Learned Feedback & Feedforward Perception & Control

Thesis by
Joseph L. Marino

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2021
Defended May 21, 2021

Joseph L. Marino
ORCID: 0000-0001-6387-8062

To my father, Steve, and my grandmother, Elizabeth.

ACKNOWLEDGEMENTS

To the late **Jerry Pine**, thank you for allowing me into the Caltech community.

To my **advisors**, Yisong Yue and Pietro Perona, thank you for your guidance, encouragement, and compassion.

To my **collaborators**, thank your for your dedication and insights.

To the **vision lab**, Steve Branson, Bo Chen, Ron Appel, Eyrun Eyjolfsdottir, Krzysztof Chalupka, David Hall, Mason McGill, Matteo Ruggero Ronchi, Grant Van Horn, Serim Ryou, Alvita Tran, Oisin Mac Aodha, Cristina Segalin, Tony Zhang, Sara Beery, Eli Cole, Jennifer Sun, and Neehar Kondapaneni: thank you for all of the wonderful memories.

To the **Yue Crew**, thank you for broadening my perspectives. Special thanks to my fellow former physicists Jeremy Bernstein and Uriah Israel.

To my **CNS cohort**, Yang Liu and Koichiro Kajikawa, thank you for your friendship.

To **IMPLiCIT**, thank you for the improvisational space to explore ideas.

To all of the other **friends**, thank you for making my time at Caltech memorable.

To the **Hairy Chests**, Matteo Ruggero Ronchi and Alessandro Zocca, thank you for the adventures and support. Our Italian coffee breaks were a highlight of the day.

To my **family**, thank you for your love.

ABSTRACT

The notions of feedback and feedforward information processing gained prominence under cybernetics, an early movement at the dawn of computer science and theoretical neuroscience. Negative feedback processing corrects errors, whereas feedforward processing makes predictions, thereby preemptively reducing errors. A key insight of cybernetics was that such processes can be applied to both perception, or state estimation, and control, or action selection. The remnants of this insight are found in many modern areas, including predictive coding in neuroscience and deep latent variable models in machine learning. This thesis draws on feedback and feedforward ideas developed within predictive coding, adapting them to improve machine learning techniques for perception (Part II) and control (Part III). Upon establishing these conceptual connections, in Part IV, we traverse this bridge, from machine learning back to neuroscience, arriving at new perspectives on the correspondences between these fields.

PUBLISHED CONTENT AND CONTRIBUTIONS

Yang, Ruihan, Yibo Yang, Joseph Marino, and Stephan Mandt (2021). "Hierarchical Autoregressive Modeling for Neural Video Compression". In: *International Conference on Learning Representations*. URL: https://openreview.net/pdf?id=TK_6nNb_C7q.
J.M. participated in the conception of the project and the writing of the manuscript.

Guerra, Alex and Joseph Marino (2020). "Sequential Autoregressive Flow-Based Policies". In: *ICML workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.
J.M. participated in the conception of the project, conducting experiments, and the writing of the manuscript.

Marino, Joseph, Lei Chen, Jiawei He, and Stephan Mandt (2020). "Improving Sequential Latent Variable Models with Autoregressive Flows". In: *Symposium on Advances in Approximate Bayesian Inference*, pp. 1–16. URL: http://proceedings.mlr.press/v118/marino20a.html.
J.M. participated in the conception of the project, implementation, analysis, and the writing of the manuscript.

Marino, Joseph, Alexandre Piché, Alessandro Davide Ialongo, and Yisong Yue (2020). "Iterative Amortized Policy Optimization". In: *Preprint*. URL: https://arxiv.org/abs/2010.10670.
J.M. participated in the conception of the project, conducting experiments, and the writing of the manuscript.

He, Jiawei, Yu Gong, Joseph Marino, Greg Mori, and Andreas Lehrmann (2019). "Variational autoencoders with jointly optimized latent dependency structure". In: *International Conference on Learning Representations*. URL: https://openreview.net/pdf?id=SJgsCjCqt7.
J.M. participated in the conception of the project, derivations, analysis, and writing of the manuscript.

Marino, Joseph (2019). "Predictive Coding, Variational Autoencoders, and Biological Connections". In: *NeurIPS Workshop on Real Neurons and Hidden Units*. URL: https://openreview.net/forum?id=SyeumQYUUH.
J.M. participated in the conception of the project and the writing of the manuscript.

Marino, Joseph, Alexandre Piché, and Yisong Yue (2019). "On the Design of Variational RL Algorithms". In: *NeurIPS Workshop on Deep Reinforcement Learning*. URL: https://drive.google.com/file/d/10hB0AS_naGNgSNG8p1kqElnc9HRmMYde/view?usp=drivesdk.
J.M. participated in the conception of the project, conducting experiments, and the writing of the manuscript.

