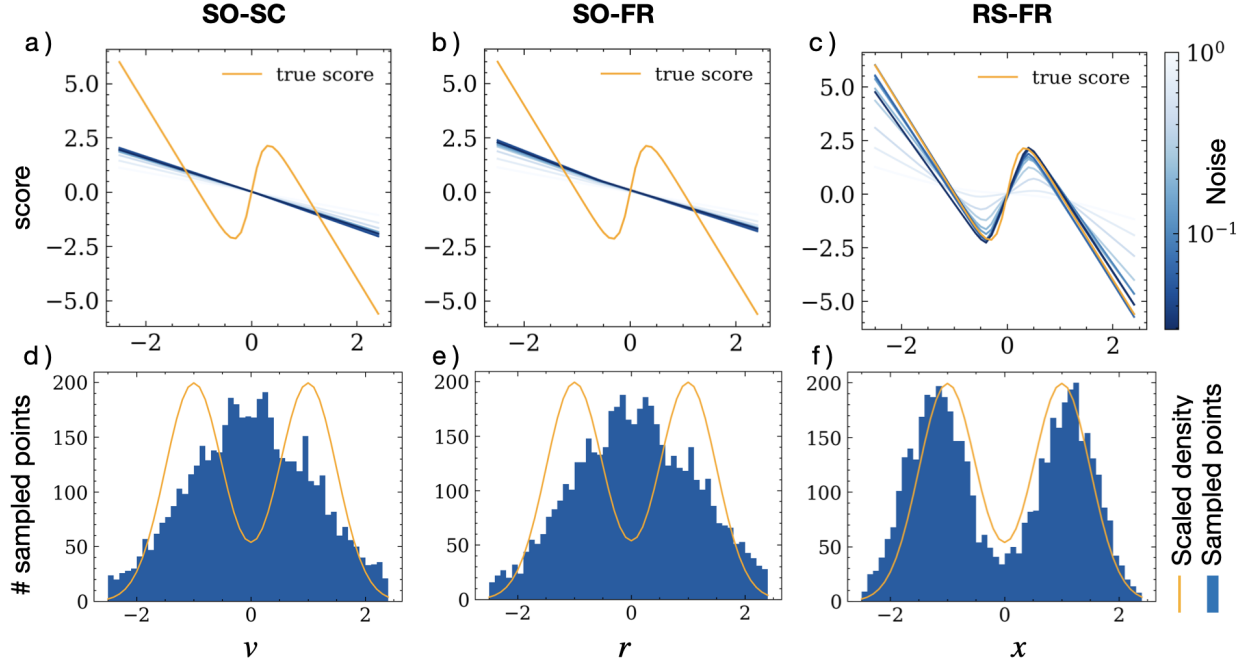


Figure 2.2: **Bimodal distribution sampling results.** The 3 tractable cases shown are Sampler-only (SO) networks with both synaptic current (SC) dynamics and firing rate (FR) dynamics and Reservoir-sampler (RS) networks with FR dynamics, which are named SO-SC, SO-FR and RS-FR respectively. a-c) The score function learned compared to the true score function (orange curve) as we gradually decrease the noise level (the darker the line, the lower the noise level). We see that RS-FR is capable of perfectly fitting the score function, while SO-SC and SO-FR are only able to fit the score function with piecewise linear functions when using the ReLU transfer function. d-f) Histogram of sampled points, and the (scaled) density function of the target distribution. Again the reservoir-sampler network is able to generate samples whose distribution matches the target distribution, while the sampler-only network is not able to do so due to the incorrectly matched score function.

distribution (Figure 2.3 left vs. middle), and captured the heavy tails of the distribution as measured by the kurtosis (Figure 2.3 right).

2.4.2 MNIST generation task

We also tested the sampling ability of our model using the MNIST dataset [Lecun et al., 1998] which contains 60,000 handwritten digits from 0 to 9. We projected MNIST images to a 300-D latent space spanned by the first 300 principal components, and trained the weights of the recurrent neural network as described in section 2.3.4 so that the RNN can sample

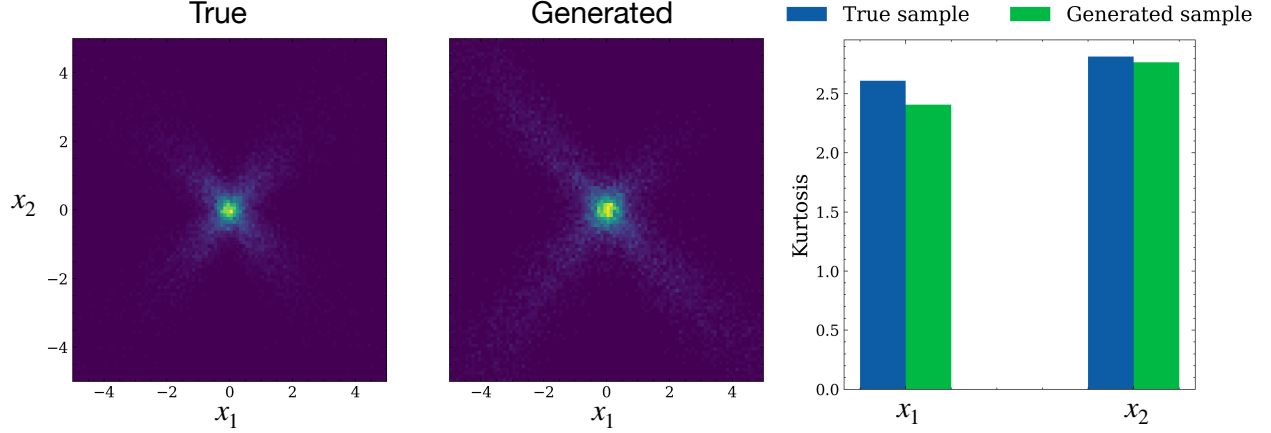


Figure 2.3: **RS-FR model learning a mixture of 2-D Laplace distribution.** Left to right: sample density from the true distribution (brighter color denotes higher density); sample density from the learned distribution; marginalized kurtosis of each dimension from the true and learned distribution.

from the latent distribution. To test the model, we generated images by applying inverse PCA projection to samples generated by the model. The schematics and generated images are shown in Figure 2.4. Note that since we are essentially using a shallow network to match the score, we should not expect comparable performance to generative models that use deep ANNs. Our main goal is to illustrate that the reservoir-sampler network using firing rate dynamics is qualitatively more expressive than other traditional neural sampling models (Appendix A.7). Finally we also note that it is highly nontrivial for recurrent neural dynamics to complete such a generative task, and to the best of our knowledge, no previous work has achieved such results.

2.5 Discussion

From the perspective of functional analysis and SDE theory, we prove that under the strict orthogonality constraint, it is essential for neural circuits to have a drift term that has the expressivity to approximate complex score functions, despite the fact that the dynamics do not have to exactly implement the Langevin dynamics. We investigated whether a population of neurons can sample from an arbitrary distribution directly and proved that the drift term

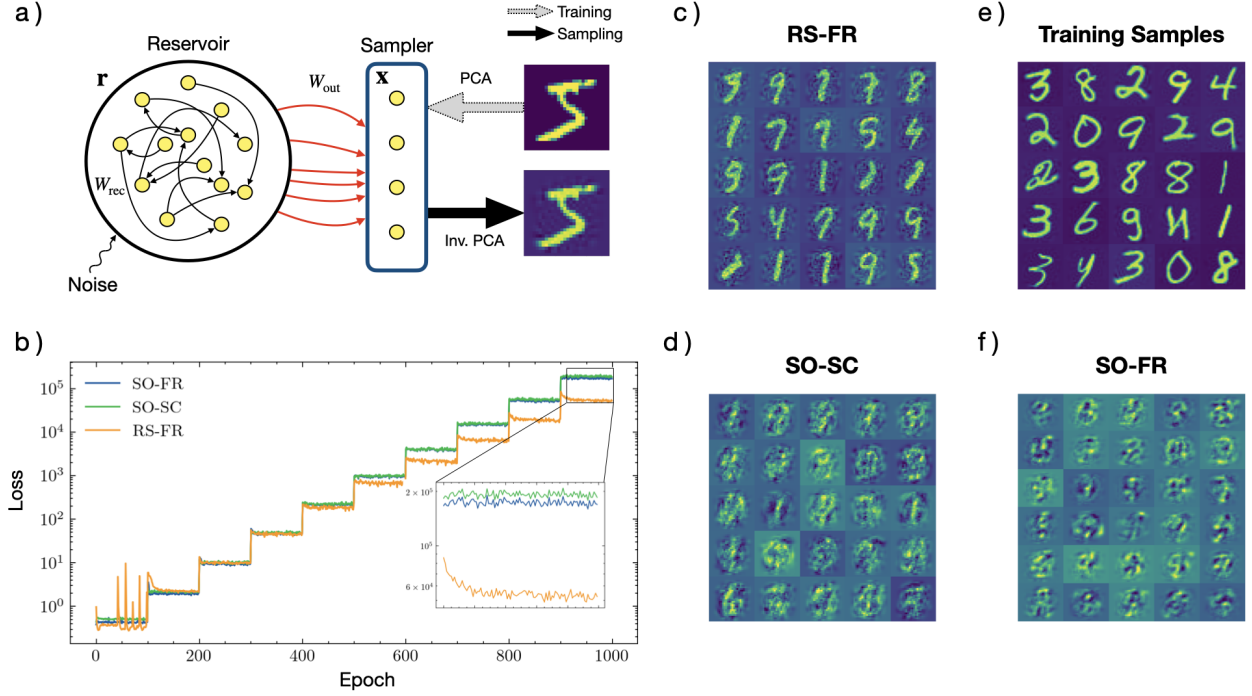


Figure 2.4: **Learning to sample the MNIST image distribution.** a) An MNIST image is projected to a 300-D latent space (orange circle) spanned by the first 300 principal components using PCA. The sampler learns to sample from the distribution of this latent space and generates images by applying inverse PCA projection to these samples. The diagram illustrates the RS-FR model. b) The loss curves for 3 different models during training. Every 100 epochs the noise level added to the training samples is reduced (Appendix A.8), and the noise increases to a higher value because the score matching loss magnitude depends on the noise level. As shown in the inset, the loss of the RS-FR model decreases throughout the training process when using the lowest fixed noise level. Meanwhile, the losses of the other two models remain unchanged. c-f) The images generated for the 3 models compared to the digit images generated from latent training samples.

of the synaptic current dynamics can only sample from a finite-dimensional function space. Although the drift term of the firing rate dynamics can approximate functions spanned by different basis functions, the number of basis functions is limited. To address this problem, we proposed the reservoir-sampler network for firing rate dynamics. We found that with learnable diffusion coefficients and a sufficiently large reservoir of hidden neurons, the output neurons described using the firing rate dynamics are able to sample from arbitrary data distributions. Our results partly answer the question of what architecture recurrent neural circuits need so that they are able to sample from complex data distributions.

Our analysis and empirical experiments affirm the universality of the heavily used deep learning architecture, artificial RNNs, in a probabilistic setting. However, this universality comes with limitations. First, we have only analytically shown the existence of weights that enable sampling from complex data distributions; there is no guarantee that one will find such weights through backpropagation. Additionally, in order to obtain the tunable diffusion coefficient during training, a matrix inverse is needed (likewise in the FORCE algorithm [Sussillo and Abbott, 2009]). Further, the question of how biological circuits compute the specific gradient and implement the denoising score-matching algorithm remains an open question. Moreover, in our current formulation, we are only able to approximate the score function with a shallow network with one hidden layer. Our preliminary experiments show that one-hidden-layer RSNs cannot readily approximate high-dimensional heavy-tailed distributions (*e.g.*, those of overcomplete sparse coding representations [Olshausen and Field, 1996]). Theoretical results show that a deep feedforward network with more than one hidden layer can find the global minimum despite the non-convex nature of the loss landscape [Du et al., 2019]. Therefore an important step for future research is how small populations of recurrent neural circuits can form hierarchical patterns or block structures in the connection matrix, to match the performance of deep diffusion models.

A second practical limitation is temporal: samples produced by the dynamics should not be interpreted as stationary samples immediately after initialization. Like other Markovian samplers, the network requires burn-in time before its output distribution is close to the target distribution. In a biological interpretation, however, this transient need not be viewed only as an inefficiency. Ambiguous perceptual stimuli such as the Necker cube provide a useful analogy. The physical input is fixed, but perception can take time to settle into one coherent three-dimensional interpretation, and it may later switch between competing interpretations. This suggests that a settling period can be part of the computational process by which an internal probabilistic neuronal state becomes cognitive.

Compared with recent diffusion models [Ho et al., 2020, Song et al., 2021], our model

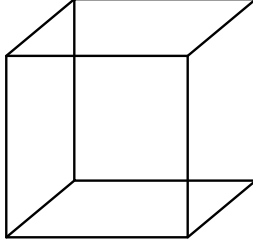


Figure 2.5: **Classic Necker cube ambiguity.** The same two-dimensional line drawing supports multiple three-dimensional percepts. This ambiguity provides cognitive evidence for the existence of probabilistic neural dynamics: the stimulus is fixed, but the internal interpretation can require time to stabilize or switch between competing modes.

has the advantage of being able to run indefinitely in time, making it a suitable candidate for modeling spontaneous activity in the brain. Moreover, while Song and Ermon [2019] optimizes the denoising score-matching loss at different noise levels jointly, we adopt a sequential learning procedure by gradually decreasing the noise level of the training samples. This procedure is more aligned with the developmental processes involved in forming visual representations in the infant brain, where visual representations are thought to be noisier (less linearly separable) initially [Bayet et al., 2020]. Our study therefore serves as a starting point for building a mechanistic model for probabilistic computation in the brain that has generative power similar to that of current AI generative models.

Biologically, there are multiple ways to interpret the reservoir and sampler neurons in an RSN. First, reservoir and sampler neurons can be seen as different types of neurons in a single population, where the dynamics of sampler neurons converge quickly to an equilibrium point. Alternatively, reservoir neurons can be interpreted as a hidden nonsynaptic signaling network (e.g., pervasive neuropeptidergic signaling [Bargmann and Marder, 2013], extensive aminergic signaling [Bentley et al., 2016], or potential extrasynaptic signaling [Yemini et al., 2021]) that is not available in the connectome. This interpretation is supported by the recent finding of a low correlation between functional activity and the structural connectome in *C. elegans* [Yemini et al., 2021]. Notably, if we consider only the structural connectome, then the resulting biophysical model of *C. elegans* corresponds to the sampler-only network, which, as our theory predicts, has limited sampling capability. Therefore, our theory is consistent

with this experimental finding.

2.6 Conclusion

In this chapter, we explore how a recurrent neural circuit can sample from complex probability distributions, an important functional motif in probabilistic brain models. We start from the basic assumption that a recurrent neural circuit can be described as an SDE. We show that a recurrently connected neural population by itself has a limited capacity to implement stochastic dynamics that can sample from complex data distributions. In contrast, we prove that the firing-rate dynamics of the output units of a recurrent neural circuit (a reservoir-sampler network) can sample from a richer range of probability distributions. These theoretical results, together with our preliminary experimental results, provide a sufficient condition for neural sampling-based models to exhibit universal sampling capability. Our results therefore provide a foundation for the next generation of sampling-based probabilistic brain models that can explain a wider range of cognitive behaviors.

Chapter 3

Token Probabilities as Progress

Rewards for Robot Learning¹

The preceding chapter asked how recurrent neural dynamics can sample from a target distribution. Here, a pretrained video-language model has already learned a conditional distribution over language given visual and textual context. This chapter asks whether the output token probability can provide a useful progress signal for robot learning.

3.1 Introduction

Recent advances in Vision-Language-Action (VLA) models have spurred significant interest in leveraging Reinforcement Learning (RL) to achieve truly generalizable real-world performance Lei et al. [2025], Chen et al. [2025a], Xiao et al. [2025]. However, real-world RL remains bottlenecked by the extreme sample inefficiency inherent in sparse reward signals. To bridge this gap, the research community has pivoted toward developing generalizable process reward models that provide fine-grained and dense feedback. Current efforts typically focus on directly fine-tuning vision-language models (VLMs) as process reward functions

¹This chapter is based on Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J. Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna, “TOPReward: Token Probabilities as Hidden Zero-Shot Rewards for Robotics” (2026) [Chen et al., 2026a].

on curated robot datasets [Duan et al., 2024, Budzianowski et al., 2025, Ma et al., 2024, Lin et al., 2025b] or training specific networks with custom-collected datasets Zhang et al. [2025], Chen et al. [2025a]. For instance, RoboDopamine [Tan et al., 2025] trains a reward model on 3,400+ hours of manipulation data with step-aware multi-view perception, but requires task-specific demonstrations for adaptation. Likewise, Lee et al. [2026] introduces RoboReward, which fine-tunes a VLM on a large-scale set of robot trajectories with human-provided success labels and progress scores. However, these approaches rely on costly data requirements: RoboDopamine requires additional demonstrations when adapting to each new task, and RoboReward reports clear gaps across different embodiments and views, indicating limited generalization guarantees. Therefore, while these efforts demonstrate promise for narrow-domain reward models, they still rely on extensive data collection and struggle to generalize beyond the training distribution.

To circumvent the high costs of task-specific fine-tuning, we investigate the use of pretrained VLMs as zero-shot reward models. Our goal is to harness the pretrained visual-language representations embedded in these models to provide generalizable instruction-conditioned progress signals. Recent literature [Rocamonde et al., 2023, Ma et al., 2024, Baumli et al., 2023] has converged on progress estimation as a useful proxy for value, as it provides dense temporal feedback for learning and adaptation. The current state-of-the-art training-free method, GVL [Ma et al., 2024], casts progress estimation as visual question-answering—but performs well only on proprietary VLMs like Gemini and GPT-4 [Budzianowski et al., 2025], collapsing on open-source alternatives. Indeed, contemporary studies [Zhang et al., 2026] suggest that open-source VLMs are not yet “robotics-ready” for progress estimation.

In this work, we challenge the prevailing assumption that open-source VLMs are unfit for reward modeling by introducing a novel formulation for zero-shot progress estimation. We hypothesize that the failure of current open-source models stems not from a lack of temporal understanding, but from the representation bottleneck of textual output—specifically, the models’ inconsistent instruction-following and their notorious bias in representing numerical

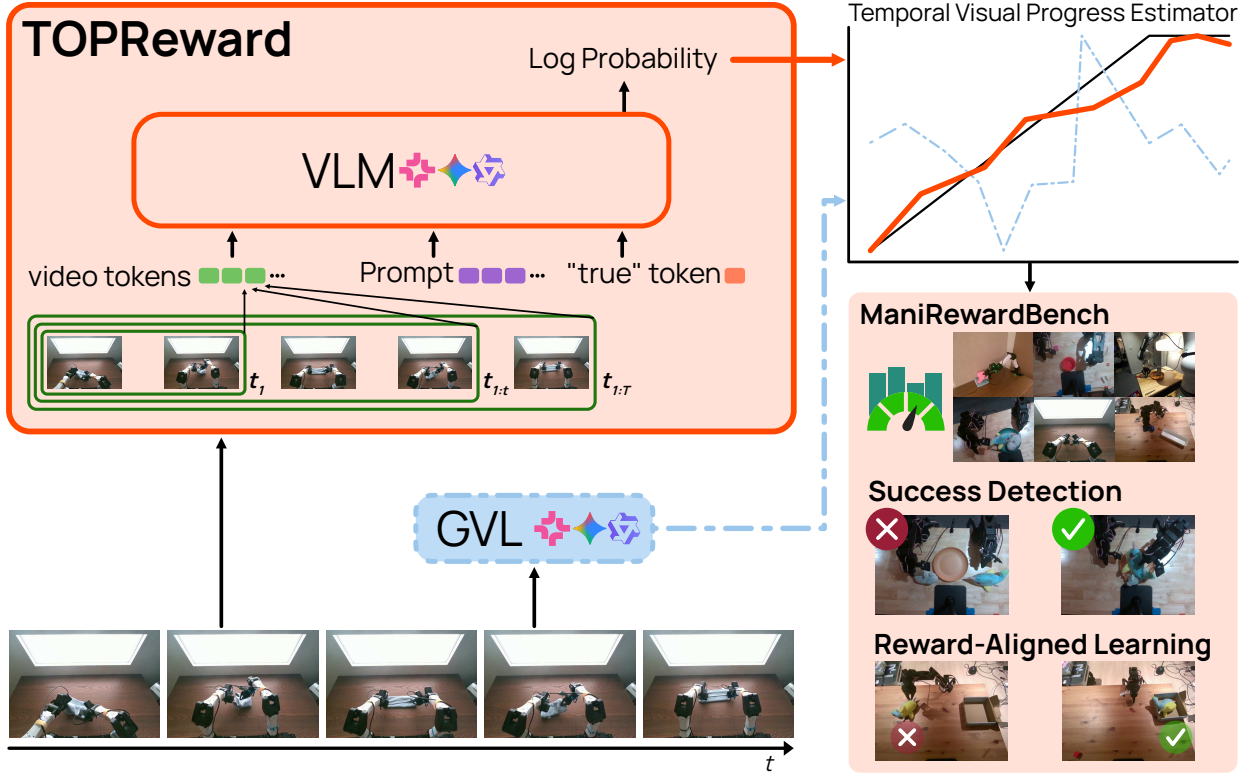


Figure 3.1: **Result highlights.** TOPReward enables effective zero-shot estimation of task progress across diverse and challenging real-world manipulation tasks, without task-specific training. Across several VLM backbones, TOPReward provides a temporally consistent visual reward signal that supports multiple downstream applications, including success detection, policy improvement, and evaluation on our in-house benchmark, ManiRewardBench.

tokens. To resolve this, we present TOPReward, a probabilistically grounded progress estimator that bypasses autoregressive text generation entirely. Instead of prompting the VLM to output the value of the completion percentage in text space, TOPReward extracts the model’s internal “belief” by analyzing the probabilistic distribution of its token logits. By measuring the shift in confidence toward task-completion tokens over time, we derive a continuous, well-posed progress signal directly from the VLM’s latent probabilistic representation. This approach requires zero additional training or fine-tuning, revealing that robust reward modeling is an emergent capability already present in pretrained video VLMs—if one looks beyond the text.

To evaluate progress estimation methods rigorously, we introduce ManiRewardBench, a benchmark comprising 130 unique real-world manipulation tasks spanning multiple robot platforms (Franka Emika arms, Single-Arm/Bimanual YAM, SO-100/101) with temporal

annotations of task progress. We demonstrate that **TOPReward** can effectively track task progress across this diverse benchmark. The raw completion probability supports cross-trajectory success detection, while normalized prefix scores provide within-trajectory progress curves for evaluation and visualization. We further use **TOPReward** to weight expert examples in offline behavior cloning. In real-world deployment on six single-arm SO-100 manipulation tasks, reward-weighted fine-tuning with **TOPReward** consistently improves success rates over standard behavior cloning, achieving up to 10 out of 10 successes on challenging tasks where baseline behavior cloning reaches only 7 out of 10.

3.2 Related Work

The reward bottleneck for VLA. Large-scale vision-language-action policies such as OpenVLA Kim et al. [2024], π_0 Black et al. [2026], MolmoAct Lee et al. [2025] and Gemini Robotics Team et al. [2025] have demonstrated strong language-conditioned manipulation capabilities across diverse embodiments, yet reliably deploying them in real-world settings remains an open problem Firoozi et al. [2025]. A natural path forward is reinforcement learning with online or offline fine-tuning, but RL in practice hinges on the availability of a reward signal—one that is traditionally hand-crafted per task in robotics, difficult to scale, and brittle under distribution shift Kober et al. [2013], Dulac-Arnold et al. [2019]. Recent efforts have applied RL to improve generalist robot policies in real-world deployment Zhang et al. [2024], Nakamoto et al. [2023], Chen et al. [2025c], Hu et al. [2025a], Ankile et al. [2025], Wagenmaker et al. [2025], Dong et al. [2025], including RL-100 Lei et al. [2025], which trains diffusion-based visuomotor policies directly on real robots using human-provided success signals, and $\pi_{0.6}^*$ Intelligence et al. [2025], which improves π_0 through real-world RL with human-annotated episode outcomes. All of these approaches, however, remain reliant on manual reward specification, underscoring the need for automated, scalable alternatives.

Learned reward models. A long line of work seeks to replace hand-crafted rewards with

learned alternatives Pomerleau [1991], Ho and Ermon [2016], Hester et al. [2017]. Embedding-based methods such as VIP Ma et al. [2022], LIV Ma et al. [2023a], and R3M Nair et al. [2022] learn visual representations that capture progress toward a goal, but require task-specific fine-tuning and offer limited language grounding. VQA-style approaches such as SuccessVQA Du et al. [2023] and related frameworks Stone et al. [2023], Huang et al. [2022] reframe reward as a binary classification problem—asking a VLM whether the task succeeded—but produce signals too coarse for dense reward shaping Lynch et al. [2020], Andrychowicz et al. [2017]. Code-generation strategies like Eureka Ma et al. [2023b] synthesize reward functions via LLMs, yet depend on access to simulation ground truth. More recently, generalist reward models such as RoboReward Lee et al. [2026] and RoboDopamine Tan et al. [2025] are trained on large-scale datasets of successes and failures to predict progress scores or distance-to-goal estimates Wu et al. [2023], Fan et al. [2022]. While these models move toward broader coverage, they still require domain-specific training data and can struggle to generalize across embodiments and environments Kalashnikov et al. [2021]. A common thread across all these approaches is the dependence on training or domain-specific resources—a requirement our method avoids.

Training-free value estimation with VLMs. A separate and more relevant line of work asks whether VLMs can estimate task progress without any additional training. Generative Value Learning (GVL) Ma et al. [2024] poses progress prediction as a temporal ordering problem: given a batch of shuffled trajectory frames, the VLM is prompted to assign per-frame progress scores, exploiting its semantic grounding to rank frames by task completion. This enables various downstream applications such as dataset filtering and advantage-weighted regression without task-specific reward engineering. The OpenGVL benchmark Budzianowski et al. [2025] evaluates this paradigm across diverse tasks and model families, revealing that open-source VLMs fall substantially short of their proprietary counterparts on temporal progress prediction. In this chapter, we hypothesize that the reason for this is not a lack of visual understanding in open-source VLMs but rather the instability

of numeric token generation—LLMs are poorly calibrated when asked to produce precise numerical outputs Wallace et al. [2019], Yuan et al. [2023]. This observation motivates a fundamental shift: rather than asking a VLM to *generate* a progress value, one can instead probe what the model already *knows* through its internal representations.

Internal representations as reward signals. A growing body of work in NLP has shown that a model’s internal activations—logits, hidden states, and embeddings—track its certainty and factual accuracy more reliably than its generated text Kadavath et al. [2022], Tian et al. [2023], Azaria and Mitchell [2023], Burns et al. [2022], Liu et al. [2023]. In robotics, recent methods have begun to leverage such representations for reward definition Rocamonde et al. [2023], Grislain et al. [2025], bypassing the instabilities of text generation. Our work, TOPReward, takes this principle further: instead of prompting a VLM to generate numeric progress estimates, we pose a binary completion query (“does this trajectory complete the task?”) and extract the probability of the affirmative token as a continuous reward signal. This formulation is zero-shot, requires no fine-tuning or domain-specific data, and yields a useful progress signal that scales to the 130-task ManiRewardBench benchmark as well as the Open X-Embodiment dataset across multiple robot platforms.

3.3 TOPReward

The stark difference between GVL’s [Ma et al., 2024] performance on Gemini versus open-source models might mislead one into believing that only the most powerful VLM can accurately estimate the progress of a robotic trajectory. Indeed, for a model to output well-formatted and accurate progress estimates for all shuffled frames, it needs strong instruction-following capability and an accurate internal representation of numerical values. However, neither of these is necessarily correlated with the model’s underlying video understanding capability. Therefore, we propose TOPReward, a method that leverages the internal understanding of the VLM to produce a progress estimator without requiring accurate, well-formatted

numerical generation.

Problem setup. We formulate the progress estimation task as follows: given an instruction x and a video trajectory $\tau_{1:T} = (I_1, \dots, I_T)$ (frames in chronological order), our goal is to produce a scalar prefix score for each $\tau_{1:t}$ that reflects accumulated evidence that the instruction has been completed.

3.3.1 Token probability as the reward

Key idea.

We use the VLM’s *internal output*, *i.e.*, *predicted token probabilities* as the reward. Concretely, we ask the model to judge whether the observed trajectory *completes* the instruction and score the probability of an affirmative answer (e.g. the token **True**).

Let p_θ be a VLM defining a next-token distribution with pretrained weights θ . We form a prompt u that grounds the judgment in the video:

“<|video|> The above video shows a robot manipulation trajectory that completes the following task: {INSTRUCTION}. Decide whether the above statement is True or not. The answer is: {a}”

and compute the probability of the answer token sequence a =“True”². We choose **True** because we found boolean tokens to show the clearest success–failure separation, with **True** exhibiting the largest absolute difference in mean token probability across episodes (see Appendix B.2 for the token-probability comparison). Denoting the (video-conditioned) textual context by $c(\tau_{1:t}, u)$, we define the reward for a prefix to be

$$r_t = \log p_\theta(a \mid c(\tau_{1:t}, u)). \quad (3.1)$$

In this way, we construct a conditional completion score for the trajectory-instruction pair while sidestepping the need for the language model to generate calibrated numerical values.

²We also explore another variant that evaluates the probability over the entire instruction which consists of multiple tokens, but found it to be less effective. See Appendix B.1 for details.

The raw score r_t is causal because it depends only on the prefix $\tau_{1:t}$ and the instruction. As we will see in Section 3.4, r_t tends to increase on successful demonstrations as visual evidence accumulates, while its absolute level is useful for comparing complete successful and failed trajectories.

3.3.2 Progress estimation from trajectory prefixes

Prefix sampling. To obtain a temporal progress curve, we evaluate Eq. (3.1) on a set of K uniformly spaced prefix lengths $\{t_k\}_{k=1}^K$ with $1 = t_1 < \dots < t_K = T$. This involves K model forwards and produces rewards $\{r_{t_k}\}$ summarizing how completion evidence accumulates over time (Figure B.6).

Normalization. The raw log-probability range is $(-\infty, 0]$, while VOC and visualizations are easier to interpret on a bounded scale. For within-trajectory evaluation, we therefore use min-max normalization to map rewards to a normalized progress score s_{t_k} within each episode:

$$s_{t_k} = \frac{r_{t_k} - \min_j r_{t_j}}{\max_j r_{t_j} - \min_j r_{t_j} + \varepsilon}, \quad (3.2)$$

with a small ε for numerical stability. This normalization is non-causal because the episode-level minimum and maximum are known only after all sampled prefixes are scored. We use it for within-trajectory progress visualization and VOC evaluation, not as the raw online reward signal.

Positive weights for downstream offline learning. When a per-step weight is needed for reward-weighted offline behavior cloning, we use the TOPReward increment to construct a positive weighting signal:

$$\Delta_{t_k} = \min \left\{ \exp(\beta \cdot (r_{t_k} - r_{t_{k-1}})), \delta_{\max} \right\}, \quad (3.3)$$

where β controls the sharpness of the weighting and $\delta_{\max}$ caps large weights for training stability. The weight is always positive: steps with decreasing TOPReward score are down-

weighted rather than treated as negative supervision, which keeps the flow-matching objective well-posed.

3.4 Experiments

We evaluate TOPReward across three main dimensions: (1) zero-shot progress estimation on large-scale robot datasets using VOC and complementary progress metrics (Section 3.4.1); (2) instruction sensitivity; and (3) downstream applications including success detection and real-world reward-weighted behavior cloning.

VLM backbones. We evaluate TOPReward on Qwen3-VL-8B and Qwen3-VL-32B [Bai et al., 2025], Molmo2-8B [Clark et al., 2026], and Gemini-2.5-Pro [et al., 2025]. Qwen3-VL and Molmo2 are open-source video-language models; Gemini-2.5-Pro serves as a proprietary baseline with logit access.

Benchmark. ManiRewardBench contains 130 unique real-world manipulation tasks collected across four robot platforms: Franka, SO-100/101, single-arm YAM, and bimanual YAM. Episodes are annotated with subtask boundaries for stage-aware progress evaluation, and a separate 23-task failure split supports success detection. Additional dataset details are in Appendix B.6.

3.4.1 Large-scale real-world evaluation

To evaluate zero-shot progress estimation, we test whether TOPReward produces accurate dense progress estimates on expert robotic trajectories from ManiRewardBench and Open X-Embodiment (OXE). We mainly compare against GVL [Budzianowski et al., 2025, Ma et al., 2024], the state-of-the-art training-free progress estimator, which prompts a VLM to assign numerical progress values to shuffled frames.

Metrics. Following standard practice [Ma et al., 2024, 2023a], we use Value-Order Correlation (VOC) to measure Spearman’s rank correlation between the chronological order of input

video frames and the predicted values,

$$\text{VOC} = \rho_S((s_{t_1}, \dots, s_{t_K}), (t_1, \dots, t_K)), \quad (3.4)$$

where ρ_S is Spearman’s rank correlation, with average ranks assigned to ties. VOC ranges from -1 to 1 : -1 indicates reverse temporal ordering and 1 indicates perfect ordering. Per-episode min–max normalization preserves this ordering unless all raw scores are tied. Because VOC measures only temporal order, it is not a standalone test of instruction grounding. We therefore also report Kendall tau-b, Pearson correlation, and mean absolute error (MAE) against the available completion annotations.

Results on Open X-Embodiment. The Open X-Embodiment (OXE) dataset [O’Neill et al., 2024] is a collection of 50 academic robot datasets spanning diverse tasks, camera configurations, and robot platforms. We select 33 high-quality OXE datasets from the LeRobot collection and use this same filtered set for every row in Table 3.1. On OXE datasets, **TOPReward** substantially outperforms GVL on open-source models such as Qwen3-VL and Molmo. Complementary progress metrics show the same trend beyond VOC across available rows. We also compare against the trained Robometer-4B progress head on the same filtered OXE split using 15 uniformly sampled frames per episode, and find that **TOPReward** is slightly better. On the proprietary Gemini-2.5-Pro, GVL performs better (0.572) while **TOPReward** achieves 0.430, reflecting the chat-template issue discussed in Appendix B.5.2.

Results on ManiRewardBench. We evaluate progress estimation on the successful-trajectory subset of **ManiRewardBench**, covering 113 tasks and 497 episodes across 4 robotic platforms. Results are shown in Table 3.2. On Qwen3-VL-8B, **TOPReward** achieves the strongest aggregate progress metrics, substantially outperforming GVL on the same backbone. On Molmo2-8B, GVL is near zero across rank and correlation metrics, while **TOPReward** recovers a useful progress signal with positive VOC, Kendall tau-b, and Pearson correlation. Compared with the trained Robometer-4B baseline, **TOPReward** is competitive across rank, correlation, and

Table 3.1: **Results on the filtered Open X-Embodiment dataset.** VOC reports mean dataset-level VOC over the same 33 filtered datasets and 20 episodes per dataset. Higher is better except MAE.

Method	Model	VOC	Kendall	Pearson	MAE
GVL	Molmo2-8B	-0.019	0.000	0.001	0.500
GVL	Qwen3-VL-8B	0.218	0.192	0.443	0.396
GVL	Gemini-2.5-Pro	0.572	0.534	0.669	0.218
TOPReward	Molmo2-8B	0.525	0.427	0.503	0.308
TOPReward	Qwen3-VL-8B	0.874	0.771	0.856	0.163
TOPReward	Qwen3-VL-32B	0.890	0.799	0.870	0.191
TOPReward	Gemini-2.5-Pro	0.430	0.332	0.399	0.321
Robometer	4B	0.838	0.765	0.840	0.180

Table 3.2: **Results on ManiRewardBench.** We report metrics averaged over the 497 successful-evaluation episodes. This progress-evaluation subset contains 113 tasks across four robot platforms: LeRobot, Franka, Bimanual YAM, and Single-arm YAM. VOC and Kendall tau-b measure rank agreement, Pearson measures linear agreement, and MAE measures normalized progress error. Higher is better except MAE.

Method	Model	VOC	Kendall	Pearson	MAE
GVL	Molmo2-8B	-0.003	-0.002	-0.003	0.514
GVL	Qwen3-VL-8B	0.316	0.261	0.326	0.392
TOPReward	Molmo2-8B	0.619	0.484	0.569	0.316
TOPReward	Qwen3-VL-8B	0.942	0.858	0.915	0.126
TOPReward	Qwen3-VL-32B	0.864	0.771	0.824	0.194
Robometer	4B	0.858	0.768	0.861	0.160

error metrics while requiring no reward-model training. Detailed per-task breakdowns and distribution plots are in Appendix B.4.

Qualitative results. We visualize representative progress traces in Figure B.5. The traces demonstrate that TOPReward produces smooth, monotonically increasing progress signals that closely track stage-aware ground-truth task completion (computed from annotated subtask boundaries) across diverse manipulation tasks. In contrast, Gemini-GVL exhibits noisier predictions with frequent non-monotonic fluctuations. Notably, TOPReward correctly captures the temporal structure of multi-step tasks, with progress plateaus corresponding to intermediate subtask completions and accelerations during active manipulation phases.

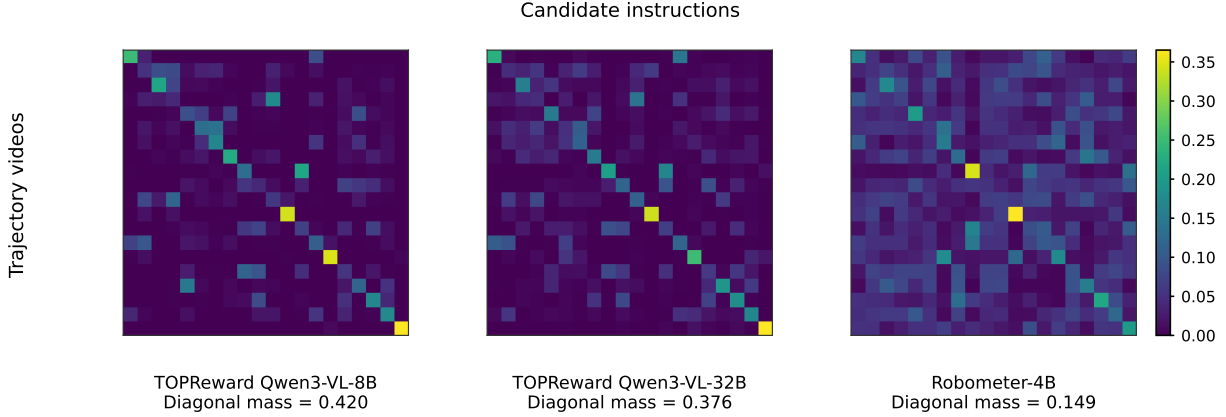


Figure 3.2: **Instruction sensitivity.** Instruction-grounding matrices for TOPReward and Robometer-4B. Rows are terminal trajectory videos and columns are candidate task instructions. For TOPReward, each cell is the terminal TOPReward score for that video–instruction pair; for Robometer-4B, each cell is its terminal reward prediction. We apply Sinkhorn normalization for visualization and report mean diagonal mass, so both rows and columns sum to one and bright diagonal cells indicate strong one-to-one matching between trajectories and their intended instructions.

3.4.2 Instruction sensitivity

Order-based metrics alone cannot establish that a reward is grounded in the language instruction: an instruction-agnostic ordering score can be highly correlated with progress on successful demonstrations while assigning the same values for every instruction. We therefore evaluate whether final-trajectory rewards discriminate the matched instruction from plausible but incorrect task descriptions. For each video, we score the same terminal trajectory under each candidate instruction and form a video-task $\times$ instruction-task matrix, shown in Figure 3.2. For TOPReward, each row entry is the terminal TOPReward score for a video–instruction pair; for Robometer-4B, each row entry is its terminal reward prediction. We convert these score matrices to positive weights and apply Sinkhorn normalization, so each row and each column sums to one. Strong diagonal structure and high mean diagonal mass in this doubly normalized matrix indicate one-to-one matching between trajectories and their intended instructions rather than instruction-agnostic temporal ordering.

Figure 3.2 summarizes instruction disambiguation using mean diagonal mass after Sinkhorn

Table 3.3: **Success detection results.** We evaluate 279 cleaned LeRobot episodes from **ManiRewardBench** (150 successful, 129 failed). **TOPReward** uses the training-free success score defined in Appendix B.3; GVL uses per-trajectory VOC; Robometer-4B uses its trained success head.

Method	Score	ROC-AUC	PR-AUC
GVL (Qwen3-VL-8B)	VOC	0.728	0.893
TOPReward (Qwen3-VL-8B)	success score	0.939	0.952
TOPReward (Qwen3-VL-32B)	success score	0.965	0.973
Robometer-4B	trained success head	0.930	0.951

normalization of the displayed score matrix. This gives a bidirectional one-to-one matching view: each trajectory distributes mass across candidate instructions, and each instruction distributes mass across candidate trajectories. The random baseline is $1/20 = 0.05$ on this 20-task split. The diagonal mass is 0.420 for TOPReward Qwen3-VL-8B, 0.376 for TOPReward Qwen3-VL-32B, and 0.149 for Robometer-4B, showing that TOPReward produces substantially stronger matched video–instruction structure than the trained reward-model baseline. As a scalar control on the 150 successful LeRobot episodes in the progress-evaluation subset, pairing each video with its correct instruction gives mean raw reward -1.68 , while pairing the same videos with mismatched instructions from other tasks drops the mean raw reward to -5.04 . Thus raw instruction-conditioned rewards and confusion matrices provide the evidence for semantic grounding beyond temporal ordering.

3.4.3 Success detection

Setup and results. We evaluate on the cleaned failure trajectory split of **ManiRewardBench**, which contains 150 successful and 129 failed LeRobot attempts. Taking inspiration from self-consistency prompting [Wang et al., 2023], we combine multiple TOPReward scores using different prompts to obtain a success score for success detection as defined in Appendix B.3.

As shown in Table 3.3, Qwen3-VL-32B reaches 0.965 ROC-AUC and Qwen3-VL-8B reaches 0.939 ROC-AUC, both exceeding the trained Robometer-4B success head without reward-model training or success-label supervision.

Table 3.4: **Real-world experiments.** Partial success score out of 10 trials for reward-weighted behavior cloning on single-arm SO-100 tasks.

Task	Pretrained	BC	TOP-RWBC (Ours)
Place toy car in box	1	2	3
Stack red cube on green cube	1.33	1	2.33
Put pen into cup	1.67	5.67	6.33
Place doll in box	0	7	10
Pick up cube	4	7	10
Put cube in cup	4	6	9

3.4.4 Real-world reward-weighted behavior cloning

To further showcase TOPReward as a signal for policy improvement, we use it to derive positive weights for an offline flow-matching behavior cloning objective. We start from a base policy π_0 pretrained on 200 hours of the publicly available single-arm SO-100 dataset (HuggingFace). For each of six real-world tasks, we collect 50 demonstrations (potentially noisy and suboptimal) and use TOPReward to compute prefix scores for the demonstrations. We convert progress increments to weights using Equation (3.3), then fine-tune π_0 with

$$\mathcal{L}_{\text{RWBC}} = \mathbb{E}_{p(a|o), q(a_t|a)} \left[\Delta_t \cdot \|v_\theta(a_t, t | o) - (a - \epsilon)\|^2 \right], \quad (3.5)$$

where $a_t = (1 - t)\epsilon + ta$, $t \sim \mathcal{U}(0, 1)$, $\epsilon \sim \mathcal{N}(0, I)$, and Δ_t is the TOPReward-derived positive weight computed with $\beta = 0.2$ and $\delta_{\max} = 2.0$ in Equation (3.3). We compare TOP-RWBC against behavior cloning (BC) on the same data, measuring partial success as the fraction of predefined subtasks completed per trial, summed over 10 trials. As shown in Table 3.4, TOP-RWBC improves over BC on all six tasks.

3.5 Limitations

TOPReward inherits the visual perception limitations of the underlying VLM: tasks requiring fine-grained spatial reasoning, precise alignment, or small-object manipulation may receive noisy progress estimates when the model cannot visually distinguish intermediate states. The

raw prefix score in Equation (3.1) is causal, and the per-episode min-max normalization in Equation (3.2) is non-causal and is used for within-trajectory evaluation and visualization. The chat-template ablation in Appendix B.5.2 further shows that formatting can affect progress estimates for the evaluated backbones. The answer token must be defined consistently with the completion query; under our prompt, `True` is selected because it provides the clearest final-step success–failure separation among the tokens examined. Deployments should therefore validate the complete query and chat-template implementation for the chosen VLM backbone.

3.6 Conclusion

We presented **TOPReward**, a zero-shot progress reward method that repurposes token probabilities of pretrained video VLMs as instruction-conditioned progress signals for robotic manipulation. By querying the model’s internal belief about instruction completion rather than requiring it to generate calibrated numerical outputs, **TOPReward** sidesteps the well-known limitations of VLMs in numerical reasoning and instruction following. Across Open X-Embodiment and our newly introduced **ManiRewardBench** benchmark, **TOPReward** substantially outperforms GVL on open-source models. It also remains competitive with robotics reward models trained on large-scale data across progress-estimation metrics, while requiring no reward-model training. The instruction-disambiguation matrix and wrong-instruction raw-reward drop further show that rewards depend on the language instruction rather than prefix position alone. Finally, **TOPReward** supports success detection without reward-model training, and **TOPReward** can be used for reward-weighted behavior cloning, yielding consistent improvements over standard BC across six real-world SO-100 manipulation tasks.

Chapter 4

Learned Parallel Decoding for Masked Diffusion Language Models¹

Chapter 3 used token probabilities as observations from which to construct a reward. This chapter turns to the generative process of language, the hallmark of human intelligence. In masked diffusion language models, choosing when to reveal each token maximizes how much computation can occur in parallel without violating the conditional independence assumption.

4.1 Introduction

The success and parallel generation capabilities of diffusion models make them attractive for language generation. There has been an ongoing effort to transfer the diffusion paradigm to the natural language domain. Earlier attempts tried to sample in the token embedding space [Li et al., 2022a, Gulrajani and Hashimoto, 2023, Gong et al., 2023] or the latent space of a language autoencoder [Lovelace et al., 2023] using continuous diffusion. In a parallel line of work, discrete diffusion in the vocabulary space is studied in [Austin et al., 2021, Campbell et al., 2022, Meng et al., 2022, He et al., 2022b, Wang and Cho, 2019]. Without

¹This chapter is based on Shirui Chen, Jiantao Jiao, Lillian J. Ratliff, and Banghua Zhu, “dUltra: Ultra-Fast Diffusion Language Models via Reinforcement Learning” (2025) [Chen et al., 2025b].

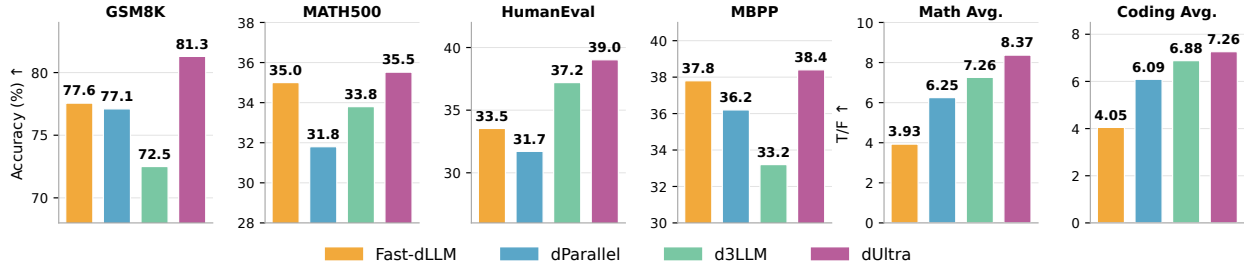


Figure 4.1: **dUltra accuracy-efficiency summary.** Across math reasoning and code generation benchmarks, dUltra consistently achieves strong task accuracy while decoding more tokens per model forward pass than Fast-dLLM, dParallel, and d3LLM. We report the main setting with generation length 256 and block size $B = 32$; complete $B = 32$ and $B = 128$ results are provided in Tables 4.1 and C.2. T/F denotes generated tokens per model forward pass, so higher values indicate more parallel decoding.

the need for an invertible token-wise embedding function, discrete diffusion language models generally perform better than continuous diffusion language models when operating in a discrete state space with a countable number of states [Lou et al., 2023, Shi et al., 2024, Sahoo et al., 2024]. Moreover, it is observed that the best performance is achieved with an absorbing source distribution, which further restricts possible sampling paths [Lou et al., 2023, Austin et al., 2021]. The diffusion models with an absorbing source distribution are also called masked diffusion language models (MDLMs) because of the introduction of a masked token. Moreover, Ou et al. [2025] and Sahoo et al. [2024] have shown that the optimal solution that minimizes negative ELBO is independent of the time variable, thus promoting time-agnostic parameterization. Recent works have scaled time-agnostic MDLMs to show that their performance can rival that of autoregressive (AR) models with similar parameter size [Nie et al., 2025, Ye et al., 2025].

However, even the best open-source masked diffusion language models (MDLMs) [Nie et al., 2025, Ye et al., 2025] face the same slow sampling problem as continuous diffusion models. Apart from system-level optimization, such as approximate KV Cache [Wu et al., 2025, Hu et al., 2025b], attempts to accelerate MDLMs have focused on designing confidence/entropy-

aware sampling methods to unmask tokens in parallel [Ben-Hamu et al., 2025, Wu et al., 2025]. These deterministic parallel decoding strategies generally favor tokens with high confidence or low entropy within a single unmasking step, leading to 3-5x faster sampling speed. Despite this decent speed-up, the number of tokens decoded in parallel within a single denoising step is still low (~ 4.5 tokens per step on GSM8K and ~ 3.3 tokens per step on MATH500) [Wu et al., 2025]. Meanwhile, commercial diffusion language models [Google DeepMind, 2025, Khanna et al., 2025] operate at a throughput of ~ 1000 tokens/s, significantly surpassing the state-of-the-art throughput of autoregressive models, which typically range from 100-300 tokens/s [Khanna et al., 2025]. Therefore, current open-source MDLMs have yet to exploit the parallel generation potential of diffusion language models to their fullest and have not achieved "diffusion supremacy" (analogous to quantum supremacy) over autoregressive models.

A fundamental observation is that when tokens are generated in parallel, MDLMs assume conditional independence between them given the masked input sequence (see also Section 4.2.1) due to the curse of dimensionality [Aaron, 2024]. If the tokens are actually dependent and ambiguous², the actual distribution sampled from will have unwanted modes, and the model may generate nonsensical text. Since acceleration methods aim to increase parallelism by decoding more tokens per step, they inevitably unmask some dependent tokens together, violating the independence assumption. Therefore, given the current paradigm of parallel decoding³, any method that accelerates sampling is implementing some form of mode filtering, i.e. the accelerated sampling procedure will try to sample from a single mode that produces sensible text out of all modes that a slower model can sample from. Appendix C.3 illustrates this behavior.

Recent work [Qian et al., 2025, Chen et al., 2025d] on accelerating diffusion LLM sampling with offline distillation demonstrates this principle as well. Specifically, d3LLM [Qian et al., 2025] distills from the sampling trajectory produced by the teacher model, effectively teaching

²An ambiguous token means that multiple words in the vocabulary are predicted to have similar and high probability.

³Unless the curse of dimensionality is solved

the student model to sample from one of the high probability teacher modes at each sampling step. Similarly, dParallel [Chen et al., 2025d] proposes certainty-forcing distillation that fine-tunes the model to be more confident on self-generated trajectories, which are fixed per prompt. This effectively encourages the model to sample from one of the high probability teacher modes marginalized by the prompts. However, this self-distillation approach faces two shortcomings. First of all, the quality of training data cannot exceed the capability of the base model. In other words, the training data only includes greedily sampled trajectories, limiting exploration and performance of the distilled model. Secondly, the self-generated trajectory can be off-policy during distillation, and the student learns in contexts more frequently encountered by the teacher model rather than by itself. Therefore, we want a method that explores the teacher model modes and trains on samples generated by the student model. On-policy reinforcement learning naturally addresses both limitations: by training on trajectories generated by the current policy, it avoids the distribution mismatch inherent in offline distillation, while the exploration encouraged by RL allows discovering better unmasking strategies beyond those demonstrated by the base model.

Another implication of the observation above is that we can accelerate sampling while reducing the need for mode filtering by planning trajectories with more conditionally independent tokens per step. A one-step generation case study in Appendix C.4 shows that LLaDA-Instruct decodes randomly masked tokens much more reliably than the same number of consecutive suffix masks, illustrating the value of conditional independence-aware planning. Our approach learns adaptive unmasking strategies throughout the full denoising trajectory to maximize the number of conditionally independent tokens decoded at each step.

Taking these motivations all together, in this work, we propose dUltra, a framework that *learns* adaptive unmasking strategies through on-policy RL combined with on-policy distillation reward. The headline accuracy-efficiency results are summarized in Figure 4.1, and an overview of the dUltra architecture is shown in Figure 4.2. Our key contributions are:

- We introduce an unmasking planner head that predicts per-token unmasking probabili-

ties using independent Bernoulli distributions, enabling learnable sampling trajectory planning.

- We develop a GRPO-based on-policy distillation framework that jointly optimizes the diffusion model and unmasking order planner using both verifiable reward and distillation reward. We also propose advantage clipping to enable effective RL training of the unmasking planner head.
- We show state-of-the-art performance in terms of both tokens per model forward pass (T/F) and number of function evaluations (NFE) on mathematical reasoning (GSM8K, MATH500) and code generation (HumanEval, MBPP) tasks.

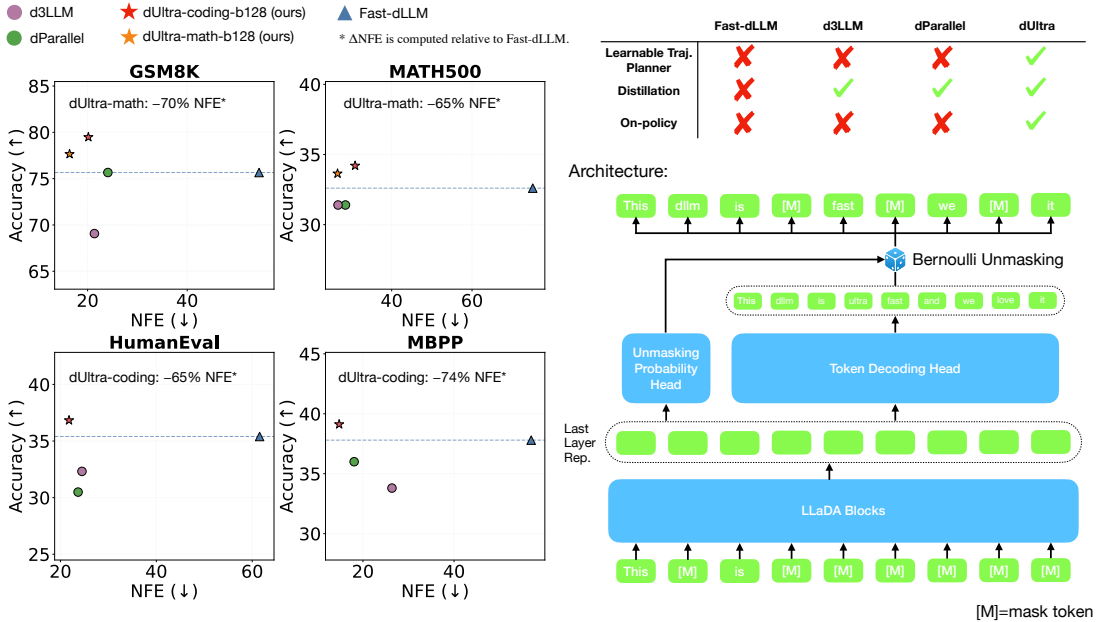


Figure 4.2: **dUltra overview.** Left: Accuracy versus number of function evaluations (NFE; denoising steps) on math and coding benchmarks for Fast-dLLM, dParallel, d3LLM, and dUltra (ours) variants. Here we use a block size of 128 and a generation length of 256. Right: dUltra architecture: a masked diffusion LM (LLaDA blocks) produces last-layer representations that feed (i) a token decoding head to output logits for masked positions and (ii) an unmasking planner head to output per-position unmasking probabilities; the Bernoulli sampled unmasking decisions update the masked sequence ([M] denotes the mask token).

4.2 Preliminaries

Continuous-time MDLMs [Shi et al., 2024, Sahoo et al., 2024] define a forward noising process $q_{t|s}$ that progressively masks tokens according to a noise schedule α_t (with $\alpha_0 = 1$, $\alpha_1 = 0$), and a learned reverse process $p_\theta(x_s|x_t)$ that predicts clean tokens from masked inputs. We use $\mathcal{X} = \{1, \dots, m\}$ as the token vocabulary with the mask token as the m -th entry, and $\text{Cat}(\boldsymbol{\pi})$ to denote a categorical distribution over the simplex Δ^m . Full single-token forward/reverse process formulations are given in Appendix C.13.

4.2.1 Sequence Factorization

For a token sequence $\mathbf{x} \in \mathcal{X}^L$ of length L , MDLM factorizes the forward and reverse processes to sample each token independently. For the forward process, $q_{t|0}(\mathbf{x}_t|\mathbf{x}_0) = \prod_{l=1}^L q_{t|0}(x_t^{(l)}|x_0^{(l)})$. For the reverse process, $q_{s|t,0}(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \prod_{l=1}^L q_{s|t,0}(x_s^{(l)}|x_t^{(l)}, x_0^{(l)} \approx \boldsymbol{\mu}_\theta^{(l)}(\mathbf{x}_t, t))$. Note that here $\boldsymbol{\mu}_\theta$ is dependent on the entire sequence, so this factorization assumes conditional independence.

4.2.2 Training Objective

For MDLM with carry-over unmasking [Sahoo et al., 2024, Shi et al., 2024, Nie et al., 2025, Ye et al., 2025], the NELBO reduces to a weighted cross-entropy over masked positions:

$$\mathcal{L}_{\text{NELBO}} = - \int_0^1 \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{\alpha'_t}{1 - \alpha_t} \sum_{i=0}^{\|\mathbf{x}\|} \mathbf{1}[\mathbf{x}_t^{(i)} = m] \cdot \log \boldsymbol{\mu}_\theta(\mathbf{x}_t; \mathbf{x}_t^{(i)} = \mathbf{x}_0^{(i)}) \right] dt, \quad (4.1)$$

where $\boldsymbol{\mu}_\theta(\mathbf{x}_t; \mathbf{x}_t^{(i)} = \mathbf{x}_0^{(i)})$ is the model’s predicted probability for the i -th token being its ground-truth value. The full NELBO decomposition is in Appendix C.13.