He, Jiawei, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal (2018). "Probabilistic video generation using holistic attribute control". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 452–467. URL: http://openaccess.thecvf.com/content_ECCV_2018/html/Jiawei_He_Probabilistic_Video_Generation_ECCV_2018_paper.html.
J.M. participated in the conception of the project, derivations, baseline comparisons, and the writing of the manuscript.

Marino, Joseph, Milan Cvitkovic, and Yisong Yue (2018). "A general method for amortizing variational filtering". In: *Advances in Neural Information Processing Systems*, pp. 7857–7868. URL: http://papers.nips.cc/paper/8011-a-general-method-for-amortizing-variational-filtering.
J.M. participated in the conception of the project, implementation, analysis, and the writing of the manuscript.

Marino, Joseph, Yisong Yue, and Stephan Mandt (2018). "Iterative Amortized Inference". In: *International Conference on Machine Learning*, pp. 3403–3412. URL: http://proceedings.mlr.press/v80/marino18a.html.
J.M. participated in the conception of the project, implementation, analysis, and the writing of the manuscript.

TABLE OF CONTENTS

Acknowledgements . . . iv
Abstract . . . v
Published Content and Contributions . . . vi
Table of Contents . . . vii
List of Illustrations . . . x
List of Tables . . . xxi

I Introduction 1
Chapter I: Introduction . . . 2
1.1 Cybernetics . . . 2
1.2 Neuroscience and Machine Learning, Convergence and Divergence . . . 3
1.3 Predictive Coding Meets Machine Learning . . . 7
1.4 The Future of Feedback & Feedforward . . . 8
Chapter II: Background . . . 15
2.1 Introduction . . . 15
2.2 Probabilistic Models . . . 15
2.3 Variational Inference . . . 23
2.4 Discussion . . . 30

II Perception 37
Chapter III: Iterative Amortized Inference . . . 38
3.1 Introduction . . . 38
3.2 Issues with Direct Inference Models . . . 39
3.3 Iterative Amortized Inference . . . 42
3.4 Iterative Inference in Latent Gaussian Models . . . 44
3.5 Experiments . . . 47
3.6 Discussion . . . 52
Chapter IV: Amortized Variational Filtering . . . 56
4.1 Introduction . . . 56
4.2 Background . . . 57
4.3 Variational Filtering . . . 60
4.4 Experiments . . . 64
4.5 Discussion . . . 70
4.6 Appendix: Filtering ELBO Derivation . . . 70
Chapter V: Improving Sequential Latent Variable Models with Autoregressive Flows . . . 76
5.1 Introduction . . . 76

5.2 Method 77
5.3 Experiments 80
5.4 Discussion 86
5.5 Appendix: ELBO Derivation 86

III Control **90**
Chapter VI: Iterative Amortized Policy Optimization 91
6.1 Introduction 91
6.2 Background 93
6.3 Iterative Amortized Policy Optimization 98
6.4 Experiments 102
6.5 Discussion 108
Chapter VII: Sequential Autoregressive Flow-Based Policies 114
7.1 Introduction 114
7.2 Autoregressive Flow-Based Policies 115
7.3 Experiments 117
7.4 Discussion 121

IV Discussion **124**
Chapter VIII: Connections to Predictive Coding & Neuroscience 125
8.1 Introduction 125
8.2 Predictive Coding 125
8.3 Connections 135
8.4 Correspondences 137
8.5 Discussion 148
Chapter IX: Conclusion 158
9.1 Iterative Estimation 158
9.2 Combining Generative Perception & Control 160
9.3 Controlling Perception 163
9.4 Bridging Neuroscience and Machine Learning 169

LIST OF ILLUSTRATIONS

Number *Page*

1.1 **Conceptual Overview**. The concepts put forth by cybernetics permeated the modern fields of theoretical neuroscience (left), machine learning (center), and control theory (right). The graph attempts to identify key concepts and several relevant works in these areas, although the graph can barely begin to capture the full breadth and depth of prior work, as well as the overlap between control theory and theoretical neuroscience. At the bottom of the graph, we have coarsely labeled how Parts II & III of this thesis fit into the conceptual landscape. 5

2.1 **Maximum Likelihood Estimation**. A data distribution, p_{data}, produces samples, $\mathbf{x}$, of a random variable X. Maximum likelihood estimation attempts to fit a model distribution, p_θ, to the empirical distribution of samples by maximizing the log-likelihood of data samples under the model (Eq. 2.3). 16

2.2 **Dependency Structures**. Circles denote random variables, with gray denoting observed variables and white denoting latent variables. Arrows denote probabilistic conditional dependencies. From left to right: autoregressive model (Eqs. 2.5 & 2.6), latent variable model (Eq. 2.7), hierarchical latent variable model (Eq. 2.10), autoregressive or sequential latent variable model (Eq. 2.11). For clarity, we have drawn a subset of the possible dependencies in the final model. 18

2.3 **Model Parameterization & Computation Graph**. The diagram depicts a simplified computation graph for a deep autoregressive Gaussian model. Green circles denote the conditional likelihood distribution at each step, while gray circles again denote the (distribution of) data observations. Smaller red circles denote each of the log-likelihood terms in the objective. Gradients w.r.t. these terms are backpropagated through the networks parameterizing the model's distribution parameters (red dotted lines). 22

2.4 **ELBO Computation Graphs**. **(a)** Basic computation graph for variational inference. Outlined circles denote distributions. Smaller red circles denote terms in the ELBO objective. Arrows, again, denote conditional dependencies. This notation can be used to express **(b)** hierarchical and **(c)** sequential models with various model dependencies. 25

2.5 **Stochastic Gradient Estimation**. **(a)** Variational inference with parametric approximate posteriors requires optimizing $\mathcal{L}$ w.r.t. the distribution parameters, $\boldsymbol{\lambda}$. This often requires estimating stochastic gradients (red dotted lines). **(b)** The score function estimator (Eq. 2.26) is applicable to any distribution, but suffers from high variance. Note: the solid red arrow denotes the per-sample objective, $l(\mathbf{x}, \mathbf{z}; \theta)$. **(c)** The pathwise derivative estimator (Eq. 2.27), in contrast, has lower variance, but is less widely applicable. 27

2.6 **Variational Autoencoder (VAE)**. VAEs combine direct amortization (Eq. 2.28) and the pathwise derivative estimator (Eq. 2.27, top) with Gaussian approximate posteriors to train deep latent variable models. In the model diagram (center), the amortized inference model (left) acts as an *encoder*, with the conditional likelihood (right) acting as a *decoder*. Each are parameterized by deep networks. 29

3.1 **The Amortization Gap.** Optimization surface of $\mathcal{L}$ (in *nats*) for a 2-D latent Gaussian model and an MNIST data example. Shown on the plots are the optimal estimate (★), the output of a direct inference model (♦), and an optimization trajectory of gradient ascent (●), initialized at the blue square at (0,0). The plot on the right shows an enlarged view near the optimum. Gradient-based optimization outperforms the direct inference model, exhibiting an amortization gap in performance: $\mathcal{L}(\bigstar) > \mathcal{L}(\bullet) > \mathcal{L}(\blacklozenge)$. 40

3.2 **Lack of Prior Information.** Naïve direct inference models, encoding only observations, cannot account for conditional priors in structured latent variable models. These arise in hierarchical (left) and sequential (right) latent variable models. For instance, in hierarchical models, bottom-up approximate posteriors at intermediate levels cannot account for conditional "top-down" priors which result from sampling higher-level latent variables. 41

3.3 **Iterative Amortized Inference**. Computation graph for a latent variable model with iterative amortized inference. Red dots, denoting errors or gradients from the ELBO objective, are used to update the current estimate of the approximate posterior parameters, $\boldsymbol{\lambda}$. Using the pathwise derivative estimator, one can update the inference model parameters, ϕ, learning to efficiently perform iterative inference. . . . 43

3.4 **Automatic Top-Down Inference.** Bottom-up direct amortized inference cannot account for conditional priors, whereas iterative amortized inference automatically accounts for these priors through gradients or errors. . . . 45

3.5 **Iterative Amortized Inference Optimization.** Optimization trajectory on $\mathcal{L}$ (in *nats*) for an iterative inference model with a 2D latent Gaussian model for a particular MNIST example. The estimate is initialized at $(0, 0)$ (cyan square), and the iterative inference model adaptively adjusts inference update step sizes to iteratively refine the approximate posterior estimate (♦) to arrive at the optimum (★). . . . 47

3.6 **Amortized vs. Gradient-Based Optimization.** Average ELBO over MNIST validation set during inference optimization as a function of (**a**) inference iterations and (**b**) inference wall-clock time. Iterative amortized inference converges *faster* than gradient-based optimizers to *better* estimates, remaining stable over hundreds of iterations, despite only being trained with 16 inference iterations. . . . 49

3.7 **Reconstructions Over Inference Iterations**. During inference (left to right), reconstruction means become gradually sharper, more closely resembling data examples (right). . . . 50