4.2.3 Connection to Any-Order Autoregressive Modeling

Any-order autoregressive models (AO-ARMs) [Uria et al., 2014, Hoogeboom et al., 2021, Shih et al., 2022] evaluate the likelihood of a sequence $\mathbf{x}$ by marginalizing over all possible generation orders. More specifically, the expected negative log-likelihood minimized by AO-ARMs during training is

$$\mathcal{L}_\theta(\mathbf{x}) = -\mathbb{E}_{\xi \sim U(S_{\|\mathbf{x}\|})} \left[\sum_{k=1}^{\|\mathbf{x}\|} \log p_\theta(\mathbf{x}^{\xi(k)} | \mathbf{x}^{\xi(<k)}) \right], \quad (4.2)$$

where $U(S_{\|\mathbf{x}\|})$ is the uniform distribution over all permutations of the initial mask tokens, and each permutation uniquely specifies a sampling order. Previous work has shown that this loss is equivalent to the NELBO loss of MDLMs with carry-over unmasking (Equation (4.1)) under mild conditions [Ou et al., 2025, Kim et al., 2025]. Therefore, MDLMs with carry-over unmasking can be interpreted as AO-ARMs that sample generation orders from a uniform distribution over all possible permutations.

4.3 Methods

4.3.1 Path Planning with Unmasking Probability

If we strictly follow the predetermined forward noising process and corresponding time reversal (Appendix C.13) during denoising, a masked token has a *fixed* probability of $\frac{1-\alpha_{s(i)}}{1-\alpha_{t(i)}}$ remaining a mask token from time $t(i) = i/T$ to time $s(i) = (i-1)/T$ given the noise schedule function α . However, there is no inherent restriction on what the noise schedule function should be or what variable it should depend on under the framework of diffusion models. While maximizing ELBO requires sampling from the forward kernel $q(x_t|x_0)$ during training, at inference time we have flexibility in designing the reverse transition kernels $q_{s|t}$, allowing us to explore different unmasking strategies beyond the training distribution. One possibility is to make the schedule state-dependent by conditioning $\frac{1-\alpha_{s(i)}}{1-\alpha_{t(i)}}$ on the current masked state

$\mathbf{x}_t$, resulting in per-token schedules $\frac{1-\alpha_{s(i)}^l(\mathbf{x}_t)}{1-\alpha_{t(i)}^l(\mathbf{x}_t)}$ for the l -th token. In this way, the reverse denoising process is essentially a Markovian process with the transition kernel $p_{\theta}(\mathbf{x}_{s(i)}|\mathbf{x}_{t(i)})$.

From the perspective of any-order autoregressive modeling, this is equivalent to learning a better distribution over sampling orders instead of the uniform distribution used in the standard negative log-likelihood. As motivated in Figure C.2, we would like to refine this distribution to concentrate the probability mass over sampling orders that maximize the average number of conditionally independent tokens to unmask per step. Therefore, we want to learn a better unmasking order planner that can decide which tokens to unmask at each denoising step.

We therefore introduce an unmasking order planner to decide which mask tokens to unmask and with what probability. To decide which tokens to unmask at each step, we model per-token unmasking decisions as independent Bernoulli distributions, mirroring the conditional independence assumption in MDLM’s token prediction factorization (Section 4.2.1). Modeling the full joint distribution over all possible position subsets would be intractable due to the exponential number of possibilities (2^n subsets for n masked tokens). While we sample unmasking decisions independently for computational feasibility, the planner can still capture dependencies between positions through self-attention when computing the unmasking probabilities. We introduce an unmasking planner head ϕ that takes the last-layer hidden representations of the base MDLM and outputs per-token unmasking probabilities:

$$p_{\text{unmask}}(x) = \sigma(\phi(h_x, x)), \quad z_x \sim \text{Bernoulli}(p_{\text{unmask}}(x)), \quad (3)$$

where h_x is the last-layer hidden representation of the sequence of tokens x , and $z_x \in \{0, 1\}$ denotes the binary decision of whether to unmask token x .

The unmasking planner head ϕ is a lightweight module consisting of a single transformer block (matching the base MDLM architecture), time embedding layers for timestep conditioning, adaptive layer normalization, and a final linear projection to scalar logits. This design keeps the computational overhead minimal while allowing the planner to capture dependencies

between masked positions and incorporate temporal information when deciding which tokens to unmask. The planner processes all masked token representations in parallel and outputs conditionally independent unmasking probabilities for each token. See Appendix C.14.1 for complete architectural details.

4.3.2 Exact Likelihood Calculation

One of the advantages of introducing an explicit unmasking planner head is that we can tractably compute the likelihood of each rollout during training, which is essential for policy gradient methods. Recall from the AO-ARM perspective (Section 4.2.3) that MDLMs define a Markovian generative process: at each step, we unmask a subset of tokens conditioned on all previously unmasked tokens. This Markovian structure makes the likelihood calculation straightforward—we simply multiply the transition probabilities at each step. Next we compute the transition probability at each denoising step.

We denote the prompt as $\mathbf{q}$ and the sequence completion after denoising step k as $\mathbf{o}^k$, and $\mathbf{o}^0$ is the completely masked sequence. At step k , we select positions $\mathbf{Pos}^{(k)} \subseteq \mathcal{X}_{\text{mask}}^{(k)}$ to unmask from the set of currently masked tokens $\mathcal{X}_{\text{mask}}^{(k)}$. Our model consists of two key components: (1) π_{token} , the token probability distribution for masked positions, which we obtain from the logits of the model output, and (2) π_{select} , the unmasking order planner that unmasks each token with probability p_{unmask} at each step. With these components, we model the parallel decoding procedure as an ancestral sampling process. We first sample the positions to unmask from π_{select} , and then sample the tokens for the selected positions from π_{token} . Therefore, the likelihood of generating $\mathbf{o}^k$ from $\mathbf{o}^{k-1}$ given prompt $\mathbf{q}$ is:

$$p(\mathbf{o}^k \mid \mathbf{q}, \mathbf{o}^{k-1}) = \pi_{\text{select}}(\mathbf{Pos}^{(k)} \mid \mathbf{q}, \mathbf{o}^{k-1}) \cdot \pi_{\text{token}}(\mathbf{o}^k \mid \mathbf{q}, \mathbf{o}^{k-1}, \mathbf{Pos}^{(k)}) \quad (4.3)$$

The token selection probability factorizes as:

$$\pi_{\text{token}}(\mathbf{o}^k \mid \mathbf{q}, \mathbf{o}^{k-1}, \mathbf{Pos}^{(k)}) = \prod_{j=1}^{\#\text{unmask}} \pi_{\theta}(\mathbf{o}_{\mathbf{Pos}_j^{(k)}}^k \mid \mathbf{q}, \mathbf{o}^{k-1}, \mathbf{Pos}^{(k)})$$

The position selection probability follows the product of independent Bernoulli distributions:

$$\pi_{\text{select}}(\mathbf{Pos}^{(k)} \mid \mathbf{q}, \mathbf{o}^{k-1}) = \prod_{x \in \mathbf{Pos}^{(k)}} p_{\text{unmask}}(x) \cdot \prod_{x \in \mathcal{X}_{\text{mask}}^{(k)} \setminus \mathbf{Pos}^{(k)}} (1 - p_{\text{unmask}}(x)) \quad (4.4)$$

In practice, one can additionally incorporate a block size hyperparameter B to restrict the set of candidate positions: instead of selecting from all masked tokens $\mathcal{X}_{\text{mask}}^{(k)}$, we constrain $\mathbf{Pos}^{(k)}$ to be a subset of the leftmost B masked tokens. The block size interpolates between completely learnable unmasking order and autoregressive-like order: larger B allows the planner to select from positions throughout the sequence, while smaller B restricts candidates to the leftmost masked tokens, enforcing a more left-to-right progression similar to autoregressive decoding, though the planner retains flexibility in ordering tokens within each block.

Let $\zeta = (\mathbf{o}^1, \dots, \mathbf{o}^K)$ denote the complete denoising path. Its likelihood is

$$p_{\text{student}}(\zeta \mid \mathbf{q}) = \prod_{k=1}^K p(\mathbf{o}^k \mid \mathbf{q}, \mathbf{o}^{k-1}) = \prod_{k=1}^K \pi_{\text{select}}(\mathbf{Pos}^{(k)} \mid \mathbf{q}, \mathbf{o}^{k-1}) \cdot \pi_{\text{token}}(\mathbf{o}^k \mid \mathbf{q}, \mathbf{o}^{k-1}, \mathbf{Pos}^{(k)}). \quad (4.5)$$

This is a path probability. The marginal probability of the final completion would require summing (4.5) over all unmasking paths that produce it. Equivalently, we can view each token’s unmasking trajectory as following a non-homogeneous geometric distribution⁴: at each denoising step, the planner assigns a state-dependent probability of unmasking, and the token remains masked until it is eventually sampled to be unmasked.

⁴Unlike the standard geometric distribution with constant success probability, the non-homogeneous variant allows the probability to vary across steps.

4.3.3 Initializing On-policy Planners with Heuristic Methods

Randomly initialized planners often unmask too many tokens at the start of training, producing low-quality rollouts before RL can explore useful trajectories. We therefore initialize the planner by supervised imitation of the confidence-based Fast-dLLM decoder [Wu et al., 2025], treating its selected positions as binary targets. This gives GRPO a strong heuristic policy as a starting point, after which on-policy optimization can improve beyond fixed confidence-based sampling. The full initialization objective is given in Appendix C.6.

4.3.4 On-policy Distillation Reward

As motivated by Figure C.1, to further accelerate sampling with parallel decoding beyond sampling conditionally independent tokens, the model must learn to sample from a single high-quality mode among multiple plausible modes to avoid generating incoherent text, i.e. the model should exhibit mode-seeking behavior. To encourage this behavior, we introduce an on-policy distillation reward that encourages the student diffusion LLM to maximize the sequence-level log-likelihood of on-policy samples evaluated by an autoregressive teacher model.

For high-quality distillation rewards, we use domain-specific teacher models: Qwen2.5-Math-7B-Instruct [Yang et al., 2024] for mathematical reasoning and Qwen2.5-Coder-7B-Instruct [Hui et al., 2024] for code generation. Given a prompt $\mathbf{q}$ and a completion $\mathbf{c}$ generated by the student model during rollout, we compute the on-policy distillation reward as follows:

$$r_{\text{distill}}(\mathbf{q}, \mathbf{c}) = \frac{1}{|\mathbf{c}|} \left(\left[\sum_{i=1}^{|\mathbf{c}|} \log p_{\text{teacher}}(c_i \mid \mathbf{q}, \mathbf{c}_{<i}) \right] - \beta \log p_{\text{student}}(\mathbf{c} \mid \mathbf{q}) \right) \quad (4.6)$$

where $|\mathbf{c}|$ is the length of the completion in tokens, $p_{\text{student}}(\mathbf{c} \mid \mathbf{q})$ is given by Equation (4.5), and β controls the weight of the entropy regularization term. When $\beta = 1$, maximizing $\mathbb{E}_{\mathbf{c} \sim p_{\text{student}}} [r_{\text{distill}}]$ corresponds to minimizing a scaled reverse KL divergence $\frac{1}{|\mathbf{c}|} D_{\text{KL}}(p_{\text{student}} \parallel p_{\text{teacher}})$. Reverse KL divergence is known to encourage mode-seeking be-

havior [Minka, 2005]: when the student places probability mass where the teacher model has low probability, the term $\log(p_{\text{student}}/p_{\text{teacher}})$ becomes large and heavily penalizes the divergence. We set $\beta = 0$ in our experiments, simplifying the reward to average teacher log-likelihood, while still encouraging mode-seeking behavior: the student learns to generate completions that the teacher assigns high probability, focusing on high-quality modes verified by the teacher model rather than exploring low-probability regions that may lead to incoherent text. Empirically, we did not find significant differences in performance when varying β .

The on-policy distillation reward is combined with verifiable rewards and efficiency rewards (i.e., negative number of denoising steps) via a weighted sum:

$$r_{\text{total}} = \lambda_{\text{task}} \cdot r_{\text{task}} + \lambda_{\text{distill}} \cdot r_{\text{distill}} + \lambda_{\text{step}} \cdot r_{\text{step}} \quad (4.7)$$

where the weights $\lambda_{\text{task}}, \lambda_{\text{distill}}, \lambda_{\text{step}}$ control the relative importance of each component. This multi-objective reward encourages the model to generate correct, coherent, and efficiently-sampled outputs. See Appendix C.14.2 for detailed reward configuration for each task.

4.3.5 GRPO Training Framework

We train the diffusion LLM and unmasking planner head using Group Relative Policy Optimization (GRPO), a policy-gradient method that operates on groups of trajectories per prompt and uses group-relative advantages as baselines. We use on-policy updates (`num_iterations=1`) as off-policy updates provide no computational benefit: unlike autoregressive LLMs that evaluate likelihoods in a single forward pass, each likelihood evaluation using Equation (4.3) requires a complete denoising process. For each group, we sample a single prompt $\mathbf{q}$ and generate G trajectories with the denoised sequence at step k of the g -th trajectory denoted as $\mathbf{o}^{g,k}$. The group-relative advantages of these trajectories are calculated as $A^g = r^g - \text{mean}(r^{1:G})$. Note that we do not normalize the advantage to avoid introducing any bias into the policy gradient estimation [Liu et al., 2025b]. Then the GRPO objective

reduces to the REINFORCE-style loss

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{\|\mathbf{o}\| \cdot G} \sum_{g=1}^G A^g \sum_{k=1}^{K^g} \log p_{\theta}(\mathbf{o}^{g,k} \mid \mathbf{q}, \mathbf{o}^{g,k-1}), \quad (4.8)$$

where $p_{\theta}(\mathbf{o}^{g,k} \mid \mathbf{q}, \mathbf{o}^{g,k-1})$ is the rollout likelihood defined in Equation (4.3), and K^g is the total number of denoising steps for the g -th trajectory. In practice, we use the ratio formulation $\frac{p_{\theta}}{\text{sg}(p_{\theta})}$ instead of $\log p_{\theta}$ for numerical stability, where $\text{sg}(\cdot)$ denotes stop-gradient. The two formulations have identical gradients.

Advantage clipping. Vanilla GRPO can collapse the planner into never unmasking any token, because low-advantage conservative trajectories uniformly penalize both bad and correct unmasking decisions. We therefore clip below-threshold advantages:

$$A^g \leftarrow \begin{cases} 0 & \text{if } A^g < C \\ A^g & \text{otherwise} \end{cases} \quad (4.9)$$

where C is the clipping threshold. This ignores below-threshold trajectories and lets the planner learn primarily from successful unmasking strategies; we set $C = 0$ in experiments. Additional intuition and stability evidence are provided in Appendix C.7 and Appendix C.8.

In summary, the complete training algorithm can be found in Appendix C.15.

4.4 Results

We evaluate dUltra on mathematical reasoning and code generation benchmarks to demonstrate its ability to achieve competitive performance with significantly reduced computational cost. All of our experiments use LLaDA-8B-Instruct as the base model. We compare against *state-of-the-art* confidence-based heuristics, such as Fast-dLLM [Wu et al., 2025], and *state-of-the-art* off-policy distillation-based methods, such as d3LLM [Qian et al., 2025] and dParallel [Chen et al., 2025d]. We report both task performance and the average number of function

evaluations (NFE), i.e., the number of denoising steps during generation.

Unlike distillation baselines, which rely on self-generated or heuristic trajectories without an external teacher signal, dUltra uses on-policy rollouts with an AR teacher model (Qwen2.5-Math/Coder-7B-Instruct) as an additional reward signal. This difference in supervision is an acknowledged advantage of dUltra and is explicitly modeled as a reward term rather than a training label; all methods still use the same base model (LLaDA-8B-Instruct) and are evaluated under identical generation settings. A detailed comparison of supervision signals is provided in Appendix C.5.

4.4.1 Training Dynamics

Figure C.3 shows the evolution of reward metrics during dUltra training on GSM8K. Correctness, efficiency, and distillation rewards increase stably throughout training, validating the GRPO framework and multi-objective reward design. Detailed reward curves are included in Appendix C.8.

Effect of Advantage Clipping on Efficiency Reward. The role of advantage clipping in training stability is illustrated in Appendix C.8. Without clipping ($C = -\infty$), the planner collapses to a degenerate policy that never unmask tokens; with $C = 0.0$, training remains stable and learns a balanced unmasking strategy.

4.4.2 Main Results

Results with Small Block Size

We first evaluate dUltra with block size $B = 32$, which provides a balance between sampling efficiency and unmasking selectivity. All tasks are evaluated in a zero-shot setting. Although the planner is trained as a Bernoulli policy, all reported inference results in Table 4.1 and Table C.2 use deterministic threshold decoding: tokens are unmasked only when their planner probability exceeds the selected threshold. This removes sampling randomness and lets

us control the accuracy-efficiency trade-off through the threshold. Table 4.1 presents our results on mathematical reasoning (GSM8K, MATH500) and code generation (HumanEval, MBPP) tasks, comparing against multiple accelerated MDLMs based on distillation, including dParallel and d3LLM. We report both task performance and the average number of function evaluations (NFE), i.e., the number of denoising steps during generation.

Method	Math				Coding			
	GSM8K		MATH500		HumanEval		MBPP	
	Acc $\uparrow$	NFE $\downarrow$ (T/F $\uparrow$)	Acc $\uparrow$	NFE $\downarrow$ (T/F $\uparrow$)	Acc $\uparrow$	NFE $\downarrow$ (T/F $\uparrow$)	Acc $\uparrow$	NFE $\downarrow$ (T/F $\uparrow$)
Fast-dLLM	<u>77.56</u>	57.26 (4.33)	<u>35.00</u>	70.56 (3.52)	33.54	62.65 (3.94)	<u>37.80</u>	50.04 (4.16)
dParallel	77.10	28.47 (7.17)	31.80	<u>35.21</u> (5.32)	31.71	<u>30.68</u> (5.87)	36.20	<u>22.77</u> (6.30)
d3LLM	72.50	<u>25.79</u> (8.39)	33.80	35.57 (<u>6.13</u>)	<u>37.20</u>	33.41 (<u>6.86</u>)	33.20	32.26 (<u>6.90</u>)
dUltra (ours)	81.29	21.21 (9.72)	35.52	33.20 (7.03)	39.02	24.25 (7.23)	38.40	20.03 (7.29)

Table 4.1: Main results with generation length 256 and block size $B = 32$. dUltra rows use deterministic threshold decoding at inference time. We report accuracy, NFE, and T/F (tokens per model forward pass); **bold** and underlined values mark the best and second-best result in each column.

dUltra achieves strong performance across all benchmarks while maintaining low NFE. Notably, dUltra outperforms both distillation baselines (d3LLM and dParallel) on GSM8K and MATH500 in accuracy, while using fewer denoising steps on GSM8K and comparable denoising steps on MATH500, consistent with our mode filtering hypothesis: on-policy RL can discover better modes than methods limited to off-policy teacher trajectories. We further ablate the learned unmasking order planner by replacing it with confidence-based sampling on the same trained checkpoints (w/o planner); the planner-based dUltra improves accuracy by +5.9 pp on GSM8K and +12–17 pp on code generation tasks (see Appendix C.11.1). This demonstrates the value of jointly training the base model and unmasking order planner for task-specific unmasking strategies.

Results with Large Block Size

To explore the impact of larger block sizes on sampling efficiency, we evaluate dUltra with $B = 128$ in Appendix C.9. Larger block sizes give the planner more freedom for longer-horizon planning by allowing it to select from a broader set of candidate positions at each denoising step, potentially enabling faster inference through increased parallelism. Appendix C.11 provides the corresponding speed-quality frontiers.

Wall-Clock Throughput

To verify that NFE reductions translate to real latency gains, we measure wall-clock throughput on a single H200 GPU for 256-token generation with $B = 128$ and no KV-cache. dUltra achieves $3.0\text{--}3.4\times$ speedup over the LLaDA baseline and $\sim 1.5\times$ speedup over d3LLM while improving accuracy on both GSM8K and MATH500; the full throughput table is in Appendix C.10.

4.4.3 Speed-Quality Trade-off

The deterministic threshold rule used for Table 4.1 and Table C.2 also provides an inference-time speed-quality knob: increasing the threshold makes decoding more conservative, while decreasing it unmask more tokens per step. We use Fast-dLLM’s factor-based parallel decoding strategy [Wu et al., 2025] for models without a planner. Sweeping thresholds for dUltra and factors for baselines yields Pareto frontiers where dUltra consistently achieves superior accuracy-efficiency trade-offs; full frontier plots are provided in Appendix C.11.

4.4.4 Analysis of Learned Unmasking Strategies

To understand what unmasking order patterns dUltra learns, Figure 4.3 visualizes the unmasking behavior across denoising steps on a code-generation example; additional math and coding visualizations are provided in Appendix C.12. The learned policies are largely

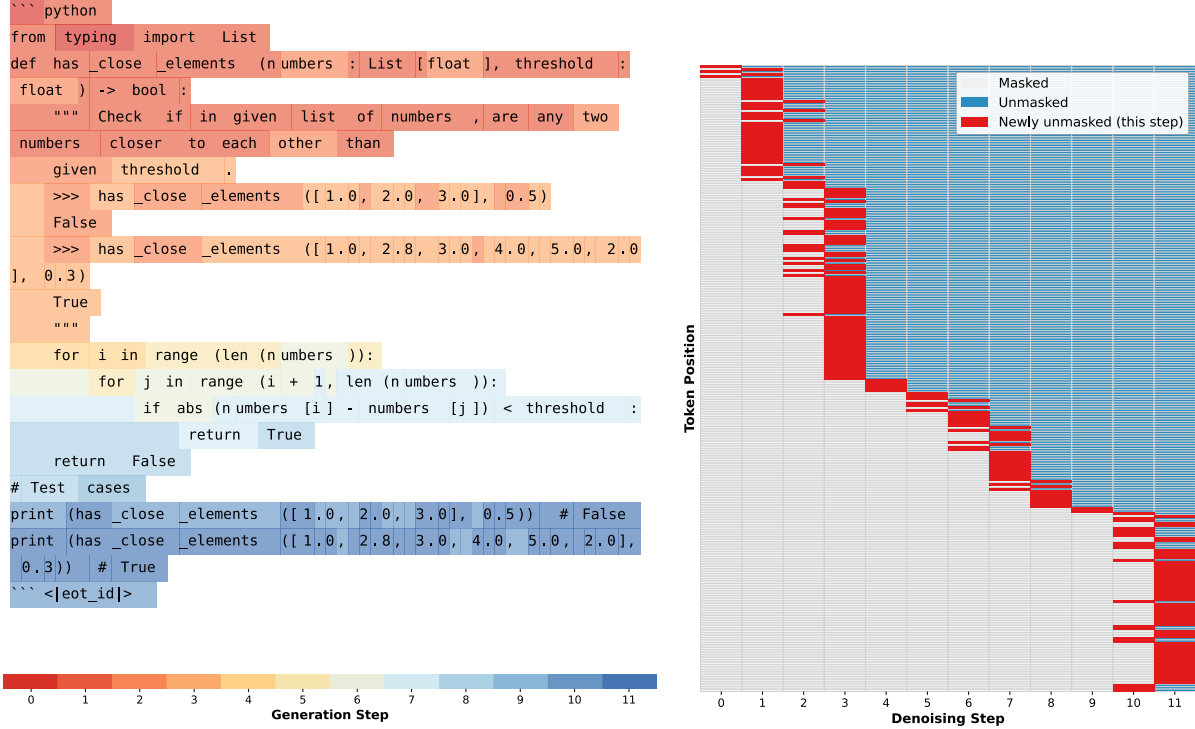


Figure 4.3: **Learned unmasking order on HumanEval.** Red bars indicate tokens newly unmasked at each denoising step. The planner mostly follows a left-to-right trend, but it also jumps to predictable code structure such as function signatures, control-flow keywords, and return statements, creating opportunities for parallel decoding while respecting stronger local dependencies.

autoregressive, but code generation exhibits more non-autoregressive jumps to structural tokens such as function signatures, control-flow keywords, and return statements. These patterns support the main motivation: the planner selectively deviates from left-to-right order when task structure creates opportunities for parallel decoding while preserving quality. Extended related work is provided in Appendix C.2.

4.5 Conclusion

We present dUltra, a framework for learning adaptive unmasking strategies that accelerate masked diffusion language models through on-policy RL and distillation rewards. dUltra outperforms distillation baselines in both accuracy and NFE, and its learned policies adapt to

task structure by exploiting more parallelism in code while preserving more sequential decoding in math. These results show that learned mode filtering and conditional-independence-aware planning can move MDLMs closer to practical parallel generation.

Part II

Dynamics and Geometry of Representation Learning

Chapter 5

Loss Sharpness and Compressed Neural Representations¹

The first part of the dissertation examined how neural systems compute with probability through recurrent sampling, token-level rewards, and a discrete diffusion process. This chapter begins the second part by shifting attention from probabilistic outputs and transitions to the geometry of the representations on which those computations depend. It asks how learning in parameter space affects the local sensitivity and compression of representations across a network.

5.1 Introduction

There has been long-standing interest in sharpness, a geometric metric in the *parameter* space that measures the flatness of the loss landscape at local minima. Flat minima refer to regions in the loss landscape where the loss function has a relatively large basin, and the loss does not change much in different directions around the minimum. Empirical studies and theoretical analyses have shown that training deep neural networks using stochastic gradient

¹This chapter is based on Shirui Chen, Stefano Recanatesi, and Eric Shea-Brown, “A Simple Connection from Loss Flatness to Compressed Neural Representations,” published in *Transactions on Machine Learning Research* (2026) [Chen et al., 2026b].

descent (SGD) with a small batch size and a high learning rate often converges to flat and wide minima [Ma and Ying, 2021, Blanc et al., 2020, Geiger et al., 2021, Li et al., 2022b, Wu et al., 2018, Jastrzebski et al., 2018, Xie et al., 2021, Zhu et al., 2019]. Many works conjecture that flat minima lead to a simpler model (shorter description length), and thus are less likely to overfit and more likely to generalize well [Hochreiter and Schmidhuber, 1997, Keskar et al., 2016, Wu et al., 2018, Yang et al., 2023]. Based on this rationale, sharpness-aware minimization (SAM) has been a popular method for improving a model’s generalization ability. However, recent work has shown that SAM does not *only* minimize sharpness to achieve superior generalization performance [Andriushchenko and Flammarion, 2022, Wen et al., 2023]. More confusingly, it remains unclear whether flatness correlates positively with the generalization capacity of the network [Dinh et al., 2017, Andriushchenko et al., 2023, Yang et al., 2021], and even when it does, the correlation is not perfect [Neyshabur et al., 2017, Jiang et al., 2019]. In particular, Dinh et al. [2017] argues that one can construct very sharp networks that generalize well through reparametrization, while Andriushchenko et al. [2023] show that even reparametrization-invariant sharpness cannot capture the relationship between sharpness and generalization.

Figure 5.1 illustrates the crux of this issue. The common intuition for why flat minima should generalize better [Hochreiter and Schmidhuber, 1997, Keskar et al., 2016] implicitly assumes that the test loss landscape is a horizontally shifted copy of the training loss (Figure 5.1a): under such a shift, the loss at a flat minimum changes little while the loss at a sharp minimum increases substantially, yielding a smaller generalization gap at the flat minimum. However, there is no principled justification for this horizontal-shift assumption. In a hypothetical setting where the test loss differs from the training loss by a random perturbation (Figure 5.1b), the generalization gap at a flat minimum could be *larger* than at a sharp minimum, contradicting the standard narrative. This suggests that sharpness, while clearly capturing meaningful geometric structure, may not be directly measuring generalization ability.

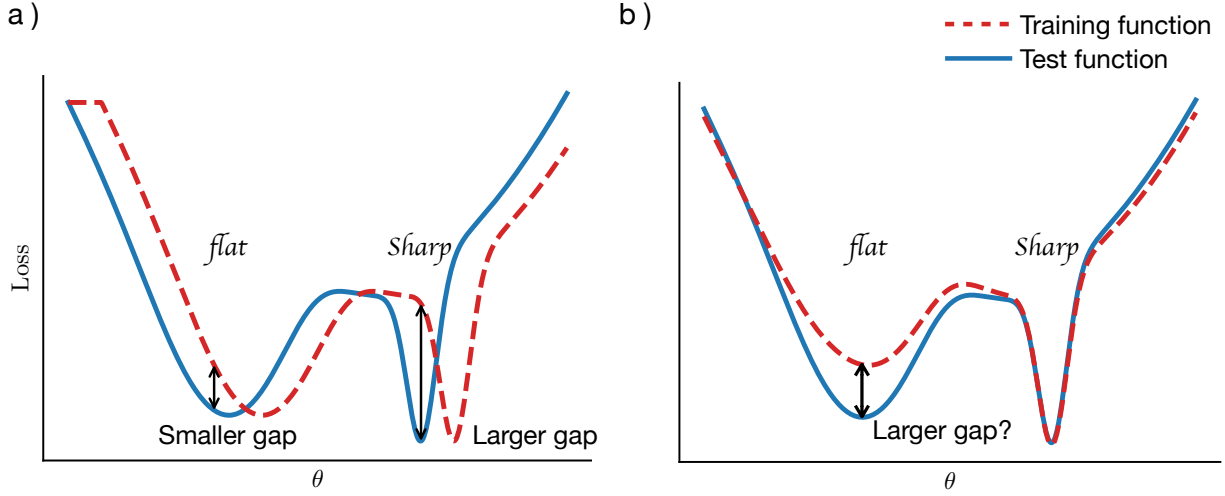


Figure 5.1: A schematic illustration of why flatness alone does not reliably predict generalization. **(a)** Under the common but unjustified assumption that the test loss is a horizontal shift of the training loss, flat minima trivially exhibit a smaller generalization gap than sharp minima. **(b)** In a hypothetical but possible scenario, the test loss differs from the training loss by an additive perturbation rather than a simple shift. Here the flat minimum can exhibit a *larger* generalization gap than the sharp minimum, demonstrating that curvature alone is insufficient to predict generalization.

In light of this paradox, we take a different perspective: rather than asking what sharpness tells us about generalization, we ask what sharpness tells us about the *representations* learned by the network. We show that there exists a more consistent relationship by investigating how sharpness near interpolation solutions in the *parameter* space influences local geometric features of neural representations in the *feature* space. By building a relationship between sharpness and the local compression of neural representations, we argue that sharpness, in its essence, measures the compression of neural representations. Specifically, we show that as sharpness decreases and the minimum flattens, sharpness-related quantities upper-bound certain compression measures, meaning that the neural representation must also undergo some degree of compression. We also note how local dimensionality is a compression metric of a distinct nature and therefore does not necessarily correlate with sharpness.

More specifically, our work makes the following novel contributions:

1. **Refined and new compression metrics with sharpness bounds.** We refine and identify feature-space quantities that quantify compression and are bounded by sharpness: local volumetric ratio (LVR) and maximum local sensitivity (MLS). The former is a new metric in this context, and the latter refines a related quantity in earlier work.² We derive explicit upper bounds showing that these metrics are constrained by sharpness-related quantities. This establishes a direct mathematical link between parameter-space geometry (sharpness) and feature-space geometry (compression), applicable to feedforward networks with quadratic loss.
2. **Improved bounds via network-wide metrics.** We improve the bound on MLS from Ma and Ying [2021] and propose Network MLS (NMLS), which considers sensitivity at all intermediate layers rather than only the input. This yields bounds that consistently predict positive correlation with sharpness across diverse experimental settings, overcoming limitations of single-layer analyses where prefactors can dominate and obscure the sharpness-compression relationship.
3. **Extension to reparametrization-invariant sharpness.** We prove that using reparametrization-invariant sharpness (i.e., unchanged under layer rescaling) tightens our bounds, addressing a key criticism of standard sharpness measures and providing a novel interpretation: reparametrization-invariant sharpness quantifies robustness of outputs to internal representations.
4. **Empirical validation across architectures.** We validate our theoretical predictions through experiments on feedforward (MLP), convolutional (VGG-11, LeNet), and transformer (ViT) architectures, demonstrating that LVR and MLS/NMLS consistently correlate with their sharpness-related bounds during training and across pretrained models.

²We collectively term MLS and NMLS “compression metrics” because these quantities measure how compressed or concentrated a set of noise-perturbed input or internal neural representations is after passing through the network.

5. Connection to neural collapse and intermediate representations. We extend our analysis to penultimate and intermediate layer representations, relating our framework to neural collapse phenomena. Our bounds apply most naturally when the number of classes exceeds feature dimension, relevant for settings like language modeling and large-scale retrieval.

On bound tightness. We emphasize that our theoretical results provide *upper bounds* rather than equalities. Moreover, these bounds depend not only on sharpness but also on weight norms and input magnitudes. Consequently, for two networks with identical sharpness but different weight scales, the bounds may differ. However, our extensive empirical validation demonstrates that despite this apparent looseness, the bounds are sufficient to predict consistent positive correlation between sharpness and compression metrics across training dynamics and diverse architectures. This empirical validation is essential to our claims and demonstrates the practical utility of our theoretical framework.

With these results, we help reveal the nature of sharpness through the interplay between key properties of trained neural networks in parameter space and feature space.

Chapter organization. Section 5.2 reviews the pioneering arguments of Ma and Ying [2021] that flatter minima can constrain the gradient of network output with respect to network input, and extends the formulation to the multidimensional input case. Section 5.3 proves that lower sharpness implies a lower upper bound on two metrics of the compression of the representation manifold in feature space: the local volumetric ratio and the maximum local sensitivity (MLS) (Section 5.3.1, Section 5.3.2). Section 5.4 empirically verifies our theory by calculating various compression metrics, their theoretical bounds, and sharpness for models during training as well as pretrained ones. Finally, Section 5.5 discusses how these conditions help explain why there are mixed results on the relationship between sharpness and generalization in the literature, by looking through the alternative lens of compressed neural representations.

Key terminology. Before proceeding, we clarify key terms used throughout this work. *Reparametrization* refers to equivalent network parameterizations that produce identical input-output mappings; for example, scaling the weights of layer l by factor α and layer $l + 1$ by $1/\alpha$ leaves the network function with ReLU nonlinearity unchanged as demonstrated in Dinh et al. [2017]. Reparametrization-invariant sharpness measures remain constant under such transformations, addressing the concern that standard sharpness can be arbitrarily changed without affecting network behavior.

The compression metrics we study are: *Local Volumetric Ratio (LVR)*, which quantifies how much a small input volume contracts as it propagates through the network; *Maximum Local Sensitivity (MLS)*, which measures the maximum rate of change of network output with respect to input perturbations; and *Network MLS (NMLS)*, which extends MLS by considering sensitivity at all intermediate layer representations rather than only the input layer.

5.2 Background and setup

Consider a feedforward neural network f with input data $\mathbf{x} \in \mathbb{R}^M$ and parameters $\boldsymbol{\theta}$. The output of the network is:

$$\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta}) , \quad (5.1)$$

where $\mathbf{y} \in \mathbb{R}^N$ ($N < M$). We consider a quadratic loss $L(\mathbf{y}, \mathbf{y}_{\text{true}}) = \frac{1}{2} \|\mathbf{y} - \mathbf{y}_{\text{true}}\|^2$, a function of the outputs and ground truth $\mathbf{y}_{\text{true}}$. In the following, we will simply write $L(\mathbf{y})$, $L(f(\mathbf{x}, \boldsymbol{\theta}))$ or simply $L(\boldsymbol{\theta})$ to highlight the dependence of the loss on the output, the network, or its parameters.

Sharpness measures how much the loss gradient changes when the network parameters are perturbed, and is defined by the sum of the eigenvalues of the Hessian:

$$S(\boldsymbol{\theta}) = \text{Tr}(H) , \quad (5.2)$$

with $H = \nabla^2 L(\theta)$ being the Hessian. The trace of the Hessian, $\text{Tr}(\nabla^2 L(\theta))$, is not the only definition of sharpness, but many sharpness minimization methods have been theoretically shown to reduce this quantity in interpolating models. Specifically, assuming that the training loss minimizers lie on a smooth manifold [Cooper, 2018, Fehrman et al., 2020], methods like Sharpness-Aware Minimization (SAM) [Foret et al., 2020] when used with batch size 1 and sufficiently small learning rate and perturbation radius [Wen et al., 2022, Bartlett et al., 2023], or Label Noise SGD with a small enough learning rate [Blanc et al., 2020, Damian et al., 2021], tend to favor interpolating solutions with a low Hessian trace. Therefore, we focus our analysis on the trace of the Hessian.

Scope and loss function considerations. Our theoretical analysis is conducted using the quadratic (MSE) loss, which allows us to derive the key relationship in Equation (5.3) connecting sharpness to network gradients. However, we note important considerations regarding other loss functions. For cross-entropy (CE) loss, Granziol [2020] showed that Hessian-based flatness measures become unreliable: solutions with large weights and low CE loss can exhibit artificially small Hessian traces, appearing deceptively "flat" despite poor generalization properties. Moreover, the Hessian rank of CE loss is bounded by the number of neurons times the number of classes, causing many eigenvalues to be zero or near-zero regardless of the solution quality. Consequently, networks with L2 regularization (which have smaller weights) can be sharper yet generalize better than unregularized networks, directly contradicting the flatness-generalization hypothesis for CE loss. This fundamentally breaks the connection between CE loss sharpness and generalization.

Despite this limitation of CE loss sharpness, our framework can still provide insights for networks trained with CE loss through two pathways. First, our theoretical results extend to logistic loss with label smoothing because using the logistic loss with label smoothing yields the same set of minimizers and flattest minimizers as a corresponding problem using mean squared error (see Lemma A.13 in Wen et al. [2023]). Second, and more practically

relevant, our compression metrics (LVR, MLS, NMLS) depend on the network function f itself rather than the training loss. Once a network is trained (with any loss function), we can analyze its compression properties by examining how it transforms inputs to outputs. The empirical success of this approach is demonstrated in Figure 5.5, where we observe meaningful correlations between compression and sharpness (computed assuming MSE loss) for 181 pretrained ViT models that were trained with CE loss. For these networks, we compute sharpness using MSE loss applied to the trained network, which allows us to differentiate between solutions even though their original CE training loss was near zero.

Following [Ma and Ying, 2021, Ratzon et al., 2023], we define $\boldsymbol{\theta}^*$ to be an “exact interpolation solution” on the zero training loss manifold in the parameter space (the zero loss manifold in what follows), where $f(\mathbf{x}_i, \boldsymbol{\theta}^*) = \mathbf{y}_i$ for all i ’s (with $i \in \{1..n\}$ indexing the training set) and $L(\boldsymbol{\theta}^*) = 0$. On the zero loss manifold, in particular, we have

$$S(\boldsymbol{\theta}^*) = \frac{1}{n} \sum_{i=1}^n \|\nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F^2, \quad (5.3)$$

where $\|\cdot\|_F$ is the Frobenius norm. We state a proof of this equality, which appears in Ma and Ying [2021] and Wen et al. [2023], in Appendix D.3. In practice, the parameter $\boldsymbol{\theta}$ will never reach an exact interpolation solution due to the gradient noise of SGD; however, Equation (5.3) is a good approximation of the sharpness when the training loss is sufficiently small (see error bounds in Theorem D.2).

To see why minimizing the sharpness of the solution leads to more compressed representations, we need to move from the parameter space to the input space. To do so we clarify the proof of Equation (4) in Ma and Ying [2021] that relates adversarial robustness to sharpness in the following. The improvements we made are summarized at the end of this section. Let $\mathbf{W}$ be the input weights (the parameters of the first linear layer) of the network, and $\bar{\boldsymbol{\theta}}$ be the rest of the parameters. Following [Ma and Ying, 2021], as the weights $\mathbf{W}$ multiply the

inputs $\mathbf{x}$, we have the following identities:

$$\begin{aligned}\|\nabla_{\mathbf{W}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})\|_F &= \sqrt{\sum_{i,j,k} J_{jk}^2 x_i^2} = \|J\|_F \|\mathbf{x}\|_2 \geq \|J\|_2 \|\mathbf{x}\|_2, \\ \nabla_{\mathbf{x}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}}) &= J\mathbf{W},\end{aligned}\tag{5.4}$$

where $J = \frac{\partial f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})}{\partial(\mathbf{W}\mathbf{x})}$ is a complex expression computed with backpropagation. From Equation (5.4) and the sub-multiplicative property of the Frobenius norm and the matrix 2-norm³, we have:

$$\|\nabla_{\mathbf{x}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})\|_2 \leq \|\nabla_{\mathbf{x}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})\|_F \leq \frac{\|\mathbf{W}\|_2}{\|\mathbf{x}\|_2} \|\nabla_{\mathbf{W}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})\|_F. \tag{5.5}$$

We call Equation (5.5) the linear stability trick. As a result, we have

$$\begin{aligned}\frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_2^k &\leq \frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F^k \\ &\leq \frac{\|\mathbf{W}\|_2^k}{\min_i \|\mathbf{x}_i\|_2^k} \frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{W}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F^k \\ &\leq \frac{\|\mathbf{W}\|_2^k}{\min_i \|\mathbf{x}_i\|_2^k} \frac{1}{n} \sum_{i=1}^n \|\nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F^k.\end{aligned}\tag{5.6}$$

This reveals the impact of flatness on the input sensitivity when $k = 2$. Equation (5.6) holds for any positive k . Thus, the effect of input perturbations is upper-bounded by the sharpness of the loss function (cf. Equation (5.3)). Note that Equation (5.6) corresponds to Equation (4) in Ma and Ying [2021] with multivariable output.

While the experiments of Ma and Ying [2021] *empirically* show a high correlation between the left-hand side of Equation (5.6) and the sharpness, Equation (5.6) does not explain such a correlation by itself because of the scaling factor $\|\mathbf{W}\|_2^k / \min_i \|\mathbf{x}_i\|_2^k$. This factor makes the right-hand side of Equation (5.6) highly variable, leading to mixed positive and/or negative correlations with sharpness under different experimental settings. In the next section, we

³ $\|AB\|_F \leq \|A\|_F \|B\|_2$, $\|AB\|_2 \leq \|A\|_2 \|B\|_2$

will improve this bound to relate sharpness to various metrics measuring robustness and compression of representations. More specifically, compared to Equation (4) of Ma and Ying [2021], we make the following improvements:

1. We replace the reciprocal of the minimum with the quadratic mean to achieve a more stable bound (Proposition 5.8). This term remains relevant as common practice in deep learning does *not* normalize the input by its 2-norm, as this would erase information about the modulus of the input.
2. While Ma and Ying [2021] only considers scalar output, we extend the result to consider networks with multivariable output throughout the chapter.
3. We introduce new metrics such as Network Volumetric Ratio and Network MLS (Definition 5.5 and Definition 5.9) and their sharpness-related bounds (Proposition 5.6 and Proposition 5.10), which have two advantages compared to prior results (cf. the right-hand side of Equation (5.6)):
 - (a) our metrics consider all linear weights so that bounds remain stable to weight changes during training.
 - (b) they avoid the gap between derivative w.r.t. the first layer weights and the derivative w.r.t. all weights, i.e. the second inequality in Eq. 6, thus tightening the bound.

Moreover, we show that the underlying theory readily extends to networks with residual connections in Appendix D.2.

5.3 From robustness to inputs to compression of representations

Building upon the linear stability bound (Equation (5.5)) established in Section 5.2, this section derives our main theoretical results connecting sharpness to compression of neural representations.

Key assumptions. Our analysis assumes: (i) the network has reached an approximate interpolation solution θ^* where training loss is near zero, allowing us to use the sharpness characterization from Equation (5.3); (ii) the network architecture consists of linear layers (including convolutional layers, which are linear operations) with nonlinear activations, though we do not assume specific activation functions for the main bounds (the theory extends to networks with residual/skip connections as detailed in Appendix D.2); and (iii) all bounds are derived using the quadratic (MSE) loss function. While our theoretical derivations require MSE loss, the compression metrics themselves (LVR, MLS, NMLS) depend only on the network function f and can be computed for networks trained with any loss function, as discussed in Section 5.2. Throughout, we work in the local regime where first-order Taylor approximations are valid.

5.3.1 Sharpness bounds local volumetric transformation in the feature space

To understand how networks compress their representations, we first study how local volumes in input space transform as they pass through the network. If a network compresses its inputs, small volumes around input points should contract as they are mapped to feature space. We quantify this via the local volumetric ratio, the ratio between the volume of a hypercube of side length h at $\mathbf{x}$, $H(\mathbf{x})$, and its image under transformation f , $f(H(\mathbf{x}), \theta^*)$. By the change of variables formula from multivariate calculus, the infinitesimal volume transformation is

characterized by the Jacobian matrix. Consider the mapping $f : \mathbb{R}^M \rightarrow \mathbb{R}^N$ with $N < M$ and Jacobian $J = \nabla_{\mathbf{x}} f \in \mathbb{R}^{N \times M}$. The volume scaling factor is:

$$\begin{aligned} d \text{vol}|_{f(\mathbf{x}, \boldsymbol{\theta}^*)} &= \lim_{h \rightarrow 0} \frac{\text{vol}(f(H(\mathbf{x}), \boldsymbol{\theta}^*))}{\text{vol}(H(\mathbf{x}))} \\ &= \sqrt{\det(JJ^T)} \quad (\text{product of singular values of } J) \\ &= \sqrt{\det(\nabla_{\mathbf{x}} f (\nabla_{\mathbf{x}} f)^T)}, \end{aligned} \tag{5.7}$$

where the equality holds because for a rectangular matrix $J \in \mathbb{R}^{N \times M}$ with $N < M$, the N -dimensional volume scaling equals the product of its N singular values $\sigma_1, \dots, \sigma_N$, which equals $\sqrt{\det(JJ^T)}$ since $JJ^T \in \mathbb{R}^{N \times N}$ is a square matrix with eigenvalues $\sigma_1^2, \dots, \sigma_N^2$. We formalize this as a definition:

Definition 5.1. *The **Local Volumetric Ratio at input $\mathbf{x}$** of a network f with parameters $\boldsymbol{\theta}$ is defined as $d \text{vol}|_{f(\mathbf{x}, \boldsymbol{\theta})} = \sqrt{\det(\nabla_{\mathbf{x}} f (\nabla_{\mathbf{x}} f)^T)}$.*

Exploiting the bound on the gradients derived earlier in Equation (5.5), we derive a similar bound for the volumetric ratio:

Lemma 5.2.

$$d \text{vol}|_{f(\mathbf{x}, \boldsymbol{\theta}^*)} \leq \left(\frac{\text{tr} \nabla_{\mathbf{x}} f (\nabla_{\mathbf{x}} f)^T}{N} \right)^{N/2} = N^{-N/2} \|\nabla_{\mathbf{x}} f(\mathbf{x}, \boldsymbol{\theta}^*)\|_F^N, \tag{5.8}$$

Proof. The inequality follows from the arithmetic-geometric mean inequality applied to the squared singular values of $\nabla_{\mathbf{x}} f$. The equality uses the trace identity $\text{tr}(\nabla_{\mathbf{x}} f (\nabla_{\mathbf{x}} f)^T) = \|\nabla_{\mathbf{x}} f\|_F^2$. See Appendix D.4 for the complete derivation. $\square$

Next, we introduce a measure of the volumetric ratio averaged across input samples.

Definition 5.3. *The **Local Volumetric Ratio (LVR)** of a network f with parameters $\boldsymbol{\theta}$ is defined as the sample mean of Local Volumetric Ratio at different input samples:*

$$dV_f(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n d \text{vol}|_{f(\mathbf{x}_i, \boldsymbol{\theta})}.$$

We now establish our first main result: an upper bound on LVR in terms of sharpness. This bound will show that flatter minima (lower sharpness) necessarily constrain the volumetric ratio, establishing a mathematical link between parameter-space flatness and feature-space compression.

Proposition 5.4. *The local volumetric ratio is upper bounded by a sharpness related quantity:*

$$dV_{f(\theta^*)} \leq \frac{N^{-N/2}}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f(\mathbf{x}_i, \theta^*)\|_F^N \leq \frac{1}{n} \sqrt{\sum_{i=1}^n \frac{\|\mathbf{W}\|_2^{2N}}{\|\mathbf{x}_i\|_2^{2N}}} \left(\frac{nS(\theta^*)}{N} \right)^{N/2} \quad (5.9)$$

for all $N \geq 1$.

The proof of the above inequalities is given in Appendix D.4.

The bound in Proposition 5.4 only considers sensitivity at the input layer. However, we can strengthen our analysis by examining compression at all intermediate layers, which allows us to incorporate all network weights rather than just the first-layer weights. This yields tighter bounds that are more stable during training. We denote the input to the l -th linear layer as $\mathbf{x}_i^l$ for $l = 1, 2, \dots, L$, where $\mathbf{x}_i^1 = \mathbf{x}_i$ is the network input. Similarly, $\mathbf{W}_l$ is the weight matrix of the l -th linear/convolutional layer. With a slight abuse of notation, we use f_l to denote the mapping from the input of the l -th layer to the final output.

Definition 5.5. *The **Network Volumetric Ratio (NVR)** is defined as the sum of the local volumetric ratios dV_{f_l} for all f_l , that is, $dV_{net} = \sum_{l=1}^L dV_{f_l}$*

By applying Equation (5.9) to every intermediate layer, we obtain a network-wide bound:

Proposition 5.6. *The network volumetric ratio is upper bounded by a sharpness related quantity:*

$$\sum_{l=1}^L dV_{f_l} \leq \frac{N^{-N/2}}{n} \sum_{l=1}^L \sum_{i=1}^n \|\nabla_{\mathbf{x}^l} f_i^l\|_F^N \leq \frac{1}{n} \sqrt{\sum_{l=1}^L \sum_{i=1}^n \frac{\|\mathbf{W}_l\|_2^{2N}}{\|\mathbf{x}_i^l\|_2^{2N}}} \cdot \left(\frac{nS(\theta^*)}{N} \right)^{N/2}. \quad (5.10)$$

Again, a detailed derivation of the above inequalities is given in Appendix D.4. Proposition 5.4 and Proposition 5.6 imply that flatter minima of the loss function in parameter space contribute to local compression of the data’s representation manifold.

5.3.2 Maximum Local Sensitivity as an allied metric to track neural representation geometry

We observe that the equality condition in the first line of Equation (5.8) rarely holds in practice. To achieve equality, we would need all singular values of the Jacobian matrix $\nabla_{\mathbf{x}}f$ to be identical. However, our experiments in Section 5.4 show that the local dimensionality decreases rapidly with training onset; this implies that $\nabla_{\mathbf{x}}f^T\nabla_{\mathbf{x}}f$ has a non-uniform eigenspectrum (i.e., some directions being particularly elongated, corresponding to a lower overall dimension). Moreover, the volume will decrease rapidly as the smallest eigenvalue vanishes. Thus, although sharpness upper bounds the volumetric ratio and often correlates reasonably with it (see Appendix D.8.2), the correlation is far from perfect.

Fortunately, considering only the maximum eigenvalue instead of the product of all eigenvalues alleviates this discrepancy (recall that $\det(\nabla_{\mathbf{x}}f\nabla_{\mathbf{x}}f^T)$ in Definition 5.1 is the product of all eigenvalues). This motivates an alternative compression metric based on the largest singular value.

Definition 5.7. *The **Maximum Local Sensitivity (MLS)** of network f is defined to be $\text{MLS}_f = \frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}}f(\mathbf{x}_i)\|_2$, which is the sample mean of the largest singular value of $\nabla_{\mathbf{x}}f$.*

Intuitively, MLS is the largest possible average local change of $f(\mathbf{x})$ when the norm of the perturbation to $\mathbf{x}$ is regularized. MLS is also referred to as *adversarial robustness* or *Lipschitz constant* of the model function in Ma and Ying [2021]. We now derive an analogous bound for MLS that mirrors Proposition 5.4 but avoids the product-of-eigenvalues issue.

Proposition 5.8. *The maximum local sensitivity is upper bounded by a sharpness-related*

quantity:

$$\text{MLS}_f = \frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_2 \leq \|\mathbf{W}\|_2 \sqrt{\frac{1}{n} \sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^2}} S(\boldsymbol{\theta}^*)^{1/2}. \quad (5.11)$$

The derivation of the above bound is included in Appendix D.5. As an alternative measure of compressed representations, we empirically show in Appendix D.8.2 that MLS has a higher correlation with sharpness than the local volumetric ratio. We include more analysis of the tightness of this bound in Appendix D.8 and discuss its connection to other works therein.

As with volumetric ratio, considering only the first layer can lead to unstable bounds dominated by prefactor variations. We therefore introduce a network-wide version that examines sensitivity at all intermediate layers, yielding bounds that more consistently predict positive correlation with sharpness across experimental settings.

Definition 5.9. *The **Network Maximum Local Sensitivity (NMLS)** of network f is defined as the sum of MLS_{f_l} for all l , i.e. $\sum_{l=1}^L \text{MLS}_{f_l}$.*

Recall that $\mathbf{x}_i^l$ is the input to the l -th linear/convolutional layer for sample $\mathbf{x}_i$ and f_l is the mapping from the input of l -th layer to the final output. Extending the analysis from Proposition 5.8 to all layers yields:

Proposition 5.10. *The network maximum local sensitivity is upper bounded by a sharpness related quantity:*

$$\text{NMLS} = \frac{1}{n} \sum_{l=1}^L \sum_{i=1}^n \|\nabla_{\mathbf{x}^l} f^l(\mathbf{x}_i^l, \boldsymbol{\theta}^*)\|_2 \leq \sqrt{\frac{1}{n} \sum_{i=1}^n \sum_{l=1}^L \frac{\|\mathbf{W}_l\|_2^2}{\|\mathbf{x}_i^l\|_2^2}} \cdot S(\boldsymbol{\theta}^*)^{1/2}. \quad (5.12)$$

The derivation is in Appendix D.5. The advantage of NMLS is that instead of only considering the robustness of the final output w.r.t. the input, NMLS considers the robustness of the output w.r.t. all hidden-layer representations. This allows us to derive a bound that not only considers the weights in the first linear layer but also all other linear weights.

We observe in Section 5.4.2 that while MLS could be negatively correlated with the right-hand side of Equation (5.11), NMLS has a positive correlation with right-hand side of Equation (5.12) consistently. We do observe that MLS is still positively correlated with sharpness (Figure D.8.3). Therefore, the only possible reason for this negative correlation is the factor before sharpness in the MLS bound involving the first-layer weights and the quadratic mean (see Equation (5.11)).

Why LVR and MLS quantify compression. Having introduced our main compression metrics, we clarify why these quantities measure compression of neural representations. In our framework, *compression* refers to the concentration or contraction of representations when inputs are locally perturbed. LVR quantifies this via volume contraction: if a small volume in input space is mapped to a smaller volume in representation space, the network has compressed that region of its representation manifold. Mathematically, $\text{LVR} < 1$ indicates volume contraction (compression), $\text{LVR} = 1$ indicates volume preservation, and $\text{LVR} > 1$ indicates volume expansion. Similarly, MLS measures compression through sensitivity: lower MLS means the network output changes less in response to input perturbations, indicating that nearby inputs are mapped to nearby (compressed) outputs. Networks with high compression have both low LVR (contracted volumes) and low MLS (reduced sensitivity).

Importantly, our theoretical bounds do not *guarantee* that networks will compress (i.e., achieve $\text{LVR} < 1$ or small MLS). The bounds only establish that flatter minima constrain these metrics to be smaller. However, our extensive empirical results in Section 5.4 demonstrate that trained networks do indeed exhibit compression: we observe decreasing LVR and MLS values during training, particularly in the late stages when networks find flatter minima. The correlation between sharpness and these compression metrics validates that flatter minima are associated with more compressed representations in practice, even though the bounds themselves are upper bounds rather than guarantees of compression.

On the role of weight norms in the bounds. A critical aspect of our bounds is their dependence on both sharpness and weight-related prefactors (e.g., $\|\mathbf{W}\|_2/\|\mathbf{x}\|_2$ ratios). This raises a natural question: does lower sharpness *always* imply more compression? Theoretically, this is not guaranteed. Consider two networks at different interpolating solutions: if one has lower sharpness but substantially larger weight norms, the bound could be looser despite the decrease in sharpness. In principle, a decrease in sharpness might not fully counteract increases in the prefactor terms. We provide detailed empirical analysis of where tightness is lost in the chain of inequalities comprising our bounds in Appendix D.8.

However, despite this ambiguity in theory, the bounds may still apply usefully in practice. One important setting where this is the case is for a *fixed network architecture during training*, when the prefactors remain relatively stable as the network evolves toward flatter minima. Then weight norms (and input statistics) do not fluctuate dramatically within a training trajectory, so the sharpness term dominates the variation in the bound. This explains the strong positive correlations we observe between sharpness and compression metrics during training (e.g., Figure 5.2, Figure 5.3). In contrast, when *comparing across different network configurations*—such as in our analysis of 181 pretrained ViT models with varying architectures, datasets, and training procedures—the prefactors can vary substantially. This is precisely why we employ adaptive sharpness (which normalizes by weight norms) and normalized MLS in Figure 5.5: these normalized metrics account for the prefactor variations and reveal that the sharpness-compression relationship holds even across heterogeneous model families. The empirical robustness of this correlation, both within training trajectories and across diverse architectures when properly normalized, demonstrates that the bounds capture meaningful relationships even despite their theoretical limitations.

5.3.3 Local dimensionality is tied to, but not bounded by, sharpness

While LVR and MLS capture compression via volume contraction and maximum sensitivity, they do not capture the *uniformity* of compression across different directions. A network might

compress representations uniformly (reducing all directions proportionally) or anisotropically (compressing certain directions more than others). This distinction is captured by local dimensionality, which measures whether the eigenvalues of the Jacobian are uniform or concentrated in a few directions. As we will show, unlike LVR and MLS, local dimensionality cannot be directly bounded by sharpness alone, as it depends on eigenvalue ratios rather than their magnitudes.