3.8 **Reduced Gradient Magnitudes**. Gradient magnitudes (*vertical axis*) over inference iterations (indexed by color on right) during training (*horizontal axis*) on RCV1. Approximate posterior mean gradient magnitudes decrease over inference iterations as estimates approach local maxima. . . . 50

3.9 **Hyperparameter Comparison**. ELBO for direct and iterative inference models on binarized MNIST for **(a)** additional inference iterations during training and **(b)** additional samples. Iterative inference models improve significantly with both quantities. . . . 51

4.1 **Variational Filtering Inference.** The diagram shows filtering inference within a sequential latent variable model, as outlined in Algorithm 2. The central gray region depicts inference optimization at step t, initialized at or near the corresponding prior (white), $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$. Sampling from the approximate posterior (blue) generates the conditional likelihood (green), $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})$, which is evaluated at the observation (gray), $\mathbf{x}_t$, to calculate the prediction error. This term is combined with the KL divergence between the approximate posterior and prior, yielding the step ELBO (red), $\mathcal{L}_t$ (Eq. 4.7). Inference optimization (E-step) involves finding the approximate posterior that maximizes the step ELBO terms. 61

4.2 **Filtering Inference Models for VRNN, SRNN, SVG, and AVF**. Each diagram shows the computational graph for inferring the approximate posterior parameters, $\boldsymbol{\lambda}^q$, at step t. Previously proposed methods rely on hand-crafted architectures of observations, hidden states, and latent variables. AVF is a simple, general filtering procedure that only requires the local inference gradient. 65

4.3 **Prediction-Update Visualization**. Test data (top), output predictions (middle), and updated reconstructions (bottom) for TIMIT using SRNN with AVF. Sequences run from left to right. The predictions made by the model already contain the general structure of the data. AVF explicitly updates the approximate posterior from the prior prediction, focusing on inference *corrections* rather than re-estimation. 66

4.4 **Improvement with Inference Iterations.** Results are shown on the TIMIT validation set using VRNN with AVF. **(a)** Average negative ELBO per step with varying numbers of inference iterations during training. Additional iterations result in improved performance. **(b)** Average relative ELBO improvement from the initial (prior) estimate at each inference iteration for a single model. Empirically, each successive iteration provides diminishing additional improvement. . . 68

5.1 **Redundancy Reduction**. **(a)** Conditional densities for $p(x_2|x_1)$. **(b)** The marginal, $p(x_2)$ differs from the conditional densities, thus, $\mathcal{I}(x_1; x_2) > 0$. **(c)** In the normalized space of y, the corresponding densities $p(y_2|y_1)$ are identical. **(d)** The marginal $p(y_2)$ is identical to the conditionals, so $\mathcal{I}(y_1; y_2) = 0$. Thus, in this case, a conditional affine transform removed the dependencies. 78

5.2 **Model Diagrams**. **(a)** An autoregressive flow pre-processes a data sequence, $\mathbf{x}_{1:T}$, to produce a new sequence, $\mathbf{y}_{1:T}$, with reduced temporal dependencies. This simplifies dynamics modeling for a higher-level sequential latent variable model, $p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T})$. Empty diamond nodes represent deterministic dependencies, not recurrent states. **(b)** Diagram of the autoregressive flow architecture. Blank white rectangles represent convolutional layers (see Appendix). The three stacks of convolutional layers within the blue region are shared. `cat` denotes channel-wise concatenation. 79

5.3 **Sequence Modeling with Autoregressive Flows**. **Top**: Pixel values (solid) for a particular pixel location in a video sequence. An autoregressive flow models the pixel sequence using an affine shift (dashed) and scale (shaded), acting as a frame of reference. **Middle**: Frames of the data sequence (top) and the resulting "noise" (bottom) from applying the shift and scale. The redundant, static background has been largely removed. **Bottom**: The noise values (solid) are modeled using a base distribution (dashed and shaded) provided by a higher-level model. By removing temporal redundancy from the data sequence, the autoregressive flow simplifies dynamics modeling. . . . 81

5.4 **Flow Visualization** for SLVM + 1-AF on Moving MNIST (left) and KTH Actions (right). 82

5.5 **Improved Generated Samples**. Random samples generated from (**a**) VideoFlow and (**b**) VideoFlow + AF, each conditioned on the first 3 frames. Using AF produces more coherent samples. The robot arm blurs for VideoFlow in samples 1 and 4 (red), but does not blur for VideoFlow + AF. 83

5.6 **Decreased Temporal Correlation**. **(a)** Affine autoregressive flows result in sequences, $\mathbf{y}_{1:T}$, with decreased temporal correlation, $\text{corr}_\mathbf{y}$, as compared with that of the original data, $\text{corr}_\mathbf{x}$. **(b)** For SLVM + 1-AF, $\text{corr}_\mathbf{y}$ decreases during training on KTH Actions. 84

5.7 **Improved Generalization**. The low-level reference frame improves generalization to unseen sequences. Train and test negative log-likelihood bound histograms for **(a)** SLVM and **(b)** SLVM + 1-AF on KTH Actions. **(c)** The generalization gap for SLVM + 1-AF remains small for varying amounts of KTH training data, while it becomes worse in the low-data regime for SLVM. 85

6.1 **Amortized Policy Optimization**. **(a)** 2D visualization of policy optimization. A direct amortized policy network fails to output an optimal estimate, resulting in an *amortization gap* in performance. An iterative amortized policy network finds an improved estimate. **(b)** Diagrams of direct and iterative amortized policy optimization. Larger circles denote distributions, and smaller red circles denote terms in the objective, $\mathcal{J}$. Dashed arrows denote amortized optimizers. Iterative amortization uses gradient-based feedback during optimization, whereas direct amortization does not. 95

6.2 **Estimating Multiple Policy Modes**. Unlike direct amortization, which is restricted to a single estimate, the stochasticity of iterative optimization allows iterative amortization to effectively sample from multiple high-value modes in the action space. This capability is shown for a particular state in `Ant-v2`, showing multiple optimization runs across two action dimensions (**Left**). Each colored square denotes an initialization. The optimizer finds both modes, with the assigned density plotted on the **Right**. This capability provides increased flexibility in action exploration. 100

6.3 **Mitigating Value Overestimation**. Using the same value estimation setup ($\beta = 1$), shown on `Ant-v2`, iterative policy optimization results in (**a**) higher value overestimation bias and (**b**) a more rapidly changing policy as compared with direct policy optimization. Increasing β helps to mitigate these issues by further penalizing variance in the value estimate. 101

6.4 **Policy Optimization**. Visualization over time steps of **(a)** one dimension of the policy distribution and **(b)** the improvement in the objective, $\Delta\mathcal{J}$, across policy optimization iterations. 102

6.5 **Performance Comparison**. Iterative amortized policy optimization performs comparably with or better than direct amortized policies across a range of MuJoCo environments. Performance curves show the mean and ± standard deviation over 5 random seeds. 103

6.6 **Amortization Gap**. Estimated amortization gaps per step for direct and iterative amortized policy optimization. Iterative amortization achieves comparable or lower gaps across environments. Gaps are estimated using stochastic gradient-based optimization over 100 random states. Curves show the mean and ± standard deviation over 5 random seeds. 103

6.7 **Additional Performance Comparison**. Results are shown on the remaining MuJoCo environments from OpenAI gym. Performance curves show the mean and ± standard deviation over 5 random seeds. 104

6.8 **Multiple Policy Modes**. **(a)** Histogram of distances between policy means (μ) across optimization runs (i and j) over seeds and states on `Walker2d-v2` at 3 million environment steps. For the state with the largest distance, **(b)** shows the projected optimization surface on each pair of action dimensions, and **(c)** shows the policy density for 10 optimization runs. 105

6.9 **Varying Iterations During Training**. Performance of iterative amortized policy optimization for varying numbers of iterations during training. Increasing the number of iterations generally results in improvements. Curves show the mean and ± standard deviation over 5 random seeds. 105

6.10 **Amortization Gap of Varying Iterations During Training**. Corresponding amortization gaps for varying numbers of iterations during training. We generally see that increasing the number of iterations generally reduces the amortization gap. 106

6.11 **Comparison with Iterative Optimizers**. Average estimated objective over policy optimization iterations, comparing with Adam (Kingma and Ba, 2014) and CEM (Rubinstein and Kroese, 2013). These iterative optimizers require over an order of magnitude more iterations to reach comparable performance with iterative amortization, making them impractical in many applications. 106

6.12 **Optimizing Model-Based Value Estimates**. **(a)** Performance comparison of direct and iterative amortization using model-based value estimates. **(b)** Planned trajectories over policy optimization iterations. **(c)** The corresponding estimated objective increases over iterations. **(d)** Zero-shot transfer of iterative amortization from model-free (MF) to model-based (MB) estimates. 108

7.1 **Autoregressive Flow-Based Policy**. An autoregressive flow-based policy converts samples from a base distribution, $\pi_\phi(\mathbf{u}_t|\mathbf{s}_t)$, using an affine transform with parameters $\beta_\theta(\mathbf{a}_{<t})$ and $\delta_\theta(\mathbf{a}_{<t})$, into actions, $\mathbf{a}_t$. The affine transform (top) acts as a feedforward policy, purely conditioned on previous actions, whereas the base distribution (bottom) acts as a feedback policy. Each are parameterized by a separate deep network, with parameters θ and ϕ, respectively. 116