Consider an input data point $\bar{\mathbf{x}}$ drawn from the training set: $\bar{\mathbf{x}} = \mathbf{x}_i$ for a specific $i \in \{1, \dots, n\}$. Let the set of all possible perturbations around $\bar{\mathbf{x}}$ in the input space be samples from an isotropic normal distribution, $\mathcal{B}(\bar{\mathbf{x}})_\alpha \sim \mathcal{N}(\bar{\mathbf{x}}, \alpha \mathcal{I})$, where $C_{\mathcal{B}(\bar{\mathbf{x}})_\alpha} = \alpha \mathcal{I}$, with $\mathcal{I}$ as the identity matrix, is the covariance matrix. We first propagate $\mathcal{B}(\bar{\mathbf{x}})_\alpha$ through the network transforming each point $\mathbf{x}$ into its corresponding image $f(\mathbf{x})$. Following a Taylor expansion for points within $\mathcal{B}(\bar{\mathbf{x}})_\alpha$ in the limit $\alpha \rightarrow 0$, we have:

$$f(\mathbf{x}) = f(\bar{\mathbf{x}}) + \nabla_{\mathbf{x}} f(\bar{\mathbf{x}}, \boldsymbol{\theta}^*)(\mathbf{x} - \bar{\mathbf{x}}) + O(\|\mathbf{x} - \bar{\mathbf{x}}\|^2) . \quad (5.13)$$

We can express the limit of the covariance matrix $C_{f(\mathcal{B}(\mathbf{x}))}$ of the output $f(\mathbf{x})$ as

$$C_f^{\text{lim}} := \lim_{\alpha \rightarrow 0} \frac{C_{f(\mathcal{B}(\mathbf{x})_\alpha)}}{\alpha} = \nabla_{\mathbf{x}} f(\bar{\mathbf{x}}, \boldsymbol{\theta}^*) \nabla_{\mathbf{x}}^T f(\bar{\mathbf{x}}, \boldsymbol{\theta}^*) . \quad (5.14)$$

Our covariance expressions capture the distribution of the samples in $\mathcal{B}(\bar{\mathbf{x}})_\alpha$ as they go through the network $f(\bar{\mathbf{x}}, \boldsymbol{\theta}^*)$. The local Participation Ratio based on this covariance is given by:

$$D_{\text{PR}}(f(\bar{\mathbf{x}})) = \lim_{\alpha \rightarrow 0} \frac{\text{Tr}[C_{f(\mathcal{B}(\mathbf{x})_\alpha)}]^2}{\text{Tr}[(C_{f(\mathcal{B}(\mathbf{x})_\alpha)})^2]} = \frac{\text{Tr}[C_f^{\text{lim}}]^2}{\text{Tr}[(C_f^{\text{lim}})^2]} \quad (5.15)$$

(Recanatesi et al. 2022, cf. nonlocal measures in Gao et al. 2017, Litwin-Kumar et al. 2017, Mazzucato et al. 2016).

Definition 5.11. The **Local Dimensionality** of a network f is defined as the sample mean of local participation ratio at different input samples: $D(f) = \frac{1}{n} \sum_{i=1}^n D_{\text{PR}}(f(\mathbf{x}_i))$

This quantity in some sense represents the sparseness of the eigenvalues of C_f^{lim} : If we let $\boldsymbol{\lambda}$ be all the eigenvalues of C_f^{lim} , then the local dimensionality can be written as $D_{\text{PR}} = (\|\boldsymbol{\lambda}\|_1 / \|\boldsymbol{\lambda}\|_2)^2$, which attains its maximum value when all eigenvalues are equal to each other, and its minimum when all eigenvalues except for the leading one are zero. Note that the quantity retains the same value when $\boldsymbol{\lambda}$ is arbitrarily scaled. As a consequence, it is hard to find a relationship between the local dimensionality and the fundamental quantity on which our bounds are based: $\|\nabla_{\mathbf{x}} f(\mathbf{x}, \boldsymbol{\theta}^*)\|_F^2$, which is $\|\boldsymbol{\lambda}\|_1$.

5.3.4 Relation to reparametrization-invariant sharpness

Dinh et al. [2017] argues that a robust sharpness metric should have the reparametrization-invariant property, meaning that scaling the neighboring linear layer weights should not change the metric. While the bounds in Proposition 5.8 and Proposition 5.10 are not strictly reparametrization-invariant, sharpness metrics that are redesigned [Tsuzuku et al., 2019] to achieve invariance can be proved to tighten our bounds (see Appendix D.6.1). Another more aggressive reparametrization-invariant sharpness is proposed in Andriushchenko et al. [2023], Kwon et al. [2021], and we again show that it upper-bounds input-invariant MLS in Appendix D.6.2. We also empirically evaluate the relative flatness [Petzka et al., 2021], which is also reparametrization-invariant in Appendix D.8.2, but no significant correlation is observed. Overall, we provide a novel perspective: reparametrization-invariant sharpness is characterized by the robustness of outputs to internal neural representations.

5.3.5 Connection to neural collapse and compressed neural representations

The neural collapse phenomenon [Papayan et al., 2020, Zhu et al., 2021] indicates that the within-class variance of the features in the penultimate layer vanishes at the terminal phase of training; allied studies have also found compressed neural representations at this and other

points internal to trained neural networks Farrell et al. 2022 (see also Farrell et al. 2023, Schwartz-Ziv and Tishby 2017). We next show that the present sharpness-based approach can describe related properties at penultimate and intermediate network layers. While neural collapse is often described as a *global* phenomenon (within-class variance across all samples of a class), our results are *local*, quantifying robustness of layer responses for nearby points in the input space. Nevertheless, a connection emerges when viewing within-class samples as local perturbations around their class mean, a perspective we develop below.

We first apply our method to study the penultimate-layer features. To accomplish this we can adapt the linear stability trick in Equation (5.5) to establish a relationship between their robustness and sharpness. More concretely, we can show that

$$\left\| \nabla_{\mathbf{x}} g(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}}) \right\|_F \leq \frac{\|\mathbf{W}\|_2}{\sigma_{\min}(\mathbf{W}_L) \|\mathbf{x}\|_2} \left\| \nabla_{\mathbf{w}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}}) \right\|_F . \quad (5.16)$$

Here again, $\mathbf{W}$ denotes the first-layer weights, $\mathbf{W}_L$ denotes the last-layer linear classifier weights, and $g(x)$ is the penultimate-layer feature. $\sigma_{\min}(\mathbf{W}_L)$ is defined as the square root of the smallest eigenvalue of $\mathbf{W}_L^T \mathbf{W}_L$. The proof is given in Appendix D.1.

To extend this analysis to representations before the penultimate layer, note that inequality (5.16) extends to any middle-layer representation, as detailed in Appendix D.1, so that very similar conclusions apply directly.

Let the feature dimension be d and the number of classes be K . Then, $\mathbf{W}_L \in \mathbb{R}^{K \times d}$, and $\sigma_{\min}(\mathbf{W}_L) = 0$ if $d > K$; otherwise, it is the smallest singular value of $\mathbf{W}_L$. An effective bound on the robustness of penultimate-layer (or other internal-layer) features is obtained when $d \leq K$, i.e., when the number of classes is at least as large as the feature dimension. This condition is naturally satisfied in settings such as language modeling, retrieval systems, and face recognition applications, where the number of classes (vocabulary size, number of items, number of identities) is typically much larger than the feature dimension.

Interestingly, these are also settings where generalized neural collapse phenomena have been observed [Jiang et al., 2023]. While our bounds are local (quantifying robustness to small

input perturbations), a potential connection to neural collapse emerges when we consider samples from the same class as local perturbations around the class mean. In neural collapse, the within-class variance of features vanishes, meaning all samples from a class produce nearly identical features. This can be viewed through our framework as extreme robustness: if the network maps all within-class samples (which differ from the class mean by small to moderate perturbations) to the same feature representation, the features exhibit strong compression and robustness properties that our bounds capture locally. This said, the relationship between our local robustness bounds and the global geometric structure of neural collapse, including other aspects not considered here (e.g., the simplex equiangular tight frame geometry) remains an open question for future investigation.

5.4 Experiments

All networks are trained with MSE (quadratic) loss except for the pretrained ViTs in Section 5.4.3.

5.4.1 Sharpness and compression metrics during training: verifying the theory

The theoretical results derived above show that when the training loss is low, measures of compression of the network’s representation are upper-bounded by a function of the sharpness of the loss function in parameter space. This links sharpness and representation compression: the flatter the loss landscape, the lower the upper bound on the representation’s compression metrics.

To empirically verify whether these bounds are sufficiently tight to show a clear relationship between sharpness and representation compression, we trained a VGG-11 network [Simonyan and Zisserman, 2015] to classify images from the CIFAR-10 dataset [Krizhevsky, 2009] and calculated the (approximate) sharpness (Equation (5.3)), the log volumetric ratio

(Equation (5.7)), MLS (Definition 5.7) and NMLS (Definition 5.9) during the training phase (Figure 5.2 and Figure 5.3).

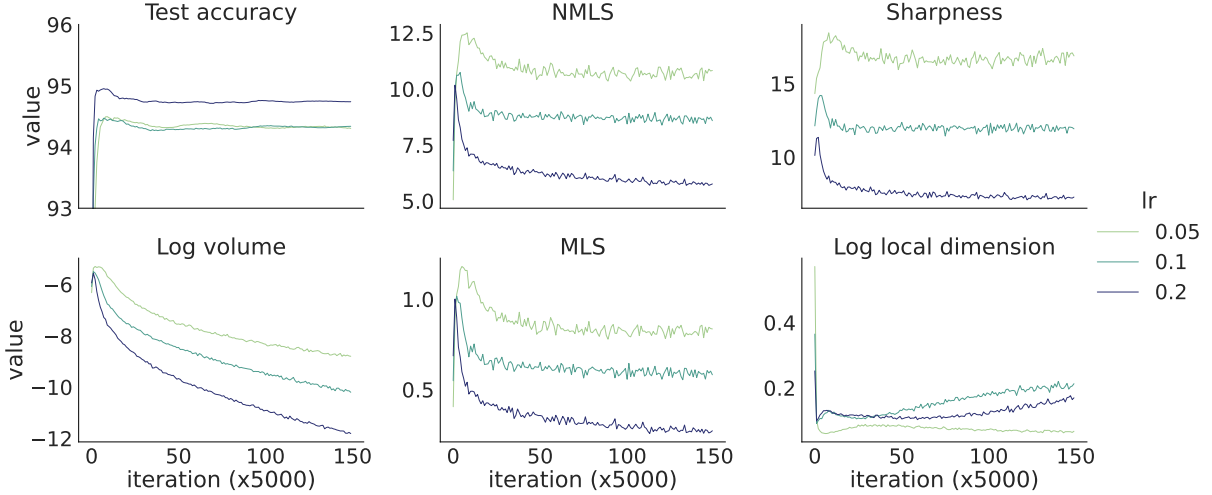


Figure 5.2: Trends in key variables across SGD training of the VGG-11 network with fixed batch size (equal to 20) and varying learning rates (0.05, 0.1 and 0.2). Higher learning rates lead to lower sharpness and hence stronger compression. From left to right: Test accuracy, NMLS, sharpness (square root of Equation (5.3)), log volumetric ratio (Equation (5.7)), MLS, and local dimensionality of the network output (Equation (5.15)).

We trained the network using SGD on CIFAR-10 images and explored the influence of two specific parameters that previous work has shown to affect the network’s sharpness: learning rate and batch size [Jastrzebski et al., 2018]. For each combination of learning rate and batch size parameters, we computed all quantities across 100 input samples and averaged across five different random initializations for network weights.

In Figure 5.2, we study the link between sharpness and representation compression with a fixed batch size (of 20). We observe that when the network reaches the interpolation regime, that is, when training loss is extremely low, the sharpness decreases with compression metrics, including MLS, NMLS, and volume. The trend is consistent across multiple learning rates for a fixed batch size, and MLS and NMLS match better with the trend of sharpness.

In Figure 5.3, we repeated the experiments while keeping the learning rate fixed at 0.1 and varying the batch size. The same broadly consistent trends emerged, linking a decrease

in the sharpness to a compression in the neural representation. However, we also found that while sharpness stops decreasing after about $5 \cdot 10^4$ iterations for a batch size of 32, the volume continues to decrease as learning proceeds. This suggests that other mechanisms, beyond sharpness, may be at play in driving the compression of volumes.

We repeated the experiments with an MLP trained on the FashionMNIST dataset [Xiao et al., 2017] (Figure D.9.5 and Figure D.9.6). The sharpness again follows the same trend as MLS and NMLS, consistent with our bound. The volume continues to decrease after the sharpness plateaus, albeit at a much slower rate, again matching our theory, while suggesting that an additional factor may be involved in its decrease.

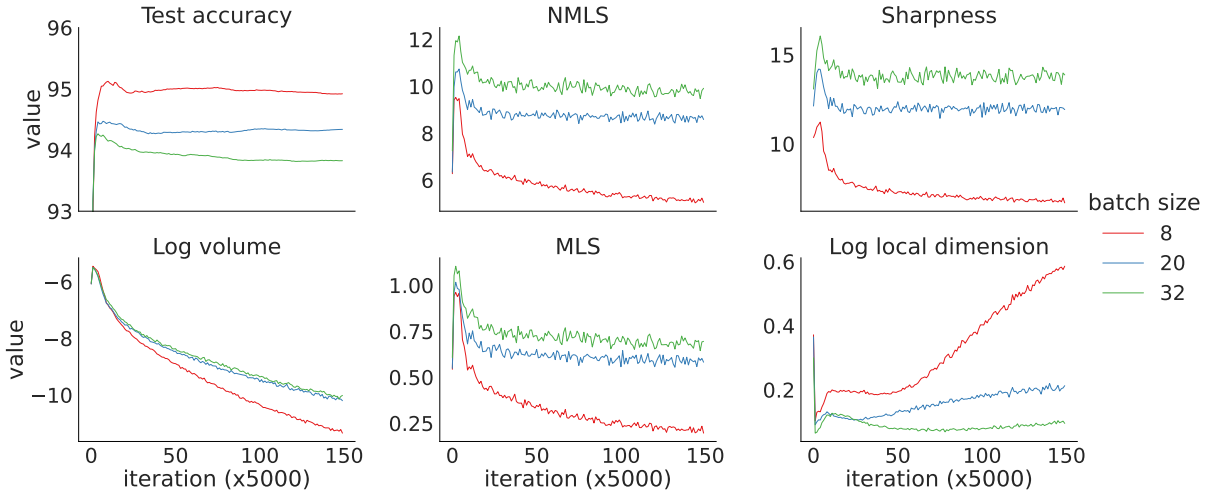


Figure 5.3: Trends in key variables across SGD training of the VGG-11 network with fixed learning rate size (equal to 0.1) and varying batch size (8, 20, and 32). Smaller batch sizes lead to lower sharpness and hence stronger compression. From left to right in row-wise order: test accuracy, NMLS, sharpness (square root of Equation (5.3)), log volumetric ratio (Equation (5.7)), MLS, and local dimensionality of the network output (Equation (5.15)).

5.4.2 Correlation between compression metrics and their sharpness-related bounds

We next test the correlation between both sides of the bounds that we derive more generally. In Figure 5.4, we show pairwise scatter plots between MLS (resp. NMLS) and the sharpness-

related bound on MLS (resp. NMLS) (we include an exhaustive set of correlation matrices in Appendix D.8.2). Interestingly, we find that quantities that only consider a single layer of weights, such as the relative flatness [Petzka et al., 2021] and the bound on the MLS (Proposition 5.8), can exhibit a negative correlation between both sides of the bound in some cases (Figures 5.4, D.8.2 and D.8.3). This demonstrates the necessity of introducing NMLS to explain the relationship between sharpness and compression. Nevertheless, we find that all compression metrics, such as MLS, NMLS, and LVR, introduced in Section 5.3, correlate well with sharpness in all of our experiments (Appendix D.8.2). Although the bound in Proposition 5.4 is loose, the log LVR still correlates positively with sharpness and MLS.

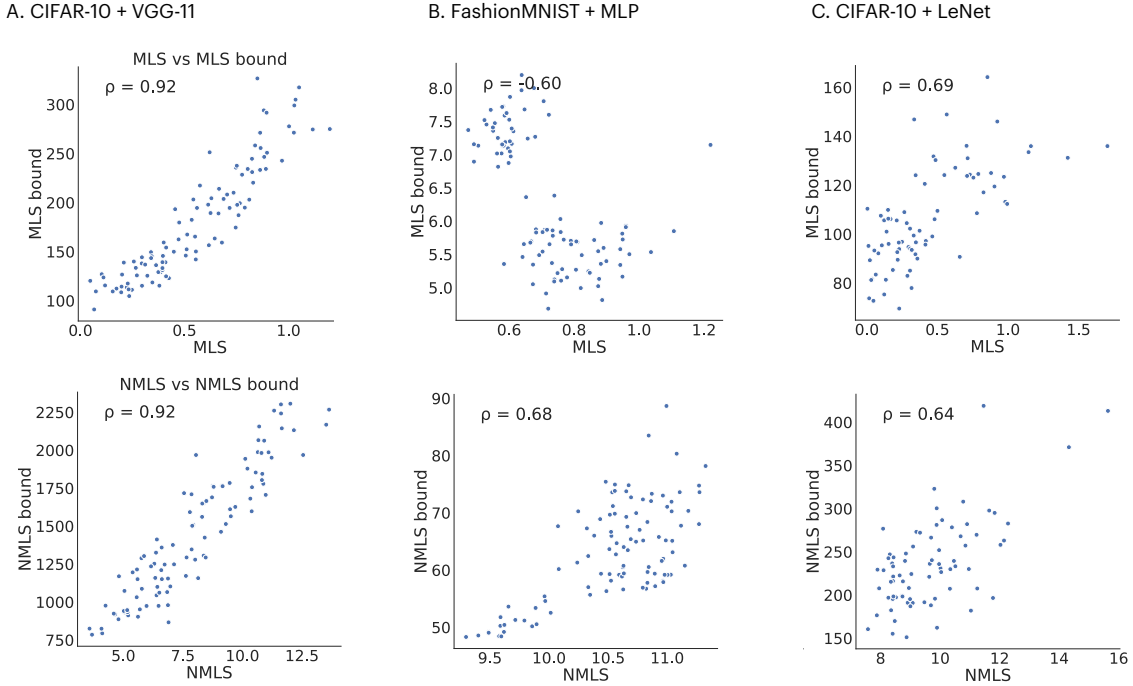


Figure 5.4: We trained 100 different models for each combination of datasets and networks by varying learning rates, batch size, and random initializations. Pairwise scatter plots between MLS (resp. NMLS) and the sharpness-related bound on MLS (resp. NMLS) are shown here. For MLS (resp. NMLS) bound see Proposition 5.8 (resp. Proposition 5.10). The Pearson correlation coefficient ρ is shown in the top-left corner for each scatter plot. See Appendix D.8.2 for the full pairwise scatter matrix.

5.4.3 Empirical evidence in Vision Transformers (ViTs)

Since our theory applies to linear and convolutional layers as well as residual layers (Appendix D.2), relationships among sharpness and compression, as demonstrated above for VGG-11 and MLP networks, it should hold more generally in modern architectures such as the Vision Transformer (ViT) and its variants. However, naive ways of evaluating the quantities discussed in previous sections are computationally prohibitive. Instead, we look at the MLS normalized by the norm of the input and the elementwise-adaptive sharpness defined in Andriushchenko et al. [2023], Kwon et al. [2021]. Both of the metrics can be estimated efficiently for large networks. Specifically, in Figure 5.5 we plot the normalized MLS against the elementwise-adaptive sharpness. The analytical relationship between the normalized MLS and the elementwise-adaptive sharpness and the details of the numerical approximation we used are given in Appendix D.6.2 and Appendix D.7 respectively. For all the models, we attach a sigmoid layer to the output logits and use MSE loss to calculate the adaptive sharpness. Figure 5.5 shows the results for 181 pretrained ViT models provided by the `timm` package [Wightman, 2019]. We observe that there is a general trend that lower sharpness indeed implies lower MLS. However, there are also outlier clusters, with data corresponding to the same model class; an interesting future direction would be to understand the mechanisms driving this outlier behavior. Interestingly, we did not observe this correlation between unnormalized metrics, indicating that weight scales should be taken into account when comparing between different models.

5.4.4 Sharpness and local dimensionality

A priori, it is unclear whether the local dimensionality of the neural representations should increase or decrease as the volume is compressed. For example, the volume could decrease while maintaining its overall form and symmetry, thus preserving its dimensionality. Alternatively, one or more of the directions in the relevant tangent space could be selectively compressed, leading to an overall reduction in dimensionality.

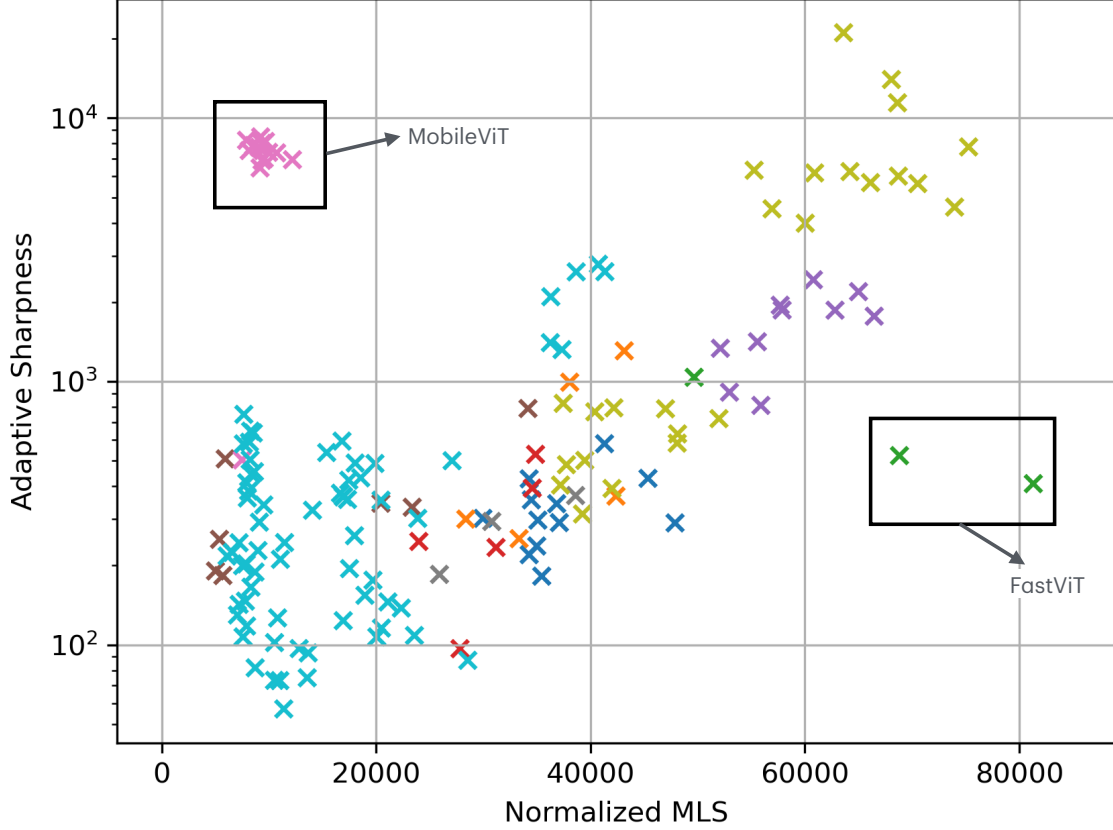


Figure 5.5: Adaptive sharpness vs Normalized MLS for 181 ViT models and variants. Different colors represent different model classes. For most models, there is a positive correlation between Sharpness and MLS. However, outlier clusters also exist, for MobileViT [Mehta and Rastegari, 2022] models in the upper left corner, and two FastViT [Vasu et al., 2023] models in the lower right corner.

Figure 5.2 and Figure 5.3 show our experiments computing the local dimensionality over the course of training. Here, we find that the local dimensionality of the representation decreases as the loss decreases to near 0, which is consistent with the viewpoint that the network compresses representations in feature space as much as possible, retaining only the directions that code for task-relevant features [Berner et al., 2020, Cohen et al., 2020]. However, the local dimensionality exhibits unpredictable behavior that cannot be explained by the sharpness once the network finds an approximate interpolation solution and training continues. Further experiments also demonstrate a weaker correlation of sharpness and local dimensionality compared to other metrics such as MLS and volume (Figures D.8.2 to D.8.4). This discrepancy is consistent with the bounds established by our theory, which

only bound the numerator of Equation (5.15). It is also consistent with the property of local dimensionality that we described in Section 5.3.3 overall: it encodes the sparseness of the eigenvalues but it does not encode the magnitude of them. This shows how local dimensionality is a distinct quality of network representations compared with volume, and is driven by mechanisms that differ from sharpness alone. We emphasize that the dimensionality we study here is a local measure, on the finest scale around a point on the “global” manifold of unit activities; dimension on larger scales (i.e., across categories or large sets of task inputs [Farrell et al., 2022, Kothapalli et al., 2022, Zhu et al., 2021, Ansuini et al., 2019, Recanatesi et al., 2019, Pappan et al., 2020]) may show different trends.

5.5 Discussion: connection to generalization

So far we have avoided remarking on implications of our results for the relationship between generalization and sharpness or compression because generalization is *not* the main focus of our work. However, our work may have implications for future research on this relationship, and we briefly discuss this here. Our theoretical results establish that sharpness directly controls compression of neural representations—specifically, how robust intermediate representations are to input perturbations. By "robustness of representations," we mean that small changes to the input produce proportionally small changes in the network’s internal feature representations (quantified by low LVR and MLS). This compression property is desirable in adversarial settings where we want representations to remain stable under input perturbations.

However, representation robustness alone does not guarantee better generalization. Ma and Ying [2021] gave a generalization bound based on adversarial robustness, but the theorem requires that most test data lie close to the training data, which demands extensive training coverage. In contrast, our correlation analysis in Appendix D.8.2 shows that sharpness consistently has a higher correlation with compression metrics than with the generalization

gap. Additional experiments in Appendix D.9 show that lower sharpness does not always imply better test accuracy.

This phenomenon is also observed in applied machine learning literature. For example, it is well-established that large learning rates lead to lower sharpness (flatter minima), a relationship we also observe in our Figure 5.2 and consistent with Cohen et al. [2020] and Wu et al. [2022]. If low sharpness directly implied better generalization, we would expect large learning rates to improve out-of-distribution (OOD) performance. However, Wortsman et al. [2022] demonstrates the opposite: large learning rates can severely hurt OOD generalization. This shows that low sharpness alone—and the compressed representations it produces—does not guarantee superior generalization.

Thus, our work offers a possible resolution to the contradictory results on the relationship between (reparametrization-invariant) sharpness and generalization [Wen et al., 2023, Andriushchenko et al., 2023]: sharpness directly determines representation compression, but whether this compression translates to superior generalization may depend on other latent factors beyond just sharpness.

One possible factor is that compression metrics (LVR, MLS, NMLS) measure robustness to *small random perturbations*—networks with low sharpness have representations that are insensitive to noise. However, good generalization sometimes requires networks to remain sensitive to *semantically meaningful differences* in the input. To illustrate this distinction, consider a large language model (LLM) performing a "needle in a haystack" test, where it must extract specific information from a long context. When the query changes meaningfully (e.g., asking for a different fact), the output should change substantially to provide the correct answer—the network must be sensitive to the semantic content of the query. Yet simultaneously, the network should be robust to small random perturbations in the text, such as typos or minor paraphrasing. Sharpness controls the latter (robustness to random noise), but whether this robustness aids generalization depends on the nature of the distribution shift at test time: if test inputs primarily differ from training via noise-like perturbations,

compression helps; if they differ in task-relevant semantic features that require discrimination, excessive compression may reduce the network’s ability to distinguish these meaningful differences.

5.6 Summary and Conclusion

This work presents a dual perspective, uniting views in both parameter and feature space, of several properties of trained neural networks. We identify two representation compression metrics that are bounded by sharpness – local volumetric ratio and maximum local sensitivity – and give new explicit formulas for these bounds. We conducted extensive experiments with feedforward, convolutional, and attention-based networks and found that the predictions of these bounds are borne out for these networks, illustrating how MLS in particular is strongly correlated with sharpness. Overall, we establish explicit links between sharpness properties in parameter spaces and compression and robustness properties in the feature space.

By demonstrating both how these links can be tight, and how and when they may also become loose, we propose that taking this dual perspective can bring more clarity to the often confusing question of what sharpness actually quantifies in practice. Indeed, many works, as reviewed in the introduction, have demonstrated how sharpness can lead to generalization, but recent studies have established contradictory results. Therefore, our work contributes to the further exploration of sharpness-based methods for improving neural network performance. Future directions include tightening the derived bounds given knowledge of the data distribution, and investigating the conditions under which representation compression translates to improved generalization.

Chapter 6

Feedforward Memory Networks and Online Synaptic Representations

Chapter 5 studied representations after optimization, relating the geometry of a trained loss landscape to compression across network layers. This chapter moves from pre-trained parameters to weights that continue to change during computation. In feedforward memory networks, synaptic plasticity makes the current weight matrix a representation of recent experience, allowing us to ask how update rules and heterogeneous timescales shape what the neural system can remember.

6.1 Introduction

We receive a continuous stream of sensory inputs in our daily lives, and we need to draw on our previous experiences when making decisions. Working memory (WM), the ability to recall recently received stimuli and operate on them, is proposed to be implemented by persistent neural activity [Funahashi, 2017, Fuster and Alexander, 1971] and activity-silent synaptic weights [Silvanto, 2017, Sreenivasan et al., 2014, Mongillo et al., 2008]. Our work focuses on how recent experiences are encoded in the synaptic traces. Classical models of memory [Hopfield, 1982] are content-addressable systems, meaning that the retrieval happens

if a partial cue is provided. These classical models mandate symmetric synaptic weights and have limited memory capacity [Ramsauer et al., 2021].

Recent transformer-like memory models such as biological key-value networks [Tyulmankov et al., 2021] and modern Hopfield networks [Ramsauer et al., 2021] store keys and values explicitly in parameters. In particular, biological key-value networks manually provide local third factors as inputs to decide whether an input should be completely ignored, and gate the synaptic change to each hidden unit to enable the selection of a single neuron for memory writing. While the modern Hopfield network and its biological counterpart [Krotov and Hopfield, 2021] provide an addressing mechanism, they do not specify how to compute the stored patterns in a biologically plausible way, nor do they specify how to selectively keep memory of information from a continuous input stream *over time* to complete more complex tasks such as those in the NeuroGym package [Molano-Mazón et al., 2022].

More recent works [Aitken and Mihalas, 2023, Tyulmankov et al., 2021, 2022] have proposed biologically plausible memory models that constantly update their synaptic weights in response to a continual stream of inputs to complete complex tasks. Both models consider a feedforward network with one hidden layer whose first-layer weights are endowed with Hebbian synaptic plasticity. This class of models, which we call feedforward memory networks (FMNs), can complete complex memory tasks such as continual familiarity detection, retrospective contextual integration, and a battery of other neuroscience tasks. However, the underlying mechanism by which these networks complete various neuroscience tasks remains elusive. To understand this class of biological models theoretically, we analyze two components of FMNs that are of independent interest. First, we assume fixed synaptic weights in the first layer and analyze the feedforward architecture based on the theory developed for modern Hopfield networks [Ramsauer et al., 2021, Krotov and Hopfield, 2021]. We show that the output of an FMN is equivalent to a linear transformation of the retrieval of memory patterns in a Hopfield network. Therefore, the performance of an FMN critically hinges upon the memory patterns that it learns through synaptic dynamics. Second, we study how the synaptic dynamics of

the first-layer weights give rise to the memory patterns of the equivalent Hopfield network. We reveal a close relationship between the performance of these FMNs and the variety of memory patterns that they learn, both theoretically and empirically. Inspired by evidence that biological synaptic plasticity can occur across a spectrum of timescales, ranging from milliseconds [Erickson et al., 2010] to seconds [Bittner et al., 2017] to hours [Becker and Tetzlaff, 2021], we hypothesize that heterogeneous synaptic parameters can facilitate the formation of varied memory patterns, thereby yielding better representations of past events that are necessary to complete temporal tasks. To test this hypothesis, we introduce synapse-wise parameters for both decay rates and plasticity rates that parametrize the feedforward network. We call the resulting model the Hetero-Plasticity Network (HPN), and show that the HebbFF network in [Tyulmankov et al., 2022] and MPN in [Aitken and Mihalas, 2023] are special cases of the HPN. Therefore, we propose in this chapter an integrated framework for understanding capabilities of all FMNs.

More specifically, our contributions are as follows:

- We show that the feedforward memory networks implicitly implement the cross-attention mechanism. We further show that the feedforward memory networks can be interpreted as a linear transformation of the retrieval of a modern Hopfield network.
- We introduce the Hetero-Plasticity Network (HPN), a biologically plausible memory model that meta-learns decay- and plasticity-rate parameters, to study the effect of heterogeneous synaptic plasticity on the learning dynamics of stored patterns. We show that MPN and HebbFF networks are special cases of HPN.
- We analytically study the learning dynamics of stored patterns in HPN. We reveal how heterogeneous decay rates and plasticity rates affect the eigenvalues and eigenvectors of the linear transition matrix of the learning dynamics.
- We empirically validate the predictions of our analysis on NeuroGym tasks and evaluate the effect of different parameter settings on network performance.

6.2 A Hopfield formulation of feedforward memory networks: Preliminaries

Without temporal synaptic dynamics, the dynamics of all FMNs, including MPN, HebbFF, and the HPN that we propose, can be simplified to a two-layer, fully connected feedforward network:

$$\begin{aligned}\mathbf{o}_t &= W_{\text{out}} \mathbf{h}_t \\ \mathbf{h}_t &= F(W_{\text{in}} \mathbf{x}_t),\end{aligned}\tag{6.1}$$

where $\mathbf{x}_t \in \mathbb{R}_i^d$ is the input (query), $W_{\text{in}} \in \mathbb{R}^{d_h \times d_i}$ is the input-weight matrix, $W_{\text{out}} \in \mathbb{R}^{d_o \times d_h}$ is the output-weight matrix, and F is the nonlinearity (e.g., $\tanh$) used by the hidden-layer neurons. Although all FMNs without synaptic dynamics can be reduced to this simple form, different FMNs perform differently on the same neuroscience tasks [Aitken and Mihalas, 2023], indicating that the learning dynamics of their synaptic plasticity are central to their performance. To understand how synaptic dynamics contribute to performance, we first need to establish the connection between the learned synaptic weights and the output of the FMN.

Inspired by previous work [Krotov and Hopfield, 2016], we propose to understand FMNs under the Hopfield formulation. Krotov and Hopfield [2016] already noted the equivalence between feedforward networks and dense associative memory, a discrete counterpart of the modern Hopfield network. We adapt the result of Krotov and Hopfield [2021] to show that the network output $\mathbf{o}_t$ in Equation (6.1) can be interpreted as a linear transformation of the pattern retrieved from a modern Hopfield network (Appendix E.2). This motivates us to study how the learning dynamics produce the memory patterns defined by W_{in} in this formulation in Section 6.3. Figure 6.1 illustrates our overall approach to understanding FMNs through their corresponding memory patterns under the Hopfield formulation.

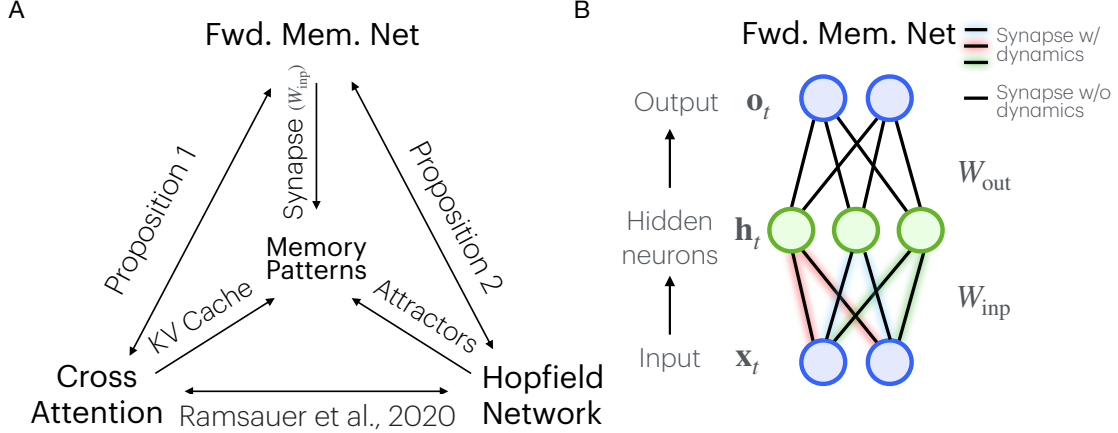


Figure 6.1: Unified Hopfield formulation to understand FMNs. **A**: The equivalence between feedforward memory networks (FMNs), modern Hopfield networks, and cross-attention mechanisms (transformer models). The memory patterns are stored in synaptic weights, attractors, and key-value caches (KV Cache) [Pope et al., 2023] respectively. This unified framework shows how the stored memory patterns in these formulations are essential to understand the learning dynamics of FMNs and make predictions about their performance. **B**: Illustration of FMNs. The only difference between different FMNs lies in the synaptic dynamics of W_{in} , which determine how memory patterns evolve through time. Different shades of input weights represent different memory patterns.

6.3 Learning dynamics of stored patterns

We introduce the Hetero-Plasticity Network (HPN) to study the effect of heterogeneous synaptic plasticity on FMN’s capacity to complete complex tasks.

HPN is a feedforward memory network with the following Hebbian synaptic dynamics in the input layer:

$$W_{\text{in}}^{(t+1)} - W^0 = \boldsymbol{\lambda} \odot (W_{\text{in}}^{(t)} - W^0) + \boldsymbol{\eta} \odot \mathbf{h}_t \mathbf{x}_t^T, \quad (6.2)$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{d_h \times d_i}$ and $\boldsymbol{\eta} \in \mathbb{R}^{d_h \times d_i}$ are the decay rates and plasticity rates, respectively, for all synapses, and W^0 is the baseline synaptic-weight matrix to which W_{ij} will converge in the absence of input. We assume $0 < \lambda_{ij} < 1$ for every synapse. In particular, with $W^0 = \mathbf{0}$, W_{in} will converge to $\mathbf{0}$ in the absence of input. We show that both the HebbFF network [Tyulmankov et al., 2021] and the MPN [Aitken and Mihalas, 2023] can be interpreted as special cases of HPN in Section 6.3.1 and Section 6.3.2. Then we study the effects of heterogeneous synaptic plasticity on the learning dynamics of stored patterns in Section 6.3.3.

6.3.1 HebbFF networks are HPNs with uniform decay and plasticity rates

In the HebbFF network, the Hebbian synaptic dynamics of the first layer is given by

$$\begin{aligned}\mathbf{M}_{t+1} &= \lambda \mathbf{M}_t + \eta \mathbf{h}_t \mathbf{x}_t^T \\ W_{\text{in}}^{(t)} &= \mathbf{M}_t + W^0,\end{aligned}\tag{6.3}$$

where $\mathbf{M}_t$ is the time-dependent synaptic modulation matrix, λ and η are scalar parameters learned through training, and W^0 is the base synaptic weights. If we expand $\mathbf{M}_t$ and apply the recurrence again,

$$\begin{aligned}W_{\text{in}}^{(t+1)} &= \lambda \mathbf{M}_t + \eta \mathbf{h}_t \mathbf{x}_t^T + W^0 \\ &= \lambda(\mathbf{M}_{t-1} + W^0) + \eta \mathbf{h}_{t-1} \mathbf{x}_{t-1}^T + (1 - \lambda)W^0 \\ &= \lambda W_{\text{in}}^{(t)} + \eta \mathbf{h}_t \mathbf{x}_t^T + (1 - \lambda)W^0.\end{aligned}\tag{6.4}$$

Therefore, HebbFFs can be written as HPNs with uniform decay rates and plasticity rates:

$$W_{\text{in}}^{(t+1)} - W^0 = \lambda(W_{\text{in}}^{(t)} - W^0) + \eta \mathbf{h}_t \mathbf{x}_t^T\tag{6.5}$$

6.3.2 MPNs are HPNs with uniform decay rates and structured heterogeneous plasticity rates

In MPN, the Hebbian synaptic dynamics of the first layer of the feedforward memory network is given by

$$\begin{aligned}\mathbf{M}_{t+1} &= \lambda \mathbf{M}_t + \eta \mathbf{h}_{t+1} \mathbf{x}_{t+1}^T \\ W_{\text{in}}^{(t+1)} &= (\mathbf{M}_t + \mathbf{1}) \odot \mathbf{W}_{\text{base}},\end{aligned}\tag{6.6}$$

where the notation is consistent with Equation (6.3). If we expand $\mathbf{M}_t$ and apply the recurrence again,

$$\begin{aligned} W_{\text{in}}^{(t+1)} &= \lambda(\mathbf{M}_{t-1} + \mathbf{1}) \odot \mathbf{W}_{\text{base}} + (1 - \lambda)W^0 \\ &\quad + \eta \mathbf{h}_t \mathbf{x}_t^T \odot W^0 \\ &= \lambda W_{\text{in}}^{(t)} + \eta W^0 \odot \mathbf{h}_t \mathbf{x}_t^T + (1 - \lambda)W^0. \end{aligned} \tag{6.7}$$

Therefore, MPN implicitly implements element-wise plasticity rates by learning ηW^0 :

$$W_{\text{in}}^{(t+1)} - W^0 = \lambda(W_{\text{in}}^{(t)} - W^0) + \eta W^0 \odot \mathbf{h}_t \mathbf{x}_t^T. \tag{6.8}$$

We want to emphasize here that the HPN formulation is more general than MPN even when one uses matrix-valued $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$ in the MPN equation. To see this, consider a case that can be achieved by HPN but cannot be achieved by the all-matrix form of MPN. Consider a case where the connectivity matrix converges to zero when no activity is present. In this case, for HPN, we simply set $\mathbf{W}_{\text{base}} = \mathbf{0}$. For HPN, we have $2n^2$ parameters from $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$. However, if we use MPN to describe the case where the connectivity matrix converges to zero when no activity is present, we must require $\mathbf{W}_{\text{base}} = \mathbf{0}$. In this case, even if one uses the matrix version of MPN with $2n^2$ parameters, due to the zero baseline connectivity matrix, the effective connectivity remains zero regardless of the presence of activity in $\mathbf{x}$ or $\mathbf{h}$. Thus, we can see that MPN is a particular case of HPN even when one uses matrix-valued $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$.

6.3.3 Temporal capacity of stored patterns

Since all historical information is stored in the synaptic traces, an important question is whether the network can extract information about recent experiences from those traces at the current time step. We use the linear Taylor approximation of the tanh nonlinearity to provide a theoretical analysis. We justify this choice of nonlinearity in Appendix E.2.4. For

the i -th stored pattern $\mathbf{w}_i$ (a row of W_{in}), we have

$$\mathbf{w}_i^{(t)} = \boldsymbol{\lambda}_i \odot \mathbf{w}_i^{(t-1)} + \boldsymbol{\eta}_i \odot \tanh(\mathbf{w}_i^{(t-1)} \mathbf{x}_{t-1}) \mathbf{x}_{t-1}^T \quad (6.9)$$

$$\approx \boldsymbol{\lambda}_i \odot \mathbf{w}_i^{(t-1)} + \boldsymbol{\eta}_i \odot (\mathbf{w}_i^{(t-1)} \mathbf{x}_{t-1}) \mathbf{x}_{t-1}^T \quad (6.10)$$

We discuss four cases where 1) $\boldsymbol{\lambda}_i$ and $\boldsymbol{\eta}_i$ are both uniform, 2) $\boldsymbol{\lambda}_i$ is heterogeneous and $\boldsymbol{\eta}_i$ is uniform, 3) $\boldsymbol{\lambda}_i$ is uniform and $\boldsymbol{\eta}_i$ is heterogeneous, and 4) both are heterogeneous. We omit the index i in the following section for simplicity. We also assume that the base synaptic plasticity is zero W.L.O.G., since we can always reparametrize $\mathbf{w}^{(t)}$ to include an offset.

Uniform synaptic plasticity

When $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$ are both uniform, we have

$$\begin{aligned} \mathbf{w}_t &= \lambda \mathbf{w}_{t-1} + \eta (\mathbf{w}_{t-1} \mathbf{x}_{t-1}) \mathbf{x}_{t-1}^T \\ &= \mathbf{w}_{t-1} (\lambda \mathbf{I} + \eta \mathbf{x}_{t-1} \mathbf{x}_{t-1}^T). \end{aligned} \quad (6.11)$$

Therefore, $\mathbf{w}_t$ is a linear dynamical system with a symmetric DPR1 (Diagonal Plus Rank-1 matrix) state transition matrix. The following lemma shows that there is at most one eigenvalue whose absolute value is greater than 1.

Lemma 6.1. *The eigenvalues of $\lambda \mathbf{I} + \eta \mathbf{x} \mathbf{x}^T$ are given by $\lambda + \eta \|\mathbf{x}\|^2$ and λ , where the geometric multiplicity of λ is $d_i - 1$.*

The proof is given in Appendix E.3.1. As a result, a non-degenerate $\mathbf{w}$ can only converge to two directions if the input $\mathbf{x}$ remains a constant vector. More specifically, we have the following proposition.

Proposition 6.2. *Assume $\mathbf{x}_t = \mathbf{x} \neq \mathbf{0}$ for all t , and $\mathbf{w}_0$ has a nonzero component along $\mathbf{x}$. Then $\left| \frac{\mathbf{w}_t \cdot \mathbf{x}}{\|\mathbf{w}_t\| \|\mathbf{x}\|} \right| \rightarrow 1$ as $t \rightarrow \infty$ when $\eta > 0$ or $\eta < -\frac{2\lambda}{\|\mathbf{x}\|^2}$, and $\mathbf{w}_t \rightarrow \mathbf{0}$ as $t \rightarrow \infty$ when $-\frac{2\lambda}{\|\mathbf{x}\|^2} < \eta < 0$.*

The proof is given in Appendix E.3.1. The proof of Proposition 6.2 shows that with each update, the components of the stored pattern orthogonal to the current input will decay relative to the component aligned with the current input. Given constant input, the absolute cosine between the stored pattern and the input converges to 1. Empirically, we found that the stored patterns learned with uniform synaptic plasticity form two clusters symmetric with respect to the origin (Figure 6.4), meaning that the learned memory patterns indeed converge to one of the two directions empirically.

Heterogeneous decay rates and uniform plasticity rates

When $\boldsymbol{\lambda}$ is heterogeneous and η is uniform, we have

$$\begin{aligned}\mathbf{w}_t &= \mathbf{w}_{t-1} \text{diag}(\boldsymbol{\lambda}) + \eta(\mathbf{w}_{t-1}\mathbf{x}_{t-1})\mathbf{x}_{t-1}^T \\ &= \mathbf{w}_{t-1}(\text{diag}(\boldsymbol{\lambda}) + \eta\mathbf{x}_{t-1}\mathbf{x}_{t-1}^T).\end{aligned}\tag{6.12}$$

Again we have a linear dynamical system with a symmetric DPR1 state transition matrix. One might expect that due to the heterogeneity of the decay rates, the transition matrix will have a more diverse eigenspectrum. However, the following proposition shows otherwise.

Proposition 6.3. *The matrix $\text{diag}(\boldsymbol{\lambda}) + \eta\mathbf{x}\mathbf{x}^T$ has at most one eigenvalue whose absolute value is greater than 1. Its largest eigenvalue $\mu_{\max}$ satisfies $\max(\boldsymbol{\lambda}) \leq \mu_{\max} \leq \max(\boldsymbol{\lambda}) + \eta\|\mathbf{x}\|^2$ when $\eta > 0$, and its smallest eigenvalue $\mu_{\min}$ satisfies $\min(\boldsymbol{\lambda}) + \eta\|\mathbf{x}\|^2 \leq \mu_{\min} \leq \min(\boldsymbol{\lambda})$ when $\eta < 0$.*

The proof is given in Appendix E.3.2. We can use the same argument used in the proof of Proposition 6.2 to show that the stored pattern generically converges, up to sign, to the eigendirection associated with the eigenvalue of largest magnitude. The eigenvector of $\text{diag}(\boldsymbol{\lambda}) + \eta\mathbf{x}\mathbf{x}^T$ corresponding to μ is given by $(\text{diag}(\boldsymbol{\lambda}) - \mu\mathbf{I})^{-1}\mathbf{x}$ [Bunch et al., 1978]¹. Using the result of Proposition 6.3, we observe that all the entries of $(\text{diag}(\boldsymbol{\lambda}) - \mu\mathbf{I})^{-1}$ have the

¹also known as the Bunch–Nielsen–Sorensen formula

same sign. In the next section, we show that this constraint does not apply to the case where the decay rate is uniform and the plasticity rates are heterogeneous. Below we will show empirically that we observe a worse performance for this parameter setting than other heterogeneous settings (Section 6.4.1), indicating that this is a significant barrier to learning.

Uniform decay rates and heterogeneous plasticity rates

When λ is uniform and $\boldsymbol{\eta}$ is heterogeneous, we have

$$\begin{aligned}\mathbf{w}_t &= \lambda \mathbf{w}_{t-1} + \mathbf{w}_{t-1} \mathbf{x}_{t-1} \mathbf{x}_{t-1}^T \text{diag}(\boldsymbol{\eta}) \\ &= \mathbf{w}_{t-1} (\lambda \mathbf{I} + \mathbf{x}_{t-1} \mathbf{x}_{t-1}^T \text{diag}(\boldsymbol{\eta})).\end{aligned}\tag{6.13}$$

Here, the transition matrix is a DPR1 matrix that is not symmetric. We can easily see that the left eigenvector of $\lambda \mathbf{I} + \mathbf{x}_{t-1} \mathbf{x}_{t-1}^T \text{diag}(\boldsymbol{\eta})$ is $\mathbf{x}_{t-1}^T \text{diag}(\boldsymbol{\eta})$, and the corresponding eigenvalue is $\lambda + \mathbf{x}_{t-1}^T \text{diag}(\boldsymbol{\eta}) \mathbf{x}_{t-1}$. The only other eigenvalue is λ , so the arguments used in the proof of Proposition 6.2 can be applied again to show that the stored patterns will converge to directions of $\pm \mathbf{x}^T \text{diag}(\boldsymbol{\eta})$. Unlike the setting where $\boldsymbol{\lambda}$ is heterogeneous and $\boldsymbol{\eta}$ is uniform, $\boldsymbol{\eta}$ here is trained without any constraints. Therefore, the stored patterns can converge to any direction in the input space given a fixed input $\mathbf{x}$, leading to better performance than the setting where $\boldsymbol{\lambda}$ is heterogeneous and $\boldsymbol{\eta}$ is uniform (Figure 6.3).

Heterogeneous decay rates and plasticity rates

When $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$ are both heterogeneous, we have

$$\begin{aligned}\mathbf{w}_t &= \mathbf{w}_{t-1} \text{diag}(\boldsymbol{\lambda}) + \mathbf{w}_{t-1} \mathbf{x}_{t-1} \mathbf{x}_{t-1}^T \text{diag}(\boldsymbol{\eta}) \\ &= \mathbf{w}_{t-1} (\text{diag}(\boldsymbol{\lambda}) + \mathbf{x}_{t-1} \mathbf{x}_{t-1}^T \text{diag}(\boldsymbol{\eta})).\end{aligned}\tag{6.14}$$

Now, the transition matrix is a DPR1 matrix that is not symmetric. In this case, it turns out that we can always find $\boldsymbol{\lambda}$, $\boldsymbol{\eta}$ and $\mathbf{x}$ such that $\text{diag}(\boldsymbol{\lambda}) + \mathbf{x}_{t-1} \mathbf{x}_{t-1}^T \text{diag}(\boldsymbol{\eta})$ has an arbitrary set of eigenvalues.

Proposition 6.4. *Assume that $\lambda_i \neq 0$, $\lambda_i \neq \lambda_j$ for $i \neq j$, and $x_i \neq 0$ for all i . Then, for any $\boldsymbol{\lambda}$ and $\mathbf{x}$, there exists an $\boldsymbol{\eta}$ such that $\text{diag}(\boldsymbol{\lambda}) + \mathbf{x}\mathbf{x}^T \text{diag}(\boldsymbol{\eta})$ has any set of eigenvalues.*

The proof is given in Appendix E.3.3. The left eigenvectors corresponding to the eigenvalues of $\text{diag}(\boldsymbol{\lambda}) + \mathbf{x}\mathbf{x}^T \text{diag}(\boldsymbol{\eta})$ are given by $\mathbf{x}^T \text{diag}(\boldsymbol{\eta})(\text{diag}(\boldsymbol{\lambda}) - \mu \mathbf{I})^{-1}$, where μ is the eigenvalue [Bunch et al., 1978]. From the proof of Proposition 6.2, we see that the stored pattern generically converges, up to sign, to the left eigendirection associated with the eigenvalue of largest magnitude. Components along other eigenvectors need not decay in magnitude because there could be more than one eigenvalue whose absolute value is greater than 1. This result shows that the network can learn a diverse set of stored patterns. Empirically, we found that the performance of the network is better when $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$ are both heterogeneous than when they are both uniform (Figure 6.2).

6.4 Results

In this section, we train variants of FMNs, including HebbFF, MPN, and HPN, with varying heterogeneity on the NeuroGym tasks [Molano-Mazón et al., 2022] to test our theory. NeuroGym consists of 16 tasks, including contextual integration, continuous integration, and a battery of other neuroscience tasks (see Appendix E.4 for a full list). We compare the performance of three types of feedforward memory networks (HPN, MPN, and HebbFF) under various parameter settings and empirically verify the equivalence between the forms of FMN derived in Section 6.3.1 and Section 6.3.2. We then compare HPNs with heterogeneous decay rates but uniform plasticity rates against HPNs with uniform decay rates but heterogeneous plasticity rates. Finally, we visualize the learned patterns to verify the theoretical analysis in Section 6.3.3. Throughout this section, we use zero baseline Hebbian plasticity for HPNs.

6.4.1 Uniform synaptic plasticity gives worse performance

We first show that switching from uniform to heterogeneous synaptic plasticity improves performance on almost all tasks for every network type under consideration (Figure 6.2). We found that networks with uniform synaptic plasticity fail to learn most tasks except those that require only binary output, while networks with heterogeneous synaptic plasticity excel on all tasks. This result is consistent with the theoretical analysis in Section 6.3.3.

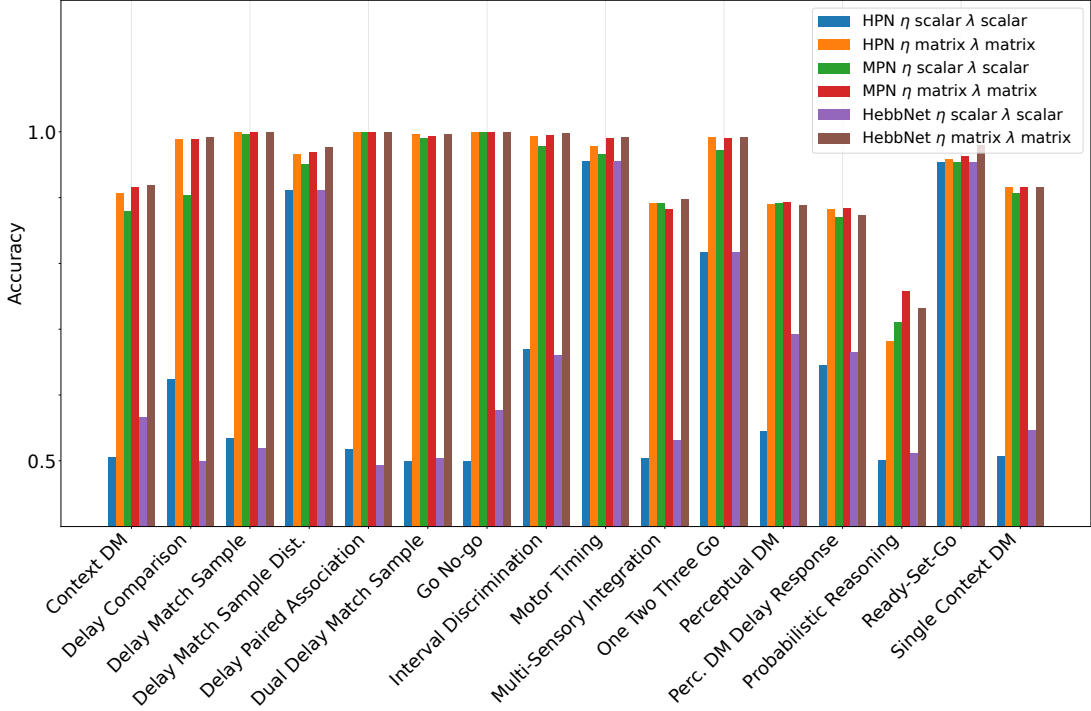


Figure 6.2: Performance of HPNs with different parameter settings on NeuroGym tasks.

6.4.2 Heterogeneous Plasticity Rates Yield Superior Performance Compared to Heterogeneous Decay Rates

Although HPNs with heterogeneous η and uniform λ and HPNs with uniform η and heterogeneous λ have the same number of parameters, our theory predicts that HPNs with uniform plasticity rates are restricted in the directions of their stored patterns. Indeed, we observe

empirically that HPNs with uniform plasticity rates perform significantly worse than HPNs with uniform decay rates (Figure 6.3).