7.2 **Performance Comparison,** 2×256 **(Default) Policy Network**. Comparison between SAC and ARSAC with an action window of 5 time steps across `dm_control` suite environments. Each curve shows the mean and standard deviation across 5 random seeds. 118

7.3 **Humanoid Performance Comparison,** 2×256 **(Default) Policy Network**. Comparison between SAC and ARSAC with varying action window sizes on `humanoid` tasks from `dm_control` suite. Each curve shows the mean and standard deviation across 5 random seeds. 119

7.4 **Performance Comparison,** 1×32 **Policy Network**. Comparison between SAC and ARSAC with an action window of 5 time steps across `dm_control` suite environments. Each curve shows the mean and standard deviation across 5 random seeds. 119

7.5 **Policy Visualization & Distillation**. **Left**: Visualization of the base distribution and autoregressive flow for various action dimensions across various environments. **Right**: Visualization after policy distillation. 120

8.1 **Spatiotemporal Predictive Coding**. **(a)** Spatial predictive coding models and removes spatial dependencies. In the domain of natural images, one version of linear predictive coding is ZCA whitening, which yields center-surround filters (left). As a result, the whitened image contains highlighted edges (right). **(b)** Temporal predictive coding models and removes temporal dependencies. In the domain of natural video, this tends to remove static backgrounds. 127

8.2 **Brain Anatomy & Cortical Circuitry**. **Left**: Sensory inputs enter first-order relays in thalamus from sensory organs. Thalamus forms reciprocal connections with neocortex. Neocortex consists of hierarchies of cortical areas, with both forward and backward connections. **Right**: Neocortex is composed of six layers (I–VI), with specific neuron classes and connections at each layer. The simplified schematic depicts two cortical columns. Black and red circles represent excitatory and inhibitory neurons respectively, with arrows denoting major connections. This basic circuit motif is repeated with slight variations throughout neocortex. 129

8.3 **Hierarchical Predictive Coding**. The diagram shows the basic computation graph for a Gaussian latent variable model with MAP inference. The insets show the weighted error calculation for the latent (left) and observed (right) variables. 131

8.4 **Hierarchical Predictive Coding & VAEs**. Computation diagrams for **(a)** hierarchical predictive coding, **(b)** VAE with direct amortized inference, and **(c)** VAE with iterative amortized inference (Chapter 3). $\mathbf{J}^\intercal$ denotes the transposed Jacobian matrix of the generative model's conditional likelihood. Red dotted lines denote gradients, and black dashed lines denote amortized inference. Hierarchical predictive coding and VAEs are highly similar in both their model formulation and inference approach. 138

8.5 **Pyramidal Neurons & Deep Networks**. Connecting deep latent variable models with predictive coding places deep networks (bottom) in correspondence with the dendrites of pyramidal neurons (top). This is in contrast with conventional one-to-one analogies of biological and artificial neurons, suggesting a larger role for non-linear dendritic computation and alternative correspondences for backpropagation. . . 140

8.6 **Pyramidal Neurons & Amortization**. In predictive coding, inference updating is implemented using forward pyramidal neurons in cortex, taking prediction errors as input. In deep latent variable models, iterative amortized inference plays a similar role, continuing the analogy of pyramidal neurons and deep networks. Interestingly, this suggests a separation of processing in apical and basal dendrites, incorporating errors from the current and lower latent level. 141

8.7 **Backpropagation**. Placing deep networks in correspondence with pyramidal neuron dendrites suggests an alternative perspective on the biological-plausibility of backpropagation. In deep latent variable models, backpropagation is only performed across variables that are directly connected through a conditional probability (left). From the perspective presented here, this corresponds to learning *within* pyramidal neurons. One possible implementation may be through backpropagating action potentials, perhaps combined with other neuromodulatory inputs (right). 142

8.8 **Computational Schematic of the Visual Pathway**. Interpreting the early visual pathway from the perspective of a latent variable model, we can assign computational functions to retina, LGN, and cortex. Retina and LGN are interpreted as implementing normalizing flows, i.e. spatiotemporal predictive coding, reducing spatial and temporal redundancy in the visual input. LGN also serves as the lowest level for hierarchical predictions, which are computed through backward connections in cortex. Using prediction errors throughout the hierarchy, forward cortical connections update latent estimates. . . 145