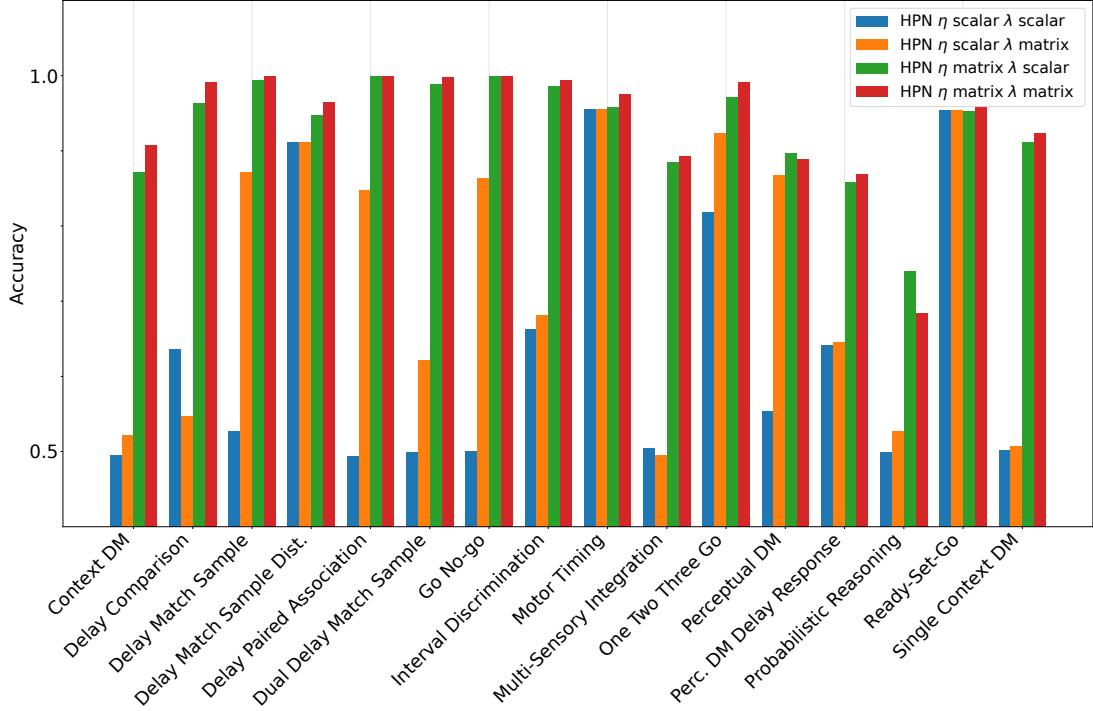


Figure 6.3: Performance of HPNs with different parameter settings on NeuroGym tasks.

6.4.3 Visualization of learned patterns

We also found that the stored patterns learned with uniform decay rates λ and plasticity rates η form two symmetric clusters with respect to the origin (Figure 6.4), consistent with the theoretical analysis in Section 6.3.3.

6.5 Discussion

In this work, we study an abstract class of feedforward memory networks (Equation (6.1) and Equation (6.2)) that store patterns in the synaptic weights of the first layer and use linear readouts of the hidden layer as outputs. This abstraction decouples the learning dynamics of

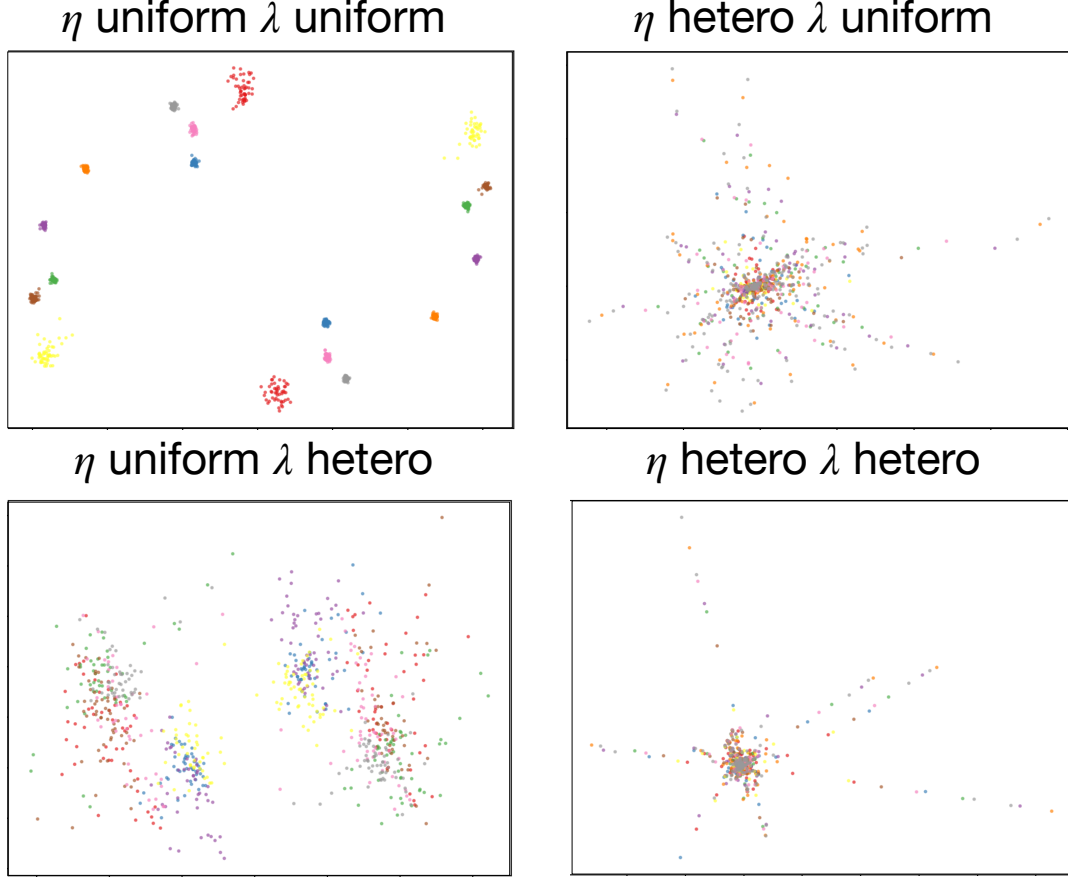


Figure 6.4: Learned memory patterns of HPNs with different parameter settings at cue time in the context-dependent decision-making task (different colors indicate different cue times). For the HPN with uniform η and λ , noise is added for visibility.

stored patterns from the retrieval computation carried out by the feedforward network. In terms of retrieval computation, our theoretical analysis demonstrates that FMN outputs are linear transformations of modern Hopfield network retrievals, linking feedforward memory networks to the well-established theory of modern Hopfield networks. This connection allows us to interpret feedforward memory networks as implementing a cross-attention mechanism. This Hopfield framework unifies memory models such as HebbFF and MPN under a common theoretical foundation, with which we study the learning dynamics of stored patterns.

We introduce the Hetero-Plasticity Network (HPN), which meta-learns decay- and plasticity-rate parameters, to study the effect of heterogeneous synaptic plasticity on the

learning dynamics of stored patterns. We show that the HebbFF network and the MPN are special cases of HPN: the MPN implicitly implements heterogeneous synaptic plasticity, while HebbFF admits a learnable bias in the Hebbian plasticity rule. We provide a theoretical analysis of the linearized learning dynamics of stored patterns in HPN and make three predictions: (1) HPNs with heterogeneous synaptic plasticity outperform those with uniform plasticity, (2) HPNs with uniform plasticity can learn at most two different stored patterns that are symmetric with respect to the origin, and (3) HPNs with heterogeneous plasticity rates outperform those with heterogeneous decay rates. Our empirical results on NeuroGym tasks confirm these predictions.

One of the key insights from our work is the importance of heterogeneous synaptic plasticity. We showed that introducing heterogeneity in decay rates and plasticity rates significantly enhances the network’s ability to retrieve and utilize stored patterns. This finding aligns with biological observations that synaptic plasticity operates over a wide range of timescales and suggests that incorporating such heterogeneity into artificial networks can improve their performance on complex tasks.

Our results elucidate how heterogeneous synaptic plasticity contributes to the network’s ability to retrieve and utilize stored patterns, providing a mechanistic understanding of the performance of different feedforward memory networks.

Despite these advances, several questions remain open for future research. One area of interest is the role of baseline Hebbian plasticity. Although we did not observe a difference in performance between networks that use zero baseline Hebbian plasticity and networks that do not, nonzero baseline Hebbian plasticity is a more biologically plausible choice. Additionally, extending our theory to incorporate recurrent connections and more sophisticated synaptic plasticity rules could further enhance the task-learning capabilities of feedforward memory networks.

6.6 Conclusion

Our theoretical analysis showed that the outputs of a broad class of memory models, which we call FMNs, are linear transformations of modern Hopfield network retrievals and have close relationships to Hopfield theory and cross-attention mechanisms. We further introduced the Hetero-Plasticity Network (HPN) as a subclass of feedforward memory networks, of which MPNs and HebbFFs are special cases, to study the effects of heterogeneous synaptic plasticity on the learning dynamics of stored patterns.

We demonstrated that HPNs with heterogeneous synaptic plasticity outperform those with uniform plasticity, can learn diverse stored patterns, and achieve better performance than networks with heterogeneous decay rates. Our empirical results on NeuroGym tasks confirmed these predictions, highlighting the importance of heterogeneous plasticity in enhancing memory retrieval and task performance.

Our study provides a comprehensive framework for understanding the role of heterogeneous synaptic plasticity in memory networks and builds a theoretical foundation for developing more advanced biological memory models.

Chapter 7

Conclusion

This dissertation brought together five case studies organized into two interconnected lines of research. Part I examined how biological and artificial neural systems model probability distributions and how those learned distributions become visible through neural dynamics, progress estimation, and generation. Part II examined how stochastic gradient descent and input-driven synaptic plasticity shape representations. Together, these complementary studies identify several precise ways in which probabilistic computation depends on internal representation and in which learning determines what those representations make available.

7.1 Synthesis

Part I examined three ways in which neural systems model and learn probability distributions. Chapter 2 treated a distribution as something characterized by the neural dynamics: under the sampling hypothesis, the frequency with which a recurrent circuit visits different states represents probability over time. It thereby took up the gap identified in the introduction between a computational account of Bayesian behavior and a neural mechanism capable of representing the required distribution. By relating stationary sampling to score-function approximation, the chapter showed that sampler-only dynamics are restricted in the distributions they can express, whereas a reservoir-sampler construction can approximate a

substantially richer class under the stated assumptions. The result therefore specifies which recurrent architectures can realize sampling-based neural codes for complex distributions.

Chapters 3 and 4 studied probabilistic modeling in artificial networks from different directions. TOPReward used progress estimation to probe the probability distribution modeled by a pretrained VLM. The systematic change in task-completion probability across trajectory prefixes, together with its dependence on the instruction, provides evidence that the network’s learned distribution reflects latent task progress even without task-specific supervised training. The resulting probability can support progress estimation, success detection, and behavior learning.

dUltra instead studied the distribution and sampling procedure of a masked diffusion language model. Because such a model represents a distribution over complete token sequences without requiring a fixed left-to-right factorization, the order in which positions are unmasked becomes part of generation. dUltra modeled this order as a state-dependent Markov process and learned per-token unmasking probabilities jointly with the token predictor. Its central contribution was to learn how probabilistic language generation should proceed under an accuracy–efficiency tradeoff.

Together, the three chapters in Part I advance probabilistic modeling at three complementary levels. The recurrent-sampling chapter works at the implementation level by identifying neural architectures capable of enacting complex distributions through stochastic dynamics. TOPReward works at the computational level by showing that a pretrained model’s conditional distribution reveals a graded judgment about latent task progress and can turn that judgment into a useful reward. dUltra works at the algorithmic level by learning a sampling policy that decides how parallel generation should allocate inference steps. At each level, probabilistic computation depends on the distinctions made available by the system’s architecture and internal representation.

Part II studies how learning processes shape representation. Chapter 5 considered an artificial network whose representations are produced by stochastic gradient descent. It

connected parameter-space sharpness to local representation-space geometry by deriving upper bounds on Local Volumetric Ratio and Maximum Local Sensitivity, including extensions across intermediate representations. This gives sharpness a specific representational interpretation as a constraint on representation robustness, while treating generalization as a further question determined by the task and data distribution.

Chapter 6 considered the biologically motivated feedforward memory networks, and representations formed by synaptic plasticity. In a feedforward memory network, plastic synaptic weights are both parameters of the current computation and a changing record of prior inputs. The Hopfield and cross-attention formulations identify the memory patterns stored in those weights, while HPN makes it possible to analyze how heterogeneous decay and plasticity rates change the diversity of patterns formed over time.

The two parts meet at the dependence stated in the introduction. A neural encoder maps sensory input into an internal representation, and a recurrent sampler must learn the distribution of those encoded states; the sampler’s architecture therefore limits, and may regularize, which representational distributions its dynamics can realize. In artificial systems, a video-language representation determines which aspects of task progress are accessible to the completion-probability readout, while a masked language model’s representation determines which token predictions can be made together. Part II shows how learning helps produce such distinctions: optimization constrains local feature sensitivity, while online plasticity determines which aspects of a continuous input stream persist as memory patterns.

These results establish a concrete research agenda: explain how biological circuits learn expressive sampling dynamics; determine how pretraining forms progress-sensitive state representations; give parallel generators the ability to revise earlier decisions; identify objectives and optimizers that produce transferable visual representations; and understand how continual experience changes those representations without erasing useful structure. The following sections develop these directions in relation to current work.

7.2 Future directions in probabilistic computation

The main open problem for recurrent neural sampling is learnability. The reservoir-sampler construction provides a denoising score-matching procedure that avoids backpropagation through time, but it leaves open how a biological circuit could compute the required gradients and implement the learning rule [Chen et al., 2023]. A next step is therefore to derive local or approximately local rules for learning the score field and to determine how errors in that field alter the distribution generated by the circuit.

Finite-time dynamics are another possible future direction. Slow mixing delays reliable inference from neural samples [Hennequin et al., 2014]. Models of bistable perception also show that statistics such as dominance durations and switching rates constrain the relative roles of noise and contextual input [Moreno-Bote et al., 2007, Shpiro et al., 2009]. A neural sampler should therefore match not only the long-run fraction of time spent in each interpretation, but also how quickly an interpretation is reached and how often the dynamics move between alternatives.

For TOPReward, the central question is how video-language pretraining produces an instruction-conditioned completion probability that tracks visible progress [Chen et al., 2026a]. Training checkpoints could reveal when this capability emerges [Biderman et al., 2023, Müller-Eberstein et al., 2023, Bayazit et al., 2026]. Comparisons between chronologically ordered and disrupted video could then test whether temporal structure is necessary, as suggested by work on videos involving multiple stages of activities [Che et al., 2026]. Finally, determining whether the model uses an encoded progress signal, rather than merely making that signal decodable, will require causal analysis. A central challenge is designing interventions that modify progress information without disrupting other information in the representation [Canby et al., 2025].

For masked diffusion language models, recent methods allow decoded tokens to be masked and predicted again [Wang et al., 2025b, Yao, 2026a]. dUltra could therefore be extended from an unmasking policy to a revision policy that decides whether each position should be

revealed, retained, or remasked. Training on the model’s own imperfect drafts would better match inference-time errors than random corruption [Yao, 2026b]. The central question is whether learned revision improves the accuracy–speed tradeoff over both unmasking-only decoding and fixed remasking schedules.

7.3 Future directions in representation learning

The central problem of representation learning is pretraining: what should a system represent before its downstream uses are known, and what learning algorithm will produce it? For vision, a useful representation must make object identity, spatial layout, motion, and physical interaction accessible to many possible tasks [Bengio et al., 2013]. Existing approaches reach this goal through different objectives, including language alignment, masked reconstruction, patch-level self-supervision, and latent prediction [Radford et al., 2021, He et al., 2022a, Oquab et al., 2024, Assran et al., 2023]. DINOv2 and V-JEPA 2 demonstrate broad transfer, but the field still needs controlled comparisons that isolate how data, loss objective, architecture, and optimizer determine what the representation retains [Oquab et al., 2024, Assran et al., 2025].

The sharpness chapter provides one starting point. It bounds local representation sensitivity using sharpness-related quantities and reports empirical correlations between the two [Chen et al., 2026b]. A broader pretraining theory should explain how objectives and optimizers jointly create representations that are robust to nuisance variation without losing task-relevant information.

JEPAs offer a concrete family of pretraining objectives: they learn representations by predicting target embeddings from partial observations [Assran et al., 2023, Bardes et al., 2024]. V-JEPA 2 shows that large-scale latent prediction can support visual tasks and, with an action-conditioned predictor, robot planning [Assran et al., 2025]. Future work should determine which prediction targets and horizons produce representations that transfer broadly.

The recurrent sampling project suggests an additional principle: the learned representation should be realizable by the dynamics that use it. A recurrent sampler’s architecture restricts which score fields and stationary latent distributions it can express [Chen et al., 2023]. I call this constraint *dynamical realizability*. It creates a possible connection between JEPAs and recurrent networks: the JEPA objective determines what is predictable in latent space, while the recurrent architecture determines which distributions over those latent states it can generate. Jointly training the encoder and sampler could reveal whether recurrent dynamics regularize the representation and help preserve multiple plausible futures.

Pretraining is not the only process that shapes representation. A complementary problem is how representations should change during continual learning. Representations learned successfully through pretraining can degrade under sequential training [Fini et al., 2022], and learning from a single continuous video stream is complicated by strongly correlated observations [Carreira et al., 2024]. HPN studies a bio-plausible architecture in which plastic weights change within an input sequence and function as memory of recent input. A promising architecture would combine a pretrained visual representation with such a stateful plastic memory. This raises a promising future direction: which information should remain in stable parameters, which should enter temporary memory, and how should the two interact?

7.4 A perspective on biological and artificial intelligence

“What I cannot create, I do not understand.” —Richard P. Feynman

My view is that biological and artificial intelligence should constrain one another. Biology supplies phenomena that theories of natural intelligence must explain, while artificial models turn proposed explanations into systems whose internal variables can be measured and manipulated [Hassabis et al., 2017, Richards et al., 2019]. This exchange is useful precisely because a working model must specify what terms such as prediction, sampling, attention, and memory mean algorithmically and how they are implemented.

Biology also demonstrates forms of learning that remain difficult for artificial systems. Organisms learn from temporally ordered experience, and their synapses change across multiple timescales [Erickson et al., 2010, Bittner et al., 2017, Becker and Tetzlaff, 2021]. These facts motivate models that combine stable knowledge with fast local adaptation. Conversely, artificial systems make proposed mechanisms experimentally separable: a component can be removed or replaced while the training data, objective, and other architectural choices are held fixed. For example, comparisons between plain and residual networks showed that identity shortcuts make substantially deeper networks easier to optimize [He et al., 2016]. Within Chapter 6, comparing uniform and heterogeneous plasticity similarly tests how synaptic diversity changes the memory patterns learned from the same task stream. Such interventions turn a proposed mechanism into a computational claim that can be tested.

Artificial intelligence constrains neuroscience in a complementary way. A verbal claim that the brain predicts, samples, attends, or compresses is too broad to be explanatory. Engineered models force those words to become algorithms with measurable states, objectives, and failure modes. When an artificial sampler requires a particular score field, or an attention-like memory requires identifiable keys and values, the formalization reveals what a proposed biological mechanism would need to provide. Success shows that a mechanism is computationally sufficient under some conditions; failure identifies missing structure. The problem of credit assignment is a central example. Work on predictive-coding, feedback-alignment, dendritic, and equilibrium-based proposals investigates how cortical circuits might approximate the error signals used by backpropagation under biological constraints [Whittington and Bogacz, 2019, Lillicrap et al., 2020]. These proposals recast the question: rather than asking whether the brain runs the textbook algorithm, they ask which locally available signals could allow feedback pathways, dendrites, and synapses to approximate credit assignment.

From a practical perspective, biological networks offer capabilities that artificial systems still seek to reproduce, including energy-efficient computation and continual adaptation. Brain-inspired spiking and event-driven frameworks have accordingly motivated neuromorphic

hardware designed to reduce the energy requirements of artificial intelligence [Roy et al., 2019]. Evolution, however, has assembled biological intelligence from molecular, cellular, circuit, and developmental mechanisms that interact across many scales. Directly imitating the full system would preserve this entanglement rather than reveal which mechanism produces which advantage. I therefore favor constructing a minimal viable artificial model: isolate one biological principle, test whether it is sufficient to yield a desired benefit, and add complexity only when the simpler model fails. The aim is the smallest explicit mechanism that reproduces a benefit such as low-power computation, continual adaptation, or robust memory.

7.5 Closing perspective

Probability describes the multi-modal nature of input data, representation determines the information available for probabilistic computation, and neural dynamics determines how this computation is implemented through inference and learning. The five projects connect these levels in both biological and artificial systems: recurrent architecture constrains neural sampling, VLMs exhibit model belief through token probability, dLLM enables probabilistic language modeling, sharpness induced by learning dynamics improve representation robustness, and heterogeneous plasticity diversifies memory. Together, they motivate a shared agenda for biological and artificial intelligence: learning representations that support a probabilistic model of the world.

Appendix A

Appendix to Chapter 2: Reservoir-Sampler Networks

A.1 Proof of proposition 2.1 via the Helmholtz-Hodge decomposition

First we give a proof for proposition 2.1:

Proposition. *Let p be the stationary distribution of the neural dynamics, and the diffusion matrix be the identity matrix. If the DF field G is strictly orthogonal to the gradient field ∇p , meaning that $G(\mathbf{v}) \cdot \nabla p(\mathbf{v}) = 0$ for all $\mathbf{v}$, then the drift term $F_\theta(\mathbf{v})$ can be written as the sum of a divergence-free field $p^{-1}G$ and a gradient field $\nabla \log p$.*

Proof. Since p is the stationary distribution of the neural dynamics, from the Fokker-Planck equation we have

$$F_\theta(\mathbf{v}) = p^{-1}G + \nabla \log p \tag{A.1}$$

where G is divergence free. Now

$$\nabla \cdot (p^{-1}G) = \nabla p^{-1} \cdot G + p^{-1} \nabla \cdot G$$

$$\begin{aligned}
&= -\frac{\nabla p \cdot G}{p^2} && G \text{ is divergence free} \\
&= 0. && G \text{ and } \nabla p \text{ are pointwise orthogonal}
\end{aligned}$$

Therefore, $p^{-1}G$ is also divergence free. $\square$

Now we give a statement of the Helmholtz-Hodge decomposition, which is an application of the Hodge decomposition theorem to vector calculus on $\mathbb{R}^n$ [Bhatia et al., 2013, Majda and Bertozzi, 2002].

Theorem A.1 (Helmholtz-Hodge Decomposition). *A smooth vector field $\mathbf{V}$ on a bounded domain of $\mathbb{R}^n$ can be uniquely decomposed into the sum of a gradient field ∇u that is normal to the boundary, a divergence-free field $\mathbf{v}$ that is parallel to the boundary and a harmonic field $\mathbf{h}$, i.e. $\mathbf{V} = \nabla u + \mathbf{v} + \mathbf{h}$. Moreover, the three components are orthogonal to each other in the L_2 sense.*

We note that the form of the decomposition in proposition 2.1 coincides with the Helmholtz-Hodge decomposition with a zero harmonic field. Therefore, we can define the orthogonal projection operator $\mathbb{P}$ that projects F_θ onto its gradient component $\nabla \log p$. Under the strict orthogonality constraint that requires G to be orthogonal to ∇p *pointwise*, to be able to sample from a broad range of distributions with $\nabla \log p$ orthogonal to the boundary, we need that for all p , there exists G_p and θ such that $F_\theta(\mathbf{v}) = p^{-1}G_p + \nabla \log p$. Applying the projection operator on both sides, we have

$$\mathbb{P}F_\theta(\mathbf{v}) = \mathbb{P}(p^{-1}G_p + \nabla \log p) = \nabla \log p. \quad (\text{A.2})$$

Moreover, if the function space $\{F_\theta\}_\theta$ admits a set of basis functions $\{e_i\}_i$, i.e. $F_\theta = \sum_i c_i e_i$, then

$$\nabla \log p = \mathbb{P}F_\theta = \sum_i c_i (\mathbb{P}e_i). \quad (\text{A.3})$$

This means that the space of $\nabla \log p$ needs to be spanned by $\{\mathbb{P}e_i\}_i$, so $\{F_\theta\}_\theta$ should also be spanned by an infinite number of basis functions $\{e_i\}_i$.

A.2 Neural dynamics not following gradient fields

For the meaning of notation, see section 2.3.2. The entries d_i of D are positive because they are decay rates, equivalently the reciprocals of positive neuronal time constants. We also adopt the model's no-self-connection assumption $W_{ii} = 0$. Without this architectural assumption, the proposition below would imply only that W is diagonal, since the gradient-field condition constrains the off-diagonal entries but not the diagonal ones.

Proposition A.2. *When $m \geq 2$, $D = \text{diag}(d_1, \dots, d_m)$ with $d_i > 0$, ϕ has a nonconstant derivative, and $W_{ii} = 0$ for all i , $\mathbf{m}_{\text{sc}}(\mathbf{v}) := D(-\mathbf{v} + W\phi(\mathbf{v}) + I)$ is a gradient field if and only if W is a zero matrix.*

Proof. ($\implies$) If $\mathbf{m}_{\text{sc}}(\mathbf{v}) = \nabla G(\mathbf{v})$ is a gradient field, then

$$\frac{\partial}{\partial v_j} \mathbf{m}_{\text{sc}}(\mathbf{v})_i = \frac{\partial}{\partial v_i} \mathbf{m}_{\text{sc}}(\mathbf{v})_j = \frac{\partial}{\partial v_i \partial v_j} G(\mathbf{v}) \quad (\text{A.4})$$

because $\mathbf{m}_{\text{sc}}(\mathbf{v})$ is smooth, and we can switch the order of partial differentiation. Written in the form of (2.7), we have for all v_i and v_j (where $i \neq j$),

$$\begin{aligned} \frac{\partial}{\partial v_j} \left(-d_i v_i + d_i \sum_{k=1}^m W_{ik} \phi(v_k) + d_i I_i \right) &= \frac{\partial}{\partial v_i} \left(-d_j v_j + d_j \sum_{k=1}^m W_{jk} \phi(v_k) + d_j I_j \right) \\ \frac{\partial}{\partial v_j} d_i W_{ij} \phi(v_j) &= \frac{\partial}{\partial v_i} d_j W_{ji} \phi(v_i) \\ d_i W_{ij} \phi(v_j) &= d_j W_{ji} \phi(v_i), \end{aligned}$$

since v_i and v_j are arbitrary and ϕ_v is nonconstant, this implies $d_i W_{ij} = d_j W_{ji} = 0$. Therefore, all off-diagonal entries of W are zero. Together with the assumption that $W_{ii} = 0$ for all i , this implies $W = 0$.

($\Leftarrow$) If $W = 0$, then $\mathbf{m}_{\text{sc}}(\mathbf{v}) = \nabla \sum_i d_i(-v_i^2/2 + I_i v_i)$ is a gradient field. $\square$

A.2.1 Hilbert projection theorem and proof of proposition 2.2

Here we state the Hilbert projection theorem directly. For a detailed proof, we refer readers to standard functional analysis textbooks [Rudin, 1991].

Theorem A.3 (Hilbert projection theorem). *Given a Hilbert space H and a nonempty closed convex set $E \subset H$, for any $x \in H$, there exists a unique $\Pi x \in E$ such that $\|\Pi x - x\| = \inf_{c \in E} \|x - c\|$.*

Here Π is called the orthogonal projection operator. Next, we give the proof of proposition 2.2.

Proof. Here we consider the Hilbert space $L_2(\mathbb{R}^m, \mathbb{R}^m; p)$, where the inner product is defined by $\langle f, g \rangle = \int_{\mathbb{R}^m} f \cdot g p(\mathbf{x}) d\mathbf{x}$, and we let $E = \{A\mathbf{v} + B\phi(\mathbf{v}) + I \mid A, B \in \mathbb{R}^{m \times m}; I \in \mathbb{R}^m\}$. Assume that $v_j, \phi(v_j) \in L_2(p)$ for every $j = 1, \dots, m$, so that $E \subset L_2(\mathbb{R}^m, \mathbb{R}^m; p)$. The set of functions E is a closed convex vector subspace of $L_2(\mathbb{R}^m, \mathbb{R}^m; p)$ because E is closed under addition and finite-dimensional. Hence, by the Hilbert projection theorem, there exists a unique $\Pi H \in E$ such that $\|(1 - \Pi)H\| = \inf_{F \in E} \|H - F\| > 0$. Because $\{\mathbf{m}_{\text{sc}}\}_\theta \subset E$ if the decay matrix D is diagonal, we have

$$\inf_{\mathbf{m}_{\text{sc}}} \|H - \mathbf{m}_{\text{sc}}\| \geq \inf_{F \in E} \|H - F\| = \|(1 - \Pi)H\| > 0. \quad (\text{A.5})$$

$\square$

A.3 Proof of theorem 2.3

First, we give a statement of the universal approximation theorem as in [Leshno et al., 1993].

Theorem A.4. (*Universal Approximation Theorem*) *The continuous transfer function ϕ is not polynomial if and only if for every $k \in \mathbb{N}, m \in \mathbb{N}$, compact set $U \subset \mathbb{R}^k$, $f \in C(U, \mathbb{R}^m)$, and $\varepsilon > 0$, there exists $n \in \mathbb{N}, W_2 \in \mathbb{R}^{n \times k}, b \in \mathbb{R}^n, W_1 \in \mathbb{R}^{m \times n}$ such that*

$$\sup_{x \in U} \|f(x) - g(x)\| < \varepsilon \quad (\text{A.6})$$

where $g(x) = W_1 \phi(W_2 x + b)$. Or equivalently, $\Sigma_n = \text{span}\{\phi(\mathbf{w} \cdot \mathbf{x} + b) : \mathbf{w} \in \mathbb{R}^k, b \in \mathbb{R}\}$ is dense in $C(\mathbb{R}^k)$ with respect to uniform convergence on compact subsets if and only if ϕ is not polynomial.

The theorem says that for any transfer function that is not polynomial (tanh for example), there is a multi-layer neural network with the transfer function that is able to approximate any continuous function to arbitrary precision on a closed and bounded subset of $\mathbb{R}^m$. The proof of theorem 2.3 proceeds in three steps:

1. Choose a large enough compact support U for $s_\theta(\mathbf{x})$ such that

$$\int_{\mathbb{R}^m \setminus U} p(\mathbf{x}) \|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2 d\mathbf{x} = \int_{\mathbb{R}^m \setminus U} p(\mathbf{x}) \|\nabla \log p(\mathbf{x})\|^2 d\mathbf{x} < \frac{\varepsilon}{2} \quad (\text{A.7})$$

2. Choose a large enough n and appropriate θ such that the universal approximation theorem can be applied to show that

$$\int_U p(\mathbf{x}) \|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2 d\mathbf{x} < \frac{\varepsilon}{2} \quad (\text{A.8})$$

3. Combine equation (A.7) and (A.8) so that we have

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2] = \int_{\mathbb{R}^m} p(\mathbf{x}) \|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2 d\mathbf{x} < \varepsilon \quad (\text{A.9})$$

We begin with two lemmas that will be helpful in the proof of the theorem. Both of them are

standard results ¹, but we include here for completeness. Let the upper incomplete Gamma function be defined by $\Gamma(s, C) = \int_C^\infty r^{s-1} e^{-r} dr$.

Lemma A.5. *If s is a positive integer, then $\Gamma(s, C) = (s-1)! e^{-C} \sum_{i=0}^{s-1} \frac{C^i}{i!}$*

Proof. This can be proved by induction. For $s = 1$,

$$\Gamma(1, C) = \int_C^\infty e^{-r} dr = e^{-C} \quad (\text{A.10})$$

Therefore the conclusion holds. For $s > 1$, we assume that for $s = k$, the conclusion holds. Then for $s = k + 1$, Through integration by parts, we have

$$\Gamma(k+1, C) = k\Gamma(k, C) + C^k e^{-C} \quad (\text{A.11})$$

Therefore

$$\begin{aligned} \Gamma(k+1, C) &= k(k-1)! e^{-C} \sum_{i=0}^{k-1} \frac{C^i}{i!} + C^k e^{-C} \\ &= k! e^{-C} \left(\frac{C^k}{k!} + \sum_{i=0}^{k-1} \frac{C^i}{i!} \right) \\ &= k! e^{-C} \sum_{i=0}^k \frac{C^i}{i!} \end{aligned} \quad (\text{A.12})$$

Therefore the conclusion holds for $s = k + 1$ as well. By induction, it holds for all positive integer s . □

Lemma A.6. $\Gamma(s, C) \rightarrow 0$ as $C \rightarrow \infty$.

Proof. When $C \rightarrow \infty$, $\Gamma(s, C) < \int_C^\infty r^{[s-1]} e^{-r} dr$. Therefore W.L.O.G., we can assume that $s - 1$ is a positive integer. By lemma A.5,

$$\lim_{C \rightarrow \infty} \Gamma(s, C) = (s-1)! \sum_{i=0}^{s-1} \lim_{C \rightarrow \infty} \frac{e^{-C} C^i}{i!} = 0 \quad (\text{A.13})$$

¹See <https://dlmf.nist.gov/8.11>.

since the exponential term e^{-C} is beyond all orders. $\square$

Now we give a formal proof of theorem 2.3.

Theorem. *Suppose that we are given a probability distribution with continuously differentiable density function $p(\mathbf{x}) : \mathbb{R}^m \rightarrow \mathbb{R}^+$ and score function $\nabla \log p(\mathbf{x})$ for which there exist constants $M_1, M_2, a, C, k > 0$ such that*

$$p(\mathbf{x}) < M_1 e^{-a\|\mathbf{x}\|} \quad (\text{A.14})$$

$$\|\nabla \log p(\mathbf{x})\|^2 < M_2 \|\mathbf{x}\|^k \quad (\text{A.15})$$

when $\|\mathbf{x}\| > L$ for large enough L . Then for any $\varepsilon > 0$, there exists a recurrent neural network whose firing-rate dynamics are given by (2.11), whose recurrent weights, output weights and the diffusion coefficient are given by $W_{\text{rec}} \in \mathbb{R}^{n \times n}$ of rank at most m , $W_{\text{out}} \in \mathbb{R}^{m \times n}$, and $\sigma \in \mathbb{R}^{n \times m}$ respectively, such that, for a large enough n , the score of the stationary distribution of the output units $s_\theta(\mathbf{x})$ satisfies

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2] < \varepsilon.$$

Proof.

Step 1 First of all, we would like to find a compact set $U_C = \{\mathbf{x} \mid \|\mathbf{x}\|_2 \leq C\}$ such that

$\int_{\mathbb{R}^m \setminus U_C} \|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x} < \frac{\varepsilon}{2}$. We can upper bound the integral by the following calculation using spherical coordinates when $C > L$.

$$\int_{\mathbb{R}^m \setminus U_C} \|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x} = \int_{\mathbb{R}^m \setminus U_C} \|\nabla \log p(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x} \quad (\text{A.16})$$

$$< \int_{\mathbb{R}^m \setminus U_C} M_1 M_2 e^{-a\|\mathbf{x}\|} \|\mathbf{x}\|^k d\mathbf{x} \quad (\text{A.17})$$

$$= \frac{2M_1 M_2 \pi^{m/2}}{\Gamma(\frac{m}{2})} \int_C^\infty r^{m+k-1} e^{-ar} dr \quad (\text{A.18})$$

$$= \frac{2M_1M_2\pi^{m/2}}{a^{m+k}\Gamma(\frac{m}{2})} \int_{aC}^{\infty} u^{m+k-1}e^{-u}du \quad (\text{A.19})$$

$$= \frac{2M_1M_2\pi^{m/2}}{a^{m+k}\Gamma(\frac{m}{2})} \Gamma(m+k, aC) \quad (\text{A.20})$$

Note that we assume $s_\theta(\mathbf{x}) = \mathbf{0}$ outside of U_C ². By lemma A.6, there exists a large enough constant $\tilde{C}$ such that $\Gamma(m+k, a\tilde{C}) < \frac{a^{m+k}\Gamma(\frac{m}{2})\varepsilon}{4M_1M_2\pi^{m/2}}$. Therefore

$$\int_{\mathbb{R}^m \setminus U_{\tilde{C}}} \|\nabla \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x} < \frac{2M_1M_2\pi^{m/2}}{a^{m+k}\Gamma(\frac{m}{2})} \cdot \frac{a^{m+k}\Gamma(\frac{m}{2})\varepsilon}{4M_1M_2\pi^{m/2}} = \frac{\varepsilon}{2} \quad (\text{A.21})$$

Step 2 Next we write down the specific form of $s_\theta(\mathbf{x})$, the score of the stationary distribution of $\mathbf{x}$ for the reservoir-sampler dynamics (2.11)

$$s_\theta(\mathbf{x}) = 2(W_{\text{out}}\sigma\sigma^TW_{\text{out}}^T)^{-1}(-\alpha\mathbf{x} + \alpha W_{\text{out}}\phi(\tilde{W}_{\text{rec}}\mathbf{x} + I)). \quad (\text{A.22})$$

Since σ is learnable, we can define $\sigma = W_{\text{out}}^T(W_{\text{out}}W_{\text{out}}^T)^{-1}$ such that $W_{\text{out}}\sigma\sigma^TW_{\text{out}}^T$ is an identity matrix³:

$$\begin{aligned} W_{\text{out}}\sigma\sigma^TW_{\text{out}}^T &= W_{\text{out}}W_{\text{out}}^T(W_{\text{out}}W_{\text{out}}^T)^{-1}(W_{\text{out}}W_{\text{out}}^T)^{-1}W_{\text{out}}W_{\text{out}}^T \\ &= \mathbf{I} \end{aligned} \quad (\text{A.23})$$

As p is continuously differentiable, by theorem A.4, there exists $W_{\text{out}}, \tilde{W}_{\text{rec}}$ such that

$$\sup_{\mathbf{x} \in U_{\tilde{C}}} \left\| W_{\text{out}}\phi(\tilde{W}_{\text{rec}}\mathbf{x} + I) - \left(\mathbf{x} + \frac{\nabla \log p(\mathbf{x})}{2\alpha} \right) \right\|^2 < \frac{\varepsilon}{8\alpha^2} \quad (\text{A.24})$$

²Equality (A.16) is the only place where this assumption is used. A similar tail bound can also be obtained if $s_\theta(\mathbf{x})$ grows at a comparable rate to $\nabla \log p(\mathbf{x})$.

³A technical subtlety is that $W_{\text{out}}W_{\text{out}}^T$ is not necessarily invertible. However, since every transformation considered here is continuous, we could add a small perturbation to W_{out} such that $W_{\text{out}}W_{\text{out}}^T$ is invertible, and our result still holds.

Therefore with those choices of W_{out} , $\widetilde{W}_{\text{rec}}$ and σ , we have

$$\begin{aligned}
\int_{U_{\widetilde{C}}} \|\nabla \log p(\mathbf{x}) - s_{\theta}(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x} &< \sup_{\mathbf{x} \in U_{\widetilde{C}}} \|\nabla \log p(\mathbf{x}) - s_{\theta}(\mathbf{x})\|^2 \\
&= \sup_{\mathbf{x} \in U_{\widetilde{C}}} \left\| \nabla \log p(\mathbf{x}) + 2\alpha\mathbf{x} - 2\alpha W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}\mathbf{x} + I) \right\|^2 \\
&= 4\alpha^2 \sup_{\mathbf{x} \in U_{\widetilde{C}}} \left\| \frac{\nabla \log p(\mathbf{x})}{2\alpha} + \mathbf{x} - W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}\mathbf{x} + I) \right\|^2 \\
&< 4\alpha^2 \cdot \frac{\varepsilon}{8\alpha^2} \\
&= \frac{\varepsilon}{2}
\end{aligned} \tag{A.25}$$

Step 3

$$\begin{aligned}
\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\|\nabla \log p(\mathbf{x}) - s_{\theta}(\mathbf{x})\|^2] &= \left(\int_{\mathbb{R}^m \setminus U_{\widetilde{C}}} + \int_{U_{\widetilde{C}}} \right) p(\mathbf{x}) \|\nabla \log p(\mathbf{x}) - s_{\theta}(\mathbf{x})\|^2 d\mathbf{x} \\
&< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon \quad \square
\end{aligned}$$

A.4 Pseudocode for training the reservoir-sampler network

A.5 Accelerated sampling with reservoir-sampler network

To construct an irreversible dynamics that accelerates sampling, we let the divergence-free (DF) field G be of the form $J\nabla p$, where $J = -J^T$ can be any skew-symmetric matrix that is either learned through training or prescribed. Then from equation (2.6) we see that the drift

Algorithm 1: Training RSN

Input: Training samples $\{\mathbf{x}_i\}$, pseudo recurrent weights $\widetilde{W}_{\text{rec}}$, output weights W_{out} , start noise level σ_1 , noise decay factor C , number of noise levels N

Output: trained recurrent weights W_{rec} , trained output weights W_{out} , trained diffusion coefficient σ

for $i = 1, \dots, N$ **do**

- while** *not done* **do**
 - Sample $\mathbf{x}$ from the training set
 - Perturb the sample with Gaussian noise $\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma_i^2 \mathcal{I})$
 - Compute the gradient of the score-matching loss
$$\frac{1}{2} \nabla_{\widetilde{W}_{\text{rec}}, W_{\text{out}}} \left\| 2\alpha(W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}\tilde{\mathbf{x}} + I) - \tilde{\mathbf{x}}) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma_i^2} \right\|^2$$
 - Update $\widetilde{W}_{\text{rec}}, W_{\text{out}}$ with gradient descent
- $\sigma_{i+1} = C\sigma_i$

$W_{\text{rec}} \leftarrow \widetilde{W}_{\text{rec}} W_{\text{out}}$

$\sigma \leftarrow W_{\text{out}}^T (W_{\text{out}} W_{\text{out}}^T)^{-1}$

return $W_{\text{rec}}, W_{\text{out}}$

term F_θ satisfies

$$F_\theta(\mathbf{v}) = \Sigma \nabla \log p(\mathbf{v}) + J \nabla \log p(\mathbf{v}) = (\Sigma + J) \nabla \log p(\mathbf{v}). \quad (\text{A.26})$$

Therefore the corresponding score matching problem is $\alpha(\Sigma + J)^{-1}(W_{\text{out}}\phi(\widetilde{W}_{\text{rec}}x + I) - x) \approx \nabla \log p(x)$. With a slight modification, the pseudo code for an accelerated RSN training algorithm is given in algorithm 2. We repeat the experiment in section 2.4.1 with the accelerated RSN. The results are shown in Figure A.1. We see that the accelerated RSN is able to sample from the distribution with a smaller noise level than the RSN.

Algorithm 2: Training accelerated RSN

Input: Training samples $\{\mathbf{x}_i\}$, pseudo recurrent weights $\widetilde{W}_{\text{rec}}$, output weights W_{out} , pseudo diffusion coefficient $\tilde{\sigma}$, skew-symmetric matrix J , start noise level σ_1 , noise decay factor C , number of noise levels N

Output: trained recurrent weights W_{rec} , trained output weights W_{out} , trained diffusion coefficient σ

for $i = 1, \dots, N$ **do**

while *not done* **do**

 Sample $\mathbf{x}$ from the training set

 Perturb the sample with Gaussian noise $\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma_i^2 \mathcal{I})$

 Compute the gradient of the score-matching loss

$$\frac{1}{2} \nabla_{\widetilde{W}_{\text{rec}}, W_{\text{out}}, \tilde{\sigma}, J} \left\| \alpha \left(\frac{1}{2} \tilde{\sigma} \tilde{\sigma}^T + J \right)^{-1} (W_{\text{out}} \phi(\widetilde{W}_{\text{rec}} \tilde{\mathbf{x}} + I) - \tilde{\mathbf{x}}) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma_i^2} \right\|^2$$

 Update $\widetilde{W}_{\text{rec}}, W_{\text{out}}, \tilde{\sigma}$ with gradient descent

 Update J with gradient descent (Optional)

$\sigma_{i+1} = C \sigma_i$

$W_{\text{rec}} \leftarrow \widetilde{W}_{\text{rec}} W_{\text{out}}$

$\sigma \leftarrow W_{\text{out}}^T (W_{\text{out}} W_{\text{out}}^T)^{-1} \tilde{\sigma}$

return $W_{\text{rec}}, W_{\text{out}}$

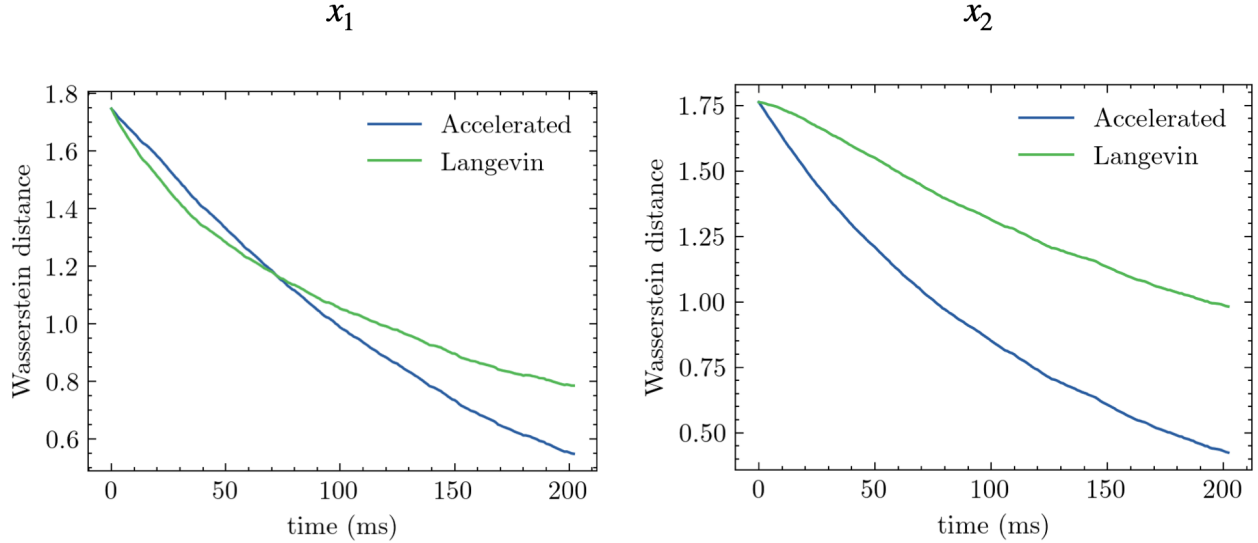


Figure A.1: **Sampling speed of accelerated RSN** We repeat the experiment described in section 2.4.1 where we try to sample from a 2-D Laplacian mixture distribution. The figure shows the Wasserstein distance between the marginalized distributions of generated samples and true samples for both dimensions. We see that compared to vanilla RSN, accelerated RSN is able to sample from the stationary distribution with faster relaxation time.

A.6 Additional experiments

A.6.1 Failure cases for sampler-only networks with hyperbolic tangent nonlinearity

Figure A.2 shows that with $\tanh$ nonlinearity, sampler-only networks are able to sample from the distribution we consider in section 2.4.1. Its exact score is $-4x + 4 \tanh(4x)$, which SO-FR can represent because its inner weight is tunable, while SO-SC only approximates it over the plotted range using its fixed basis $\{x, \tanh(x), 1\}$. More generally, our theory predicts that each sampler-only network cannot approximate score functions outside its respective finite-dimensional function space. Indeed, if we consider a slightly different Gaussian mixture distribution, $p_{\text{data}}(x) = \frac{1}{2}(\mathcal{N}(-1, 0.0625) + \mathcal{N}(1, 0.25))$, then neither sampler-only network recovers the score function as shown in Figure A.3, while RSN still succeeds in recovering the score function.

A.6.2 Sampling using autoencoder

We show that we can replace the fixed linear PCA projection used in section 2.4.2 with pretrained nonlinear autoencoder. We test the algorithm on both the MNIST dataset (figure A.4) and the CIFAR-10 dataset (figure A.5).

A.7 Related work

In this section, we examine the neural dynamics considered in [Dong et al., 2022, Aitchison and Lengyel, 2016, Masset et al., 2022, Echeveste et al., 2020] and discuss how our work is related.

Dong et al. [2022] uses Hamiltonian Langevin dynamics to accelerate sampling in continuous attractor networks. The Hamiltonian dynamics below are considered in the chapter (cf.

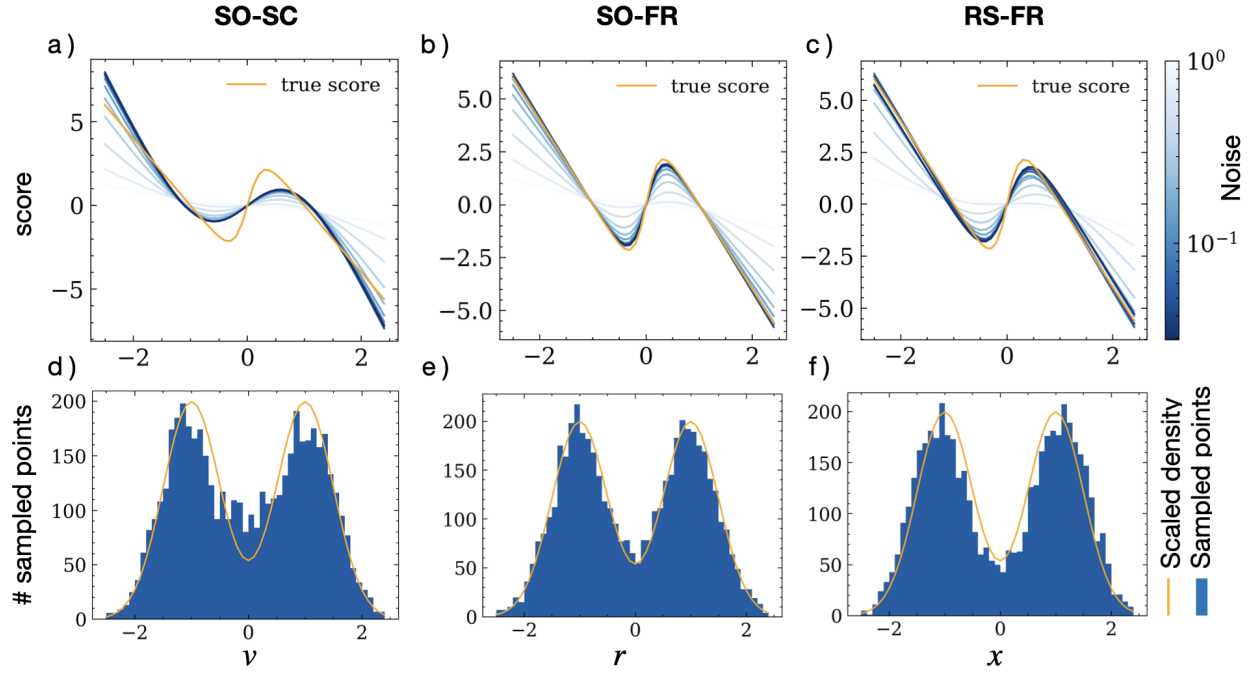


Figure A.2: **Networks with hyperbolic tangent nonlinearity applied to a 1-D Gaussian mixture distribution with a symmetric score function.** See Figure 2.2 for an explanation of the abbreviations. Here we use $\tanh$ nonlinearity for sampler-only networks (SO-SC and SO-FR). SO-FR and RS-FR can represent the exact score, while SO-SC approximates it over the plotted range.

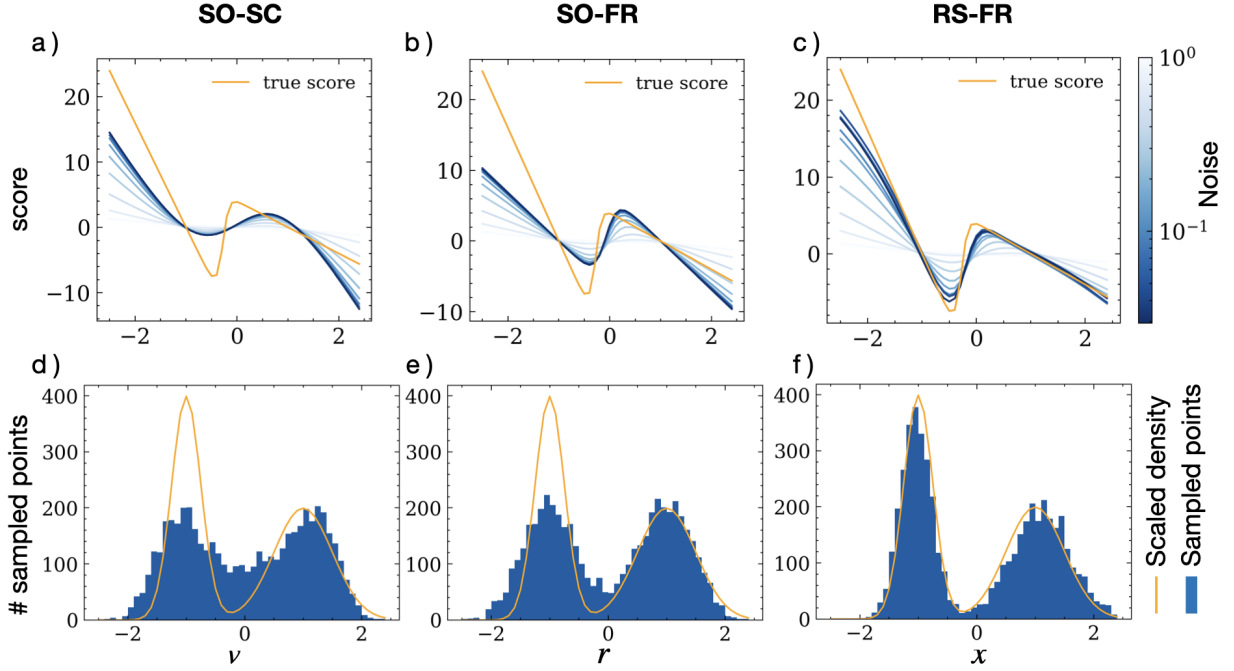


Figure A.3: **Sampler-only network with hyperbolic tangent nonlinearity fails to sample from a 1-D Gaussian mixture distribution with a non-symmetric score function.** See Figure 2.2 for an explanation of the abbreviations. Here we use $\tanh$ nonlinearity for sampler-only networks (SO-SC and SO-FR). Only RSN is able to match the score function and sample from the distribution.

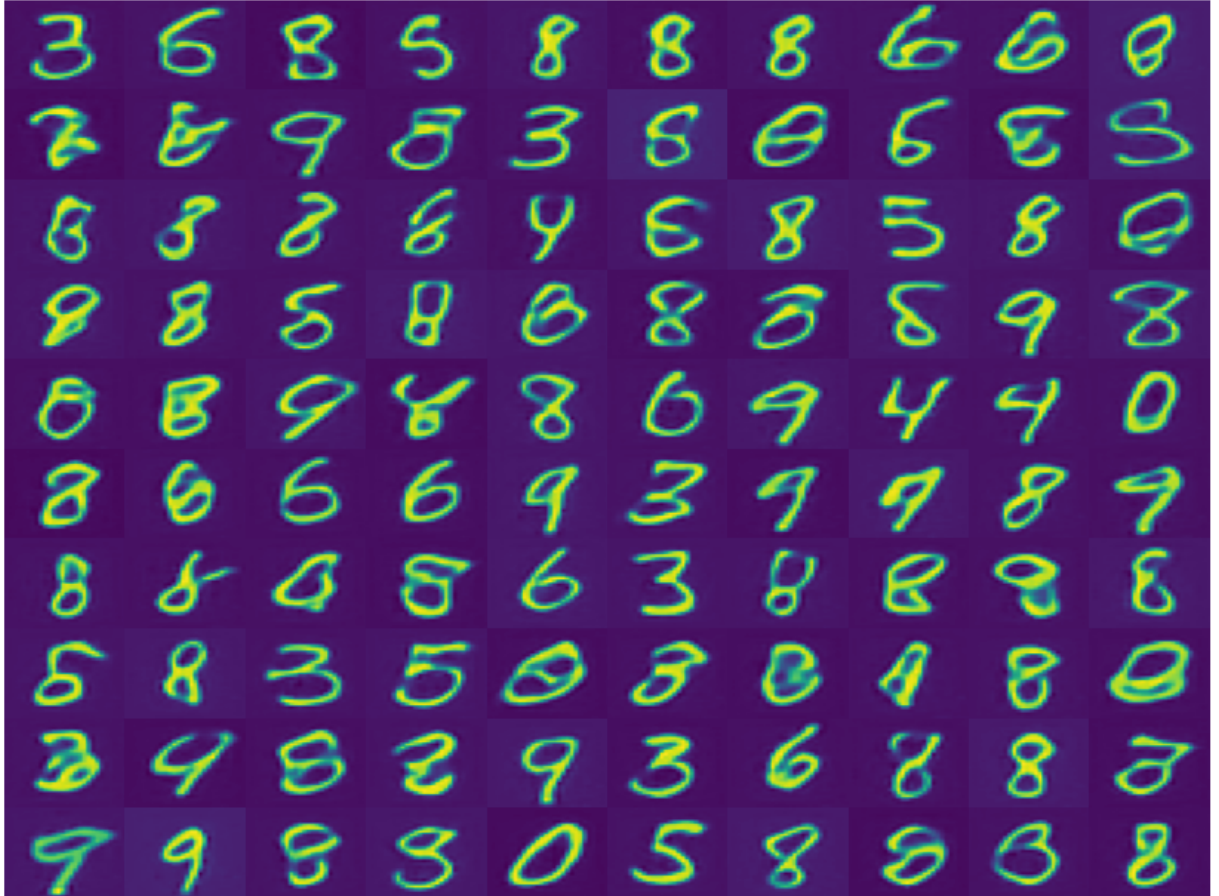


Figure A.4: Samples produced by our reservoir sampler network when trained to fit the latent distribution produced by a non-linear convolution autoencoder from MNIST. The distribution was 32-dimensional. The reservoir sampler network had 500 reservoir neurons.



Figure A.5: Samples produced by our reservoir sampler network when trained to fit the latent distribution produced by a non-linear convolution autoencoder from CIFAR-10. We show both the original images and the reconstructed images to evaluate the sampling quality. The distribution was 128-dimensional. The reservoir sampler network had 20000 reservoir neurons.

eq (19) and (20)):

$$\begin{cases} \tau_s \frac{ds}{dt} &= \boldsymbol{\alpha} \mathbf{y} \\ \tau_z \frac{dz}{dt} &= -\boldsymbol{\beta} \boldsymbol{\alpha}^{-1} \mathbf{y} + \Lambda(\mathbf{s}^\circ - \mathbf{s}) + \sqrt{\tau_z} \boldsymbol{\sigma}_y \boldsymbol{\xi}_t. \end{cases} \quad (\text{A.27})$$

The overdamped Langevin dynamics that samples from the same marginal stationary distribution over $\mathbf{s}$ as (A.27):

$$\frac{d\mathbf{s}}{dt} = \Lambda(\mathbf{s}^\circ - \mathbf{s}) + \sqrt{2} \boldsymbol{\xi}_t, \quad (\text{A.28})$$

where $\mathbf{s}^\circ$ is an observation generated by a latent feature $\mathbf{s}$ according to a Gaussian distribution, and the linear term $\Lambda(\mathbf{s}^\circ - \mathbf{s})$ comes from the external input. So the network dynamics in Dong et al. [2022] themselves do not generate samples without the external input, and the study does not clarify how to supply the network with nonlinear score functions.

Aitchison and Lengyel [2016] have shown that Hamiltonian Monte Carlo (HMC) naturally maps onto the dynamics of excitatory-inhibitory neural networks. However, the neural dynamics considered are entirely linear in terms of the network variables (cf. eq (23) and (24)). Therefore, the dynamics are again equivalent to the synaptic current dynamics with an identity transfer function in section 2.3.2. The same applies to the analysis in Masset et al. [2022], which also only considers linear neural dynamics. By proposition 2.2 and the decomposition discussed in Kwon et al. [2005], these dynamics can only sample from probability distributions whose score functions are linear, i.e. Gaussians.

The work by Echeveste et al. [2020] provides a unifying model for several dynamical phenomena in sensory cortices by training a nonlinear recurrent excitatory-inhibitory neural circuit model to perform sampling-based probabilistic inference. The following synaptic neural dynamics (cf. equation (8)) are used, where the nonlinear transfer function is supralinear, i.e., $\phi(u) = k[u]^n$:

$$\tau_i \frac{du_i}{dt} = -u_i + \sum_{j=1}^N w_{ij} \phi(u_j) + h_i(t) + \eta_i(t). \quad (\text{A.29})$$

As discussed in section 2.3.2, while they are richer than linear dynamics, dynamics of this form still have limited ability to sample from complex probability distributions. Moreover, we

note that the training criterion considered in Echeveste et al. [2020] only matches the mean and the variance of the stationary distribution. These points are in contrast to our work here, where we show that the reservoir-sampler network, which extends the dynamics (A.29), can be trained via alternate methods to sample from complex probability distributions by matching their score functions.

A.8 Details of numerical experiments

All experiments were run on one NVIDIA Quadro RTX 6000 GPU. We always use the model saved at the last training iteration.

A.8.1 Training details for section 2.4.1

For the 1-D bimodal Gaussian distribution experiment (figure 2.2), we used 1000 reservoir neurons (only in the reservoir sampler network) and 1 sampler neuron. We trained all 3 networks (SO-FR, SO-SC, RS-FR) for 79000 iterations with the Adam optimizer. The learning rate was 0.0001, and the batch size was 128. We started by adding Gaussian noise with standard deviation (std) $\sigma_1 = 1$ to the training samples. We decreased the std every 7900 iterations by a constant factor, i.e. $\sigma_{i+1}/\sigma_i = C$, such that $\sigma_{10} = .01$. When simulating the stochastic neural dynamics, we used a step size of 10^{-4} , and ran 10000 steps.

For the mixture of 2-D Laplace distributions experiment (figure 2.3), we used 1000 reservoir neurons (only in the reservoir sampler network) and 2 sampler neurons. We trained the RS-FR network for 62,500 iterations with the Adam optimizer. The learning rate was 0.0001, and the batch size was 128. We started by adding Gaussian noise with std $\sigma_1 = 1$ to the training samples. We decreased the std every 6250 iterations by a constant factor, i.e. $\sigma_{i+1}/\sigma_i = C$, such that $\sigma_{10} = .01$. When simulating the stochastic neural dynamics, we used a step size of 10^{-4} , and ran 20000 steps.

A.8.2 Training details for section 2.4.2

For the MNIST experiment (Figure 2.2), we used 20000 reservoir neurons (only in the reservoir sampler network) and 300 sampler neurons. We trained all 3 networks (SO-FR, SO-SC, RS-FR) for 1000 epochs with the Adam optimizer. The learning rate was 0.001, and the batch size was 64. We started by adding Gaussian noise with std $\sigma_1 = 1$ to the training samples. We decreased the std every 100 epochs by a constant factor, i.e. $\sigma_{i+1}/\sigma_i = C$, such that $\sigma_{10} = .01$. When simulating the stochastic neural dynamics, we used a step size of 10^{-6} , and ran 30000 steps.

A.9 Statement of code availability

All code is available on GitHub.

Appendix B

Appendix to Chapter 3: TOPReward

B.1 Alternative Reward Formulation

In addition to the main formulation of TOPReward presented in Section 3.3, we also experimented with an alternative reward formulation that evaluates the probability of generating the entire instruction given the video trajectory. Specifically, we construct the following prompt,

“<|video|> The above video shows a robot manipulation trajectory that completes the following task: {instruction}.”

We then define the reward for a video prefix $\tau_{1:t}$ to be

$$r_t = \sum_i \log p_\theta(\mathbf{inst}_i \mid c(\tau_{1:t}, u, \mathbf{inst}_{<i})), \quad (\text{B.1})$$

where $\mathbf{inst}_i$ is the i -th token of the instruction, and u represents the prompt between the video and the instruction. However, we found this alternative formulation to be less effective than the main formulation presented in Section 3.3. We hypothesize that this is because the model will assign high probability to entities in the instruction if they ever appear in the robot video trajectory. For example, if there is an apple in the video, and the instruction is

to peel the apple, then the model will assign high probability to **apple** in the instruction, defeating the purpose of progress estimation. In contrast, our main formulation only requires the model to judge whether the trajectory completes the instruction, which prevents such distraction in probability evaluation.

B.2 Why the True Token?

We choose **True** as the affirmative completion token rather than alternatives (e.g., **Yes**) because it is a single token in our evaluated vocabularies and yields the largest, most consistent separation between successful and failed trajectories at the final step. Figure B.1 shows the top tokens by absolute difference in mean final-step token probability; **True** exhibits the largest gap.

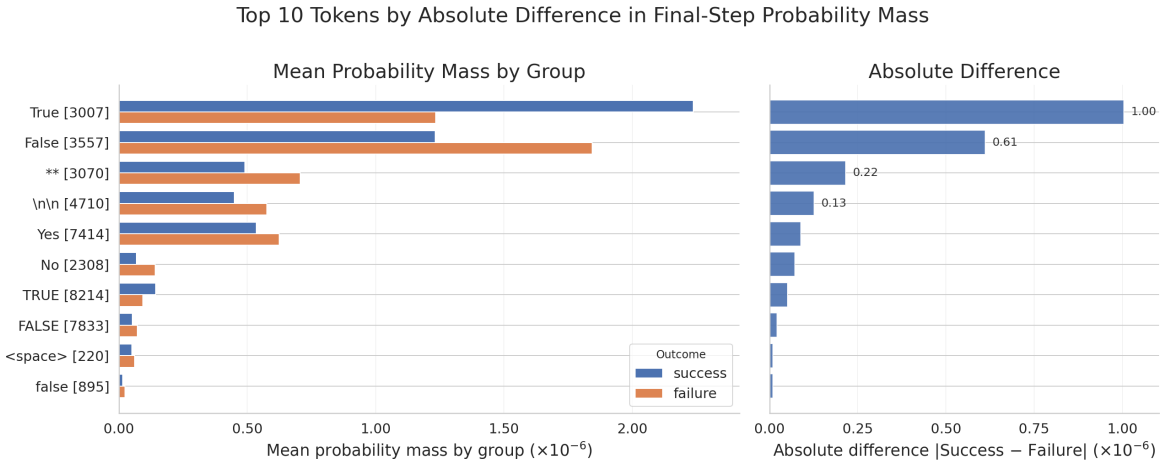


Figure B.1: Top 10 tokens by absolute difference in mean final-step token probability between successful and failed trajectories. The affirmative token **True** shows the largest separation, motivating its use as the completion token in TOPReward. Left: mean token probability by group; right: absolute difference in mean token probability.

B.3 Success Detection Score

For success detection, we convert TOPReward token probabilities into trajectory-level completion scores. Let $r_{t_k}^+ = \log p_\theta(\text{True} \mid c(\tau_{1:t_k}, x))$ and $r_{t_k}^- = \log p_\theta(\text{False} \mid c(\tau_{1:t_k}, x))$ be the

completion and non-completion log-probabilities for the k -th sampled prefix, and define the prefix completion margin $m_{t_k} = r_{t_k}^+ - r_{t_k}^-$. We first measure how much this margin increases from the beginning to the end of the trajectory,

$$s_{\text{top}}(\tau, x) = \frac{1}{|\mathcal{L}|} \sum_{k \in \mathcal{L}} m_{t_k} - \frac{1}{|\mathcal{E}|} \sum_{k \in \mathcal{E}} m_{t_k}, \quad (\text{B.2})$$

where $\mathcal{E}$ and $\mathcal{L}$ denote the first and last three sampled prefixes, respectively. We also score the full video with a direct binary completion query,

$$s_{\text{yn}}(\tau, x) = \log p_{\theta}(\text{Yes} \mid c_{\text{yn}}(\tau, x)) - \log p_{\theta}(\text{No} \mid c_{\text{yn}}(\tau, x)), \quad (\text{B.3})$$

where c_{yn} is the full-video prompt asking whether the trajectory completed the instruction. We use a fixed equal-weight combination of the standardized scores,

$$s_{\text{succ}}(\tau, x) = s_{\text{top}}(\tau, x) + s_{\text{yn}}(\tau, x), \quad (\text{B.4})$$

This uses no learned classifier or fitted task-specific reward model.

B.4 Dataset-level breakdown

This section provides dataset-level details that complement the aggregate results in Tables 3.1 and 3.2. Figure B.2 summarizes dataset-level VOC, and Table B.1 reports dataset-level VOC for GVL (0-shot) and TOPReward (TOPR) for each dataset and model backbone, using the same 33 high-quality OXE datasets as the main table.

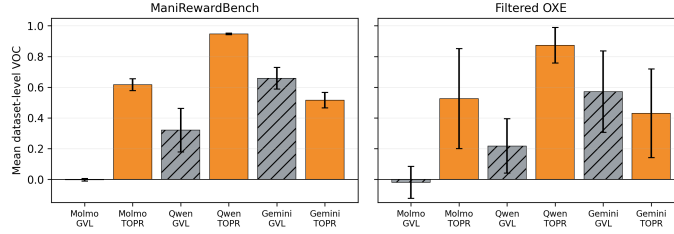


Figure B.2: **VOC comparison across datasets.** Mean dataset-level VOC for GVL (0-shot) and TOPReward across two evaluation sets: filtered OXE (33 datasets, 20 episodes each) and the ManiRewardBench successful progress-evaluation subset (4 datasets, 113 tasks, 497 episodes). Error bars denote standard deviation across datasets within each evaluation set.

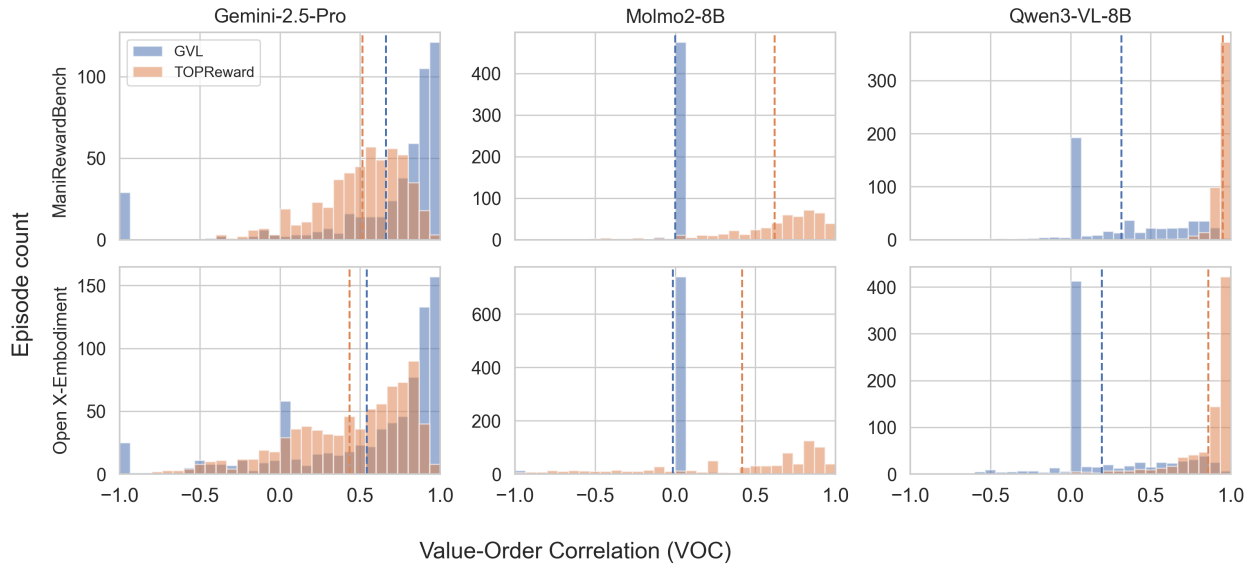


Figure B.3: Per-episode VOC distributions, broken down by evaluation set (ManiReward Bench vs Open X-Embodiment) and model backbone.

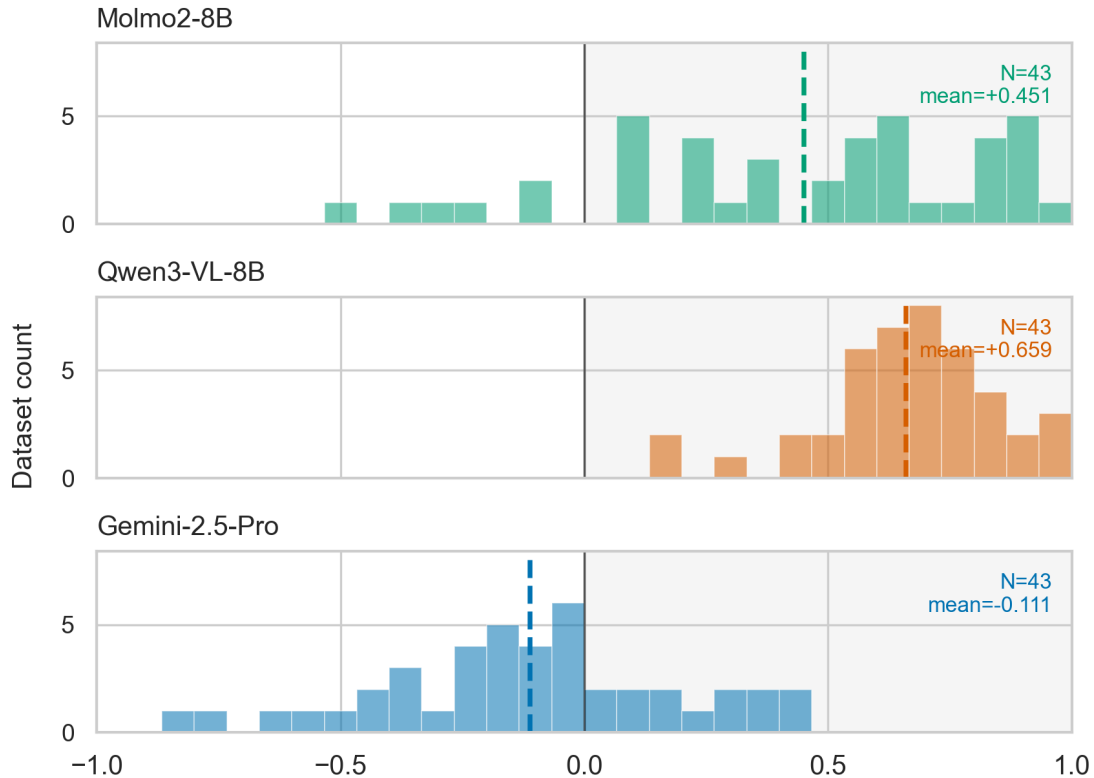


Figure B.4: Distribution of dataset-level $\Delta\text{VOC} = \text{VOC}(\text{TOPReward}) - \text{VOC}(\text{GVL})$, shown separately for each model backbone. Positive values indicate **TOPReward** outperforms GVL; the dashed line marks the per-model mean.

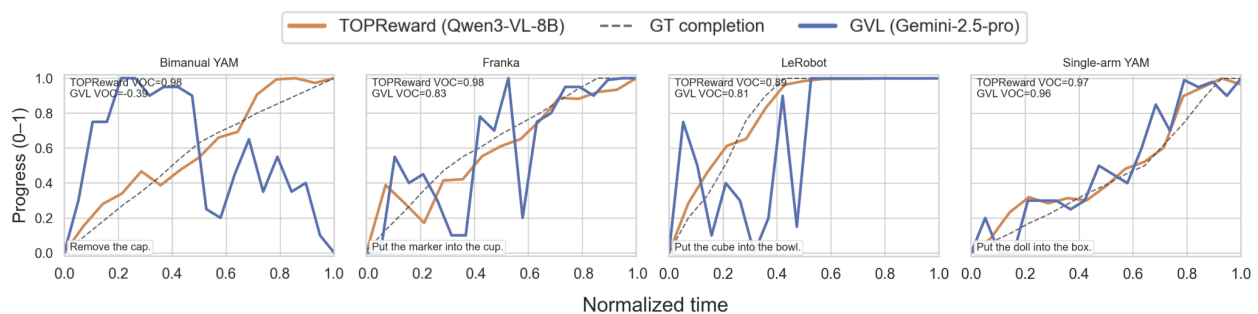


Figure B.5: **Progress traces for ManiRewardBench.** Example progress traces predicted by **TOPReward** (orange) compared to stage-aware ground-truth completion (dashed) from **ManiRewardBench**, computed from annotated subtask boundaries. We also overlay Gemini-GVL (blue) on the same episodes when available.

Dataset	Qwen3-VL-8B			Gemini-2.5-Pro			Molmo2-8B		
	GVL	TOPR	Δ	GVL	TOPR	Δ	GVL	TOPR	Δ
ManiRewardBench									
ManiRewardBench_bimanual_yam	0.164	0.947	0.783	0.566	0.546	-0.021	0.007	0.565	0.558
ManiRewardBench_franka	0.242	0.942	0.700	0.695	0.448	-0.247	0.000	0.662	0.662
ManiRewardBench_lerobot	0.332	0.954	0.622	0.620	0.578	-0.041	-0.001	0.595	0.597
ManiRewardBench_single_yam	0.544	0.945	0.401	0.752	0.488	-0.264	-0.017	0.642	0.659
Open X-Embodiment									
aloha_mobile_cabinet	0.199	0.822	0.623	0.814	0.422	-0.392	0.000	0.211	0.211
aloha_mobile_wash_pan	0.127	0.977	0.850	0.856	0.734	-0.121	0.012	0.893	0.880
aloha_mobile_wipe_wine	0.243	0.961	0.718	0.699	0.825	0.125	0.000	0.804	0.804
aloha_static_candy	0.414	0.963	0.549	0.697	0.803	0.105	0.000	0.842	0.842
aloha_static_coffee	0.101	0.977	0.876	0.877	0.114	-0.764	0.012	0.400	0.388
aloha_static_cups_open	0.271	0.837	0.565	0.524	0.765	0.241	0.000	0.864	0.864
aloha_static_pro_pencil	0.000	0.963	0.963	0.457	-0.029	-0.486	0.000	0.225	0.225
aloha_static_screw_driver	0.212	0.937	0.725	0.833	0.171	-0.662	0.000	0.925	0.925
aloha_static_vinh_cup	0.332	0.916	0.584	0.836	0.723	-0.113	0.000	0.971	0.971
aloha_static_vinh_cup_left	-0.057	0.903	0.960	0.458	0.898	0.441	-0.050	0.707	0.757
aloha_static_ziploc_slide	0.301	0.978	0.677	0.845	0.394	-0.452	0.000	0.918	0.918
austin_buds_dataset	0.347	0.954	0.607	0.777	-0.055	-0.832	0.000	0.908	0.908
austin_sirius_dataset	0.242	0.854	0.612	0.704	0.701	-0.003	0.000	0.119	0.119
berkeley_fanuc_manipulation	0.201	0.866	0.665	0.422	0.333	-0.089	0.000	-0.077	-0.077
berkeley_rpt	0.399	0.983	0.584	0.303	0.600	0.297	0.000	0.605	0.605
berkeleymvp	0.240	0.966	0.727	0.744	0.564	-0.180	0.000	0.472	0.472
cmu_franka_exploration_dataset	0.000	0.626	0.626	0.622	0.272	-0.350	0.000	0.325	0.325
d1r_edan_shared_control	0.166	0.867	0.700	0.651	0.583	-0.068	-0.600	0.770	1.370
d1r_sara_grid_clamp	0.000	0.740	0.740	0.253	-0.166	-0.419	0.000	0.562	0.562
jaco_play	0.090	0.907	0.818	0.513	0.508	-0.005	0.000	-0.299	-0.299
kaist_nonprehensile	0.158	0.914	0.756	0.491	0.349	-0.142	0.000	0.251	0.251
nyudoor	0.608	0.802	0.194	0.872	0.729	-0.142	0.000	0.650	0.650
nyufranka	0.232	0.875	0.642	0.772	0.494	-0.278	0.059	0.865	0.806
stanford_hydra_dataset	0.164	0.973	0.809	0.379	0.557	0.178	0.000	0.091	0.091
stanford_kuka_multimodal_dataset	0.122	0.821	0.700	-0.390	-0.055	0.335	0.000	0.493	0.493
stanford_robocook	0.035	0.443	0.408	0.329	0.521	0.191	0.000	0.582	0.582
taco_play	0.015	0.779	0.764	0.050	0.024	-0.026	0.000	0.342	0.342
tokyo_u_lsno	0.215	0.977	0.762	0.690	0.501	-0.188	0.000	0.719	0.719
ucsd_kitchen_dataset	0.183	0.718	0.536	0.542	-0.006	-0.547	0.000	0.228	0.228
ucsd_pick_and_place_dataset	0.090	0.819	0.729	0.683	0.520	-0.163	0.000	0.605	0.605
utokyo_pr2_opening_fridge	0.485	0.957	0.472	0.372	0.644	0.272	-0.075	0.849	0.924
utokyo_pr2_tabletop_manipulation	0.784	0.946	0.162	0.696	0.485	-0.211	0.000	0.121	0.121
utokyo_xarm_bimanual	0.266	0.815	0.549	0.495	0.277	-0.218	0.000	0.386	0.386

Table B.1: Filtered datasets (alphabetical) comparing GVL (0-shot) vs TOPReward (TOPR) and their difference, broken down by model backbone. Bold indicates the higher VOC between the two methods for that dataset/model.

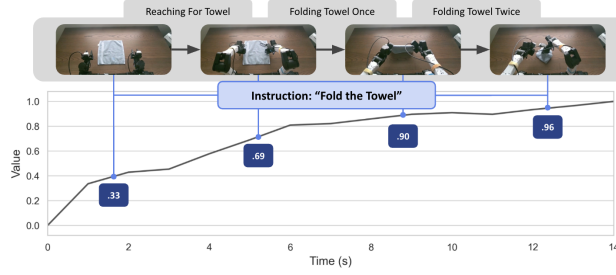


Figure B.6: **Qualitative example of “Fold the Towel”**: Instruction-conditioned progress estimation on a real trajectory. The curve shows TOPReward’s predicted completion value over time, with annotated values at selected frames corresponding to semantic subtasks.

B.5 Additional qualitative results and ablations

B.5.1 Real-world policy details

For the reward-weighted behavior cloning objective in Equation (3.5), we use $\beta = 0.2$ and $\delta_{\max} = 2.0$ in Equation (3.3).

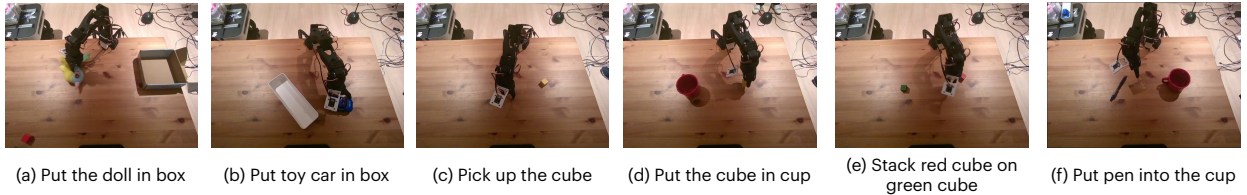


Figure B.7: The six real-world single-arm SO-100 manipulation tasks used for reward-weighted behavior cloning evaluation.

B.5.2 Chat-template ablation

The Gemini API enforces a chat template, while our preferred open-source inference setting scores the prompt directly without one. To test this factor, we wrap the prompt in Section 3.3.1 with a chat template and evaluate the probability of the answer being **True**. Table B.2 shows that chat formatting substantially reduces VOC for both Qwen3-VL-8B and Molmo2-8B.



Figure B.8: **Qualitative comparison on “Place doll in box.”** The pretrained policy and behavior cloning (BC) both fail, while TOP-RWBC, fine-tuned with reward weights from TOPReward, succeeds consistently. Frames are uniformly sampled from evaluation rollouts.

Table B.2: **Effect of Chat Template on TOPReward VOC.** Chat template degrades Qwen3-VL-8B performance by nearly 50% and Molmo2-8B by 20%, demonstrating that the logit-based formulation is sensitive to prompt formatting.

Dataset	Qwen3-VL-8B		Molmo2-8B	
	Base	+Chat	Base	+Chat
Bimanual YAM	0.947	0.269	0.570	0.408
Franka	0.943	0.528	0.696	0.615
Single-arm YAM	0.946	0.703	0.691	0.546
Mean	0.945	0.500	0.652	0.523
Δ	−47.1%		−19.8%	

B.6 Additional details of ManiRewardBench

The benchmark includes 130 diverse tasks that capture a wide range of everyday manipulation activities, such as stacking objects, sorting items, and interacting with containers. The dataset is manually collected using *Franka*, *SO-100/101*, *bimanual YAM*, and *single-arm YAM*.

Example challenging tasks in ManiRewardBench include:

Multi-step / Reasoning tasks.

- “*Push the puzzles to spell word GO*” (LeRobot) — spatial reasoning combined with sequential multi-object manipulation.
- “*Build a pyramid*” (Bimanual YAM) — multi-step stacking with precise positioning, requiring four distinct subtasks.
- “*Group the cubes by color*” (Bimanual YAM) — requires color perception and categorical reorganization of multiple objects.
- “*Put the cubes of the same colors together*” (Franka) — color-conditional sorting across multiple objects.
- “*Remove the block and stack the green cube on the red cube*” (LeRobot) — obstacle removal followed by color-conditional stacking.
- “*Pack and close the box*” (Franka) — multi-phase task involving packing objects then closing the container.

Fine manipulation / Precise control.

- “*Align the cubes horizontally*” (Bimanual YAM) — fine spatial alignment, corresponding to the longest execution durations in the dataset.
- “*Rotate the banana by 90 degrees*” / “*Rotate the marker by 45 degrees*” (Franka) — precise rotation control with specified angles.

- “*Make the screw points to the glue*” (Bimanual YAM) — precise orientation alignment of two distinct objects.
- “*Pour tea*” (Franka) — requires controlled pouring motion and spatial orientation awareness.

Deformable object handling.

- “*Fold the towel*” / “*Fold towel*” (Franka / Bimanual YAM) — deformable material manipulation requiring careful grasp and fold planning.
- “*Stack one cloth on top of another*” (Single-arm YAM) — soft object stacking with non-rigid geometry.

Abstract / Symbolic tasks.

- “*Press enter and then space key*” (Franka) — keyboard interaction requiring sequential key presses.
- “*Set table*” (Bimanual YAM) — open-ended task requiring understanding of table-setting conventions.

The following table summarizes the statistics of each dataset in **ManiRewardBench**.

Table B.3: Summary of **ManiRewardBench** datasets. 6 tasks appear in both the Lerobot and Lerobot failed splits, giving 130 unique tasks across the full benchmark.

Dataset	Episodes	Tasks
Lerobot	150	22
Lerobot failed	156	23
Franka	150	51
Bimanual YAM	97	20
Single-arm YAM	100	20
Total	653	136 (130 unique)

We briefly describe each dataset below:

- **Lerobot Bimanual dataset:** Successful bimanual LeRobot manipulation demos (push, put, remove, stack tasks), 5–10 episodes per task.
- **Lerobot failure dataset:** Mixed failed and successful trajectory examples with the same task types, ~ 7 episodes per task.
- **Franka dataset:** Franka robot demos with a diverse set of 51 instructions (rotation, cleaning, packing, pick-and-place), mostly 3 episodes per task.
- **Bimanual YAM dataset:** Bimanual YAM manipulation (fold, stack, build, open, etc.), 5 episodes per task.
- **Single-arm YAM dataset:** Single-arm YAM manipulation (put, remove, stack), 5 episodes per task.

B.6.1 Subtask Annotation

For each task, episodes are manually labeled and segmented into a sequence of predefined subtasks. Each task is associated with an ordered list of subtasks that represent stages of execution (e.g., reaching for an object, grasping it, or placing it). For every subtask, annotators specify a `start_second` (the time in seconds when the subtask begins) and an `end_second` (the time in seconds when it ends). Subtasks are non-overlapping and strictly ordered in time, with each subtask beginning immediately after the previous one ends.

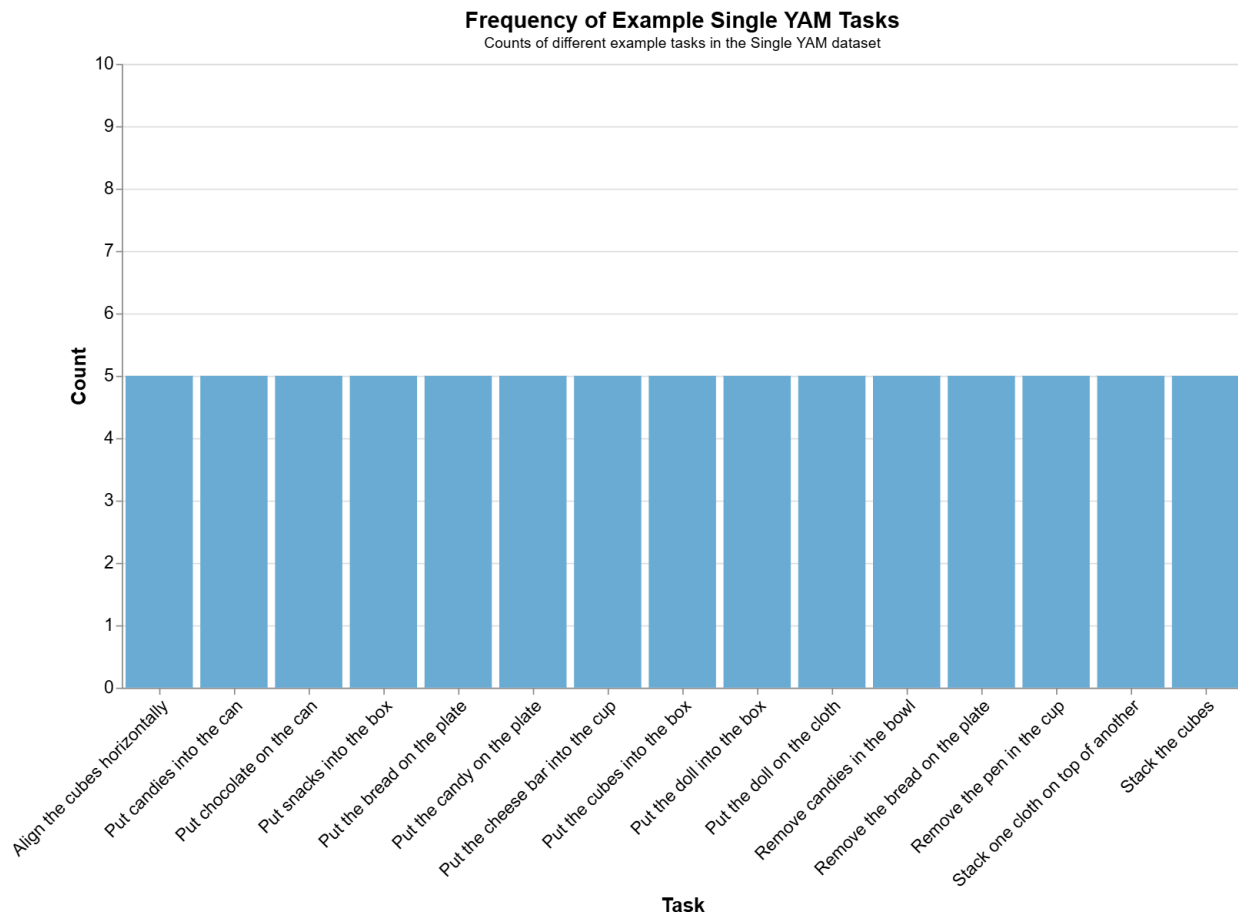


Figure B.9: Counts of different example tasks in the single-arm YAM dataset.

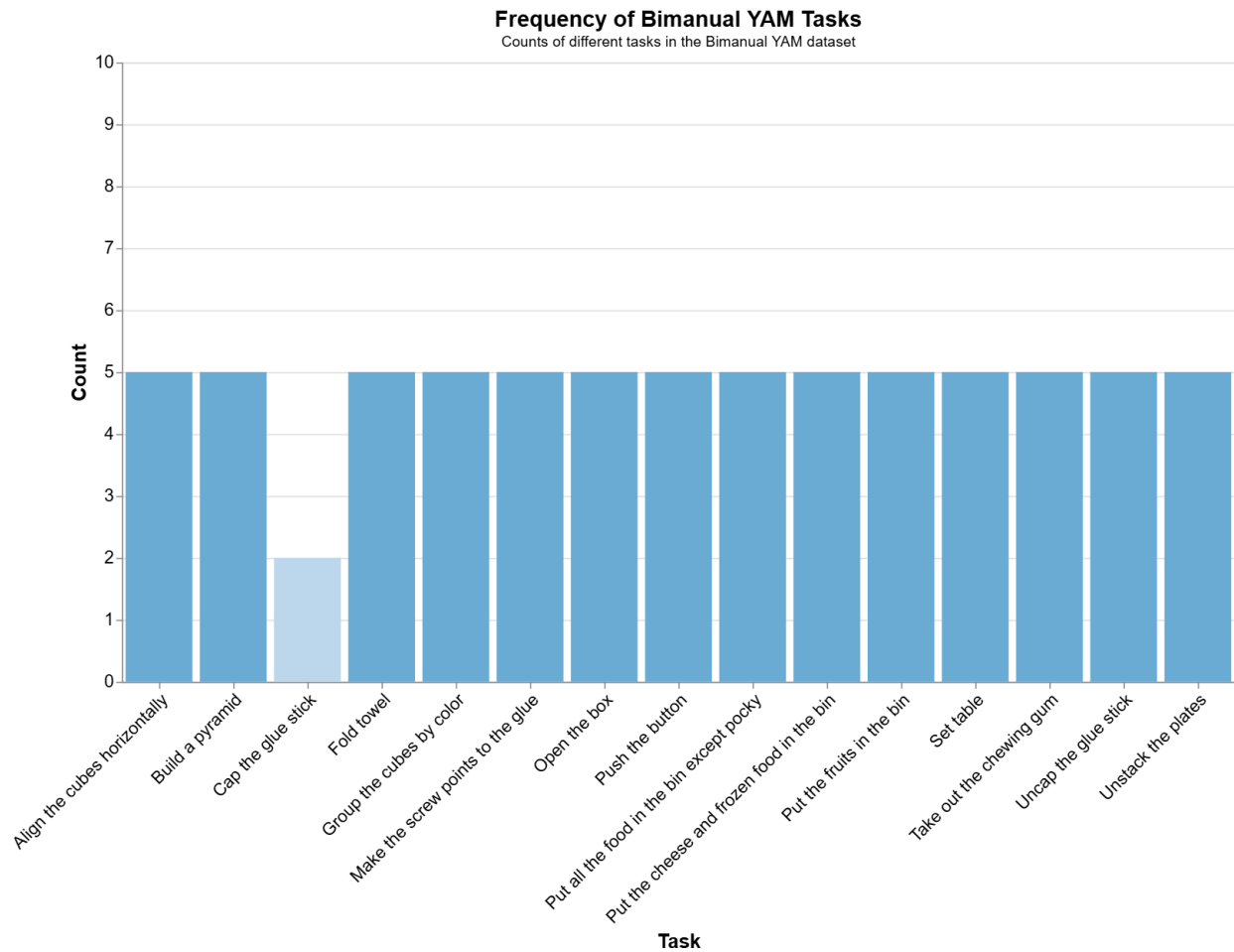


Figure B.10: Counts of different example tasks in the bimanual YAM dataset.

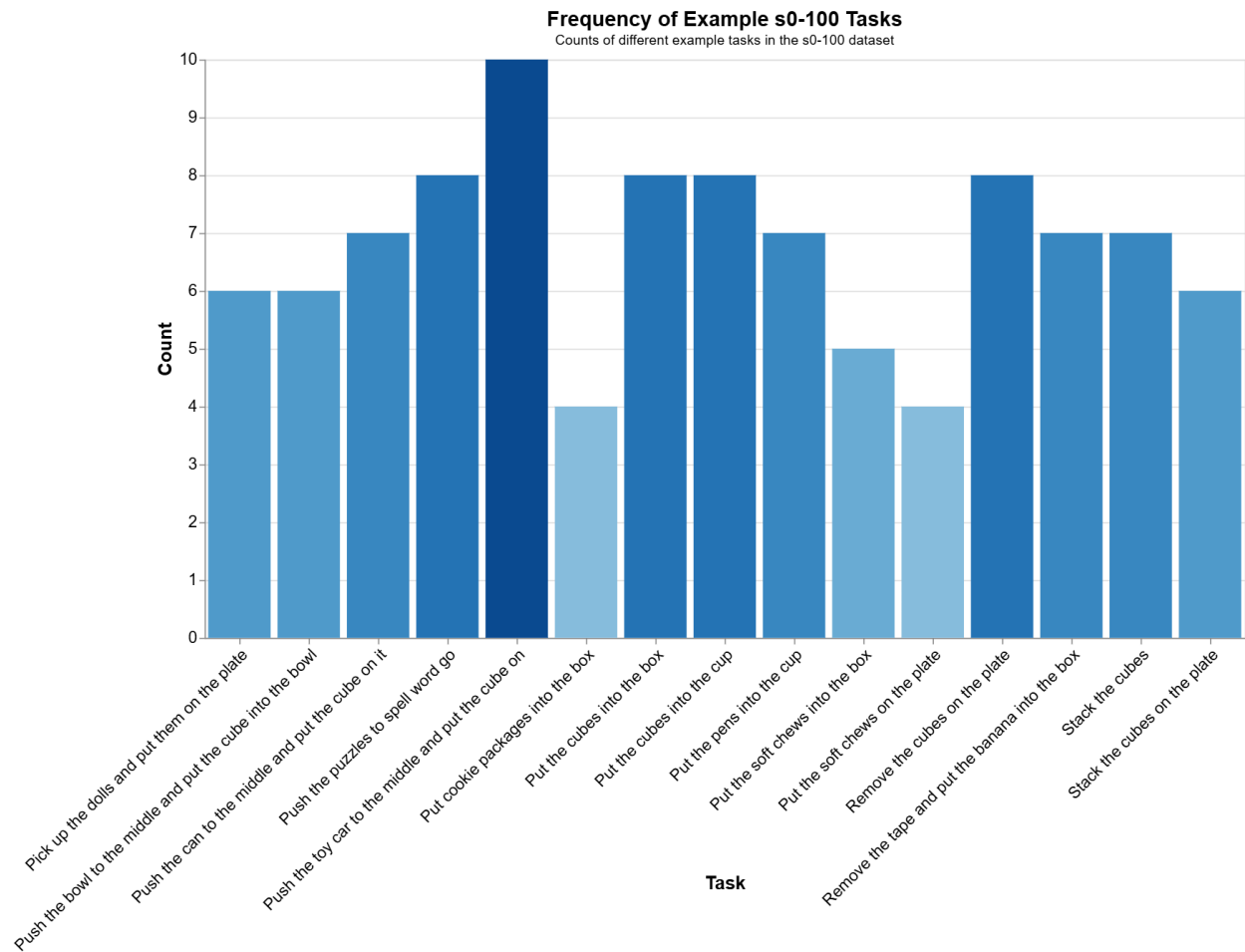


Figure B.11: Counts of different example tasks in the SO-100 dataset.

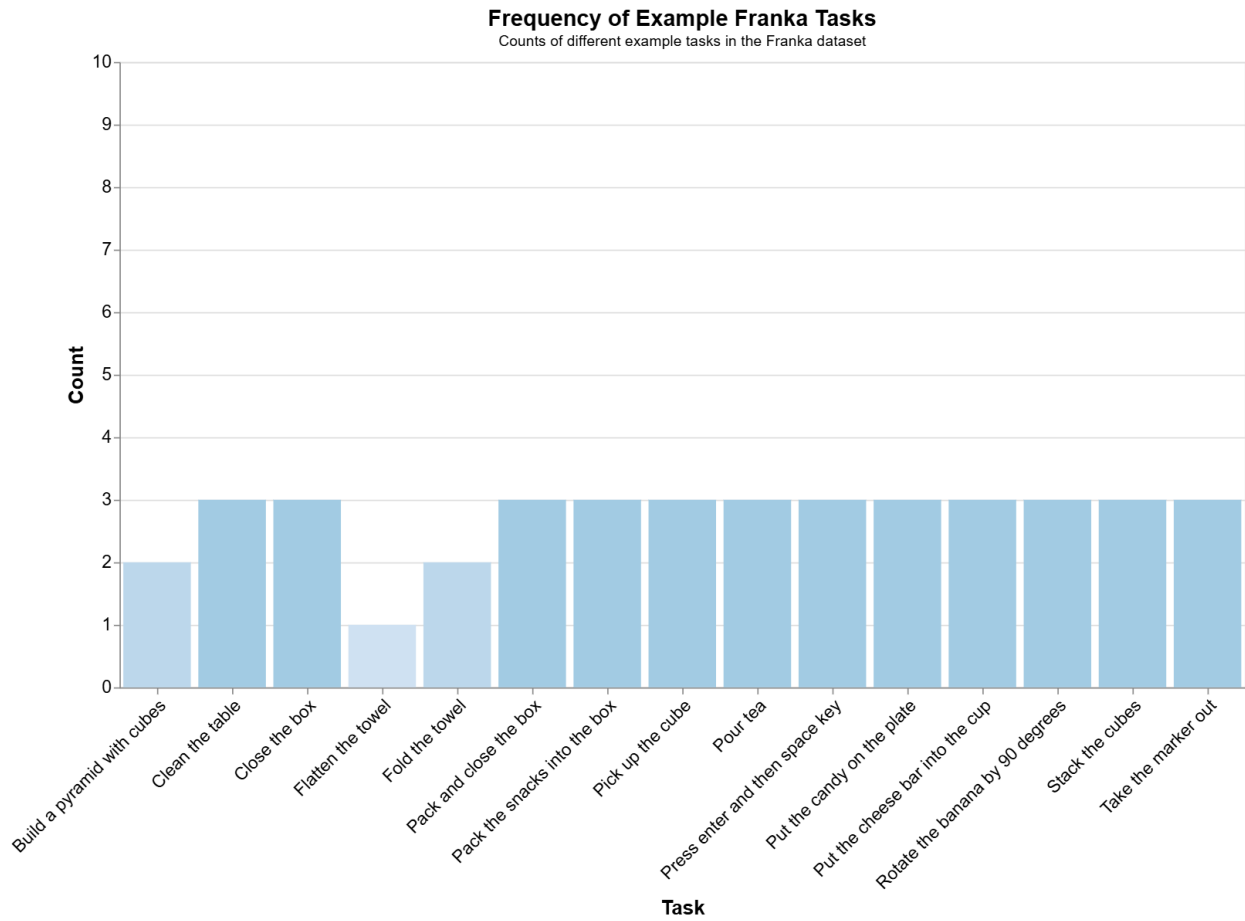


Figure B.12: Counts of different example tasks in the Franka dataset.

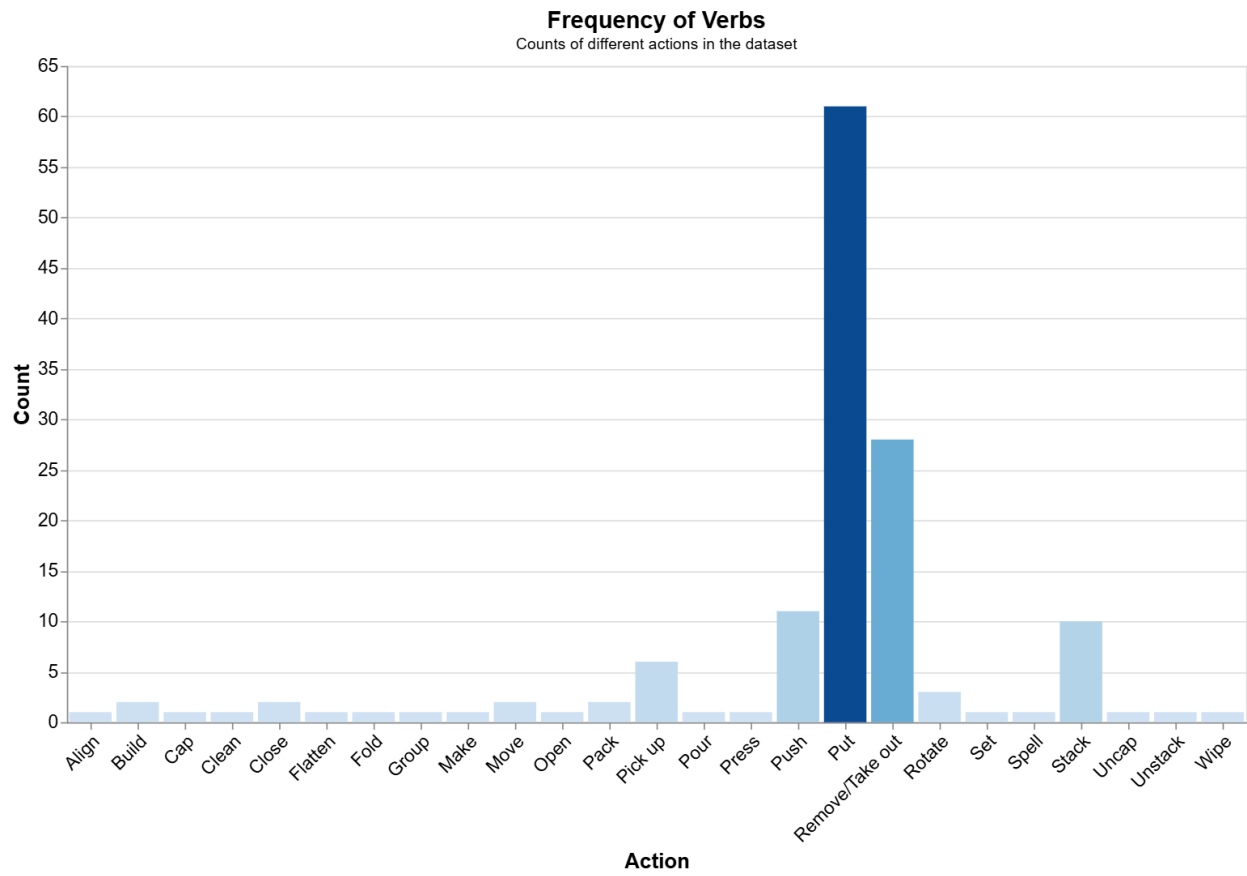


Figure B.13: Frequency of verbs

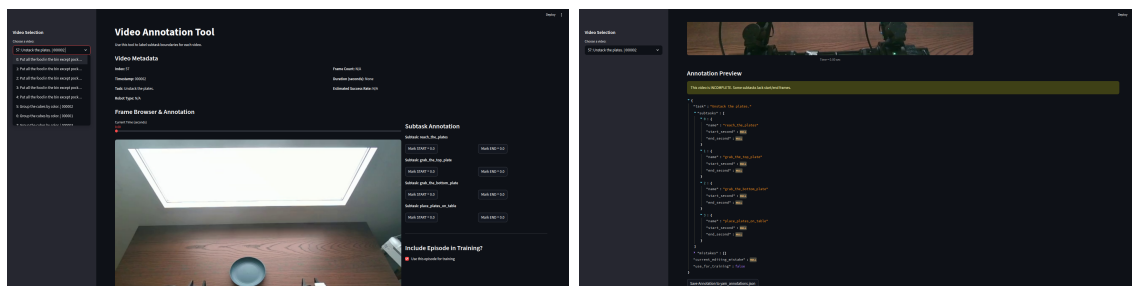
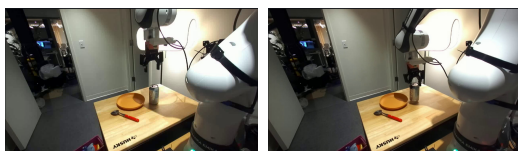


Figure B.14: Screenshot of the Annotation Tool.



start_second: 0.0 end_second: 3.9

(a) Grab the can



start_second: 4.0 end_second: 6.4

(b) Place the can in the plate



start_second: 6.5 end_second: 9.5

(c) Grab the spoon



start_second: 9.6 end_second: 11.4

(d) Place the spoon in the plate

Figure B.15: Expert demonstration with annotation for the task "Clean the table".

Appendix C

Appendix to Chapter 4: dUltra

C.1 Limitations

The on-policy distillation reward uses AR teacher models (Qwen2.5-Math/Coder-7B-Instruct); choosing domains where strong teachers exist is part of the experimental design rather than an inherent constraint of the framework. We do not include a head-to-head empirical comparison with the concurrent work of Jazbec et al. [2025] because their code and checkpoints were not public at the time of writing; an indirect comparison based on their reported numbers is given in Related Work. Throughputs in Table C.3 are measured on a single hardware type (H200) with identical inference kernels across all methods.

C.2 Related Work

Reinforcement Learning for MDLMs. Recent work has explored post-training diffusion language models (dLLMs) with reinforcement learning to improve their reasoning and generation capabilities. Early examples include **d1**, which adapts GRPO-style optimization to the (approximate) likelihood structure of dLLMs [Zhao et al., 2025b], and DiffuCoder, which targets code generation for LLaDA-style masked diffusion models [Gong et al., 2025]. A growing line of follow-up work focuses on improving the stability and bias/variance trade-offs

of policy-gradient estimators for dLLMs [Tang et al., 2025, Rojas et al., 2025, Lin et al., 2025a, Wang et al., 2025a, Zhu et al., 2025, Yang et al., 2025]. LightningRL [Hu et al., 2026] is especially related in its goal of improving the speed-quality frontier of block-wise dLLMs with RL. Its main reported results, however, use the SDAR backbone, whereas our experiments are built on LLaDA-8B-Instruct; including the LightningRL headline numbers in the main table would therefore conflate method differences with base-model differences. Methodologically, LightningRL samples rollouts with confidence-threshold remasking, whereas dUltra learns the token-level unmasking policy through an explicit planner head. Notably, Wang et al. [2025c], Zhao et al. [2025a] try to obtain more accurate likelihood by using more intermediate decoding steps. More broadly, prior RL work either improves reasoning capability or tunes speed-quality behavior around an existing sampler, while dUltra directly learns an adaptive token-level unmasking policy and enables high quality mode filtering via RL. Close to our work is DCOLT [Huang et al., 2025], which also learns an unmasking order policy with RL. However, their unmasking policy does not enable parallel decoding due to the adoption of the Plackett-Luce model. Zhao et al. [2025a] learns adaptive confidence thresholds for unmasking with RL, but their method still relies on logit-based heuristics for token selection rather than learning a flexible unmasking policy as in our work. Concurrent work Jazbec et al. [2025] also learns an unmasking policy head with GRPO, but there are several important distinctions. First, they learn the unmasking policy from the logits rather than the last-layer representation. Second, they keep the base model fixed rather than jointly training it with the unmasking policy. Third, they do not introduce an on-policy distillation reward to encourage mode-seeking behavior. As a result, they report only marginal benefits over logit-based heuristics, while we achieve better performance in both accuracy and NFE even when compared to SOTA distillation-based methods.

MDLM Acceleration. Most existing acceleration methods for masked dLLMs are either system level, such as enabling KV cache and employing token dropping [Ma et al., 2025, Liu et al., 2025a, Hu et al., 2025b, Wu et al., 2025], or algorithmic level methods that design

better sampling strategies. Among the latter, there are two approaches: (i) training-free logit-based heuristic decoders [Wu et al., 2025, Israel et al., 2025, Wei et al., 2025, Ben-Hamu et al., 2025] and (ii) *distillation*-based approaches that fit off-policy trajectories produced by a base model [Chen et al., 2025d, Qian et al., 2025]. Heuristic samplers often rely on confidence, entropy, or related uncertainty signals computed from logits, and have recently enabled substantial inference speedups in dLLMs via parallel decoding and KV caching [Wu et al., 2025, Hu et al., 2025b]. Distillation-based methods, such as pseudo-trajectory distillation in d3LLM [Qian et al., 2025], learnable parallel decoding in dParallel [Chen et al., 2025d], and distribution-matching distillation in DiDi-Instruct [Zheng et al., 2025], can further increase decoding efficiency by fine-tuning the base model to sample multiple steps of the base model in one step, which essentially filters the modes of the base model. DMax [Chen et al., 2026c] provides another recent acceleration route by reformulating masked decoding as progressive self-refinement with soft parallel decoding. These methods depend on the quality and coverage of the teacher trajectories, specialized training distribution, or refinement dynamics, which may become off-policy as training progresses. Our approach explicitly addresses this mismatch by using on-policy rollouts with GRPO, while also encouraging mode-seeking behavior.

Denoising Trajectory Planning. Beyond heuristic confidence-based planning, recent works have explored learned planning modules for MDLM trajectories. These approaches differ fundamentally in their planning strategies. P2 [Peng et al., 2025] introduces explicit planning and denoising sub-stages, enabling iterative refinement of existing unmasked tokens. DCOLT [Huang et al., 2025] employs the Plackett-Luce model to generate sequential unmasking orders optimized for accuracy rather than parallel decoding efficiency. DiFFPO [Zhao et al., 2025a] learns adaptive per-prompt thresholds via off-policy RL but still selects tokens based on confidence heuristics. Concurrently, Jazbec et al. [2025] learns an unmasking policy head with GRPO on token confidences from logits.

C.3 Mode Filtering Illustration

Parallel decoding can expose unwanted modes when several dependent ambiguous tokens are sampled independently in the same denoising step. Figure C.1 illustrates this phenomenon and motivates learning unmasking trajectories that decode only compatible sets of tokens in parallel.

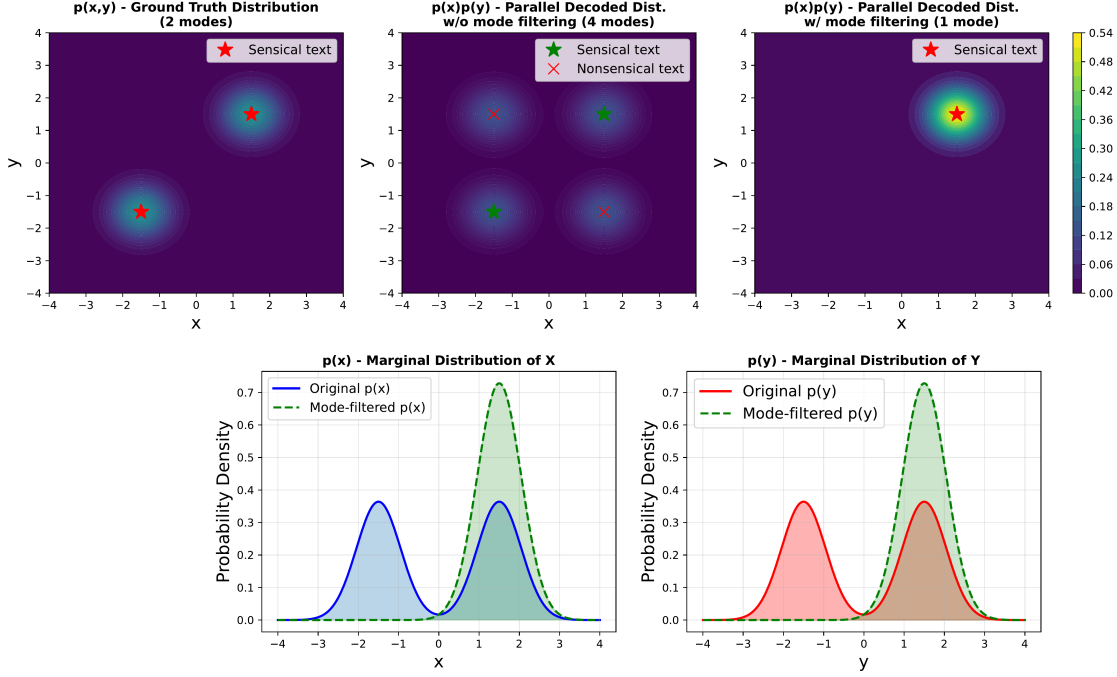


Figure C.1: Parallel decoding of dependent ambiguous tokens can lead to unwanted modes.

C.4 Motivating Case Study

To illustrate the potential of conditional independence-aware planning, we compare the quality of one-step generation of LLaDA-Instruct under different masking strategies in Figure C.2. More specifically, we compare random masking, where we randomly select a certain percentage of tokens in the ground truth to mask, and autoregressive masking, where we mask the same percentage of tokens only at the end of the ground truth sequence. We observe that LLaDA-Instruct is able to decode a large percentage (30%) of masked tokens with no noticeable

quality degradation with random masking, while masking the same percentage of mask tokens at the end of the sequence leads to poor generation quality. This is expected as randomly masked tokens usually have more contextual information making them less conditionally dependent on other masked tokens, while consecutive tokens at the end of the sequence are highly dependent on each other, even if given the partial answer.