8.9 **Adding & Removing Dependencies with Normalizing Flows**. Normalizing flows provides a general mathematical framework for removing (left) or adding (right) probabilistic dependencies. Using lateral interactions, one can move between a normalized (top) or un-normalized (bottom) space. Normalized spaces have benefits for compression, whereas un-normalized spaces are more expressive. Neural systems may employ these transforms for sensory and motor processing. 147

9.1 **Thesis Summary**. Graphical representations of perception (**a–c**) and control (**d–e**) using feedback (**a, b, d**) and feedforward (**c, e**) processes. 159

9.2 **Combining Generative Perception & Control**. Graphical representation combining generative perception (Part II) and control (Part III). Internal perceptual latent variables are a hierarchical variable in the control policy. Importantly, the agent maximizes the task-relevant *information gain* from the environment's state observation, rather than the conditional log-likelihood. 162

9.3 **State Distributions & Rewards.** A quadratic reward function, $r(S)$, can be equivalently expressed as a Gaussian desired state distribution, $p_d(S)$. 166

9.4 **Comparator Circuit**. Using the error between the current state and the input reference signal (or setpoint), a compensator-effector function updates the output control. The feedback takeoff function converts this control output into the current state. An ideal compensator-effector minimizes the discrepancy between the current state and the reference signal. Adapted from Wiener, 1948. 167

LIST OF TABLES

Number *Page*

1.1 **Thesis Organization.** 8

3.1 **Negative Log-Likelihood** on binarized MNIST (in *nats*) for direct and iterative amortized inference. 52

3.2 **Negative Log-Likelihood** on CIFAR-10 (in *bits/dim.*) for direct and iterative amortized inference. 52

3.3 **Perplexity** on RCV1 for direct and iterative amortized inference. . . . 52

4.1 **Average negative ELBO per step** (in *nats*) on the TIMIT speech dataset for SRNN and VRNN with the respective originally proposed filtering procedures (baselines) and with AVF. 69

4.2 **Average negative ELBO per step** (in *nats per dimension*) on the KTH Actions video dataset for SVG with the originally proposed filtering procedure (baseline) and with AVF. 69

4.3 **Average negative ELBO per step** (in *nats*) on polyphonic music datasets for SRNN with and without AVF. Results from Fraccaro, S. K. Sønderby, et al., 2016 are provided for comparison, however, our model implementation differs in several aspects (see Marino, Cvitkovic, and Yue, 2018). 69

5.1 **Quantitative Comparison.** Average test and (train) negative log-likelihood in *nats per dimension* for Moving MNIST, BAIR Robot Pushing, and KTH Actions. Lower values are better. 85

8.1 Proposed Neural Correspondences of Hierarchical Predictive Coding. 133

Part I

Introduction

Chapter 1

INTRODUCTION

This thesis operates at multiple levels. At the lowest level, this thesis presents two core techniques for improving probabilistic modeling, applied to both state estimation, or *perception*, and action selection, or *control*. At a higher level, this thesis fortifies a bridge between neuroscience and machine learning through the theory of predictive coding. And at a higher level still, this thesis is an attempt to help re-integrate the concepts and framework of *cybernetics* back into the current scientific discourse.

1.1 Cybernetics

The field of cybernetics (Wiener, 1948; Ashby, 1956) was an agglomeration of various disciplines, encompassing aspects of what would become computer science, neuroscience, control theory, and the social sciences. Despite their diversity, a common theme unified these interdisciplinary investigations: *feedback*. Broadly, feedback is the process of *feed*ing the output of a system *back* into the system itself. Such processes come in two flavors, with positive feedback amplifying a signal and negative feedback attenuating a signal. Negative feedback is particularly useful in engineering applications (Astrom and R. M. Murray, 2008), where it can reduce the error between two signals. A related concept is that of *feedforward* processes, which attempt to predict a signal, preemptively reducing error. These ideas around information processing formed the basis of cybernetics' formulation of perception and control, both in biological and non-biological systems, in terms of feedback and feedforward processes.

Perception and control, though seemingly disparate concepts, are intimately related. In both cases, given a model of how the corresponding variable (states or actions) affects observed outcomes, one can 1) *infer* an estimate of the variable that would result in an observation, and 2) *learn* an improved model to better predict observations. Inference and learning, by minimizing prediction errors, are negative feedback processes, while the model itself, formulated across time, may involve feedforward processes. Cybernetics formalized these techniques using probabilistic models, i.e., models that estimate the likelihood of random outcomes, and variational calculus, an optimization technique for estimating functions, including probability distributions (Wiener, 1948). By using these techniques to evaluate and minimize

error signals, such systems can exhibit goal-directed, or *teleological*, behavior, in which desired outcomes appear to drive action. This provided a novel perspective in understanding how goal-directed behavior can arise in biological organisms composed of ordinary matter (Rosenblueth, Wiener, and Bigelow, 1943), as well as a path toward constructing more capable machines.