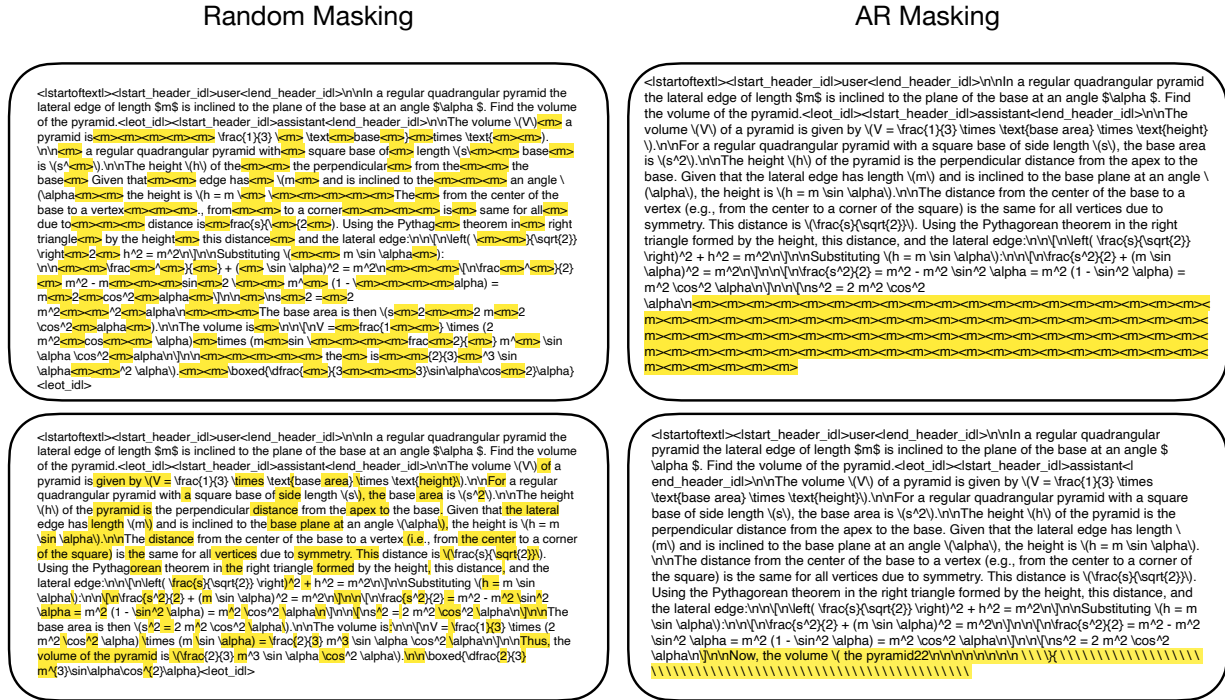


Figure C.2: Given the same prompt and ground truth, we mask 30% of tokens using both random masking and autoregressive (AR) masking. The left column shows the masked input sequence with random masking (<m> represents the mask token), and the corresponding one-step decoding result. The right columns show the masked input sequence with AR masking, and the corresponding one-step decoding result. We use LLaDA-8B-Instruct for both generations.

C.5 Baseline Supervision Comparison

Table C.1 summarizes the supervision and training requirements of each method.

Method	Training	Teacher signal	Rollout policy	Base model updated	Planner learned
Fast-dLLM	None	None (logit heuristic)	—	No	No
dParallel	SFT	Base-model completions + ground-truth filter	Off-policy	Yes	No
d3LLM	SFT	Base-model order + ground-truth tokens	Off-policy	Yes	No
dUltra	GRPO	AR teacher (Qwen2.5)	On-policy	Yes	Yes

Table C.1: Comparison of supervision and training requirements across acceleration methods. dUltra is the only method that (i) learns an explicit unmasking planner, (ii) trains with on-policy rollouts, and (iii) uses an external AR teacher as a reward signal. This additional supervision is an intentional design choice that enables mode-seeking behavior beyond what self-distillation can provide.

C.6 Planner Initialization

We find that if we initialize the unmasking planner head randomly, the expected number of tokens unmasked at each denoising step $n = \sum_i p_{\text{unmask}}(x_i)$ can be excessively large, leading to degenerate performance at the onset of training and preventing RL from effectively exploring trajectories with more sampling steps and better performance. We therefore initialize the planner before applying RL so that the RL algorithm uses a heuristic logit-based planner as a baseline to optimize the sampling order.

More specifically, we fine-tune the unmasking planner head using supervised learning to mimic the confidence-aware parallel decoding method proposed in Fast-dLLM [Wu et al., 2025]. We formulate this planner fine-tuning as a binary classification problem: given a partially masked input, the planner predicts which tokens should be unmasked at the current step according to the Fast-dLLM confidence criterion. For a set of masked positions $\mathcal{I}_{\text{block}}$, we optimize the unmasking planner head ϕ using binary cross-entropy loss with position weighting to handle class imbalance:

$$\mathcal{L}_{\text{planner}} = \frac{1}{|\mathcal{I}_{\text{block}}|} \sum_{i \in \mathcal{I}_{\text{block}}} \text{BCE}(\phi_i(\mathbf{h}), y_i; w_{\text{pos}} = \frac{\#\text{negatives}}{\#\text{positives}}), \quad (\text{C.1})$$

where $\phi_i(\mathbf{h})$ is the predicted unmasking probability for the i -th masked token, $y_i \in \{0, 1\}$ indicates whether token i should be unmasked according to Fast-dLLM, and w_{pos} weights tokens that should be unmasked to compensate for class imbalance.

C.7 Advantage Clipping Details

Vanilla GRPO can lead to the degenerate solution where all unmasking probabilities collapse to 0, even with the exact likelihood computed in Section 4.3.2. While this gives high distillation reward due to extremely slow sampling, it defeats the purpose of acceleration. We hypothesize that this occurs because trajectories with low advantage may be overly conservative: if the model could have unmasked 5 tokens but the planner only unmasked 3, the trajectory receives low advantage due to inefficiency. Penalizing this trajectory uniformly reduces the likelihood of all unmasking decisions, including the 3 tokens that were correctly unmasked.

Advantage clipping avoids this failure mode by setting below-threshold advantages to zero, focusing updates on above-threshold trajectories. This is useful because the space of possible unmasking orders is exponentially large compared to valid high-quality unmasking orders, making it more effective to exploit successful discoveries than to learn uniformly from failed attempts. In our experiments, setting $C = 0$ stabilizes training.

C.8 Training Dynamics and Stabilization

Figure C.3 shows the evolution of reward metrics during dUltra training on GSM8K. The correctness reward is measured by the correctness of generated solutions, and receives 2.0 if the answer is correct and 0.0 otherwise. The efficiency reward is defined as the negative number of denoising steps (NFE) times a constant. The distillation reward is the average per-token log-likelihood of the final rollout under the teacher model.

Figure C.4 illustrates the role of advantage clipping in training stability. Without advantage clipping ($C = -\infty$), the training quickly becomes unstable: the average NFE diverges to extreme values as the learned planner collapses to a degenerate policy that never unmask tokens. This occurs because negative advantages from poor trajectories can dominate the gradient signal, causing the planner to become overly conservative. With

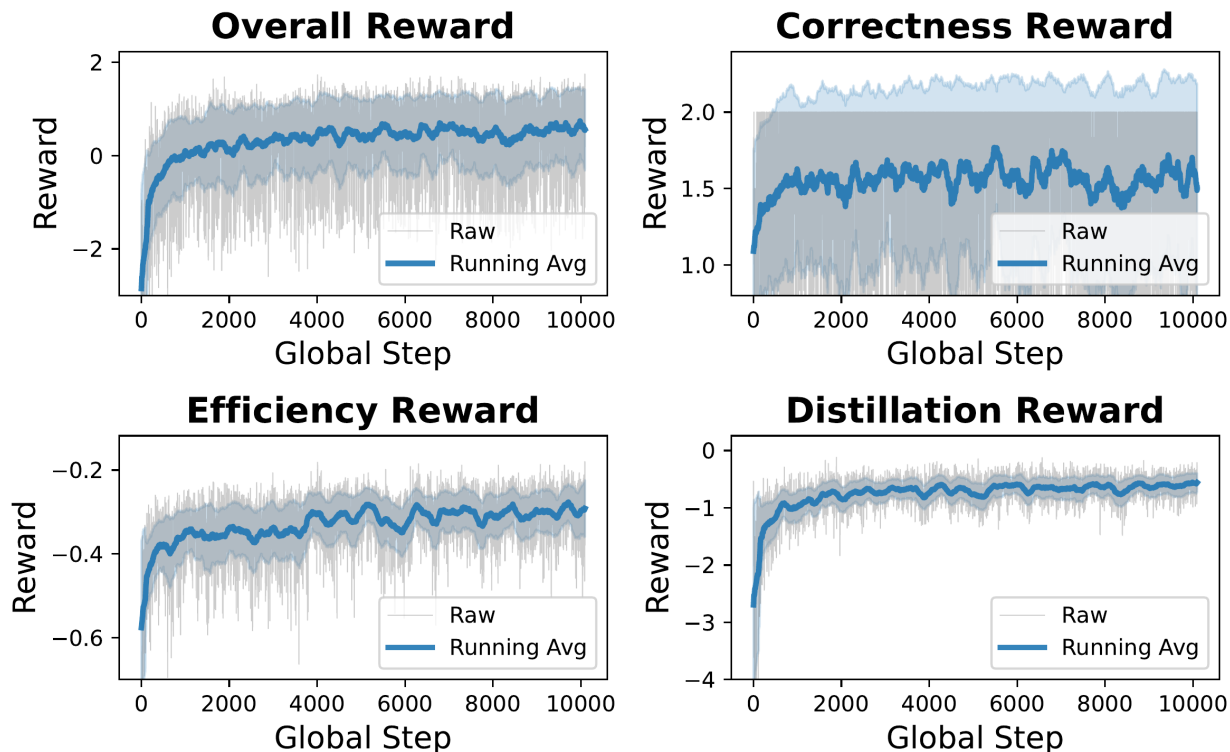


Figure C.3: Training dynamics on GSM8K.

appropriate advantage clipping ($C = 0.0$), training remains stable and the model learns a balanced unmasking strategy. The clipping mechanism ensures that only rollouts with above-average returns contribute to policy updates, preventing the model from over-fitting to failure modes.

C.9 Additional Main Results with Large Block Size

Table C.2 presents the full results with block size 128 with all other settings the same as results with block size 32. As in Table 4.1, all dUltra inference results use deterministic threshold decoding rather than random Bernoulli sampling. Larger block sizes enable faster sampling with minimal accuracy loss; dUltra achieves 16.38 NFE on GSM8K (23% reduction vs. $B = 32$) with only 3.6 pp accuracy drop.

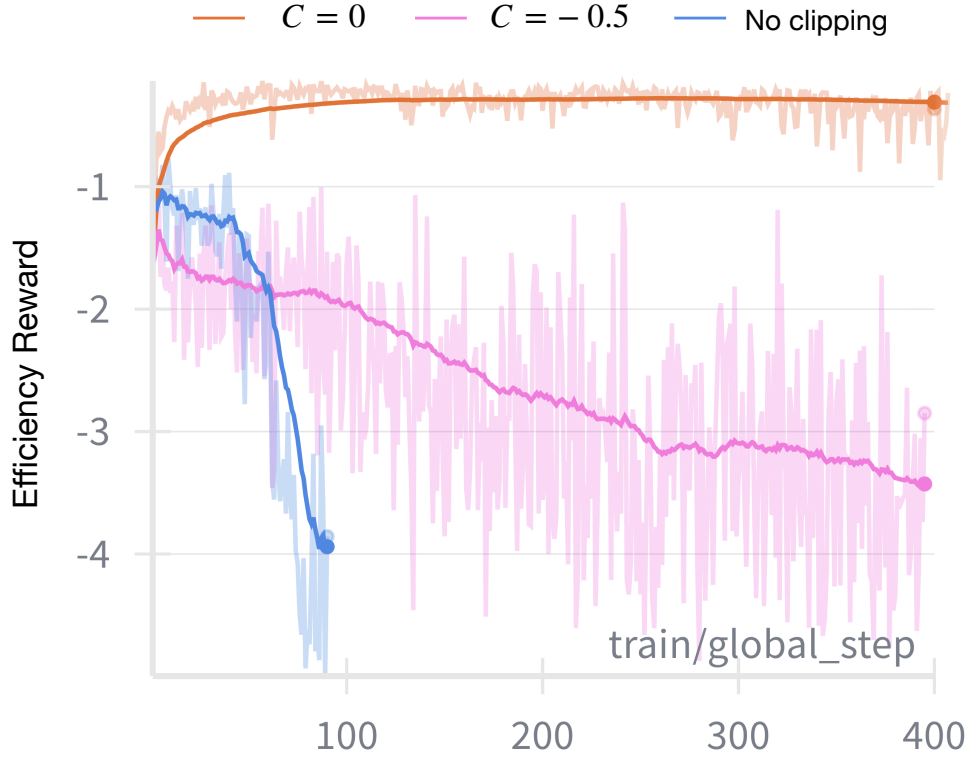


Figure C.4: Effect of advantage clipping value C on training stability. Without advantage clipping, the average NFE quickly diverges as the planner collapses to never unmasking any tokens. With advantage clipping, training remains stable and the model learns to reduce NFE while maintaining task performance.

C.10 Wall-Clock Throughput

To verify that NFE reductions translate to real latency gains, we measure wall-clock throughput (tokens/second) on a single H200 GPU for 256-token generation with block size $B = 128$. We compare against the LLaDA baseline (Fast-dLLM) and distillation-based acceleration methods. No KV-cache is used.

C.11 Pareto Frontiers

At inference time, we eliminate Bernoulli sampling randomness by deterministically unmasking only tokens whose planner probability exceeds a threshold; this is the rule used for both

Method	Math				Coding			
	GSM8K		MATH500		HumanEval		MBPP	
	Acc↑	NFE↓ (T/F↑)	Acc↑	NFE↓ (T/F↑)	Acc↑	NFE↓ (T/F↑)	Acc↑	NFE↓ (T/F↑)
Fast-dLLM	<u>75.66</u>	54.43 (4.56)	<u>32.60</u>	75.72 (3.30)	<u>35.40</u>	61.54 (4.08)	<u>37.80</u>	56.93 (4.07)
dParallel	<u>75.66</u>	24.07 (8.23)	31.40	28.49 (5.45)	30.49	<u>23.68</u> (6.26)	36.00	<u>18.07</u> (7.63)
d3LLM	69.07	<u>21.37</u> (9.59)	31.40	<u>26.65</u> (7.03)	32.32	<u>24.49</u> (7.34)	33.80	<u>26.37</u> (7.66)
dUltra (ours)	77.65	16.38 (13.96)	33.64	26.53 (9.25)	36.83	21.79 (9.43)	39.12	14.78 (9.97)

Table C.2: Main results with generation length 256 and block size $B = 128$. dUltra uses deterministic threshold decoding at inference time. We report accuracy, NFE, and T/F (tokens per model forward pass); **bold** and underlined numeric values mark the best and second-best results for each metric.

Method	GSM8K			MATH500		
	Acc↑	Tok/s↑	Speedup↑	Acc↑	Tok/s↑	Speedup↑
LLaDA baseline	<u>75.7%</u>	252	1.0×	<u>32.6%</u>	176	1.0×
dParallel	<u>75.7%</u>	459	1.8×	31.4%	298	1.7×
d3LLM	69.1%	<u>532</u>	<u>2.1×</u>	31.4%	<u>378</u>	<u>2.1×</u>
dUltra (ours)	77.7%	851	3.4×	32.8%	524	3.0×

Table C.3: Wall-clock throughput on a single H200 GPU (256-token generation, $B = 128$). dUltra achieves 3.0–3.4× speedup over the LLaDA baseline and $\sim 1.5\times$ over the best distillation baseline (d3LLM), while simultaneously improving accuracy on both tasks. **Bold** and underlined values mark the best and second-best results for each metric; tied second-best values are both underlined.

Table 4.1 and Table C.2. We use Fast-dLLM’s factor-based parallel decoding strategy [Wu et al., 2025] for models without a planner. We sweep planner thresholds for dUltra models with planners and Fast-dLLM factors for dUltra without planners, and show the resulting Pareto frontiers in Figure C.5. No-planner ablations replace the learned planner with confidence-based decoding while keeping the base model fixed.

C.11.1 Ablation Results Without the Planner

Table C.4 reports the no-planner ablation obtained by removing the learned planner from dUltra and decoding with confidence-based sampling. These rows quantify the accuracy, NFE, and T/F trade-offs when the planner is replaced by a fixed confidence-based decoding

rule.

Method	Math				Coding			
	GSM8K		MATH500		HumanEval		MBPP	
	Acc↑	NFE↓ (T/F↑)	Acc↑	NFE↓ (T/F↑)	Acc↑	NFE↓ (T/F↑)	Acc↑	NFE↓ (T/F↑)
dUltra w/o planner ($B = 32$)	75.44	15.31 (13.39)	34.00	24.67 (9.45)	23.78	16.18 (10.19)	20.40	12.18 (11.28)
dUltra w/o planner ($B = 128$)	69.67	10.46 (22.16)	32.80	19.36 (12.80)	21.95	13.18 (15.00)	21.60	8.27 (17.65)

Table C.4: No-planner ablation for dUltra. Each row removes the learned planner and uses confidence-based decoding, reporting the benchmark-matched dUltra result for each task group.

C.12 Learned Unmasking Strategies

To understand what unmasking order patterns dUltra learns, we visualize the unmasking behavior across denoising steps on both math and coding tasks (Figure C.6 and Figure C.7). Each heatmap shows when tokens are unmasked during the denoising process, with red bars indicating newly unmasked tokens at each step. The vertical axis represents token positions, while the horizontal axis shows denoising steps. For both visualizations, we use a block size of 128 to allow more flexible unmasking orders.

The visualizations reveal that the major trend across both tasks is largely autoregressive: tokens are predominantly unmasked in left-to-right order, as evidenced by the diagonal progression from top-left to bottom-right in both heatmaps. This suggests that despite the theoretical flexibility of masked diffusion models, the autoregressive order often serves as a good enough prior. This also explains the success of AR models and speculative decoding in practice.

Nevertheless, we observe that code generation exhibits more non-autoregressive behavior compared to mathematical reasoning. In Figure C.7, we observe visible jumps where the model unmask function signatures (`def has_close_elements`), control flow keywords (`for`, `if`), and return statements earlier than strict left-to-right order would dictate. These structural elements provide crucial context for generating subsequent code and can be predicted with

high confidence from the prompt alone. The model exploits this hierarchical dependency structure to achieve greater parallelism: by unmasking the skeleton of the code early, it creates multiple independent sub-problems (filling in loop bodies, conditions, etc.) that can be decoded in parallel.

In contrast, mathematical reasoning in Figure C.6 follows a smoother, more sequential progression. While there are minor deviations—such as unmasking numbers or operators that can be inferred from the problem statement—the overall pattern adheres closely to autoregressive order. This reflects the fundamentally sequential nature of step-by-step reasoning: each calculation step typically depends on previous results, limiting opportunities for parallel decoding.

Overall, these unmasking patterns also validate our motivation: the learned planner creates masking distributions that locally resemble the random masking strategy from our case study (Figure C.2), selectively deviating from autoregressive order to unmask conditionally independent tokens in parallel while maintaining quality. The degree of non-autoregressive behavior adapts to task structure, with code exploiting more parallelism opportunities than sequential mathematical reasoning. Since consecutive tokens are usually highly dependent on each other in the true data distribution, the model’s ability to generate many consecutive tokens in parallel demonstrates the mode filtering behavior discussed in the introduction: parallel generation of dependent tokens independently can be realized by filtering the modes of the joint distribution, trading diversity for speed.

C.13 MDLM Background

C.13.1 Single-Token Forward and Reverse Process

Continuous-time MDLMs [Shi et al., 2024, Sahoo et al., 2024] define a forward noising process for $t, s \in [0, 1]$ with $t > s$:

$$\begin{aligned} q_{t|s}(x_t|x_s = m) &= \text{Cat}(\mathbf{e}_m), \\ q_{t|s}(x_t|x_s = x_0) &= \text{Cat}\left(\frac{\alpha_t}{\alpha_s}\mathbf{e}_{x_0} + \frac{\alpha_s - \alpha_t}{\alpha_s}\mathbf{e}_m\right), \end{aligned} \tag{C.2}$$

where α_t is the noise schedule with $\alpha_0 = 1$ and $\alpha_1 = 0$. The marginal is $q_{t|0}(x_t|x_0) = \text{Cat}(\alpha_t\mathbf{e}_{x_0} + (1 - \alpha_t)\mathbf{e}_m)$. The time reversal for $s < t$ given x_0 is

$$q_{s|t,0}(x_s|x_t, x_0) = \begin{cases} \text{Cat}(\mathbf{e}_{x_0}), & x_t = x_0 \\ \text{Cat}\left(\frac{\alpha_s - \alpha_t}{1 - \alpha_t}\mathbf{e}_{x_0} + \frac{1 - \alpha_s}{1 - \alpha_t}\mathbf{e}_m\right) & x_t = m. \end{cases} \tag{C.3}$$

Similar to DDPM [Ho et al., 2020], the model $\boldsymbol{\mu}_\theta$ predicts the clean sample x_0 :

$$p_\theta(x_s|x_t) = q_{s|t,0}(x_s|x_t, \mathbf{e}_{x_0} \approx \boldsymbol{\mu}_\theta(x_t, t)). \tag{C.4}$$

C.13.2 NELBO Derivation

The approximation in Equation (C.4) is optimized by minimizing the NELBO. Given discretization steps T , with $s(i) = (i - 1)/T$ and $t(i) = i/T$, the NELBO decomposes as:

$$\mathcal{L}_{\text{NELBO}} = \mathbb{E}_q \left[\underbrace{-\log p_\theta(x | x_{t(1)})}_{\mathcal{L}_{\text{recons}}} + \underbrace{\sum_{i=2}^T D_{\text{KL}}[q(x_{s(i)} | x_{t(i)}, x) \| p_\theta(x_{s(i)} | x_{t(i)})]}_{\mathcal{L}_{\text{diffusion}}} + \underbrace{D_{\text{KL}}[q(x_{t(T)} | x) \| p_\theta(x_{t(T)})]}_{\mathcal{L}_{\text{prior}}} \right]. \tag{C.5}$$

For MDLMs with carry-over unmasking [Sahoo et al., 2024, Shi et al., 2024], this simplifies to the weighted cross-entropy in Eq. 4.1 of the main paper.

C.14 Experimental Setup

C.14.1 Implementation Details

Model Architecture

Our model extends the LLaDA-Instruct model [Nie et al., 2025] with an unmasking planner head. The architecture consists of:

Base Model. We use LLaDA [Nie et al., 2025], a masked diffusion language model with Llama-like architecture adapted for discrete diffusion. The model uses a special mask token and employs time-agnostic parametrization following recent findings that the optimal ELBO solution is independent of the time variable [Ou et al., 2025, Sahoo et al., 2024].

Unmasking Planner Head. Following the architecture in Huang et al. [2025], we introduce an unmasking planner head that takes the last-layer hidden states and outputs per-token unmasking probabilities:

$$p_{\text{unmask}}(x) = \text{sigmoid}(\text{head}(h_x)) \quad (\text{C.6})$$

where h_x is the hidden state for token position x . The head consists of:

- $L = 1$ transformer blocks (LLaDABlock layers)
- Adaptive Layer Normalization (AdaLN) with timestep conditioning
- Mask embeddings
- Final linear projection to scalar logit

Since the unmasking planner head is small (a single transformer block), it adds only a small computational overhead relative to the base model.

Training Configuration:

- Unmasking planner head fine-tuning: Learning rate 1e-6, batch size 4, cosine schedule with warmup
- GRPO: Block length 32/128, temperature 0.1, advantage clipping, group size 12, learning rate 5e-6
- Optimizer: AdamW with weight decay 0.01
- Hardware: Single H200 GPU

C.14.2 Reward Function

Our training uses a multi-component reward function that combines task-specific correctness verification, format compliance, and efficiency incentives. The total reward is computed as a weighted sum:

$$r_{\text{total}} = \sum_i \lambda_i \cdot r_i \quad (\text{C.7})$$

where r_i are individual reward components and λ_i are their weights. Below we detail the default configuration for mathematical reasoning and code generation tasks.

Mathematical Reasoning Tasks (GSM8K, MATH500)

Correctness Reward ($\lambda = 1.0$): Extracts the final answer from LaTeX `\boxed{\}` notation and verifies mathematical equivalence against ground truth using `math_verify` library for symbolic comparison. Returns 2.0 for correct answers, 0.0 otherwise.

Format Reward ($\lambda = 1.0$): Rewards proper use of LaTeX boxed notation (`\boxed{answer}`) for presenting the final answer. Returns 0.5 when answers are properly boxed, 0.0 otherwise. This encourages the model to follow standard mathematical writing conventions.

Efficiency Reward ($\lambda = 1.0$): Incentivizes faster sampling by penalizing the number of denoising steps: $r_{\text{step}} = -N_{\text{steps}}/50$, where N_{steps} is the total number of denoising iterations.

On-Policy Distillation Reward ($\lambda = 1.0$): Computes the average per-token log-likelihood of the generated completion under **Qwen2.5-Math-7B-Instruct** teacher model: $r_{\text{distill}} = \frac{1}{L} \sum_{i=1}^L \log p_{\text{teacher}}(c_i \mid \mathbf{q}, \mathbf{c}_{<i})$, where L is the completion length. This encourages mode-seeking behavior by rewarding completions that the teacher assigns high probability.

Code Generation Tasks (**HumanEval**, **MBPP**)

Correctness Reward ($\lambda = 3.0$): Executes the generated code against test cases with a 10-second timeout per test. Returns 1.0 if all tests pass, 0.0 for any wrong outputs, and -0.05 for runtime errors (excluding syntax/indentation errors which return 0.0). Code is extracted from markdown fences and wrapped to support LeetCode-style class solutions.

Code Fence Format Reward ($\lambda = 1.0$): Encourages proper markdown code formatting with incremental scoring: +0.25 for any fence, +0.25 for ‘‘`python` fence, +0.25 for closing fence, +0.25 for no trailing text. Small penalty (0.001 per character) for extra content after closing fence. Maximum score: 1.0.

Efficiency Reward ($\lambda = 0.1$): Same as math tasks.

On-Policy Distillation Reward ($\lambda = 1.0$): Uses **Qwen2.5-Coder-7B-Instruct** teacher model. Computation is identical to math tasks but with domain-specific teacher.

C.15 Training Algorithm

Each training iteration consists of the rollout-and-advantage phase and the gradient-update phase below. The two phases are repeated until θ and ϕ converge or the maximum number of epochs is reached.

Algorithm 3: GRPO rollout and advantage computation

Input: diffusion LLM model θ , unmasking planner head ϕ , dataset $\mathcal{D}$, reward_func,

advantage clipping threshold C

Sample questions $q \sim \mathcal{D}$

for $g \leftarrow 1$ **to** G **do**

// Generate a group of G trajectories

 Initialize x^g with q and mask tokens

$k \leftarrow 0$ *// number of steps*

while x^g has mask tokens **do**

 Run the model forward once to get the logits l and last-layer hidden state h

 Calculate unmasking probability $\pi_{\text{unmask}} = \text{sigmoid}(\phi(h))$ for each mask token

 Calculate token probability $\pi_{\text{token}} = \text{softmax}(l)$

 Sample the mask tokens to unmask $\mathbf{Pos}^{(k)} \sim \text{MultivariateBernoulli}(\pi_{\text{unmask}})$

 Sample the value of unmasked tokens $\mathbf{x}_{\mathbf{Pos}^{(k)}}^g \sim \pi_{\text{token}}$

$k \leftarrow k + 1$

$k^g \leftarrow k$

$r^g \leftarrow \text{reward_func}(x^g)$ *// Compute the rewards*

for $g \leftarrow 1$ **to** G **do**

$A^g \leftarrow r^g - \text{mean}(r^{1:G})$ *// Compute advantages*

if $A^g < C$ **then**

$A^g \leftarrow 0$ *// Advantage clipping*

$N \leftarrow \max(k^g)$

Algorithm 4: GRPO gradient accumulation and parameter update

Input: model θ , trajectories and advantages from Algorithm 3

// Loop through the group

for $g \leftarrow 1$ to G do

 // Loop through denoising steps

 for $n \leftarrow 1$ to N do

 if $k^g > n$ then

 // Compute π_θ and losses

$$\pi(\mathbf{x}^{g,n} \mid \mathbf{x}^{g,n-1}) = \pi_{\text{select}}(\mathbf{Pos}^{(n)} \mid \mathbf{x}^{g,n-1}) \cdot \pi_{\text{token}}(\mathbf{x}^{g,n} \mid \mathbf{x}^{g,n-1}, \mathbf{Pos}^{(n)})$$

 // Equation (4.3)

$$\mathcal{L}_{\theta,n} \leftarrow -\frac{1}{\|\mathbf{x}\| \cdot G} \sum_{g=1}^G \frac{\pi(\mathbf{x}^{g,n} \mid \mathbf{x}^{g,n-1})}{\text{sg}(\pi(\mathbf{x}^{g,n} \mid \mathbf{x}^{g,n-1}))} A^g \quad // \text{ sg is stop gradient}$$

 Accumulate gradient $\nabla_\theta \mathcal{L}_{\theta,n}$

Update θ with accumulated gradients $\sum_{n=1}^N \nabla_\theta \mathcal{L}_{\theta,n}$

Zero out accumulated gradients

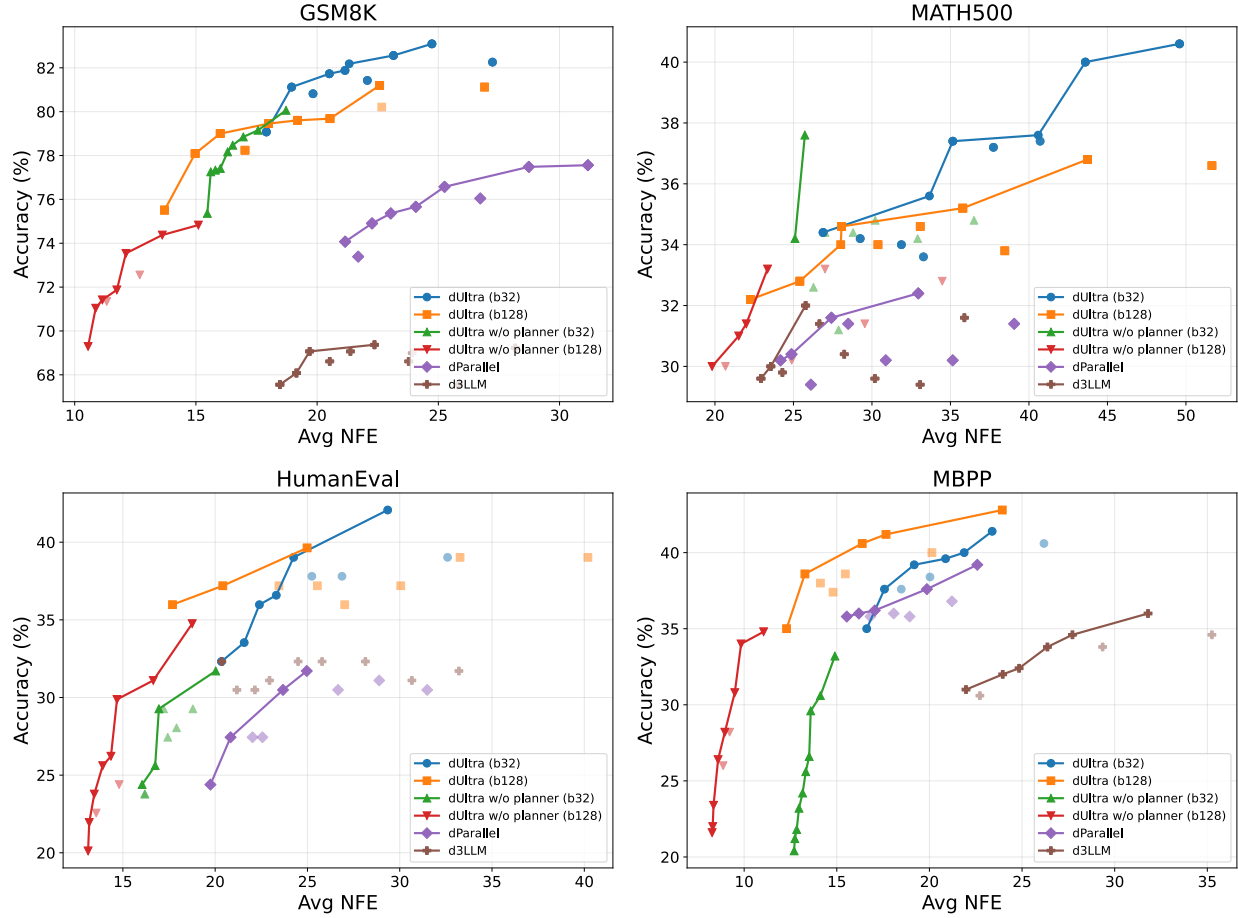


Figure C.5: Accuracy-efficiency Pareto frontiers across math reasoning and code generation benchmarks. We compare dUltra against baseline methods on GSM8K (top-left), MATH500 (top-right), HumanEval (bottom-left), and MBPP (bottom-right). The x-axis shows average NFE (lower is faster), while the y-axis shows task accuracy. Each point represents a different threshold for dUltra and a different factor for models without planners. The no-planner dUltra ablations directly use the output logits for confidence-based token selection. dUltra consistently achieves strong accuracy-efficiency trade-offs compared to distillation-based methods, demonstrating the effectiveness of our learned unmasking strategies via on-policy distillation.

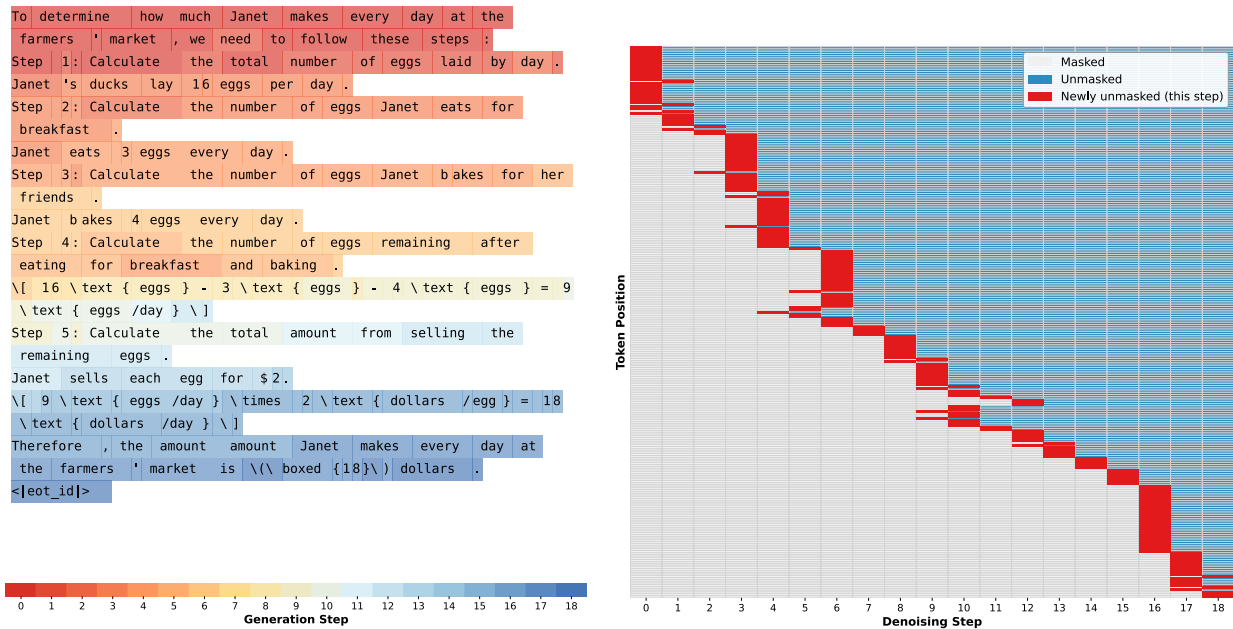


Figure C.6: Visualization of learned unmasking patterns on GSM8K. Red bars show newly unmasked tokens at each denoising step. The pattern exhibits a predominantly autoregressive (left-to-right) trend with minor deviations. An interesting observation is that sometimes, "Step" and ":" are generated first without the numbering in between, and the numbering is generated in the next step.

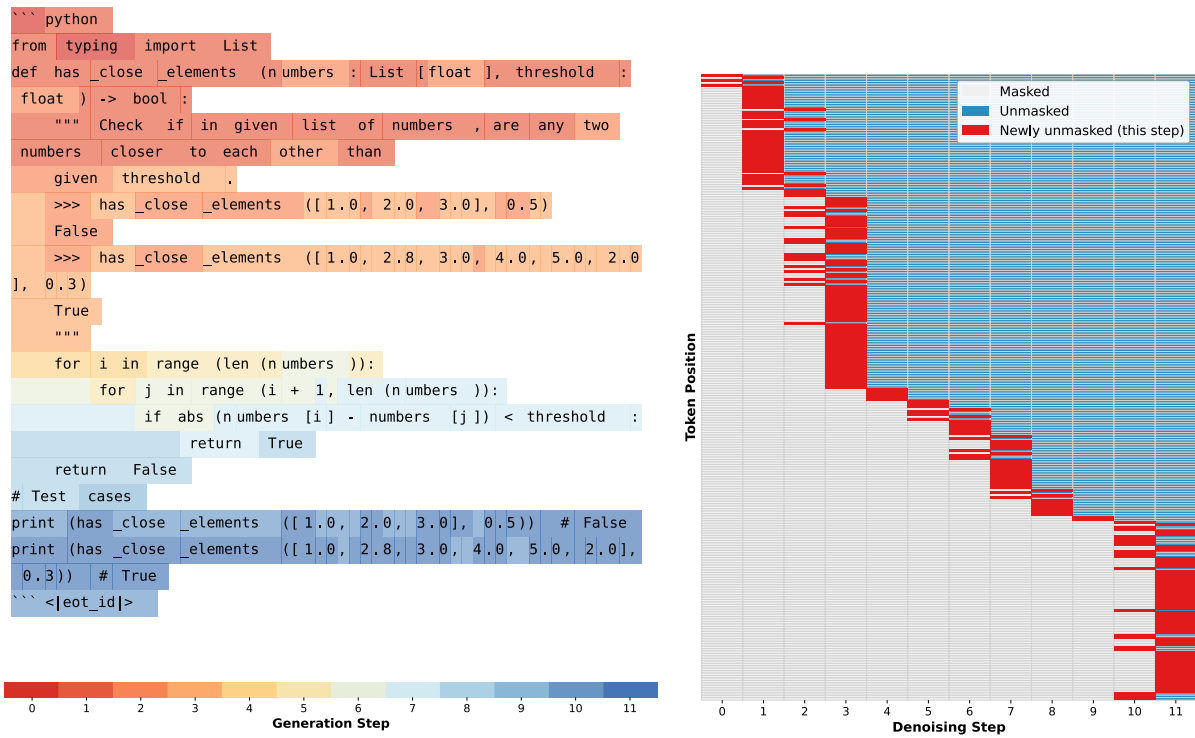


Figure C.7: Visualization of learned unmasking patterns on HumanEval. While the overall trend remains autoregressive, code generation exhibits more non-autoregressive behavior with visible jumps to unmask function signatures, control flow keywords, and structural elements earlier than strict left-to-right order.

Appendix D

Appendix to Chapter 5: Sharpness and Compression

D.1 Linear stability trick on penultimate-layer and middle-layer features

We define $g(x)$ to be the penultimate-layer features such that $f(x) = \mathbf{W}_L g(\mathbf{W}x) + b$. With slight abuse of notation, we define $J = \frac{\partial g(\mathbf{W}\mathbf{x})}{\partial \mathbf{W}\mathbf{x}}$. Similar to Equation (5.4), we have

$$\begin{aligned} \left\| \nabla_{\mathbf{W}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}}) \right\|_F &= \|\mathbf{W}_L J\|_F \|\mathbf{x}\|_2 \\ \nabla_{\mathbf{x}} g(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}}) &= J\mathbf{W} \end{aligned} \tag{D.1}$$

Lemma D.1. $\|AB\|_F^2 \geq \lambda_{\min}(A^T A) \|B\|_F^2$, where $\lambda_{\min}$ is the smallest eigenvalue.

Proof. By the definition of Frobenius norm,

$$\|AB\|_F^2 = \text{Tr}(ABB^T A^T) = \text{Tr}(A^T ABB^T). \tag{D.2}$$

From Fang et al. [1994], we have that for positive semidefinite matrices P and Q ,

$$\lambda_{\min}(P) \operatorname{Tr}(Q) \leq \operatorname{Tr}(PQ) \quad (\text{D.3})$$

Therefore,

$$\operatorname{Tr}(A^T A B B^T) \geq \lambda_{\min}(A^T A) \operatorname{Tr}(B B^T) = \lambda_{\min}(A^T A) \|B\|_F^2 \quad (\text{D.4})$$

□

As a result, $\|\mathbf{W}_L J\|_F \geq \sqrt{\lambda_{\min}(\mathbf{W}_L^T \mathbf{W}_L)} \|J\|_F$. Let d be the feature dimension, and K be the number of classes, and $\mathbf{W}_L \in \mathbb{R}^{K \times d}$. Then, $\lambda_{\min}(\mathbf{W}_L^T \mathbf{W}_L)$ vanishes when $d > K$, otherwise $\sqrt{\lambda_{\min}(\mathbf{W}_L^T \mathbf{W}_L)} = \sigma_{\min}(\mathbf{W}_L)$, the smallest singular value of $\mathbf{W}_L$. Therefore,

$$\begin{aligned} \|\nabla_{\mathbf{x}} g(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})\|_F &= \|J\mathbf{W}\|_F \\ &\leq \|J\|_F \|\mathbf{W}\|_2 \\ &\leq \frac{\|\mathbf{W}_L J\|_F}{\sqrt{\lambda_{\min}(\mathbf{W}_L^T \mathbf{W}_L)}} \|\mathbf{W}\|_2 \\ &= \frac{\|\nabla_{\mathbf{W}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})\|_F}{\|\mathbf{x}\|_2 \sqrt{\lambda_{\min}(\mathbf{W}_L^T \mathbf{W}_L)}} \|\mathbf{W}\|_2 \end{aligned} \quad (\text{D.5})$$

More generally, if we consider arbitrary intermediate-layer representations, we write $f(x) = h \circ g(\mathbf{W}x)$, where g is the transformation from linear transformed input to the intermediate-layer representations and h is the mapping from the representations to the output of the network. Then Equation (D.1) becomes

$$\begin{aligned} \|\nabla_{\mathbf{W}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})\|_F &= \|J_h J_g\|_F \|\mathbf{x}\|_2 \\ \nabla_{\mathbf{x}} g(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}}) &= J_g \mathbf{W} , \end{aligned} \quad (\text{D.6})$$

where $J_g = \frac{\partial g(\mathbf{W}\mathbf{x})}{\partial \mathbf{W}\mathbf{x}}$ and $J_h = \frac{\partial f(\mathbf{W}\mathbf{x})}{\partial g(\mathbf{W}\mathbf{x})}$. Therefore, if J_h has full column rank, similar to

Equation (D.5), we have

$$\|\nabla_{\mathbf{x}}g(\mathbf{W}\mathbf{x};\bar{\boldsymbol{\theta}})\|_F \leq \frac{\|\nabla_{\mathbf{w}}f(\mathbf{W}\mathbf{x};\bar{\boldsymbol{\theta}})\|_F}{\|\mathbf{x}\|_2 \sqrt{\lambda_{\min}(J_h^T J_h)}} \|\mathbf{W}\|_2 \quad (\text{D.7})$$

D.2 Adaptation of Inequality 5.6 to Residual Layers

We need to slightly adapt the proof in Equations (5.4) and (5.5). Consider a network whose first layer has a residual connection: $y = g(x + f(Wx))$, where f is the nonlinearity with bias (e.g. $f(x) = \tanh(x + b)$), and g is the rest of the mappings in the network. Then we have

$$\begin{aligned} \|\nabla_W g(x + f(Wx))\|_F &= \|JK\|_F \|x\|_2 \\ \nabla_x g(x + f(Wx)) &= J + JKW \end{aligned} \quad (\text{D.8})$$

where $J = \frac{\partial g(x+f(Wx))}{\partial(x+f(Wx))}$ and $K = \frac{\partial f(Wx)}{\partial(Wx)}$.

Therefore, $\|\nabla_x g(x + f(Wx))\|_2 \leq \|J\|_2 + \|JK\|_2 \|W\|_2 \leq \|J\|_2 + \frac{\|\nabla_W g(x+f(Wx))\|_F}{\|x\|_2} \|W\|_2$.

Now, we get the bound for the *difference* between MLS of input and the MLS of input to the next layer:

$$\|\nabla_x g(x + f(Wx))\|_2 - \|J\|_2 \leq \frac{\|\nabla_W g(x + f(Wx))\|_F}{\|x\|_2} \|W\|_2 \quad (\text{D.9})$$

Notice that if we apply this inequality to every residual layer in the network, and sum the left-hand side, we will get a telescoping sum on the left-hand side. Assuming the last layer is linear with weights W_L , we get $\|\nabla_x g(x + f(W_1x))\|_2 - \|W_L\|_2 \leq \sum_{l=1}^{L-1} \frac{\|W_l\|_2}{\|x_l\|_2} \|\nabla_W g_l(x_l + f(W_l x_l))\|_F$. The right-hand side is bounded by sharpness due to Cauchy, see also Equation (D.26).

D.3 Proof of Equation (5.3)

Lemma D.2. *If $\boldsymbol{\theta}$ is an approximate interpolation solution, i.e. $\|f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i\| < \varepsilon$ for $i \in \{1, 2, \dots, n\}$, and second derivatives of the network function $\|\nabla_{\boldsymbol{\theta}}^2 f(\mathbf{x}_i, \boldsymbol{\theta})\| < M$ is*

bounded, then

$$S(\boldsymbol{\theta}^*) = \frac{1}{n} \sum_{i=1}^n \|\nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F^2 + O(\varepsilon) \quad (\text{D.10})$$

Proof. Using basic calculus we get

$$\begin{aligned} S(\boldsymbol{\theta}) &= \text{Tr}(\nabla^2 L(\boldsymbol{\theta})) \\ &= \frac{1}{2n} \sum_{i=1}^n \text{Tr}(\nabla_{\boldsymbol{\theta}}^2 \|f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i\|^2) \\ &= \frac{1}{2n} \sum_{i=1}^n \text{Tr} \nabla_{\boldsymbol{\theta}} (2(f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta})) \\ &= \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m \frac{\partial}{\partial \theta_j} ((f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta}))_j \\ &= \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m \frac{\partial}{\partial \theta_j} (f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\theta_j} f(\mathbf{x}_i, \boldsymbol{\theta}) \\ &= \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m \left\| \nabla_{\theta_j} f(\mathbf{x}_i, \boldsymbol{\theta}) \right\|_2^2 + (f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\theta_j}^2 f(\mathbf{x}_i, \boldsymbol{\theta}) \\ &= \frac{1}{n} \sum_{i=1}^n \|\nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta})\|_F^2 + \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m (f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\theta_j}^2 f(\mathbf{x}_i, \boldsymbol{\theta}). \end{aligned}$$

Therefore

$$\left| S(\boldsymbol{\theta}) - \frac{1}{n} \sum_{i=1}^n \|\nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta})\|_F^2 \right| < \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |(f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\theta_j}^2 f(\mathbf{x}_i, \boldsymbol{\theta})| < mM\varepsilon = O(\varepsilon). \quad (\text{D.11})$$

□

In other words, when the network reaches zero training error and enters the interpolation phase (i.e., it fits all training data correctly), Equation (5.3) will be a good enough approximation of the sharpness because the quadratic training loss is sufficiently small.

D.4 Proof of Equation (5.8), Proposition 5.4 and Proposition 5.6

D.4.1 Proof of Equation (5.8)

We prove the bound on the local volumetric ratio using the arithmetic-geometric mean inequality. Let $J = \nabla_{\mathbf{x}} f \in \mathbb{R}^{N \times M}$ have singular values $\sigma_1, \dots, \sigma_N \geq 0$. By the AM-GM inequality applied to the squared singular values:

$$\begin{aligned}
\sqrt{\det(JJ^T)} &= \sqrt{\prod_{i=1}^N \sigma_i^2} = \prod_{i=1}^N \sigma_i \\
&\leq \left(\frac{1}{N} \sum_{i=1}^N \sigma_i^2 \right)^{N/2} \quad (\text{AM-GM inequality}) \\
&= \left(\frac{\text{tr}(JJ^T)}{N} \right)^{N/2} \quad (\text{eigenvalues of } JJ^T \text{ are } \sigma_i^2) \\
&= \left(\frac{\text{tr}(J^T J)}{N} \right)^{N/2} \quad (\text{trace identity: } \text{tr}(AB) = \text{tr}(BA)) \\
&= \left(\frac{\|J\|_F^2}{N} \right)^{N/2} \quad (\text{Frobenius norm: } \|J\|_F^2 = \text{tr}(J^T J)).
\end{aligned} \tag{D.12}$$

This completes the proof of Equation (5.8).

D.4.2 Proof of Proposition 5.4 and Proposition 5.6

For notational simplicity, we write $f_i := f(\mathbf{x}_i, \boldsymbol{\theta}^*)$ in what follows. By Equation (5.5) and the Cauchy–Schwarz inequality, we have

$$\begin{aligned}
\frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f_i\|_F^k &\leq \|\mathbf{W}\|_2^k \frac{1}{n} \sum_{i=1}^n \frac{\|\nabla_{\mathbf{w}} f_i\|_F^k}{\|\mathbf{x}_i\|_2^k} \\
&\leq \frac{1}{n} \|\mathbf{W}\|_2^k \sqrt{\sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^{2k}}} \cdot \sqrt{\sum_{i=1}^n \|\nabla_{\mathbf{w}} f_i\|_F^{2k}}.
\end{aligned} \tag{D.13}$$

Since the input-weight matrix $\mathbf{W}$ is just one part of all the weights ($\boldsymbol{\theta}$) of the network, we have $\|\nabla_{\mathbf{W}} f_i\|_F^k \leq \|\nabla_{\boldsymbol{\theta}} f_i\|_F^k$.

We next show the correctness of Proposition 5.4 with a standard lemma.

Lemma D.3. *For vector $\mathbf{x}$, $\|\mathbf{x}\|_p \geq \|\mathbf{x}\|_q$ for $1 \leq p \leq q \leq \infty$.*

Proof. First we show that for $0 < k < 1$, we have $(|a| + |b|)^k \leq |a|^k + |b|^k$. It's trivial when either a or b is 0. So W.L.O.G, we can assume that $|a| < |b|$, and divide both sides by $|b|^k$. Therefore it suffices to show that for $0 < t < 1$, $(1+t)^k < t^k + 1$. Let $f(t) = (1+t)^k - t^k - 1$, then $f(0) = 0$, and $f'(t) = k(1+t)^{k-1} - kt^{k-1}$. Because $k-1 < 0$, $1+t > 1$ and $t < 1$, $t^{k-1} > 1 > (1+t)^{k-1}$. Therefore $f'(t) < 0$ and $f(t) < 0$ for $0 < t < 1$. Combining all cases, we have $(|a| + |b|)^k \leq |a|^k + |b|^k$ for $0 < k < 1$. By induction, we have $(\sum_n |a_n|)^k \leq \sum_n |a_n|^k$.

Now we can prove the lemma using the conclusion above,

$$\left(\sum_n |x_n|^q\right)^{1/q} = \left(\sum_n |x_n|^q\right)^{p/q \cdot 1/p} \leq \left(\sum_n (|x_n|^q)^{p/q}\right)^{1/p} = \left(\sum_n |x_n|^p\right)^{1/p}$$

□

Now we can prove Proposition 5.4

Proposition. *The local volumetric ratio is upper bounded by a sharpness related quantity:*

$$dV_{f(\boldsymbol{\theta}^*)} \leq \frac{N^{-N/2}}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F^N \leq \frac{1}{n} \sqrt{\sum_{i=1}^n \frac{\|\mathbf{W}\|_2^{2N}}{\|\mathbf{x}_i\|_2^{2N}}} \left(\frac{nS(\boldsymbol{\theta}^*)}{N}\right)^{N/2} \quad (\text{D.14})$$

for all $N \geq 1$.

Proof. Take the x_i in Lemma D.3 to be $\|\nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F^2$ and let $p = 1, q = k$, then we get

$$\left(\sum_{i=1}^n (\|\nabla_{\boldsymbol{\theta}} f_i\|_F^2)^k\right)^{1/k} \leq \sum_{i=1}^n \|\nabla_{\boldsymbol{\theta}} f_i\|_F^2. \quad (\text{D.15})$$

Therefore,

$$\begin{aligned}
\frac{1}{n} \|\mathbf{W}\|_2^k \sqrt{\sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^{2k}}} \cdot \sqrt{\sum_{i=1}^n \|\nabla_{\mathbf{W}} f_i\|_F^{2k}} &\leq n^{k/2-1} \|\mathbf{W}\|_2^k \sqrt{\sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^{2k}}} \left(\frac{1}{n} \sum_{i=1}^n \|\nabla_{\boldsymbol{\theta}} f_i\|_F^2 \right)^{k/2} \\
&= n^{k/2-1} \|\mathbf{W}\|_2^k \sqrt{\sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^{2k}}} S(\boldsymbol{\theta}^*)^{k/2}
\end{aligned} \tag{D.16}$$

□

Next, we show that the first inequality in Equation (D.16) can be tightened by considering all linear layer weights.

Proposition. *The network volumetric ratio is upper bounded by a sharpness related quantity:*

$$\sum_{l=1}^L dV_{f_l} \leq \frac{N^{-N/2}}{n} \sum_{l=1}^L \sum_{i=1}^n \|\nabla_{\mathbf{x}^l} f_i^l\|_F^N \leq \frac{1}{n} \sqrt{\sum_{l=1}^L \sum_{i=1}^n \frac{\|\mathbf{W}_l\|_2^{2N}}{\|\mathbf{x}_i^l\|_2^{2N}}} \cdot \left(\frac{nS(\boldsymbol{\theta}^*)}{N} \right)^{N/2}. \tag{D.17}$$

Proof. Recall that the input to the l -th linear layer is x_i^l for $l = 1, 2, \dots, L$. In particular, x_i^1 is the input to the entire network. Similarly, $\mathbf{W}_l$ is the weight matrix of the l -th linear or convolutional layer. With a slight abuse of notation, we use f^l to denote the mapping from the activity of the l -th layer to the final output, and $f_i^l := f^l(\mathbf{x}_i, \boldsymbol{\theta}^*)$. We can apply the Cauchy–Schwarz inequality again to obtain

$$\begin{aligned}
\frac{1}{n} \sum_{l=1}^L \sum_{i=1}^n \|\nabla_{\mathbf{x}^l} f_i^l\|_F^k &\leq \frac{1}{n} \sum_{l=1}^L \sqrt{\sum_{i=1}^n \frac{\|\mathbf{W}_l\|_2^{2k}}{\|\mathbf{x}_i^l\|_2^{2k}}} \cdot \sqrt{\sum_{i=1}^n \|\nabla_{\mathbf{W}_l} f_i^l\|_F^{2k}} \\
&\leq \sqrt{\frac{1}{n} \sum_{l=1}^L \sum_{i=1}^n \frac{\|\mathbf{W}_l\|_2^{2k}}{\|\mathbf{x}_i^l\|_2^{2k}}} \cdot \sqrt{\frac{1}{n} \sum_{l=1}^L \sum_{i=1}^n \|\nabla_{\mathbf{W}_l} f_i^l\|_F^{2k}}.
\end{aligned} \tag{D.18}$$

Using Lemma D.3 again we have

$$\begin{aligned} \left(\sum_{l=1}^L (\|\nabla_{\mathbf{w}_l} f_i\|_F^2)^k \right)^{1/k} &\leq \sum_{l=1}^L \|\nabla_{\mathbf{w}_l} f_i\|_F^2 \leq \|\nabla_{\boldsymbol{\theta}} f_i\|_F^2, \\ \left(\sum_{i=1}^n (\|\nabla_{\boldsymbol{\theta}} f_i\|_F^2)^k \right)^{1/k} &\leq \sum_{i=1}^n \|\nabla_{\boldsymbol{\theta}} f_i\|_F^2 = nS(\boldsymbol{\theta}^*), \end{aligned} \quad (\text{D.19})$$

The last inequality allows $\boldsymbol{\theta}$ to include parameters beyond the listed linear weights. Therefore from Equation (D.18), we have

$$\frac{1}{n} \sum_{l=1}^L \sum_{i=1}^n \|\nabla_{\mathbf{x}^l} f_i^l\|_F^k \leq \sqrt{\frac{1}{n} \sum_{l=1}^L \sum_{i=1}^n \frac{\|\mathbf{W}_l\|_2^{2k}}{\|\mathbf{x}_i^l\|_2^{2k}}} \cdot \sqrt{n^{k-1} S(\boldsymbol{\theta}^*)^k} \quad (\text{D.20})$$

□

D.5 Proof of Proposition 5.8 and Proposition 5.10

Below we give the proof of Proposition 5.8.

Proposition. *The maximum local sensitivity is upper bounded by a sharpness related quantity:*

$$\text{MLS} = \frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_2 \leq \|\mathbf{W}\|_2 \sqrt{\frac{1}{n} \sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^2} S(\boldsymbol{\theta}^*)^{1/2}}. \quad (\text{D.21})$$

Proof. From Equation (5.5), we get

$$\text{MLS} = \frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f_i\|_2 \leq \|\mathbf{W}\|_2 \frac{1}{n} \sum_{i=1}^n \frac{\|\nabla_{\mathbf{w}} f_i\|_F}{\|\mathbf{x}_i\|_2}. \quad (\text{D.22})$$

Now the Cauchy-Schwarz inequality tells us that

$$\left(\sum_{i=1}^n \frac{\|\nabla_{\mathbf{w}} f_i\|_F}{\|\mathbf{x}_i\|_2} \right)^2 \leq \left(\sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^2} \right) \cdot \left(\sum_{i=1}^n \|\nabla_{\mathbf{w}} f_i\|_F^2 \right). \quad (\text{D.23})$$

Therefore

$$\begin{aligned}
\text{MLS} &\leq \|\mathbf{W}\|_2 \sqrt{\frac{1}{n} \sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^2}} \cdot \sqrt{\frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{W}} f_i\|_F^2} \\
&\leq \|\mathbf{W}\|_2 \sqrt{\frac{1}{n} \sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^2}} \cdot S(\boldsymbol{\theta}^*)^{1/2}.
\end{aligned} \tag{D.24}$$

□

Now we can prove Proposition 5.10.

Proposition. *The network maximum local sensitivity is upper bounded by a sharpness related quantity:*

$$\text{NMLS} = \frac{1}{n} \sum_{l=1}^L \sum_{i=1}^n \left\| \nabla_{\mathbf{x}^l} f^l(\mathbf{x}_i^l, \boldsymbol{\theta}^*) \right\|_2 \leq \sqrt{\frac{1}{n} \sum_{i=1}^n \sum_{l=1}^L \frac{\|\mathbf{W}_l\|_2^2}{\|\mathbf{x}_i^l\|_2^2}} \cdot S(\boldsymbol{\theta}^*)^{1/2}. \tag{D.25}$$

Proof. We can apply Equation (D.24) to every linear layer and again apply the Cauchy-Schwarz inequality to obtain

$$\begin{aligned}
\text{NMLS} &= \frac{1}{n} \sum_{l=1}^L \sum_{i=1}^n \left\| \nabla_{\mathbf{x}} f_l(\mathbf{x}_i^l, \boldsymbol{\theta}^*) \right\|_2 \\
&\leq \sum_{l=1}^L \left(\sqrt{\frac{1}{n} \sum_{i=1}^n \frac{\|\mathbf{W}_l\|_2^2}{\|\mathbf{x}_i^l\|_2^2}} \sqrt{\frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{W}_l} f_i^l\|_F^2} \right) \\
&\leq \sqrt{\frac{1}{n} \sum_{i=1}^n \sum_{l=1}^L \frac{\|\mathbf{W}_l\|_2^2}{\|\mathbf{x}_i^l\|_2^2}} \sqrt{\frac{1}{n} \sum_{i=1}^n \sum_{l=1}^L \|\nabla_{\mathbf{W}_l} f_i^l\|_F^2} \\
&\leq \sqrt{\frac{1}{n} \sum_{i=1}^n \sum_{l=1}^L \frac{\|\mathbf{W}_l\|_2^2}{\|\mathbf{x}_i^l\|_2^2}} \cdot S(\boldsymbol{\theta}^*)^{1/2}.
\end{aligned} \tag{D.26}$$

Note that the gap in the last inequality is significantly smaller than that of Equation (D.24) since now we consider all linear weights. □

D.6 Reparametrization-invariant sharpness and input-invariant MLS

In this section, we demonstrate that various reparametrization-invariant sharpness metrics proposed in the literature can be understood as measuring robustness of outputs to inputs and internal representations through MLS-like quantities. As discussed in Section 3.4, this provides a novel perspective: reparametrization-invariant sharpness is fundamentally characterized by the robustness of outputs to internal neural representations. We examine two different approaches from the literature—Tsuzuku et al. [2019]’s matrix-normalized and normalized sharpness (Section D.6.1), and Kwon et al. [2021], Andriushchenko et al. [2023]’s elementwise-adaptive sharpness (Section D.6.2)—showing how each relates to different variants of MLS.

D.6.1 Reparametrization-invariant sharpness in Tsuzuku et al. [2019]

In this appendix, we show that the reparametrization-invariant sharpness metrics introduced in Tsuzuku et al. [2019] can be seen as an effort to tighten the bound that we derived above. We formalize two metrics from their work and establish their connections to MLS.

Definition D.4 (Matrix-normalized sharpness). *Following Tsuzuku et al. [2019], the **matrix-normalized sharpness** of a neural network is defined as*

$$S_{\text{matrix}}(\boldsymbol{\theta}) = \sum_{l=1}^L \|\mathbf{W}_l\|_2 \sqrt{S(\mathbf{W}_l)}, \quad (\text{D.27})$$

where $S(\mathbf{W}_l)$ is the trace of Hessian of the loss w.r.t. the weights of the l -th layer, and $\|\mathbf{W}_l\|_2$ is the spectral norm of the weight matrix at layer l .

Proposition D.5. *At exact interpolation under MSE, the matrix-normalized sharpness*

upper-bounds weighted MLS across layers:

$$\sum_{l=1}^L \bar{\mathbf{x}}^l \cdot \text{MLS}^l \leq S_{\text{matrix}}(\boldsymbol{\theta}), \quad (\text{D.28})$$

where $\bar{\mathbf{x}}^l = \left(\frac{1}{n} \sum_{i=1}^n \frac{1}{\|\mathbf{x}_i^l\|_2^2} \right)^{-\frac{1}{2}}$ is the inverse of the average input norm at layer l , and MLS^l is the MLS at layer l .

Proof. At exact interpolation, from Equation (D.24) we have

$$\begin{aligned} \sum_{l=1}^L \bar{\mathbf{x}}^l \cdot \text{MLS}^l &\leq \sum_{l=1}^L \|\mathbf{W}_l\|_2 \sqrt{\frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{W}_l} f_i^l\|_F^2} \\ &= \sum_{l=1}^L \|\mathbf{W}_l\|_2 \sqrt{S(\mathbf{W}_l)} = S_{\text{matrix}}(\boldsymbol{\theta}). \end{aligned} \quad (\text{D.29})$$

Note that a similar inequality holds if we use Frobenius norm instead of 2-norm of the weights. $\square$

Definition D.6 (Normalized sharpness [Tsuzuku et al., 2019]). *The **normalized sharpness** of a neural network is defined as the solution to the optimization problem:*

$$S_{\text{normalized}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\sigma}, \boldsymbol{\sigma}'} \sum_{i,j} \left(\frac{\partial^2 L}{\partial W_{i,j} \partial W_{i,j}} (\sigma_i \sigma'_j)^2 + \frac{W_{i,j}^2}{4\lambda^2 (\sigma_i \sigma'_j)^2} \right), \quad (\text{D.30})$$

where $\boldsymbol{\sigma} \in \mathbb{R}_{>0}^d$ and $\boldsymbol{\sigma}' \in \mathbb{R}_{>0}^M$ are scaling vectors for $\mathbf{W} \in \mathbb{R}^{d \times M}$ and λ is a hyperparameter.

Proposition D.7. *The normalized sharpness optimization problem is equivalent to minimizing an upper bound on a scale-invariant MLS-like quantity:*

$$S_{\text{normalized}}(\boldsymbol{\theta}) \geq \frac{1}{\lambda} \left\| \text{diag}(\boldsymbol{\sigma}'^{-1}) \nabla_{\mathbf{x}} f \right\|_F \left\| \text{diag}(\boldsymbol{\sigma}') \mathbf{x} \right\|_2, \quad (\text{D.31})$$

where the quantity on the right is invariant under the transformation

$$\mathbf{W}\mathbf{x} \rightarrow \mathbf{W} \text{diag}(\boldsymbol{\sigma}'^{-1}) (\text{diag}(\boldsymbol{\sigma}') \mathbf{x}).$$

Proof. Note that by Lemma D.2, $\frac{\partial^2 L}{\partial W_{i,j} \partial W_{i,j}} \approx \|\nabla_{\mathbf{W}_{i,j}} f\|_2^2$. Moreover, we have

$$\begin{aligned}
& \sum_{i,j} \|\nabla_{\mathbf{W}_{i,j}} f\|^2 (\sigma_i \sigma'_j)^2 + \sum_{i,j} \frac{W_{i,j}^2}{4\lambda^2 (\sigma_i \sigma'_j)^2} \\
& \geq \frac{1}{\lambda} \sqrt{\sum_{i,j} (\nabla_{\mathbf{W}_{i,j}} f)^2 (\sigma_i \sigma'_j)^2} \\
& \quad \cdot \sqrt{\sum_{i,j} \frac{W_{i,j}^2}{(\sigma_i \sigma'_j)^2}} \\
& \geq \frac{1}{\lambda} \|\text{diag}(\boldsymbol{\sigma}) J\|_F \|\text{diag}(\boldsymbol{\sigma}') \mathbf{x}\|_2 \|\text{diag}(\boldsymbol{\sigma}^{-1}) \mathbf{W} \text{diag}(\boldsymbol{\sigma}'^{-1})\|_F \\
& \geq \frac{1}{\lambda} \|\text{diag}(\boldsymbol{\sigma}'^{-1}) W^T J\|_F \|\text{diag}(\boldsymbol{\sigma}') \mathbf{x}\|_2 \\
& = \frac{1}{\lambda} \|\text{diag}(\boldsymbol{\sigma}'^{-1}) \nabla_{\mathbf{x}} f\|_F \|\text{diag}(\boldsymbol{\sigma}') \mathbf{x}\|_2,
\end{aligned} \tag{D.32}$$

where $J = \frac{\partial f(\mathbf{W}\mathbf{x}; \boldsymbol{\theta})}{\partial(\mathbf{W}\mathbf{x})}$ (see some of the calculations in Equation (5.4)). Therefore, the optimization problem Equation (D.30) is equivalent to choosing $\boldsymbol{\sigma}, \boldsymbol{\sigma}'$ to minimize the upper bound on a scale-invariant MLS-like quantity. The quantity is invariant under the transformation of the first layer

$$\mathbf{W}\mathbf{x} \rightarrow \mathbf{W} \text{diag}(\boldsymbol{\sigma}'^{-1})(\text{diag}(\boldsymbol{\sigma}') \mathbf{x}),$$

where $\text{diag}(\boldsymbol{\sigma}') \mathbf{x}$ becomes the new input. $\square$

For simplicity, we do not scale the original dataset in our work and only compare MLS within the same dataset. As a result, we can characterize those reparametrization-invariant sharpness metrics by the robustness of output to the input. If we consider all linear weights in the network, then those metrics indicate the robustness of output to internal network representations.

In summary, we have shown that Tsuzuku et al. [2019]’s reparametrization-invariant sharpness metrics can be interpreted through our MLS framework: their matrix-normalized sharpness corresponds exactly to the upper bound on weighted MLS across layers, while their normalized sharpness minimizes an upper bound on a scale-invariant MLS-like quantity. This establishes that these metrics, while designed for reparametrization invariance, are

fundamentally measuring the robustness of outputs to inputs and internal representations.

D.6.2 Reparametrization-invariant sharpness upper-bounds input-invariant MLS

Having shown that Tsuzuku et al. [2019]’s metrics tighten our standard MLS/NMLS bounds, we now examine a different approach to reparametrization-invariant sharpness. In this subsection, we prove a formal result relating elementwise-adaptive sharpness from Kwon et al. [2021], Andriushchenko et al. [2023] to a different variant of MLS: input-invariant MLS. This demonstrates that different reparametrization-invariant sharpness metrics from the literature can be understood as measuring robustness of outputs to inputs through different MLS variants.

In this appendix, we consider the adaptive average-case n-sharpness considered in Kwon et al. [2021], Andriushchenko et al. [2023]:

$$S_{\text{avg}}^\rho(\mathbf{w}, |\mathbf{w}|) \triangleq \frac{2}{\rho^2} \mathbb{E}_{S \sim P_n, \delta \sim \mathcal{N}(0, \rho^2 \text{diag}(|\mathbf{w}|^2))} [L_S(\mathbf{w} + \delta) - L_S(\mathbf{w})], \quad (\text{D.33})$$

which is shown to be *elementwise* adaptive sharpness in Andriushchenko et al. [2023]. They also show that for a thrice differentiable loss, $L(w)$, the average-case elementwise adaptive sharpness can be written as

$$S_{\text{avg}}^\rho(\mathbf{w}, |\mathbf{w}|) = \mathbb{E}_{S \sim P_n} \left[\text{Tr} \left(\nabla^2 L_S(\mathbf{w}) \odot |\mathbf{w}| |\mathbf{w}|^\top \right) \right] + O(\rho). \quad (\text{D.34})$$

Definition D.8. We define the ***Elementwise-Adaptive Sharpness*** S_{adaptive} to be

$$S_{\text{adaptive}}(\mathbf{w}) \triangleq \lim_{\rho \rightarrow 0} S_{\text{avg}}^\rho(\mathbf{w}, |\mathbf{w}|) = \mathbb{E}_{S \sim P_n} \left[\text{Tr} \left(\nabla^2 L_S(\mathbf{w}) \odot |\mathbf{w}| |\mathbf{w}|^\top \right) \right] \quad (\text{D.35})$$

In this appendix, we focus on the property of S_{adaptive} instead of the approximation Equation (D.34). Adapting the proof of *Theorem D.2*, we have the following lemma.

Lemma D.9. *If $\boldsymbol{\theta}$ is an approximate interpolation solution, i.e. $\|f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i\| < \varepsilon$ for $i \in \{1, 2, \dots, n\}$, $|\boldsymbol{\theta}_j|^2 \|\nabla_{\boldsymbol{\theta}_j}^2 f(\mathbf{x}_i, \boldsymbol{\theta})\| < M$ for all j , and L is MSE loss, then*

$$S_{\text{adaptive}}(\boldsymbol{\theta}^*) = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |\boldsymbol{\theta}_j|^2 \|\nabla_{\boldsymbol{\theta}_j} f(\mathbf{x}_i, \boldsymbol{\theta})\|_2^2 + O(\varepsilon), \quad (\text{D.36})$$

where m is the number of parameters.

Proof. Using basic calculus we get

$$\begin{aligned} S_{\text{adaptive}}(\boldsymbol{\theta}) &= \frac{1}{2n} \sum_{i=1}^n \text{Tr}(\nabla_{\boldsymbol{\theta}}^2 \|f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i\|^2 \odot |\boldsymbol{\theta}| |\boldsymbol{\theta}|^\top) \\ &= \frac{1}{2n} \sum_{i=1}^n \text{Tr} \nabla_{\boldsymbol{\theta}} (2(f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta})) \odot |\boldsymbol{\theta}| |\boldsymbol{\theta}|^\top \\ &= \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |\boldsymbol{\theta}_j|^2 \frac{\partial}{\partial \boldsymbol{\theta}_j} ((f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i, \boldsymbol{\theta}))_j \\ &= \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |\boldsymbol{\theta}_j|^2 \frac{\partial}{\partial \boldsymbol{\theta}_j} (f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\boldsymbol{\theta}_j} f(\mathbf{x}_i, \boldsymbol{\theta}) \\ &= \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |\boldsymbol{\theta}_j|^2 \|\nabla_{\boldsymbol{\theta}_j} f(\mathbf{x}_i, \boldsymbol{\theta})\|_2^2 + |\boldsymbol{\theta}_j|^2 (f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\boldsymbol{\theta}_j}^2 f(\mathbf{x}_i, \boldsymbol{\theta}) \\ &= \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |\boldsymbol{\theta}_j|^2 \|\nabla_{\boldsymbol{\theta}_j} f(\mathbf{x}_i, \boldsymbol{\theta})\|_2^2 + \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |\boldsymbol{\theta}_j|^2 (f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \nabla_{\boldsymbol{\theta}_j}^2 f(\mathbf{x}_i, \boldsymbol{\theta}) \end{aligned}$$

Therefore

$$\begin{aligned} &\left| S_{\text{adaptive}}(\boldsymbol{\theta}) - \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |\boldsymbol{\theta}_j|^2 \|\nabla_{\boldsymbol{\theta}_j} f(\mathbf{x}_i, \boldsymbol{\theta})\|_2^2 \right| \\ &< \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |(f(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{y}_i)^T \boldsymbol{\theta}_j|^2 \nabla_{\boldsymbol{\theta}_j}^2 f(\mathbf{x}_i, \boldsymbol{\theta})| \\ &< mM\varepsilon = O(\varepsilon). \end{aligned} \quad (\text{D.37})$$

□

Definition D.10. We define the **Input-invariant MLS** of a network $f : \mathbb{R}^M \rightarrow \mathbb{R}^N$ to be

$$\frac{1}{n} \sum_{i=1}^n \sum_{p=1}^M \left\| \nabla_{x_p^{(i)}} f \right\|_2^2 (x_p^{(i)})^2, \quad (\text{D.38})$$

where $x_p^{(i)}$ is the p -th entry of i -th training sample.

At exact interpolation, the adaptive sharpness upper bounds the input-invariant MLS.

Proposition D.11. Assuming that the conditions of Theorem D.9 hold with $\varepsilon = 0$, then elementwise-adaptive sharpness upper-bounds input-invariant MLS:

$$\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m |\theta_j|^2 \left\| \nabla_{\theta_j} f(\mathbf{x}_i, \boldsymbol{\theta}) \right\|_2^2 \geq \frac{1}{nd} \sum_{i=1}^n \sum_{p=1}^M \left\| \nabla_{x_p^{(i)}} f \right\|_2^2 (x_p^{(i)})^2, \quad (\text{D.39})$$

Proof. Now we adapt the linear stability trick. Let $\mathbf{W} \in \mathbb{R}^{d \times M}$ be the first-layer weights. Since these weights are a subset of the m parameters, Cauchy–Schwarz gives

$$\begin{aligned} \sum_{j=1}^m |\theta_j|^2 \left\| \nabla_{\theta_j} f(\mathbf{x}, \boldsymbol{\theta}) \right\|_2^2 &\geq \sum_{p=1}^M \sum_{k=1}^d W_{kp}^2 x_p^2 \|J_{:k}\|_2^2 \\ &\geq \frac{1}{d} \sum_{p=1}^M \left\| \nabla_{x_p} f \right\|_2^2 x_p^2 \end{aligned} \quad (\text{D.40})$$

where, as in Equation (5.4), $J = \frac{\partial f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})}{\partial(\mathbf{W}\mathbf{x})}$ and $\nabla_{\mathbf{x}} f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}}) = J\mathbf{W}$. Taking the sample mean proves the proposition. □

D.7 Numerical approximation of normalized MLS and elementwise-adaptive sharpness

In this appendix, we detail how we approximate the normalized MLS and adaptive sharpness in Section 5.4.3. Note that for all network f the last layer is the sigmoid function, so the output is bounded in $(0, 1)$, and we use MSE loss to be consistent with the rest of the chapter.

For the adaptive sharpness, we adopt the definition in Andriushchenko et al. [2023] and use the sample mean to approximate the expectation in Equation (D.33). Therefore, for network $f(\mathbf{w})$,

$$S_{\text{adaptive}}(f) = \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m L(\mathbf{x}_i; \mathbf{w} + \delta_j) - L(\mathbf{x}_i; \mathbf{w}), \quad (\text{D.41})$$

where $\delta \sim \mathcal{N}(0, 0.01 \text{diag}(|w|^2))$.

For normalized MLS, we first reiterate the definition from the main text. We use normalized MLS below as an approximation to the input-invariant MLS (Definition D.10), because the latter is computationally prohibitive for modern large ViTs. On the other hand, there is an efficient way to estimate normalized MLS as detailed below.

Definition D.12. *We define the **normalized MLS** as $\frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i\|_2^2 \|\nabla_{\mathbf{x}_i} f\|_2^2$*

Therefore, to approximate normalized MLS, we need to approximate $\|\nabla_{\mathbf{x}_i} f\|_2$. By definition of matrix 2-norm,

$$\|\nabla_{\mathbf{x}} f\|_2 = \sup_{\delta} \frac{\|\nabla_{\mathbf{x}} f \delta\|_2}{\|\delta\|_2} \approx \max_{\delta} \frac{\|f(\mathbf{x} + \delta) - f(\mathbf{x})\|_2}{\|\delta\|_2}. \quad (\text{D.42})$$

To solve this optimization problem, we start from a randomly sampled vector δ that has the same shape as the network input, and we update δ using gradient descent.

D.8 Empirical analysis of the bound

D.8.1 Tightness of the bound

In this section, we mainly explore the tightness of the bound in Equation (5.11) for reasons discussed in Section 5.3.2. First we rewrite Equation (5.11) as

$$\begin{aligned}
\text{MLS} &= \frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{x}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_2 &&:= A \\
&\leq \frac{\|\mathbf{W}\|_2}{n} \sum_{i=1}^n \frac{\|\nabla_{\mathbf{w}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F}{\|\mathbf{x}_i\|_2} &&:= B \\
&\leq \|\mathbf{W}\|_2 \sqrt{\frac{1}{n} \sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^2}} \sqrt{\frac{1}{n} \sum_{i=1}^n \|\nabla_{\mathbf{w}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F^2} &&:= C \\
&\leq \|\mathbf{W}\|_2 \sqrt{\frac{1}{n} \sum_{i=1}^n \frac{1}{\|\mathbf{x}_i\|_2^2}} S(\boldsymbol{\theta}^*)^{1/2} &&:= D
\end{aligned} \tag{D.43}$$

Thus Equation (5.11) consists of 3 different steps of relaxations. We analyze them one by one:

1. ($A \leq B$) The equality holds when $\|JW\|_2 = \|W\|_2 \|J\|_2$ and $\|J\|_F = \|J\|_2$, where $J = \frac{\partial f(\mathbf{W}\mathbf{x}; \bar{\boldsymbol{\theta}})}{\partial(\mathbf{W}\mathbf{x})}$. The former equality requires a leading right singular vector of J to align with a leading left singular vector of W . The latter requires J to have zero singular values except for the largest singular value. Since J depends on the specific neural network architecture and training process, we test the tightness of this bound empirically (Figure D.8.1).
2. ($B \leq C$) The equality requires $\frac{\|\nabla_{\mathbf{w}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F}{\|\mathbf{x}_i\|_2}$ to be the same for all i . In other words, the bound is tight when $\frac{\|\nabla_{\mathbf{w}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F}{\|\mathbf{x}_i\|_2}$ does not vary too much from sample to sample.
3. ($C \leq D$) The equality holds if the model is linear, i.e. $\boldsymbol{\theta} = \mathbf{W}$.

We empirically verify the tightness of the above bounds in Figure D.8.1

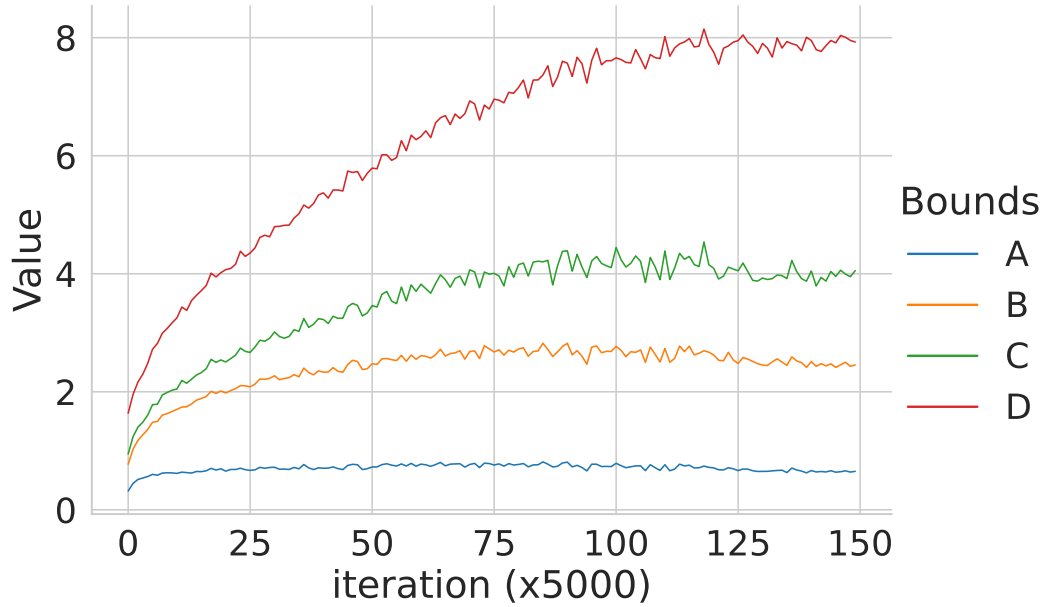


Figure D.8.1: **Empirical tightness of the bounds.** We empirically verify that the inequalities in Equation (D.43) hold and test their tightness. The results are shown for a fully connected feedforward network trained on the FashionMNIST dataset. The quantities A, B, C, and D are defined in Equation (D.43). We see that the gap between C and D is large compared to the gap between A and B or B and C. This indicates that partial sharpness $\|\nabla_{\mathbf{w}} f(\mathbf{x}_i, \boldsymbol{\theta}^*)\|_F$ (sensitivity of the loss w.r.t. only the input weights) is more indicative of the change in the maximum local sensitivity (A). Indeed, correlation analysis shows that bound C is positively correlated with MLS while bound D, perhaps surprisingly, is negatively correlated with MLS (Figure D.8.3).

D.8.2 Correlation analysis

We empirically show how different metrics correlate with each other, and how these correlations can be predicted from our bounds. We train 100 VGG-11 networks with different batch sizes, learning rates, and random initialization to classify images from the CIFAR-10 dataset, and plot pairwise scatter plots between different quantities at the end of the training: local dimensionality, sharpness (square root of Equation (5.3)), log volume (Equation (5.7)), MLS (Equation (5.11)), NMLS (Equation (5.12)), generalization gap (gen gap), D (Equation (D.43)), bound (right-hand side of Equation (5.12)) and relative sharpness [Petzka et al., 2021] (see Figure D.8.2). We only include CIFAR-10 data with 2 labels to ensure that the final training

accuracy is close to 100%.

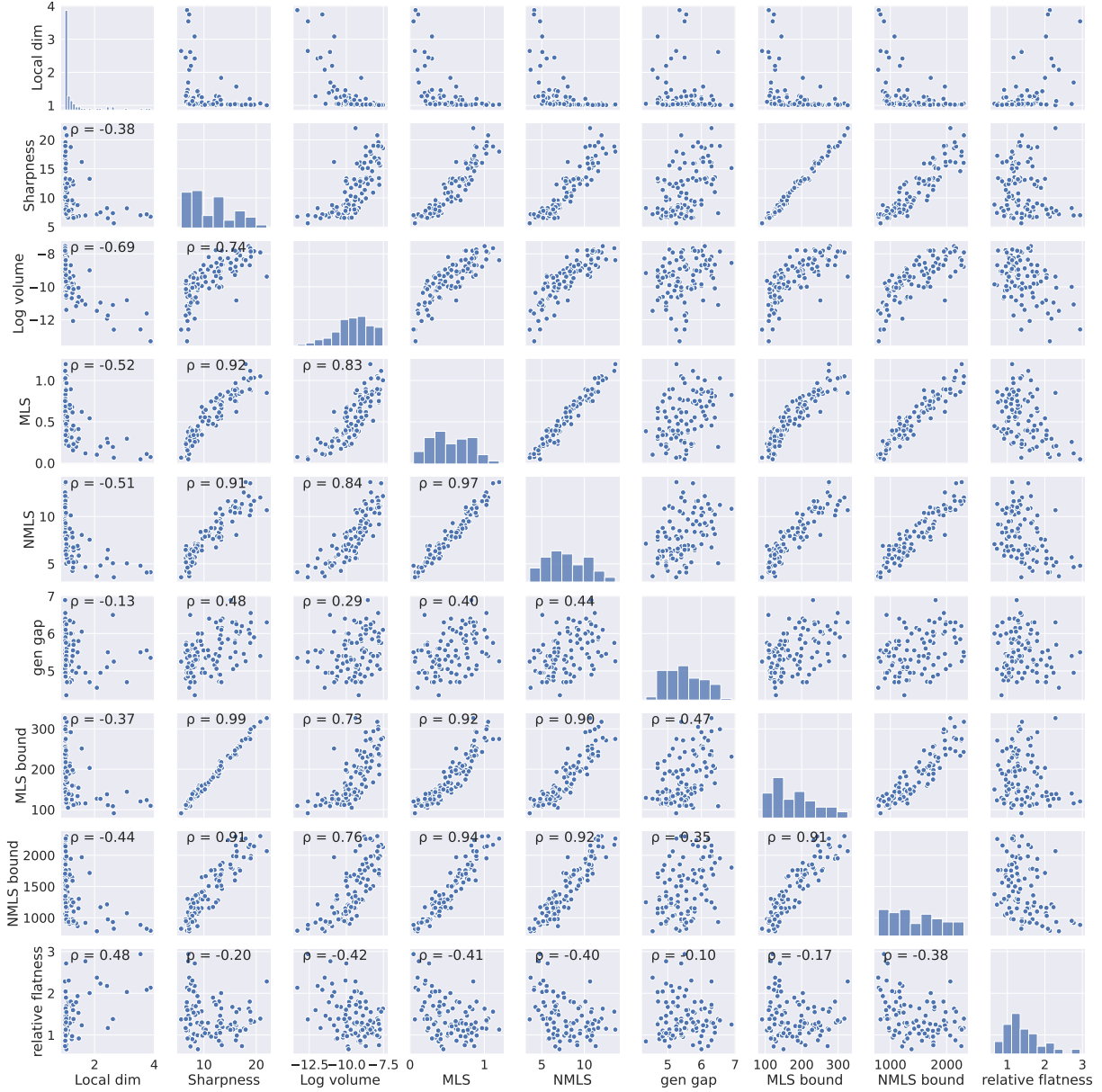


Figure D.8.2: **Pairwise correlation among different metrics.** We trained 100 different VGG-11 networks on the CIFAR-10 dataset using vanilla SGD with different learning rates, batch sizes, and random initializations and plot pairwise scatter plots between different quantities: local dimensionality, sharpness (square root of Equation (5.3)), log volume (Equation (5.7)), MLS (Equation (5.11)), NMLS (Equation (5.12)), generalization gap (gen gap), MLS bound (Proposition 5.8), NMLS bound (Proposition 5.10) and relative sharpness ([Petzka et al., 2021]). The Pearson correlation coefficient ρ is shown in the top-left corner for each pair of quantities. See Appendix D.8.2 for a summary of the findings in this figure.

We repeat the analysis on MLPs and LeNets trained on the FashionMNIST dataset and the CIFAR-10 dataset (Figure D.8.3 and Figure D.8.4). We find that

1. The bound over NMLS, MLS, and NMLS introduced in Equation (5.12) and Equation (5.11) consistently correlates positively with the generalization gap.
2. Although the bound in Equation (5.9) is loose, log volume correlates well with sharpness and MLS.
3. Sharpness is positively correlated with the generalization gap, indicating that little reparametrization effect [Dinh et al., 2017] is happening during training, i.e. the network weights do not change too much during training. This is consistent with observations in Ma and Ying [2021].
4. The bound derived in Equation (5.12) correlates positively with NMLS in all experiments.
5. MLS that only consider the first layer weights can sometimes negatively correlate with the bound derived in Equation (5.11) (Figure D.8.3).
6. Relative flatness that only consider the last layer weights introduced in [Petzka et al., 2021] shows weak (even negative) correlation with the generalization gap. Note that “relative flatness” is a misnomer that is easier understood as “relative *sharpness*”, and is supposed to be *positively* correlated with the generalization gap.

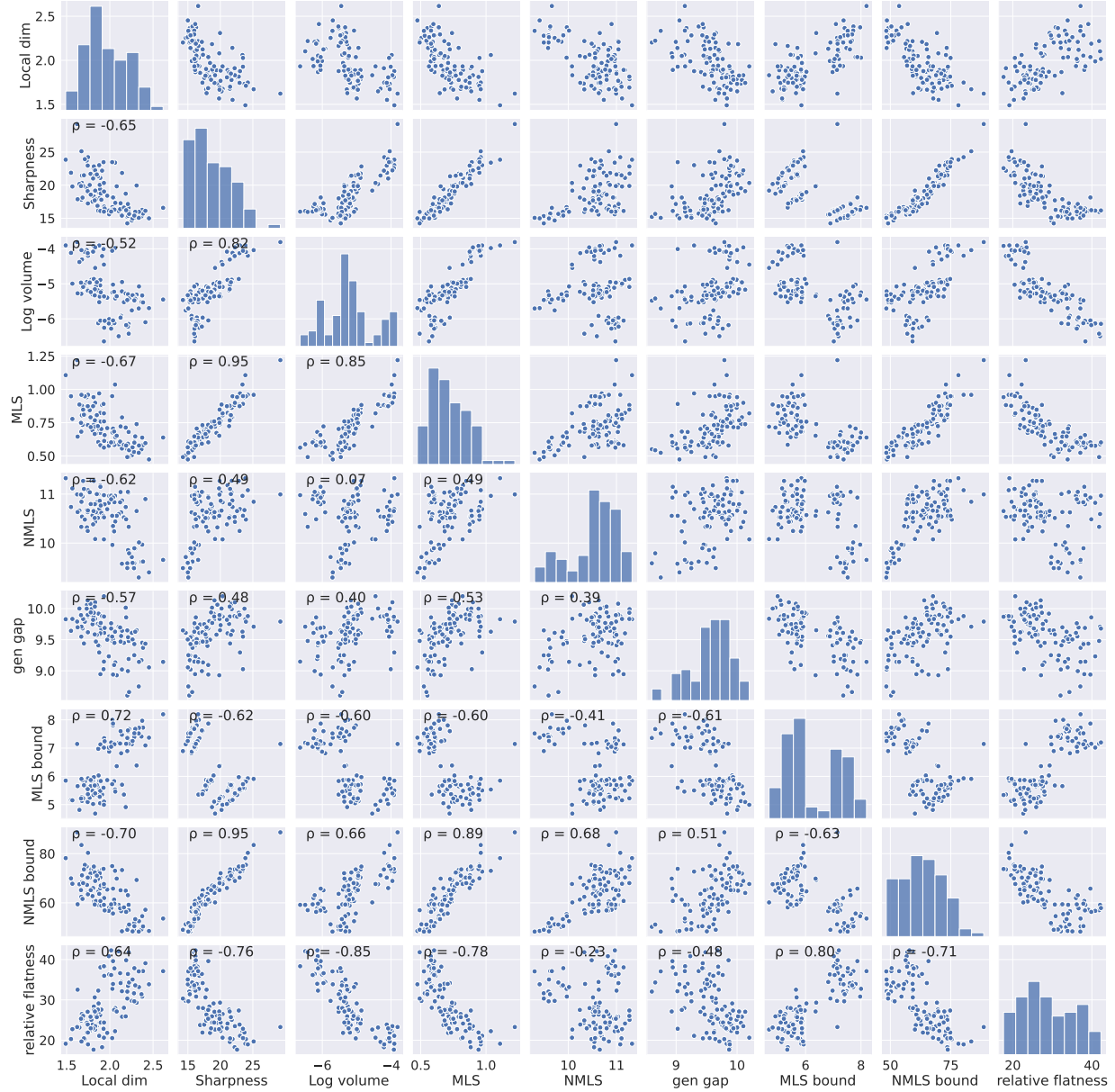


Figure D.8.3: **Pairwise correlation among different metrics.** We trained 100 different 4-layer MLPs on the FashionMNIST dataset using vanilla SGD with different learning rates, batch size, and random initializations and plot pairwise scatter plots between different quantities: local dimensionality, sharpness (square root of Equation (5.3)), log volume (Equation (5.7)), MLS (Equation (5.11)), NMLS (Equation (5.12)), generalization gap (gen gap), MLS bound (Proposition 5.8), NMLS bound (Proposition 5.10) and relative sharpness ([Petzka et al., 2021]). The Pearson correlation coefficient ρ is shown in the top-left corner for each pair of quantities. See Appendix D.8.2 for a summary of the findings in this figure.



Figure D.8.4: **Pairwise correlation among different metrics.** We trained 100 different LeNets on the CIFAR-10 dataset using vanilla SGD with different learning rates, batch size, and random initializations and plot pairwise scatter plots between different quantities: local dimensionality, sharpness (square root of Equation (5.3)), log volume (Equation (5.7)), MLS (Equation (5.11)), NMLS (Equation (5.12)), generalization gap (gen gap), MLS bound (Proposition 5.8), NMLS bound (Proposition 5.10) and relative sharpness ([Petzka et al., 2021]). The Pearson correlation coefficient ρ is shown in the top-left corner for each pair of quantities. See Appendix D.8.2 for a summary of the findings in this figure.

D.9 Additional experiments

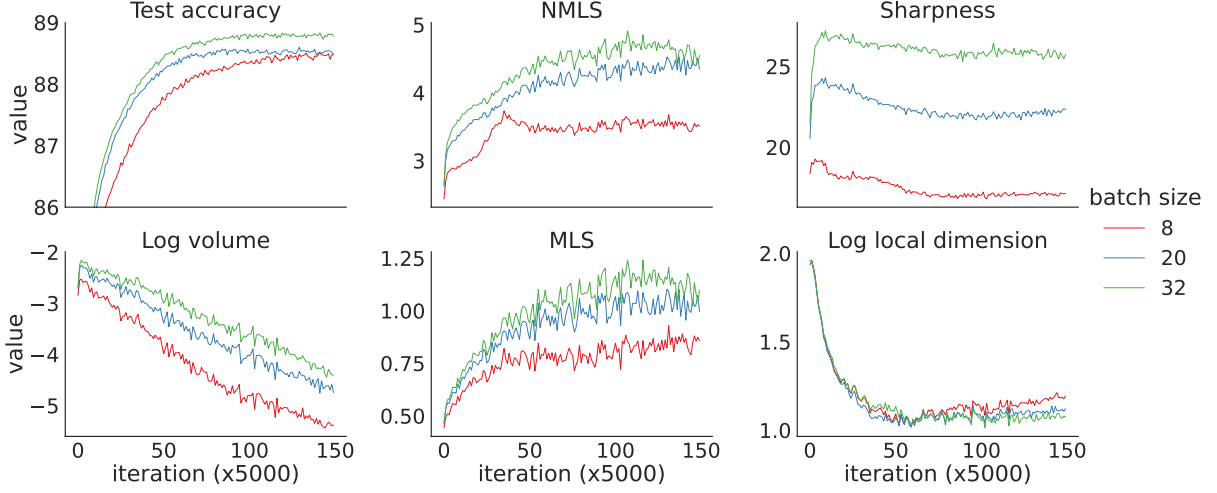


Figure D.9.5: Trends in key variables across SGD training of a 4-layer MLP with fixed learning rate (equal to 0.1) and varying batch size (8, 20, and 32). MLS/NMLS closely follows the trend of sharpness during the training. From left to right: test accuracy, NMLS, sharpness (square root of Equation (5.3)), log volumetric ratio (Equation (5.7)), MLS (Equation (5.11)), and local dimensionality of the network output (Equation (5.15)).

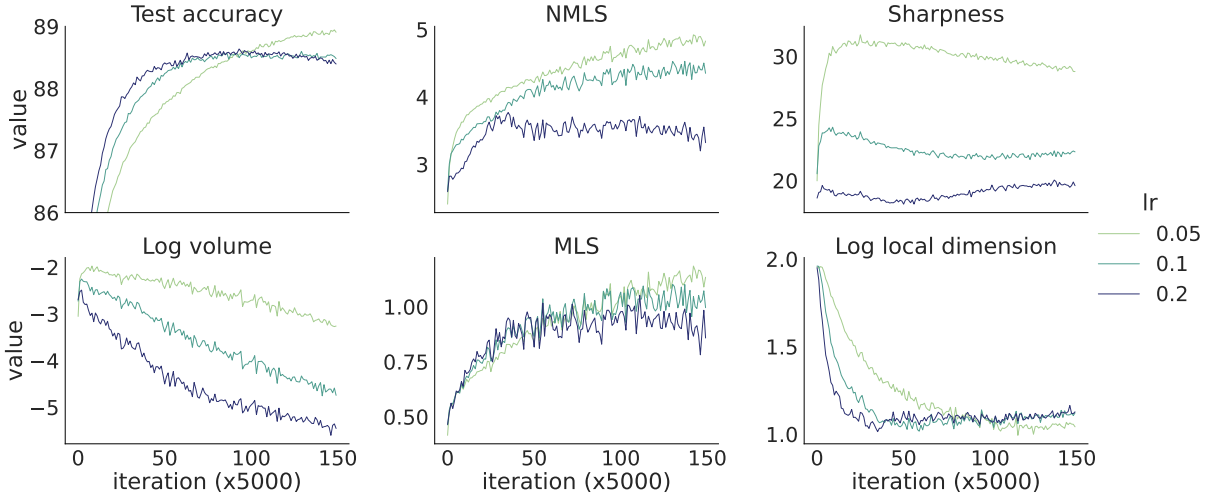


Figure D.9.6: Trends in key variables across SGD training of a 4-layer MLP with fixed batch size (equal to 20) and varying learning rates (0.05, 0.1 and 0.2). MLS/NMLS closely follows the trend of sharpness during the training. From left to right: test accuracy, NMLS, sharpness (square root of Equation (5.3)), log volumetric ratio (Equation (5.7)), MLS (Equation (5.11)), and local dimensionality of the network output (Equation (5.15)).

D.10 Sharpness and compression on test set data

Even though Equation (5.3) is exact for interpolation solutions only (i.e., those with zero loss), we found that the test loss is small enough (Figure D.10.7) so that it should be a good approximation for test data as well. Therefore we analyzed our simulations to study trends in sharpness and volume for these held-out test data as well (Figure D.10.7). We observed that this sharpness increased rather than diminished as a result of training. We hypothesized that sharpness could correlate with the difficulty of classifying testing points. This was supported by the fact that the sharpness of misclassified test data was even greater than that of all test data. Again we see that MLS has the same trend as the sharpness. Despite this increase in sharpness, the volume followed the same pattern as the training set. This suggests that compression in representation space is a robust phenomenon that can be driven by additional phenomena beyond sharpness. Nevertheless, the compression still is weaker for misclassified test samples that have higher sharpness than other test samples. Overall, these results emphasize an interesting distinction between how sharpness evolves for training vs. test data.

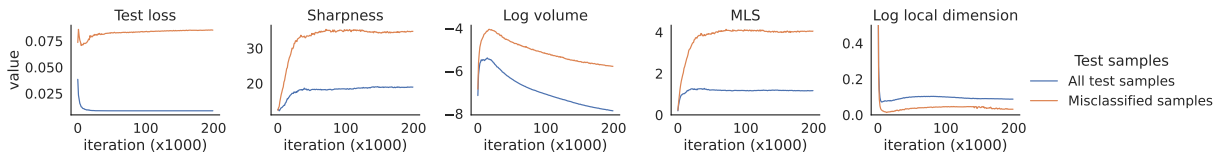


Figure D.10.7: Trends in key variables across SGD training of the VGG-11 network with fixed learning rate (equal to 0.1) and batch size (equal to 20) for samples of the test set. After the loss is minimized, we compute sharpness and volume on the test set. Moreover, the same quantities are computed separately over the entire test set or only on samples that are misclassified. In order from left to right in row-wise order: test loss, sharpness (Equation (5.2)), log volumetric ratio (Equation (5.7)), MLS, and local dimensionality of the network output (Equation (5.15)).

D.11 Computational resources and code availability

All experiments can be run on one NVIDIA Quadro RTX 6000 GPU. The code is available at <https://github.com/chinsengi/sharpness-compression>.

Appendix E

Appendix to Chapter 6: Feedforward Memory Networks

E.1 Biological motivation of feedforward memory networks

Equation (6.2) is the discrete-time Euler update of a standard leaky Hebbian plasticity ODE,

$$\frac{dW_{ij}}{dt} = -\frac{1}{\tau_{ij}}(W_{ij} - W_{ij}^0) + \beta_{ij} h_i(t) x_j(t), \quad (\text{E.1})$$

where the *leak* term with time constant τ_{ij} captures passive synaptic decay and homeostatic/heterosynaptic normalization, and the *Hebbian* term $\propto h_i x_j$ is the rate-based limit of spike-timing dependent plasticity (STDP) under common coarse-graining assumptions (e.g., Poisson spiking and balanced pre-post windows). With step size Δt , the identification

$$\lambda_{ij} = e^{-\Delta t/\tau_{ij}}, \quad \eta_{ij} = \beta_{ij} \Delta t$$

yields Equation (6.2), so λ and η have direct mechanistic interpretations: (i) $\lambda_{ij} \in (0, 1)$ sets the retention of a fast plastic trace, akin to decay of eligibility traces or short-term synaptic states; and (ii) η_{ij} sets the effective Hebbian learning gain, absorbing neuromodulatory gating (e.g., dopaminergic/cholinergic) that scales plasticity without changing the leak.

Relation to facilitation/depression. Short-term facilitation/depression (STP) models (e.g., Tsodyks–Markram) produce a first-order relaxation of effective synaptic efficacy around an operating point. Linearizing those dynamics gives an autoregressive form whose discrete update matches Equation (6.2): faster depression corresponds to smaller τ_{ij} (thus smaller λ_{ij}), whereas facilitating synapses exhibit larger τ_{ij} (thus $\lambda_{ij} \approx 1$) so that Hebbian increments persist longer. The sign and magnitude of η_{ij} set the LTP/LTD balance (allowing anti-Hebbian entries when appropriate).

Heterogeneity. Allowing λ_{ij} and η_{ij} to vary across synapses captures known diversity from receptor composition (AMPA/NMDA ratios), spine morphology, dendritic location, and modulatory receptor density, all of which jointly determine trace decay and plasticity gain. The baseline W_{ij}^0 represents consolidated synaptic strength; in the absence of input, $W_{ij} \rightarrow W_{ij}^0$.

Optional modeling note. If desired, neuromodulatory/state-dependent gating can be made explicit by writing $\eta_{ij} \rightarrow g_t \eta_{ij}$ with a scalar or elementwise gate g_t , without changing the analysis below.

E.2 Modern Hopfield network

In this section, we summarize the mathematical formulation in Krotov and Hopfield [2021] where a biologically plausible version of the modern Hopfield network is proposed. See Figure E.2.1 for an illustration. Similar to Manohar et al. [2019], two types of neurons are considered: feature neurons and memory neurons, where the feature neurons are selective to sensory input patterns, and memory neurons (conjunctive neurons in Manohar et al. [2019])

are hidden neurons whose synaptic connection strength with feature neurons encodes working memory.

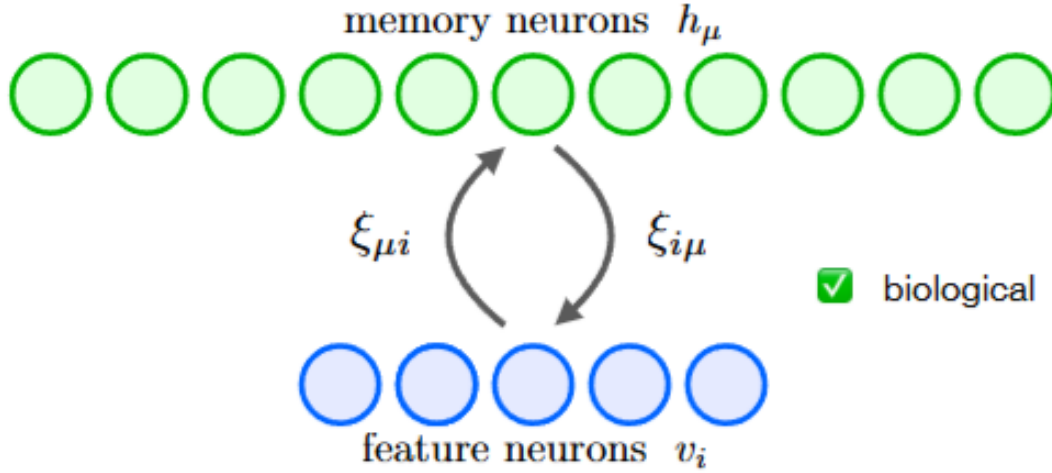


Figure E.2.1: Biological interpretation of modern Hopfield networks. Adapted from Krotov and Hopfield [2021].

E.2.1 Notations

- N_h : The number of memory neurons in the network.
- N_f : The number of feature neurons.
- h_μ : State of μ -th memory neuron.
- v_i : State of i -th feature neuron.
- f_μ : Output of μ -th memory neuron.
- g_i : Output of i -th feature neuron.
- $\xi_{\mu i}$: The synaptic weight from feature neuron i to neuron μ . All weights are symmetric.
- I_i : The input to the i -th neuron.
- τ_h : the decay rate of memory neurons.

- τ_f : the decay rate of feature neurons.

E.2.2 Neural dynamics and Energy function

The dynamics of the network illustrated in Figure E.2.1 can be written as

$$\begin{cases} \tau_f \frac{dv_i}{dt} = \sum_{\mu=1}^{N_h} \xi_{i\mu} f_\mu - v_i + I_i \\ \tau_h \frac{dh_\mu}{dt} = \sum_{i=1}^{N_f} \xi_{i\mu} g_i - h_\mu \end{cases} \quad (\text{E.2})$$

where notations are defined in Appendix E.2.1. The dynamics minimize the following energy function along its trajectories, given that the Hessian matrices of the Lagrangian functions are positive semi-definite:

$$E(t) = \left[\sum_{i=1}^{N_f} (v_i - I_i) g_i - L_v \right] + \left[\sum_{\mu=1}^{N_h} h_\mu f_\mu - L_h \right] - \sum_{\mu,i} f_\mu \xi_{\mu i} g_i. \quad (\text{E.3})$$

The Lagrangian functions L_v and L_h are defined such that

$$\begin{aligned} f_\mu &= \frac{\partial L_h}{\partial h_\mu} \\ g_i &= \frac{\partial L_v}{\partial v_i} \end{aligned} \quad (\text{E.4})$$

To see this we can take the derivative of $E(t)$ w.r.t. t ,

$$\begin{aligned}
\frac{dE}{dt} &= \sum_{i,j} (v_i - I_i) \frac{\partial^2 L_v}{\partial v_i \partial v_j} \frac{dv_j}{dt} + \sum_{\mu,\nu} h_\mu \frac{\partial^2 L_h}{\partial h_\mu \partial h_\nu} \frac{dh_\nu}{dt} \\
&\quad - \sum_{\mu,\nu} \frac{dh_\nu}{dt} \frac{\partial^2 L_h}{\partial h_\nu \partial h_\mu} \left(\sum_i \xi_{i\mu} g_i \right) \\
&\quad - \sum_{i,j} \frac{dv_j}{dt} \frac{\partial^2 L_v}{\partial v_j \partial v_i} \left(\sum_\mu \xi_{i\mu} f_\mu \right) \\
&= - \sum_{i,j} \frac{dv_j}{dt} \frac{\partial^2 L_v}{\partial v_j \partial v_i} \left[\sum_\mu \xi_{i\mu} f_\mu + I_i - v_i \right] \\
&\quad - \sum_{\mu,\nu} \frac{dh_\nu}{dt} \frac{\partial^2 L_h}{\partial h_\nu \partial h_\mu} \left[\sum_i \xi_{i\mu} g_i - h_\mu \right] \\
&= -\tau_f \sum_{i,j=1}^{N_f} \frac{dv_i}{dt} \frac{\partial^2 L_v}{\partial v_i \partial v_j} \frac{dv_j}{dt} - \tau_h \sum_{\mu,\nu=1}^{N_h} \frac{dh_\mu}{dt} \frac{\partial^2 L_h}{\partial h_\mu \partial h_\nu} \frac{dh_\nu}{dt} \leq 0,
\end{aligned} \tag{E.5}$$

given that the Hessian matrices of L_v and L_h are positive semi-definite. Note that terms cancel in the first equality above:

$$\begin{aligned}
\frac{\partial L_v}{\partial t} &= \sum_{i=1}^{N_f} \frac{\partial(v_i - I_i)}{\partial t} g_i \\
\frac{\partial L_h}{\partial t} &= \sum_{i=1}^{N_f} \frac{\partial h_\mu}{\partial t} f_\mu.
\end{aligned} \tag{E.6}$$

E.2.3 Relation to attention mechanism

Now we consider a special choice of Lagrangian functions that connects modern Hopfield networks with the attention mechanism:

$$L_h = \beta^{-1} \log \left(\sum_\mu \exp(\beta h_\mu) \right) \tag{E.7}$$

$$L_v = \frac{1}{2} \sum_i v_i^2. \tag{E.8}$$

Note that here L_h can be the integral of $f \tanh dh_\mu = \log(\cosh(h_\mu)) + C$, which is also convex. Therefore, according to the definition in Equation (E.4), we have

$$f_\mu = \frac{\partial L_h}{\partial h_\mu} = \text{softmax}(\beta h)_\mu = \frac{\exp(\beta h_\mu)}{\sum_\nu \exp(\beta h_\nu)} \quad (\text{E.9})$$

$$g_i = \frac{\partial L_v}{\partial v_i} = v_i. \quad (\text{E.10})$$

When additionally assuming that $\tau_f \gg \tau_h$, i.e. dynamics of h_μ converges to the equilibrium instantly and $h_\mu = \sum_i \xi_{\mu i} g_i$, the corresponding energy function becomes (I_i are also assumed to be 0)

$$\begin{aligned} E(t) &= \left[\sum_{i=1}^{N_f} (v_i - I_i) g_i - L_v \right] + \left[\sum_{\mu=1}^{N_h} h_\mu f_\mu - L_h \right] - \sum_{\mu,i} f_\mu \xi_{\mu i} g_i \\ &= \left[\sum_i v_i^2 - \frac{1}{2} \sum_i v_i^2 \right] + \left[\sum_{\mu,i} \xi_{\mu i} g_i f_\mu - \right. \\ &\quad \left. \beta^{-1} \log \left(\sum_\mu \exp(\beta h_\mu) \right) \right] - \sum_{\mu,i} f_\mu \xi_{\mu i} g_i \\ &= \frac{1}{2} \sum_i v_i^2 - \beta^{-1} \log \left(\sum_\mu \exp \left(\beta \sum_i \xi_{\mu i} v_i \right) \right). \end{aligned} \quad (\text{E.11})$$

This is exactly the energy function in [Ramsauer et al., 2021]. We next show that under the assumption of Equation (E.11), we also have the same update rule. Plugging h_i into the network dynamics E.2, we have

$$\tau_f \frac{dv_i}{dt} = \sum_{\mu=1}^{N_h} \xi_{i\mu} \text{softmax} \left(\beta \sum_i \xi_{\mu i} v_i \right) - v_i. \quad (\text{E.12})$$

Discretizing the dynamics with $\tau_f = dt$, we get

$$v_i^{(t+1)} = \sum_{\mu=1}^{N_h} \xi_{i\mu} \text{softmax} \left(\beta \sum_i \xi_{\mu i} v_i^{(t)} \right) \quad (\text{E.13})$$

which corresponds well to the update rule discussed in Ramsauer et al. [2021], where they argue that the update rule is the attention mechanism used in the transformer after multiplying

the update rule with $\mathbf{W}_V$.

E.2.4 Element-wise nonlinearity

In this section, we show that element-wise nonlinearity can be used in lieu of the softmax function which implement inhibition within the hidden layer neurons. Let

$$\begin{aligned} L_h &= \sum_{\mu} G(h_{\mu}) \\ L_v &= \sum_i v_i^2. \end{aligned} \tag{E.14}$$

Again, according to the definition in Equation (E.4), we have

$$f_{\mu} = \frac{\partial L_h}{\partial h_{\mu}} = G'(h_{\mu}) := F(h_{\mu}) \tag{E.15}$$

$$g_i = \frac{\partial L_v}{\partial v_i} = v_i. \tag{E.16}$$

Recall that in order for the dynamics to minimize the energy function, we need the Hessian matrices of L_v and L_h to be positive semi-definite. This is satisfied if $F'(h_{\mu}) \geq 0$ for all h_{μ} . This means that as long as F is a non-decreasing function, the dynamics will minimize the energy function. In particular, $F(x) = \tanh(x + b)$ is such a function that is used in most of this chapter.

E.2.5 Feedforward memory networks implement cross-attention

We now show that Equation (6.1), with some remapping of variables, implements the cross-attention mechanism [Vaswani et al., 2023]. We first ignore the learning process and assume

that the stored patterns ¹ are fixed. The cross-attention mechanism can be written as

$$Z = \text{softmax} \left(QK^T / \sqrt{d_i} \right) V \quad (\text{E.17})$$

$$Q = XW_Q \quad (\text{E.18})$$

$$K = YW_K \quad (\text{E.19})$$

$$V = YW_V. \quad (\text{E.20})$$

where $X \in \mathbb{R}^{1 \times d_i}$, $W_Q \in \mathbb{R}^{d_i \times d_k}$, $Y \in \mathbb{R}^{d_h \times d_Y}$, $W_K \in \mathbb{R}^{d_Y \times d_k}$ and $W_V \in \mathbb{R}^{d_Y \times d_o}$. To see that the cross-attention mechanism maps onto Equation (6.1), we further let

$$F(\mathbf{x}) = \text{softmax} \left(\frac{\mathbf{x}}{\sqrt{d_i}} \right) \quad (\text{E.21})$$

$$\mathbf{x}_t^T = X \quad (\text{E.22})$$

$$W_{\text{in}} = YW_K W_Q^T \quad (\text{E.23})$$

$$W_{\text{out}}^T = V = YW_V. \quad (\text{E.24})$$

Then we have

$$\begin{aligned} \mathbf{o}_t^T &= \mathbf{h}_t^T W_{\text{out}}^T \\ &= \text{softmax} \left(\mathbf{x}_t^T W_{\text{in}}^T \right) W_{\text{out}}^T \\ &= \text{softmax} \left(XW_Q W_K^T Y^T / \sqrt{d_i} \right) W_{\text{out}}^T \\ &= \text{softmax} \left(QK^T / \sqrt{d_i} \right) V \end{aligned} \quad (\text{E.25})$$

which is exactly the cross-attention mechanism.

A subtle question is, given Y , W_{in} and W_{out} , can we find such W_K , W_Q and W_V that the mapping Equation (E.23) and Equation (E.24) holds? We just need to count the number of unknowns and the number of equations. For Equation (E.23), we can choose fixed W_K and

¹in attention mechanism these are called the key values, but for consistency, we call them patterns here.

see that the number of unknowns is $d_i d_k$ and the number of equations is $d_i d_h$. Therefore we can choose a large enough d_k such that a solution is guaranteed to exist. For Equation (E.24), the number of unknowns is $d_Y d_o$ and the number of equations is $d_h d_o$. Again, we can choose $d_Y \geq d_h$ to guarantee a solution. The other direction is straightforward: given X, Y, W_Q, W_K, W_V , we have a configuration of the FMN according to Equation (E.23) and Equation (E.24). In conclusion, FMN is equivalent to the cross-attention mechanism given $d_k \geq d_h$ and $d_Y \geq d_h$, as summarized by the following proposition.

Proposition E.1. *Given the input $\mathbf{x}_t \in \mathbb{R}_i^d$, the input weights $W_{\text{in}} \in \mathbb{R}^{d_h \times d_i}$, the output $W_{\text{out}} \in \mathbb{R}^{d_o \times d_h}$ in the FMN defined in Equation (6.1), and $Y \in \mathbb{R}^{d_h \times d_Y}$, there exist $W_Q \in \mathbb{R}^{d_i \times d_k}$, $W_K \in \mathbb{R}^{d_Y \times d_k}$ and $W_V \in \mathbb{R}^{d_Y \times d_o}$ such that the cross-attention mechanism is implemented: $\mathbf{o}_t^T = \text{softmax}\left(QK^\top / \sqrt{d_i}\right) V$, $Q = \mathbf{x}_t^T W_Q$, $K = YW_K$, $V = YW_V$, provided that $d_k \geq d_h$, $d_Y \geq d_h$ and the nonlinearity F of the FMN is chosen accordingly. Conversely, given Y , W_Q , W_K and W_V , the FMN can be implemented by the mapping $W_{\text{in}} = YW_K W_Q^T$ and $W_{\text{out}}^T = V = YW_V$.*

E.2.6 Feedforward memory networks and Hopfield networks

From Appendix E.2.3 we know that the update rule of the Hopfield network is the cross-attention mechanism; therefore, a connection can be established between feedforward memory networks and modern Hopfield networks as well. We first decompose W_{out} into the product of a linear projection and the transpose of the input weights,

$$W_{\text{out}} = P W_{\text{in}}^T, \quad (\text{E.26})$$

where $P \in \mathbb{R}^{d_o \times d_i}$ and $W_{\text{in}} \in \mathbb{R}^{d_h \times d_i}$. For arbitrary W_{out} , this decomposition is only possible when $d_i \geq d_o$ given that the columns of W_{in} span the rows of W_{out} . We further denote the one-step update outcome of the modern Hopfield network with stored patterns Ξ and the

state (query) pattern $\mathbf{x}$ as

$$\text{Hop}(\mathbf{x}, \Xi) = \Xi F(\beta \Xi^T \mathbf{x}). \quad (\text{E.27})$$

Then we have

$$\begin{aligned} \mathbf{o}_t &= W_{\text{out}} F(W_{\text{in}} \mathbf{x}_t) \\ &= P W_{\text{in}}^T F(W_{\text{in}} \mathbf{x}_t) \\ &= \beta P \text{Hop}(\mathbf{x}_t, W_{\text{in}}^T / \beta). \end{aligned} \quad (\text{E.28})$$

Therefore, when the output dimension is smaller than the input dimension ($d_i \geq d_o$), $\mathbf{o}_t$ is effectively a linear projection of the pattern retrieved from the input weights. This is the case for all of the experiments in Section 6.4. Note that the analysis in Appendix E.2.5 and Appendix E.2.6 applies to a broad range of nonlinearity functions (see Appendix E.2.4). To summarize,

Proposition E.2. *If the output weights W_{out} of a feedforward memory network can be decomposed as $W_{\text{out}} = P W_{\text{in}}^T$, where $P \in \mathbb{R}^{d_o \times d_i}$, then the output $\mathbf{o}_t$ computes a linear projection of the pattern retrieved from a modern Hopfield network [Ramsauer et al., 2021] with stored patterns $\Xi = W_{\text{in}}^T$ and query $\mathbf{x}_t$. More formally, $\mathbf{o}_t = P \text{Hop}(\mathbf{x}_t, \Xi)$, where $\text{Hop}(\mathbf{x}, \Xi) = \Xi F(\Xi^T \mathbf{x})$.*

E.3 Proofs

E.3.1 Proof of Proposition 6.2

For this section, we assume that $\|\mathbf{x}\| > 0$.

Lemma E.3. *The eigenvalues of $\lambda \mathbf{I} + \eta \mathbf{x} \mathbf{x}^T$ are given by $\lambda + \eta \|\mathbf{x}\|^2$ and λ , where the geometric multiplicity of λ is $d_i - 1$.*

Proof. It suffices to find the eigenvalues of $\eta \mathbf{x} \mathbf{x}^T$. Since it is a rank-1 matrix, we only need to find the non-zero eigenvalue. Observe that $\eta \mathbf{x} \mathbf{x}^T \mathbf{x} = \eta \|\mathbf{x}\|^2 \mathbf{x}$. Therefore, $\eta \|\mathbf{x}\|^2$ is an

eigenvalue of $\eta \mathbf{x} \mathbf{x}^T$. Since any eigenvector of $\eta \mathbf{x} \mathbf{x}^T$ is also an eigenvector of $\lambda \mathbf{I}$, $\lambda \mathbf{I} + \eta \mathbf{x} \mathbf{x}^T$ has two different eigenvalues λ and $\lambda + \eta \|\mathbf{x}\|^2$. Since any vector in the subspace that is orthogonal to $\mathbf{x}$ is an eigenvector with eigenvalue λ , the geometric multiplicity of λ is $d_i - 1$. $\square$

Now we prove Proposition 6.2.

Proposition. Assume $\mathbf{x}_t = \mathbf{x} \neq \mathbf{0}$ for all t , and $\mathbf{w}_0$ has a nonzero component along $\mathbf{x}$. Then $\left| \frac{\mathbf{w}_t \cdot \mathbf{x}}{\|\mathbf{w}_t\| \|\mathbf{x}\|} \right| \rightarrow 1$ as $t \rightarrow \infty$ when $\eta > 0$ or $\eta < -\frac{2\lambda}{\|\mathbf{x}\|^2}$, and $\mathbf{w}_t \rightarrow \mathbf{0}$ as $t \rightarrow \infty$ when $-\frac{2\lambda}{\|\mathbf{x}\|^2} < \eta < 0$.

Proof. From the proof of Lemma 6.1, we know that $\lambda \mathbf{I} + \eta \mathbf{x} \mathbf{x}^T$ has an orthonormal eigenbasis that consists of $\mathbf{x}/\|\mathbf{x}\|$ and a set of orthonormal basis that spans the eigenspace associated with λ . Let the corresponding decomposition of $\mathbf{w}_0$ be $\mathbf{w}_0^T = \sum_{i=1}^{d_i} \alpha_i \mathbf{v}_i$, where $\mathbf{v}_i$ are the eigenbasis of $\lambda \mathbf{I} + \eta \mathbf{x} \mathbf{x}^T$ and α_i are the corresponding coefficients. Let $\mathbf{v}_1 = \mathbf{x}/\|\mathbf{x}\|$.

Since $\mathbf{x}_t = \mathbf{x}$ for all t , we have

$$\begin{aligned} \mathbf{w}_t^T &= (\lambda \mathbf{I} + \eta \mathbf{x} \mathbf{x}^T)^t \mathbf{w}_0^T \\ &= \sum_{i=1}^{d_i} \alpha_i (\lambda \mathbf{I} + \eta \mathbf{x} \mathbf{x}^T)^t \mathbf{v}_i \\ &= \alpha_1 (\lambda + \eta \|\mathbf{x}\|^2)^t \mathbf{v}_1 + \sum_{i=2}^{d_i} \alpha_i \lambda^t \mathbf{v}_i. \end{aligned} \tag{E.29}$$

Therefore cosine of the angle between $\mathbf{w}_t$ and $\mathbf{x}$ is given by

$$\begin{aligned} \cos \theta &= \frac{\mathbf{w}_t^T \mathbf{x}}{\|\mathbf{w}_t\| \|\mathbf{x}\|} \\ &= \frac{\alpha_1 (\lambda + \eta \|\mathbf{x}\|^2)^t}{\sqrt{\alpha_1^2 (\lambda + \eta \|\mathbf{x}\|^2)^{2t} + \sum_{i=2}^{d_i} \alpha_i^2 \lambda^{2t}}}. \end{aligned} \tag{E.30}$$

If $\eta > 0$ or $\eta < -2\lambda/\|\mathbf{x}\|^2$, the absolute cosine approaches 1; for a negative dominant eigenvalue, its sign alternates. If $-2\lambda/\|\mathbf{x}\|^2 < \eta < 0$, $\|\mathbf{w}_t\| \rightarrow 0$. $\square$

E.3.2 Proof of Proposition 6.3

We first state the Weyl's inequalities [Horn and Johnson, 1994].

Theorem E.4 (Weyl). *For Hermitian $A, B \in \mathbb{C}^{n \times n}$ and $i, j = 1 : n$,*

$$\lambda_{i+j-1}(A+B) \leq \lambda_i(A) + \lambda_j(B), \quad i+j \leq n+1, \quad (3)$$

$$\lambda_i(A) + \lambda_j(B) \leq \lambda_{i+j-n}(A+B), \quad i+j \geq n+1. \quad (4)$$

Corollary E.5. *Let $B = \eta \mathbf{x} \mathbf{x}^T$ be a symmetric rank-1 matrix, and $A = \text{diag}(\boldsymbol{\lambda})$ be a diagonal matrix with $\{\lambda_i\}$ as diagonal entries. Then the eigenvalues of $A+B$ interlace with the eigenvalues of A .*

Proof. Let the eigenvalues of A be $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{d_i}$, and the eigenvalues of $A+B$ be $\mu_1 \geq \mu_2 \geq \dots \geq \mu_{d_i}$. The only non-zero eigenvalue of B is $\eta \|\mathbf{x}\|^2$.

Case 1 $\eta > 0$

using Weyl's inequality we have

$$\begin{aligned} \mu_{j+d_i-d_i} &\geq \lambda_j + 0 \\ \mu_{j+2-1} &\leq \lambda_j + 0 \end{aligned} \quad (\text{E.31})$$

Therefore $\mu_j \geq \lambda_j \geq \mu_{j+1}$ for $j \leq d_i - 1$ and $\mu_{d_i} \geq \lambda_{d_i}$. Putting all together, we have

$$\mu_1 \geq \lambda_1 \geq \mu_2 \geq \lambda_2 \geq \dots \geq \mu_{d_i} \geq \lambda_{d_i}.$$

Case 2 $\eta < 0$

Now the largest eigenvalue of B is 0. Therefore, using Weyl's inequality we have

$$\begin{aligned} \mu_{j+d_i-1-d_i} &\geq \lambda_j + 0 \\ \mu_{j+1-1} &\leq \lambda_j + 0 \end{aligned} \quad (\text{E.32})$$

Therefore $\mu_j \leq \lambda_j \leq \mu_{j-1}$ for $j \geq 2$ and $\mu_1 \leq \lambda_1$. Putting all together, we have $\mu_{d_i} \leq \lambda_{d_i} \leq \mu_{d_i-1} \leq \lambda_{d_i-1} \leq \dots \leq \mu_1 \leq \lambda_1$.

□

Now we prove Proposition 6.3.

Proposition. *The matrix $\text{diag}(\boldsymbol{\lambda}) + \eta \mathbf{x} \mathbf{x}^T$ has at most one eigenvalue whose absolute value is greater than 1. Its largest eigenvalue $\mu_{\max}$ satisfies $\max(\boldsymbol{\lambda}) \leq \mu_{\max} \leq \max(\boldsymbol{\lambda}) + \eta \|\mathbf{x}\|^2$ when $\eta > 0$, and its smallest eigenvalue $\mu_{\min}$ satisfies $\min(\boldsymbol{\lambda}) + \eta \|\mathbf{x}\|^2 \leq \mu_{\min} \leq \min(\boldsymbol{\lambda})$ when $\eta < 0$.*

Proof. A direct consequence of Corollary E.5 is that $\text{diag}(\boldsymbol{\lambda}) + \eta \mathbf{x} \mathbf{x}^T$ has at most one eigenvalue whose absolute value is greater than 1. Additionally, when $\eta > 0$, $\lambda_1 \leq \mu_1 \leq \lambda_1 + \eta \|\mathbf{x}\|^2$, and when $\eta < 0$, $\lambda_{d_i} + \eta \|\mathbf{x}\|^2 \leq \mu_{d_i} \leq \lambda_{d_i}$. □

E.3.3 Proof of Proposition 6.4

The proof is based on an important theorem that underlies the pole placement method in control theory, which we restate here.

Definition E.6. *A pair $(\mathbf{A}, \mathbf{b})$ is said to be controllable if $[\mathbf{b} \ \mathbf{A}\mathbf{b} \ \mathbf{A}^2\mathbf{b} \ \dots \ \mathbf{A}^{n-1}\mathbf{b}]$ has full row rank.*

Theorem E.7 (Chen [1984]). *Given a controllable pair $(\mathbf{A}, \mathbf{b})$, there exists $\mathbf{k}$ such that the eigenvalues of $\mathbf{A} + \mathbf{b}\mathbf{k}$ can be arbitrarily assigned.*

Proposition. *Assume that $\lambda_i \neq 0$, $\lambda_i \neq \lambda_j$ for $i \neq j$, and $x_i \neq 0$ for all i . Then, for any $\boldsymbol{\lambda}$ and $\mathbf{x}$, there exists an $\boldsymbol{\eta}$ such that $\text{diag}(\boldsymbol{\lambda}) + \mathbf{x} \mathbf{x}^T \text{diag}(\boldsymbol{\eta})$ has any set of eigenvalues.*

Proof. Given Theorem E.7, we just need to show that $(\text{diag}(\boldsymbol{\lambda}), \mathbf{x})$ is controllable. The controllability matrix is given by $\mathcal{C} = [\mathbf{x} \ \text{diag}(\boldsymbol{\lambda})\mathbf{x} \ \text{diag}(\boldsymbol{\lambda})^2\mathbf{x} \ \dots \ \text{diag}(\boldsymbol{\lambda})^{d_i-1}\mathbf{x}]$. Since $\mathbf{x}$ is non-zero, $\det(\mathcal{C}) = \det(\text{Vand}(\text{diag}(\boldsymbol{\lambda}))) \prod_i x_i$, where $\text{Vand}(\text{diag}(\boldsymbol{\lambda}))$ is the Vandermonde

matrix. Since $\lambda_i \neq \lambda_j$ for $i \neq j$, $\det(\text{Vand}(\text{diag}(\boldsymbol{\lambda}))) \neq 0$. Therefore, the controllability matrix has full row rank. Therefore, there exists a $\mathbf{k} = \mathbf{x}^T \text{diag}(\boldsymbol{\eta})$ such that $\text{diag}(\boldsymbol{\lambda}) + \mathbf{x}\mathbf{x}^T \text{diag}(\boldsymbol{\eta})$ has any set of eigenvalues. Note that $\boldsymbol{\eta}$ is given by $\boldsymbol{\eta} = \mathbf{k} \text{diag}(\mathbf{x})^{-1}$. $\square$

E.4 List of tasks

See Table E.1 for information about the tasks used. Full task details are available in the official documentation.

Task	Observation Size	Action Size
ContextDecisionMaking-v0	7	3
DelayComparison-v0	2	3
DelayMatchSample-v0	3	3
DelayMatchSampleDistractor1D-v0	33	2
DelayPairedAssociation-v0	5	2
DualDelayMatchSample-v0	7	3
GoNogo-v0	3	2
IntervalDiscrimination-v0	3	3
MotorTiming-v0	4	2
MultiSensoryIntegration-v0	5	3
OneTwoThreeGo-v0	3	2
PerceptualDecisionMaking-v0	3	3
PerceptualDecisionMakingDelayResponse-v0	3	3
ProbabilisticReasoning-v0	41	3
ReadySetGo-v0	3	2
SingleContextDecisionMaking-v0	5	3

Table E.1: Task Information. Note that the dimension of the input to the network is 10 for all tasks since all original inputs go through a fixed random linear transformation. The action size is the dimension of the output of the network.

E.5 Training details

We follow the setup in [Aitken and Mihalas, 2023] and apply a fixed random linear transformation to all inputs. The network is trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The network is trained until the running average of the

validation loss saturates or increases. The network is trained on a single NVIDIA 2080 Ti GPU. The code is implemented in PyTorch [Paszke et al., 2019]. We initialize all parameters with Xavier initialization.

Bibliography

Lou Aaron. Language modeling by estimating the ratios of the data distribution. Blog post, 2024. URL <https://aaronlou.com/blog/2024/discrete-diffusion/>.

Laurence Aitchison and Máté Lengyel. The Hamiltonian Brain: Efficient Probabilistic Inference with Excitatory-Inhibitory Neural Circuit Dynamics. *PLOS Computational Biology*, 12(12):e1005186, December 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1005186. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005186>. Publisher: Public Library of Science.

Kyle Aitken and Stefan Mihalas. Neural population dynamics of computing with synaptic modulations. *eLife*, 12:e83035, February 2023. ISSN 2050-084X. doi: 10.7554/eLife.83035. URL <https://doi.org/10.7554/eLife.83035>. Publisher: eLife Sciences Publications, Ltd.

Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*, pages 639–668. PMLR, 2022.

Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization. *arXiv preprint arXiv:2302.07011*, 2023.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder,

- Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.
- Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE, 2025.
- Alessio Ansuini, Alessandro Laio, Jakob H. Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.
- P. Ao. Potential in stochastic differential equations: novel construction. *Journal of Physics A: Mathematical and General*, 37(3):L25, January 2004. ISSN 0305-4470. doi: 10.1088/0305-4470/37/3/L01. URL <https://dx.doi.org/10.1088/0305-4470/37/3/L01>.
- Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023. URL <https://arxiv.org/abs/2301.08243>.
- Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zhohus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL <https://arxiv.org/abs/2506.09985>.
- Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2021. URL <https://arxiv.org/abs/2107.03006>.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it’s lying. *arXiv preprint arXiv:2304.13734*, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibao Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL <https://arxiv.org/abs/2511.21631>.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video, 2024. URL <https://arxiv.org/abs/2404.08471>.

Cornelia I Bargmann and Eve Marder. From the connectome to brain function. *Nature methods*, 10(6):483–490, 2013.

Peter L Bartlett, Philip M Long, and Olivier Bousquet. The dynamics of sharpness-aware minimization: Bouncing across ravines and drifting towards wide minima. *Journal of Machine Learning Research*, 24(316):1–36, 2023.

Kate Baumli, Satinder Baveja, Feryal Behbahani, Harris Chan, Gheorghe Comanici, Sebastian Flennerhag, Maxime Gazeau, Kristian Holsheimer, Dan Horgan, Michael Laskin, et al. Vision-language models as a source of rewards. *arXiv preprint arXiv:2312.09187*, 2023.

Deniz Bayazit, Aaron Mueller, and Antoine Bosselut. Crosscoding through time: Tracking

- emergence and consolidation of linguistic representations throughout LLM pretraining. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1353–1377. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.60. URL <https://aclanthology.org/2026.acl-long.60/>.
- Laurie Bayet, Benjamin D. Zinszer, Emily Reilly, Julia K. Cataldo, Zoe Pruitt, Radoslaw M. Cichy, Charles A. Nelson, and Richard N. Aslin. Temporal dynamics of visual representations in the infant brain, August 2020. URL <https://www.biorxiv.org/content/10.1101/2020.02.26.947911v3>. Pages: 2020.02.26.947911 Section: New Results.
- Jeffrey M. Beck, Wei Ji Ma, Roozbeh Kiani, Tim Hanks, Anne K. Churchland, Jamie Roitman, Michael N. Shadlen, Peter E. Latham, and Alexandre Pouget. Probabilistic population codes for Bayesian decision making. *Neuron*, 60(6):1142–1152, December 2008. ISSN 0896-6273. doi: 10.1016/j.neuron.2008.09.021. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2742921/>.
- Moritz F P Becker and Christian Tetzlaff. The biophysical basis underlying the maintenance of early phase long-term potentiation. *PLoS Computational Biology*, 2021. doi: 10.1371/journal.pcbi.1008813.
- Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated Sampling from Masked Diffusion Models via Entropy Bounded Unmasking, May 2025. URL <http://arxiv.org/abs/2505.24857>. arXiv:2505.24857 [cs] version: 1.
- Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8): 1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.
- Barry Bentley, Robyn Branicky, Christopher L Barnes, Yee Lian Chew, Eviatar Yemini,

- Edward T Bullmore, Petra E Vértés, and William R Schafer. The multilayer connectome of *Caenorhabditis elegans*. *PLoS computational biology*, 12(12):e1005283, 2016.
- Pietro Berkes, Gergo Orban, Máté Lengyel, and József Fiser. Spontaneous Cortical Activity Reveals Hallmarks of an Optimal Internal Model of the Environment. *Science*, 331(6013): 83–87, January 2011. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1195870. URL <https://www.science.org/doi/10.1126/science.1195870>.
- Julius Berner, Philipp Grohs, and Arnulf Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020. Publisher: SIAM.
- Harsh Bhatia, Gregory Norgard, Valerio Pascucci, and Peer-Timo Bremer. The Helmholtz-Hodge Decomposition—A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1386–1404, August 2013. ISSN 1941-0506. doi: 10.1109/TVCG.2012.316. Conference Name: IEEE Transactions on Visualization and Computer Graphics.
- Harsh Bhatia, Valerio Pascucci, and Peer-Timo Bremer. The Natural Helmholtz-Hodge Decomposition for Open-Boundary Flow Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(11):1566–1578, November 2014. ISSN 1941-0506. doi: 10.1109/TVCG.2014.2312012. Conference Name: IEEE Transactions on Visualization and Computer Graphics.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usven Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine*

- Learning Research*, pages 2397–2430. PMLR, 2023. URL <https://proceedings.mlr.press/v202/biderman23a.html>.
- Katie C. Bittner, Aaron D. Milstein, Christine Grienberger, S. Romani, and J. Magee. Behavioral time scale synaptic plasticity underlies cal place fields. *Science*, 357:1033 – 1036, 2017. doi: 10.1126/science.aan3846.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0 : A vision-language-action flow model for general robot control, 2026. URL <https://arxiv.org/abs/2410.24164>.
- Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, pages 483–513. PMLR, 2020.
- Adam Block, Youssef Mroueh, and Alexander Rakhlin. Generative Modeling with Denoising Auto-Encoders and Langevin Sampling, October 2022. URL <http://arxiv.org/abs/2002.00107>. arXiv:2002.00107 [cs, math, stat].
- Martin Boerlin, Christian K. Machens, and Sophie Denève. Predictive Coding of Dynamical Variables in Balanced Spiking Networks. *PLOS Computational Biology*, 9(11):e1003258, November 2013. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1003258. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003258>. Publisher: Public Library of Science.
- Paweł Budzianowski, Emilia Wiśnios, Gracjan Góral, Igor Kulakov, Viktor Petrenko, and Krzysztof Walas. Opengvl—benchmarking visual temporal progress for data curation. *arXiv preprint arXiv:2509.17321*, 2025.

- Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural Dynamics as Sampling: A Model for Stochastic Computation in Recurrent Networks of Spiking Neurons. *PLOS Computational Biology*, 7(11):e1002211, November 2011. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1002211. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002211>. Publisher: Public Library of Science.
- James R Bunch, Christopher P Nielsen, and Danny C Sorensen. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31(1):31–48, 1978.
- Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.
- Camille E. Rullán Buxó and Jonathan W. Pillow. Poisson balanced spiking networks. *PLOS Computational Biology*, 16(11):e1008261, November 2020. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1008261. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008261>. Publisher: Public Library of Science.
- Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.
- Marc E. Canby, Adam Davies, Chirag Rastogi, and Julia Hockenmaier. How reliable are causal probing interventions? In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 857–878. Asian Federation of Natural Language Processing and Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.ijcnlp-long.47. URL <https://aclanthology.org/2025.ijcnlp-long.47/>.
- João Carreira, Michael King, Viorica Patraucean, Dilara Gokay, Catalin Ionescu, Yi Yang, Daniel Zoran, Joseph Heyward, Carl Doersch, Yusuf Aytar, Dima Damen, and Andrew Zisserman. Learning from one continuous video stream. In *Proceedings of the*

- IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28751–28761. IEEE/CVF, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Carreira_Learning_from_One_Continuous_Video_Stream_CVPR_2024_paper.html.
- S. Cerrai. *Second Order PDE’s in Finite and Infinite Dimension: A Probabilistic Approach*. Number no. 1762 in Lecture Notes in Mathematics. Springer, 2001. ISBN 978-3-540-42136-8. URL <https://books.google.com/books?id=pD-JeUpKrXMC>.
- Chengan Che, Chao Wang, Xinyue Chen, Sophia Tsoka, and Luis C. Garcia-Peraza-Herrera. A stitch in time: Learning procedural workflow via self-supervised Plackett–Luce ranking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17000–17010. IEEE/CVF, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Che_A_Stitch_in_Time_Learning_Procedural_Workflow_via_Self-Supervised_Plackett-Luce_CVPR_2026_paper.html.
- Chi-Tsong Chen. *Linear system theory and design*. Saunders college publishing, 1984.
- Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Yide Shentu, and Philipp Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. *arXiv preprint arXiv:2509.25358*, 2025a.
- Shirui Chen, Linxing Preston Jiang, Rajesh P. N. Rao, and Eric Shea-Brown. Expressive probabilistic sampling in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 36. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6e2a1a8a037f9a06004fe651054e8938-Abstract-Conference.html.
- Shirui Chen, Jiantao Jiao, Lillian J. Ratliff, and Banghua Zhu. dUltra: Ultra-fast diffusion language models via reinforcement learning, 2025b. URL <https://arxiv.org/abs/2512.21446>.

- Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J. Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. TOPReward: Token probabilities as hidden zero-shot rewards for robotics, 2026a. URL <https://arxiv.org/abs/2602.19313>.
- Shirui Chen, Stefano Recanatesi, and Eric Shea-Brown. A simple connection from loss flatness to compressed neural representations. *Transactions on Machine Learning Research*, 2026b. ISSN 2835-8856. URL <https://openreview.net/forum?id=GgpQbU9bFR>.
- Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025c.
- Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dparallel: Learnable parallel decoding for dllms, 2025d. URL <https://arxiv.org/abs/2509.26488>.
- Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. Dmax: Aggressive parallel decoding for dllms, 2026c. URL <https://arxiv.org/abs/2604.08302>.
- Alexandre J. Chorin and Jerrold E. Marsden. *A Mathematical Introduction to Fluid Mechanics*, volume 4 of *Texts in Applied Mathematics*. Springer, New York, NY, 1993. ISBN 978-1-4612-6934-2 978-1-4612-0883-9. doi: 10.1007/978-1-4612-0883-9. URL <http://link.springer.com/10.1007/978-1-4612-0883-9>.
- Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Winson Han, Ali Farhadi, and Ranjay Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding, 2026. URL <https://arxiv.org/abs/2601.10611>.
- Uri Cohen, SueYeon Chung, Daniel D. Lee, and Haim Sompolsky. Separability and geometry

- of object manifolds in deep neural networks. *Nature communications*, 11(1):746, 2020. Publisher: Nature Publishing Group UK London.
- Yaim Cooper. The loss landscape of overparameterized neural networks. *arXiv preprint arXiv:1804.10200*, 2018.
- Alex Damian, Tengyu Ma, and Jason D Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34:27449–27461, 2021.
- Anna Dawid and Yann LeCun. Introduction to latent variable energy-based models: a path toward autonomous machine intelligence. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10):104011, October 2024. ISSN 1742-5468. doi: 10.1088/1742-5468/ad292b. URL <http://dx.doi.org/10.1088/1742-5468/ad292b>.
- Peter Dayan and Laurence F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, August 2005. ISBN 978-0-262-54185-5. Google-Books-ID: fLT4DwAAQBAJ.
- Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.
- Perry Dong, Suvir Mirchandani, Dorsa Sadigh, and Chelsea Finn. What matters for batch online reinforcement learning in robotics? *arXiv preprint arXiv:2505.08078*, 2025.
- Xingsi Dong, Zilong Ji, Tianhao Chu, Tiejun Huang, Wenhao Zhang, and Si Wu. Adaptation Accelerating Sampling-based Bayesian Inference in Attractor Neural Networks. In *Advances in Neural Information Processing Systems*, volume 35, October 2022. URL <https://openreview.net/forum?id=Y0Bm5tL92lg>.
- Kenji Doya, Shin Ishii, Alexandre Pouget, and Rajesh P. N. Rao, editors. *Bayesian Brain:*

- Probabilistic Approaches to Neural Coding*. MIT Press, Cambridge, MA, 2006. ISBN 9780262042383. URL <https://mitpress.mit.edu/9780262042383/bayesian-brain/>.
- Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient Descent Finds Global Minima of Deep Neural Networks, May 2019. URL <http://arxiv.org/abs/1811.03804>. arXiv:1811.03804 [cs, math, stat].
- Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando De Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.
- Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024.
- Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.
- Rodrigo Echeveste, Laurence Aitchison, Guillaume Hennequin, and Máté Lengyel. Cortical-like dynamics in recurrent circuits optimized for sampling-based probabilistic inference | Nature Neuroscience. *Nature Neuroscience*, 23(9):1138–1149, September 2020. ISSN 1546-1726. doi: 10.1038/s41593-020-0671-1. URL <https://www.nature.com/articles/s41593-020-0671-1>.
- M. A. Erickson, Lauren A. Maramara, and J. Lisman. A single brief burst induces glur1-dependent associative short-term potentiation: A potential mechanism for short-term memory. *Journal of Cognitive Neuroscience*, 22:2530–2540, 2010. doi: 10.1162/jocn.2009.21375.
- Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning,

- multimodality, long context, and next generation agentic capabilities, 2025. URL <https://arxiv.org/abs/2507.06261>.
- Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.
- Yuguang Fang, K.A. Loparo, and Xiangbo Feng. Inequalities for the trace of matrix product. *IEEE Transactions on Automatic Control*, 39(12):2489–2490, 1994. doi: 10.1109/9.362841.
- Matthew Farrell, Stefano Recanatesi, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Gradient-based learning drives robust representations in recurrent neural networks by balancing compression and expansion. *Nature Machine Intelligence*, 4(6):564–573, 2022. Publisher: Nature Publishing Group UK London.
- Matthew Farrell, Stefano Recanatesi, and Eric Shea-Brown. From lazy to rich to exclusive task representations in neural networks and neural codes. *Current opinion in neurobiology*, 83:102780, 2023.
- Benjamin Fehrman, Benjamin Gess, and Arnulf Jentzen. Convergence rates for the stochastic gradient descent method for non-convex objective functions. *Journal of Machine Learning Research*, 21(136):1–48, 2020.
- Dylan Festa, Amir Aschner, Aida Davila, Adam Kohn, and Ruben Coen-Cagli. Neuronal variability reflects probabilistic inference tuned to natural image statistics. *Nature Communications*, 12(1):3635, June 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-23838-x. URL <https://www.nature.com/articles/s41467-021-23838-x>. Number: 1 Publisher: Nature Publishing Group.
- Enrico Fini, Victor G. Turrise da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *Proceedings of the*

- IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630. IEEE/CVF, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Fini_Self-Supervised_Models_Are_Continual_Learners_CVPR_2022_paper.html.
- Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 44(5):701–739, 2025.
- Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.
- Shintaro Funahashi. Working memory in the prefrontal cortex. *Brain sciences*, 7(5):49, 2017.
- Joaquin M Fuster and Garrett E Alexander. Neuron activity related to short-term memory. *Science*, 173(3997):652–654, 1971.
- Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, page 214262, 2017.
- Mario Geiger, Leonardo Petrini, and Matthieu Wyart. Landscape and training regimes in deep learning. *Physics Reports*, 924:1–18, 2021. ISSN 0370-1573. doi: 10.1016/j.physrep.2021.04.001. URL <https://www.sciencedirect.com/science/article/pii/S0370157321001290>.
- Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

- Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models, 2023. URL <https://arxiv.org/abs/2210.08933>.
- Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation, 2025. URL <https://arxiv.org/abs/2506.20639>.
- Google DeepMind. Gemini Diffusion is our new experimental research model. <https://blog.google/technology/google-deepmind/gemini-diffusion/>, May 2025. Accessed on July 25, 2025.
- Diego Granziol. Flatness is a false friend, 2020. URL <https://arxiv.org/abs/2006.09091>.
- Thomas L. Griffiths and Joshua B. Tenenbaum. Optimal predictions in everyday cognition. *Psychological Science*, 17(9):767–773, September 2006. ISSN 0956-7976. doi: 10.1111/j.1467-9280.2006.01780.x.
- Thomas L. Griffiths, Nick Chater, Charles Kemp, Amy Perfors, and Joshua B. Tenenbaum. Probabilistic models of cognition: exploring representations and inductive biases. *Trends in Cognitive Sciences*, 14(8):357–364, August 2010. ISSN 1364-6613. doi: 10.1016/j.tics.2010.05.004. URL <https://www.sciencedirect.com/science/article/pii/S1364661310001129>.
- Clemence Grislain, Hamed Rahimi, Olivier Sigaud, and Mohamed Chetouani. I-failsense: Towards general robotic failure detection with vision-language models. *arXiv preprint arXiv:2509.16072*, 2025.
- Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-based diffusion language models, 2023. URL <https://arxiv.org/abs/2305.18619>.

- Ralf M. Haefner, Pietro Berkes, and József Fiser. Perceptual Decision-Making as Probabilistic Inference by Neural Sampling. *Neuron*, 90(3):649–660, May 2016. ISSN 0896-6273. doi: 10.1016/j.neuron.2016.03.020. URL <https://www.sciencedirect.com/science/article/pii/S0896627316300113>.
- Demis Hassabis, Dhharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017. doi: 10.1016/j.neuron.2017.06.011.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.
- Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022a. URL https://openaccess.thecvf.com/content/CVPR2022/html/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.html.
- Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models, 2022b. URL <https://arxiv.org/abs/2211.15029>.
- Guillaume Hennequin, Laurence Aitchison, and Mate Lengyel. Fast Sampling-Based Inference in Balanced Neuronal Networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL <https://proceedings.neurips.cc/paper/2014/hash/a7d8ae4569120b5bec12e7b6e9648b86-Abstract.html>.
- Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew

- Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John P. Agapiou, Joel Z. Leibo, and Andrius Gruslys. Learning from demonstrations for real world reinforcement learning. *ArXiv*, abs/1704.03732, 2017. URL <https://api.semanticscholar.org/CorpusID:15254659>.
- Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html>.
- Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.
- Emiel Hooeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions, 2021. URL <https://arxiv.org/abs/2102.05379>.
- John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
- Roger A Horn and Charles R Johnson. *Topics in matrix analysis*. Cambridge university press, 1994.
- Neil M.T. Houlsby, Ferenc Huszár, Mohammad M. Ghassemi, Gergo Orbán, Daniel M. Wolpert, and Máté Lengyel. Cognitive Tomography Reveals Complex, Task-Independent Mental Representations. *Current Biology*, 23(21):2169–2175, November 2013. ISSN 0960-9822. doi: 10.1016/j.cub.2013.09.012. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3898796/>.

- Patrik Hoyer and Aapo Hyvärinen. Interpreting Neural Response Variability as Monte Carlo Sampling of the Posterior. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002. URL <https://proceedings.neurips.cc/paper/2002/hash/a486cd07e4ac3d270571622f4f316ec5-Abstract.html>.
- Jiaheng Hu, Rose Hendrix, Ali Farhadi, Aniruddha Kembhavi, Roberto Martín-Martín, Peter Stone, Kuo-Hao Zeng, and Kiana Ehsani. Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3617–3624. IEEE, 2025a.
- Yanzhe Hu, Yijie Jin, Pengfei Liu, Kai Yu, and Zhijie Deng. Lightningrl: Breaking the accuracy-parallelism trade-off of block-wise dllms via reinforcement learning, 2026. URL <https://arxiv.org/abs/2603.13319>.
- Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed S. Abdelfattah, Jae sun Seo, Zhiru Zhang, and Udit Gupta. Accelerating diffusion language model inference via efficient kv caching and guided diffusion, 2025b. URL <https://arxiv.org/abs/2505.21467>.
- Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.
- Yanping Huang and Rajesh PN Rao. Neurons as Monte Carlo Samplers: Bayesian Inference and Learning in Spiking Networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL <https://proceedings.neurips.cc/paper/2014/hash/dc58e3a306451c9d670adcd37004f48f-Abstract.html>.
- Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. Reinforcing the diffusion chain of lateral thought with diffusion language models, 2025. URL <https://arxiv.org/abs/2505.10446>.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL <https://arxiv.org/abs/2409.12186>.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi_{0.6}^*$: a vla that learns from experience, 2025. URL <https://arxiv.org/abs/2511.14759>.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion llms via adaptive parallel decoding, 2025. URL <https://arxiv.org/abs/2506.00413>.

Stanisław Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three Factors Influencing Minima in SGD, September 2018. URL <http://arxiv.org/abs/1711.04623>. arXiv:1711.04623 [cs, stat].

Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrise, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models, 2025. URL <https://arxiv.org/abs/2512.09106>.

Jiachen Jiang, Jinxin Zhou, Peng Wang, Qing Qu, Dustin Mixon, Chong You, and Zhihui Zhu.

Generalized neural collapse for a large number of classes. *arXiv preprint arXiv:2310.05351*, 2023.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. Train for the Worst, Plan for the Best: Understanding Token Ordering in Masked Diffusions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 30749–30768. PMLR, June 2025. URL <https://openreview.net/forum?id=DjJmre5IkP>.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas

- Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL <https://arxiv.org/abs/2406.09246>.
- David C. Knill and Alexandre Pouget. The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 27(12):712–719, December 2004. ISSN 0166-2236. doi: 10.1016/j.tins.2004.10.007. URL <https://www.sciencedirect.com/science/article/pii/S0166223604003352>.
- Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.
- Vignesh Kothapalli, Ebrahim Rasromani, and Vasudev Awatramani. Neural collapse: A review on modelling principles and generalization. *arXiv preprint arXiv:2206.04041*, 2022.
- Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- Dmitry Krotov and John Hopfield. Large Associative Memory Problem in Neurobiology and Machine Learning, April 2021. URL <http://arxiv.org/abs/2008.06996>. arXiv:2008.06996 [cond-mat, q-bio, stat].
- Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.
- Chulan Kwon, Ping Ao, and David J. Thouless. Structure of stochastic dynamics near fixed points. *Proceedings of the National Academy of Sciences*, 102(37):13029–13033, September 2005. doi: 10.1073/pnas.0506347102. URL <https://www.pnas.org/doi/10.1073/pnas.0506347102>. Publisher: Proceedings of the National Academy of Sciences.

- Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pages 5905–5914. PMLR, 2021.
- Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791. Conference Name: Proceedings of the IEEE.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015. doi: 10.1038/nature14539.
- Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.
- Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026.
- Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. RL-100: Performant robotic manipulation with real-world reinforcement learning, 2025. URL <https://arxiv.org/abs/2510.14830>.
- Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feed-forward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, January 1993. ISSN 0893-6080. doi: 10.1016/S0893-6080(05)80131-5. URL <https://www.sciencedirect.com/science/article/pii/S0893608005801315>.
- Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. In S. Koyejo, S. Mo-

- hamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 4328–4343. Curran Associates, Inc., 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/1be5bc25d50895ee656b8c2d9eb89d6a-Paper-Conference.pdf.
- Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. What happens after SGD reaches zero loss? –a mathematical framework, 2022b. URL <http://arxiv.org/abs/2110.06914>.
- Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020. doi: 10.1038/s41583-020-0277-3.
- Nianyi Lin, Jiajie Zhang, Lei Hou, and Juanzi Li. Boundary-guided policy optimization for memory-efficient rl of diffusion large language models. *arXiv preprint arXiv:2510.11683*, 2025a.
- Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. Failsafe: Reasoning and recovery from failures in vision-language-action models. *arXiv preprint arXiv:2510.01642*, 2025b.
- Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and LF Abbott. Optimal degrees of synaptic connectivity. *Neuron*, 93(5):1153–1164, 2017.
- Kevin Liu, Stephen Casper, Dylan Hadfield-Menell, and Jacob Andreas. Cognitive dissonance: Why do language model outputs disagree with internal representations of truthfulness? *arXiv preprint arXiv:2312.03729*, 2023.
- Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyuan Wei, Shaobo Wang, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025a.

- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution, June 2023. URL <http://arxiv.org/abs/2310.16834>. arXiv:2310.16834.
- Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation, 2023. URL <https://arxiv.org/abs/2212.09462>.
- Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. Pmlr, 2020.
- Chao Ma and Lexing Ying. On linear stability of SGD and input-smoothness of neural networks, 2021. URL <http://arxiv.org/abs/2105.13462>.
- Wei Ji Ma and Mehrdad Jazayeri. Neural Coding of Uncertainty and Probability. *Annual Review of Neuroscience*, 37(1):205–220, 2014. doi: 10.1146/annurev-neuro-071013-014017. URL <https://doi.org/10.1146/annurev-neuro-071013-014017>. [__eprint: https://doi.org/10.1146/annurev-neuro-071013-014017](https://doi.org/10.1146/annurev-neuro-071013-014017).
- Wei Ji Ma, Jeffrey M. Beck, Peter E. Latham, and Alexandre Pouget. Bayesian inference with probabilistic population codes. *Nature Neuroscience*, 9(11):1432–1438, November 2006. ISSN 1546-1726. doi: 10.1038/nn1790. URL <https://www.nature.com/articles/nn1790>. Number: 11 Publisher: Nature Publishing Group.
- Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025.

- Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.
- Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pages 23301–23320. PMLR, 2023a.
- Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023b.
- Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *The Thirteenth International Conference on Learning Representations*, 2024.
- Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. *Advances in neural information processing systems*, 28, 2015.
- Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. URL <https://arxiv.org/abs/2603.19312>.
- Andrew J. Majda and Andrea L. Bertozzi. *Vorticity and incompressible flow*. Camb. Texts Appl. Math. Cambridge University Press, Cambridge, 2002. ISBN 978-0-521-63057-3 978-0-521-63948-4. doi: 10.1017/CBO9780511613203.
- Sanjay G. Manohar, Nahid Zokaei, Sean J. Fallon, Tim P. Vogels, and Masud Husain. Neural mechanisms of attending to items in working memory. *Neuroscience & Biobehavioral Reviews*, 101:1–12, June 2019. ISSN 0149-7634. doi: 10.1016/j.neubiorev.2019.03.017. URL <https://www.sciencedirect.com/science/article/pii/S0149763419300624>.

- David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, San Francisco, 1982. ISBN 0-7167-1284-9.
- Paul Masset, Jacob A. Zavatone-Veth, J. Patrick Connor, Venkatesh N. Murthy, and Cengiz Pehlevan. Natural gradient enables fast sampling in spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 35, October 2022. URL <https://openreview.net/forum?id=Yopob26XjmL>.
- Luca Mazzucato, Alfredo Fontanini, and Giancarlo La Camera. Stimuli Reduce the Dimensionality of Cortical Activity. *Frontiers in Systems Neuroscience*, 10, February 2016. ISSN 1662-5137. doi: 10.3389/fnsys.2016.00011. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4756130/>.
- Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer, 2022. URL <https://arxiv.org/abs/2110.02178>.
- Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. *Advances in Neural Information Processing Systems*, 35:34532–34545, 2022.
- Tom Minka. Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research, Cambridge, UK, 2005. URL <https://www.microsoft.com/en-us/research/publication/divergence-measures-and-message-passing/>.
- Manuel Molano-Mazón, João Barbosa, Jordi Pastor-Ciurana, Marta Fradera, Ru-Yuan Zhang, Jérémy Forest, Jorge del Pozo Lerida, Ji-An Li, Christopher J. Cueva, Jaime de la Rocha, Devika Narain, and Guangyu Robert Yang. Neurogym: An open resource for developing and sharing neuroscience tasks, 2022. URL <https://doi.org/10.31234/osf.io/aqc9n>.
- Gianluigi Mongillo, Omri Barak, and Misha Tsodyks. Synaptic theory of working memory. *Science*, 319(5869):1543–1546, 2008.

- Rubén Moreno-Bote, John Rinzel, and Nava Rubin. Noise-induced alternations in an attractor network model of perceptual bistability. *Journal of Neurophysiology*, 98(3):1125–1139, 2007. doi: 10.1152/jn.00116.2007.
- Max Müller-Eberstein, Rob van der Goot, Barbara Plank, and Ivan Titov. Subspace chronicles: How linguistic information emerges, shifts and interacts during language model training. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13190–13208. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.879. URL <https://aclanthology.org/2023.findings-emnlp.879/>.
- Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT Press, Cambridge, MA, 2022. ISBN 978-0-262-04682-4. URL <https://probml.github.io/pml-book/book1.html>.
- Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.
- Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.
- Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL <https://arxiv.org/abs/2502.09992>.

- Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996. ISSN 1476-4687. doi: 10.1038/381607a0. URL <https://www.nature.com/articles/381607a0>. Number: 6583 Publisher: Nature Publishing Group.
- Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.
- Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. URL <https://arxiv.org/abs/2304.07193>.
- Gergo Orban, Pietro Berkes, József Fiser, and Máté Lengyel. Neural Variability and Sampling-Based Probabilistic Representations in the Visual Cortex. *Neuron*, 92(2):530–543, October 2016. ISSN 0896-6273. doi: 10.1016/j.neuron.2016.09.038. URL [https://www.cell.com/neuron/abstract/S0896-6273\(16\)30639-0](https://www.cell.com/neuron/abstract/S0896-6273(16)30639-0). Publisher: Elsevier.
- Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data, 2025. URL <https://arxiv.org/abs/2406.03736>.
- Vardan Papayan, XY Han, and David L Donoho. Prevalence of neural collapse during the

- terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.
- Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrod Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path planning for masked diffusion model sampling, 2025. URL <https://arxiv.org/abs/2502.03540>.
- Henning Petzka, Michael Kamp, Linara Adilova, Cristian Sminchisescu, and Mario Boley. Relative flatness and generalization. *Advances in neural information processing systems*, 34:18420–18432, 2021.
- Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.
- Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624, 2023.
- Hong Qian. The zeroth law of thermodynamics and volume-preserving conservative system in equilibrium with stochastic damping. *Physics Letters A*, 378(7):609–616, January 2014. ISSN 0375-9601. doi: 10.1016/j.physleta.2013.12.028. URL <https://www.sciencedirect.com/science/article/pii/S0375960113011766>.
- Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. d3llm: Ultra-fast diffusion llm using pseudo-trajectory distillation. *ArXiv preprint*, to appear, 2025. URL <https://github.com/hao-ai-lab/d3LLM>.

- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. URL <https://proceedings.mlr.press/v139/radford21a.html>.
- Hubert Ramsauer, Bernhard Schöfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need, 2021.
- Rajesh P. N. Rao, Bruno A. Olshausen, and Michael S. Lewicki, editors. *Probabilistic models of the brain: Perception and neural function*. Probabilistic models of the brain: Perception and neural function. The MIT Press, Cambridge, MA, US, 2002. ISBN 978-0-262-18224-9. doi: 10.7551/mitpress/5583.001.0001. Pages: x, 324.
- Aviv Ratzon, Dori Derdikman, and Omri Barak. Representational drift as a result of implicit regularization, 2023. URL <https://www.biorxiv.org/content/10.1101/2023.05.04.539512v3>. Pages: 2023.05.04.539512 Section: New Results.
- Keith Rayner, Kathryn H. Chace, Timothy J. Slattery, and Jane Ashby. Eye movements as reflections of comprehension processes in reading. *Scientific Studies of Reading*, 10(3): 241–255, 2006a. doi: 10.1207/s1532799xssr1003_3.
- Keith Rayner, Simon P. Liversedge, and Sarah J. White. Eye movements when reading disappearing text: The importance of the word to the right of fixation. *Vision Research*, 46(3):310–323, 2006b. doi: 10.1016/j.visres.2005.06.018.
- Keith Rayner, Timothy J. Slattery, Denis Drieghe, and Simon P. Liversedge. Eye movements and word skipping during reading: Effects of word length and predictability. *Journal of*

Experimental Psychology: Human Perception and Performance, 37(2):514–528, 2011. doi: 10.1037/a0020990.

Stefano Recanatesi, Matthew Farrell, Madhu Advani, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Dimensionality compression and expansion in deep neural networks. *arXiv preprint arXiv:1906.00443*, 2019.

Stefano Recanatesi, Serena Bradde, Vijay Balasubramanian, Nicholas A Steinmetz, and Eric Shea-Brown. A scale-dependent measure of system dimensionality. *Patterns*, 3(8), 2022.

Blake A. Richards, Timothy P. Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, Colleen J. Gillon, Danijar Hafner, Adam Kepecs, Nikolaus Kriegeskorte, Peter Latham, Grace W. Lindsay, Kenneth D. Miller, Richard Naud, Christopher C. Pack, Panayiota Poirazi, Pieter Roelfsema, Joao Sacramento, Andrew Saxe, Benjamin Scellier, Anna C. Schapiro, Walter Senn, Greg Wayne, Daniel Yamins, Friedemann Zenke, Joel Zylberberg, Denis Therien, and Konrad P. Kording. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, 2019. doi: 10.1038/s41593-019-0520-2.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*, 2023.

Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. Improving reasoning for diffusion language models via group diffusion policy optimization. *arXiv preprint arXiv:2510.08554*, 2025.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. URL <http://arxiv.org/abs/1505.04597>. arXiv:1505.04597 [cs].

- Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575:607–617, 2019. doi: 10.1038/s41586-019-1677-2.
- W. Rudin. *Functional Analysis*. International series in pure and applied mathematics. McGraw-Hill, 1991. ISBN 978-0-07-054236-5. URL https://www.google.com/books/edition/Functional_Analysis/Sh_vAAAAAAAJ?hl=en.
- Camille Rullán Buxó and Cristina Savin. A sampling-based circuit for optimal decision making. In *Advances in Neural Information Processing Systems*, volume 34, pages 14163–14175. Curran Associates, Inc., 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/76444b3132fda0e2aca778051d776f1c-Abstract.html>.
- Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models, June 2024. URL <http://arxiv.org/abs/2406.07524>. arXiv:2406.07524.
- Cristina Savin and Sophie Denève. Spatio-temporal Representations of Uncertainty in Spiking Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL <https://proceedings.neurips.cc/paper/2014/hash/4e2545f819e67f0615003dd7e04a6087-Abstract.html>.
- Elizabeth R. Schotter, Bernhard Angele, and Keith Rayner. Parafoveal processing in reading. *Attention, Perception, & Psychophysics*, 74(1):5–35, 2012. doi: 10.3758/s13414-011-0219-2.
- Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and Generalized Masked Diffusion for Discrete Data, June 2024. URL <http://arxiv.org/abs/2406.04329>. arXiv:2406.04329.
- Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way, 2022. URL <https://arxiv.org/abs/2205.13554>.

- Asya Shpiro, Rubén Moreno-Bote, Nava Rubin, and John Rinzel. Balance between noise and adaptation in competition models of perceptual bistability. *Journal of Computational Neuroscience*, 27(1):37–54, 2009. doi: 10.1007/s10827-008-0125-3.
- Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.
- Juha Silvanto. Working memory maintenance: Sustained firing or synaptic mechanisms? *Trends in Cognitive Sciences*, 21(3):152–154, 2017. ISSN 1364-6613. doi: <https://doi.org/10.1016/j.tics.2017.01.009>. URL <https://www.sciencedirect.com/science/article/pii/S1364661317300153>.
- Eero P Simoncelli and Bruno A Olshausen. Natural Image Statistics and Neural Representation. *Annual Review of Neuroscience*, 24(1):1193–1216, 2001. doi: 10.1146/annurev.neuro.24.1.1193. URL <https://doi.org/10.1146/annurev.neuro.24.1.1193>. _eprint: <https://doi.org/10.1146/annurev.neuro.24.1.1193>.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
- Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html>.
- Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, February 2021. URL <http://arxiv.org/abs/2011.13456>. arXiv:2011.13456 [cs, stat].
- Kartik K Sreenivasan, Clayton E Curtis, and Mark D’Esposito. Revisiting the role of persistent neural activity during working memory. *Trends in cognitive sciences*, 18(2): 82–89, 2014.

- Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Sean Kirmani, Brianna Zitkovich, Fei Xia, et al. Open-world object manipulation using pre-trained vision-language models. *arXiv preprint arXiv:2303.00905*, 2023.
- David Sussillo and L. F. Abbott. Generating Coherent Patterns of Activity from Chaotic Neural Networks. *Neuron*, 63(4):544–557, August 2009. ISSN 0896-6273. doi: 10.1016/j.neuron.2009.07.018. URL <https://www.sciencedirect.com/science/article/pii/S0896627309005479>.
- Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, et al. Robo-dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*, 2025.
- Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted Policy Optimization for Reasoning in Diffusion Language Models, July 2025. URL <http://arxiv.org/abs/2507.08838>. arXiv:2507.08838 [cs].
- Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D’Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew,

Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL <https://arxiv.org/abs/2503.20020>.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*, 2023.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis, 2019.

Danil Tyulmankov, Ching Fang, Annapurna Vadaparty, and Guangyu Robert Yang. Biological learning in key-value memory networks, 2021.

Danil Tyulmankov, Guangyu Robert Yang, and L. F. Abbott. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. *Neuron*, 110(3): 544–557.e8, February 2022. ISSN 1097-4199. doi: 10.1016/j.neuron.2021.11.009.

- Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator, 2014. URL <https://arxiv.org/abs/1310.1757>.
- Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization, 2023. URL <https://arxiv.org/abs/2303.14189>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.
- Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Computation*, 23(7):1661–1674, July 2011. ISSN 0899-7667. doi: 10.1162/NECO_a_00142. Conference Name: Neural Computation.
- Saurabh Vyas, Matthew D. Golub, David Sussillo, and Krishna V. Shenoy. Computation Through Neural Population Dynamics. *Annual Review of Neuroscience*, 43(1):249–275, 2020. doi: 10.1146/annurev-neuro-092619-094115. URL <https://doi.org/10.1146/annurev-neuro-092619-094115>. __eprint: <https://doi.org/10.1146/annurev-neuro-092619-094115>.
- Eszter Vértés and Maneesh Sahani. Flexible and accurate inference and learning for deep generative models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL <https://proceedings.neurips.cc/paper/2018/hash/955cb567b6e38f4c6b3f28cc857fc38c-Abstract.html>.
- Andrew Wagenmaker, Mitsuhiro Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.
- Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do nlp models know numbers? probing numeracy in embeddings. *arXiv preprint arXiv:1909.07940*, 2019.

- Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model, 2019. URL <https://arxiv.org/abs/1902.04094>.
- Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, et al. Spg: Sandwiched policy gradient for masked diffusion language models. *arXiv preprint arXiv:2510.09541*, 2025a.
- Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling, 2025b. URL <https://arxiv.org/abs/2503.00307>. Advances in Neural Information Processing Systems.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR 2023*, 2023. URL <https://arxiv.org/abs/2203.11171>.
- Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*, 2025c.
- Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast sampling: The three golden principles, 2025. URL <https://arxiv.org/abs/2506.10848>.
- Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*, 2022.
- Kaiyue Wen, Zhiyuan Li, and Tengyu Ma. Sharpness minimization algorithms do not only minimize sharpness to achieve better generalization, 2023.
- James C. R. Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23(3):235–250, 2019. doi: 10.1016/j.tics.2018.12.005.

- Ross Wightman. Pytorch image models. <https://github.com/rwightman/pytorch-image-models>, 2019.
- Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7959–7971, June 2022.
- Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding, 2025. URL <https://arxiv.org/abs/2505.22618>.
- Haiyu Wu, Randall Balestriero, and Morgan Levine. Visreg: Variance-invariance-sketching regularization for japa training, 2026. URL <https://arxiv.org/abs/2606.02572>.
- Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.
- Lei Wu, Chao Ma, and Weinan E. How SGD Selects the Global Minima in Over-parameterized Learning: A Dynamical Stability Perspective. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper_files/paper/2018/hash/6651526b6fb8f29a00507de6a49ce30f-Abstract.html.
- Lei Wu, Mingze Wang, and Weijie Su. The alignment property of sgd noise and how it helps select flat minima: A stability analysis, 2022.
- Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. URL <http://arxiv.org/abs/1708.07747>.

- Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, et al. Self-improving vision-language-action models with data generation via residual rl. *arXiv preprint arXiv:2511.00091*, 2025.
- Zeke Xie, Issei Sato, and Masashi Sugiyama. A Diffusion Theory For Deep Learning Dynamics: Stochastic Gradient Descent Exponentially Favors Flat Minima, January 2021. URL <http://arxiv.org/abs/2002.03495>. arXiv:2002.03495 [cs, stat].
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL <https://arxiv.org/abs/2409.12122>.
- Ling Yang, Ye Tian, Bowen Li, Xincheng Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models, 2025. URL <https://arxiv.org/abs/2505.15809>.
- Ning Yang, Chao Tang, and Yuhai Tu. Stochastic gradient descent introduces an effective landscape-dependent regularization favoring flat solutions. *Physical Review Letters*, 130(23):237101, 2023. doi: 10.1103/PhysRevLett.130.237101. URL <https://link.aps.org/doi/10.1103/PhysRevLett.130.237101>. Publisher: American Physical Society.
- Yaoqing Yang, Liam Hodgkinson, Ryan Theisen, Joe Zou, Joseph E Gonzalez, Kannan Ramchandran, and Michael W Mahoney. Taxonomizing local versus global structure in neural network loss landscapes. *Advances in Neural Information Processing Systems*, 34: 18722–18733, 2021.
- Lin Yao. Remask, don’t replace: Token-to-mask refinement in diffusion large language models, 2026a. URL <https://arxiv.org/abs/2604.18738>.
- Lin Yao. Self-generated error training for token editing in diffusion language models, 2026b. URL <https://arxiv.org/abs/2606.17175>.

- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL <https://arxiv.org/abs/2508.15487>.
- Eviatar Yemini, Albert Lin, Amin Nejatbakhsh, Erdem Varol, Ruoxi Sun, Gonzalo E Mena, Aravinthan DT Samuel, Liam Paninski, Vivek Venkatachalam, and Oliver Hobert. Neuropal: a multicolor atlas for whole-brain neuronal identification in *c. elegans*. *Cell*, 184(1):272–288, 2021.
- Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*, 2023.
- Jesse Zhang, Minh Heo, Zuxin Liu, Erdem Biyik, Joseph J Lim, Yao Liu, and Rasool Fakoor. Extract: Efficient policy learning by extracting transferable robot skills from offline data. *arXiv preprint arXiv:2406.17768*, 2024.
- Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations. *arXiv preprint arXiv:2505.10911*, 2025.
- Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. Progresslm: Towards progress reasoning in vision-language models. *arXiv preprint arXiv:2601.15224*, 2026.
- Hanyang Zhao, Dawen Liang, Wenpin Tang, David Yao, and Nathan Kallus. Diffpo: Training diffusion llms to reason fast and furious via reinforcement learning, 2025a. URL <https://arxiv.org/abs/2510.02212>.
- Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning, 2025b. URL <https://arxiv.org/abs/2504.12216>.

- Haoyang Zheng, Xinyang Liu, Cindy Xiangrui Kong, Nan Jiang, Zheyuan Hu, Weijian Luo, Wei Deng, and Guang Lin. Ultra-fast language generation via discrete diffusion divergence instruct, 2025. URL <https://arxiv.org/abs/2509.25035>.
- Yuchen Zhu, Wei Guo, Jaemoo Choi, Petr Molodyk, Bo Yuan, Molei Tao, and Yongxin Chen. Enhancing reasoning for diffusion llms via distribution matching policy optimization. *arXiv preprint arXiv:2510.08233*, 2025.
- Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects, June 2019. URL <http://arxiv.org/abs/1803.00195>. arXiv:1803.00195 [cs, stat].
- Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems*, 34:29820–29834, 2021.