With these new techniques and unifying perspectives, cybernetics helped usher in a new era of interdisciplinary research, aiming to establish common computational principles underlying both biological and non-biological systems. At this intersection came the first computational models of neuron function (McCulloch and Pitts, 1943; Rosenblatt, 1958), a formal definition of information (Wiener, 1942; Shannon, 1948) (with connections to neural systems (Barlow, 1961)), and algorithms for negative feedback perception and control (MacKay, 1956; Kalman, 1960). While further advances continued in these directions (see Prieto et al. (2016) and references therein), the field of cybernetics disbanded due to a variety of conspiring factors (Conway and Siegelman, 2006), with the new techniques and ideas surviving in the offshoots of theoretical neuroscience, machine learning, control theory, etc. Figure 1.1 outlines the progression of a subset of these ideas that are relevant to this thesis.

1.2 Neuroscience and Machine Learning, Convergence and Divergence

As the descendant fields of cybernetics progressed in their respective areas, a renewed dialogue formed between neuroscience and machine learning in the 1980s–1990s. Neuroscientists, bolstered by new physiological and functional analyses, began making traction in studying neural systems in probabilistic and information-theoretic terms (Laughlin, 1981; Srinivasan, Laughlin, and Dubs, 1982; Barlow, 1989; Bialek et al., 1991). In machine learning, improvements in probabilistic modeling (Pearl, 1986) and artificial neural networks (Rumelhart, Hinton, and Williams, 1986) combined with ideas from statistical mechanics (Hopfield, 1982; Ackley, Hinton, and Sejnowski, 1985) to yield new classes of models and training techniques. This convergence of ideas, primarily centered around perception, resulted in new theories of neural processing and improvements in their mathematical underpinnings.

In particular, the notion of *predictive coding* emerged within neuroscience (Srinivasan, Laughlin, and Dubs, 1982; Rao and Ballard, 1999). In its most general form, predictive coding postulates that neural circuits are fundamentally engaged in estimating probabilistic models of other neural activity and the surrounding environment, with feedback and feedforward processes playing a central role. These

models were initially formulated in early sensory areas, e.g., retina (Srinivasan, Laughlin, and Dubs, 1982) and thalamus (Dong and Atick, 1995), using feedforward processes to predict future neural activity. Similar notions were also extended to higher-level sensory processing in neocortex. In a series of papers by David Mumford (Mumford, 1991; Mumford, 1992), top-down neural projections (from higher-level to lower-level sensory areas) were hypothesized to convey hierarchical sensory predictions, whereas bottom-up neural projections were hypothesized to convey prediction errors. Through a negative feedback process, these errors would then update state estimates. These ideas were formalized and analyzed by Rao and Ballard, 1999, formulating a simplified artificial neural network model of images, reminiscent of a Kalman filter (Kalman, 1960).

Feedback and feedforward processes also featured prominently in machine learning. Indeed, the primary training algorithm for artificial neural networks, backpropagation (Rumelhart, Hinton, and Williams, 1986), literally *feeds* (propagates) the output prediction errors *back* through the network, i.e., negative feedback. During this period, the technique of variational inference was rediscovered within machine learning (Hinton and Van Camp, 1993; Neal and Hinton, 1998), recasting approximate probabilistic inference using variational calculus. This technique proved essential in formulating the Helmholtz machine (Dayan, Hinton, et al., 1995; Dayan and Hinton, 1996), a hierarchical probabilistic model parameterized by artificial neural networks. Similar advances were made in autoregressive probabilistic models (Frey, Hinton, and Dayan, 1996; Y. Bengio and S. Bengio, 2000), using artificial neural networks to form sequential feedforward predictions, as well as new classes of invertible probabilistic models (Comon, 1994; Parra, Deco, and Miesbach, 1995; Deco and Brauer, 1995; Bell and Sejnowski, 1997). Unfortunately, as the field moved toward simpler, more tractable models in the late 1990s, funding became scarce, and the resulting chill of an "AI winter" slowed progress in these areas.

These new ideas regarding variational inference and probabilistic models, particularly the Helmholtz machine (Dayan, Hinton, et al., 1995), influenced predictive coding. Specifically, Karl Friston utilized variational inference to formulate hierarchical dynamical models of neocortex (Friston, 2005; Friston, 2008). In line with Mumford's proposal (Mumford, 1992), these models contain multiple levels of variables, with each level attempting to predict its future activity (feedforward) as well as the activity at lower levels, closer to the input data. Through variational inference, prediction errors across levels facilitate updating higher-level estimates (negative

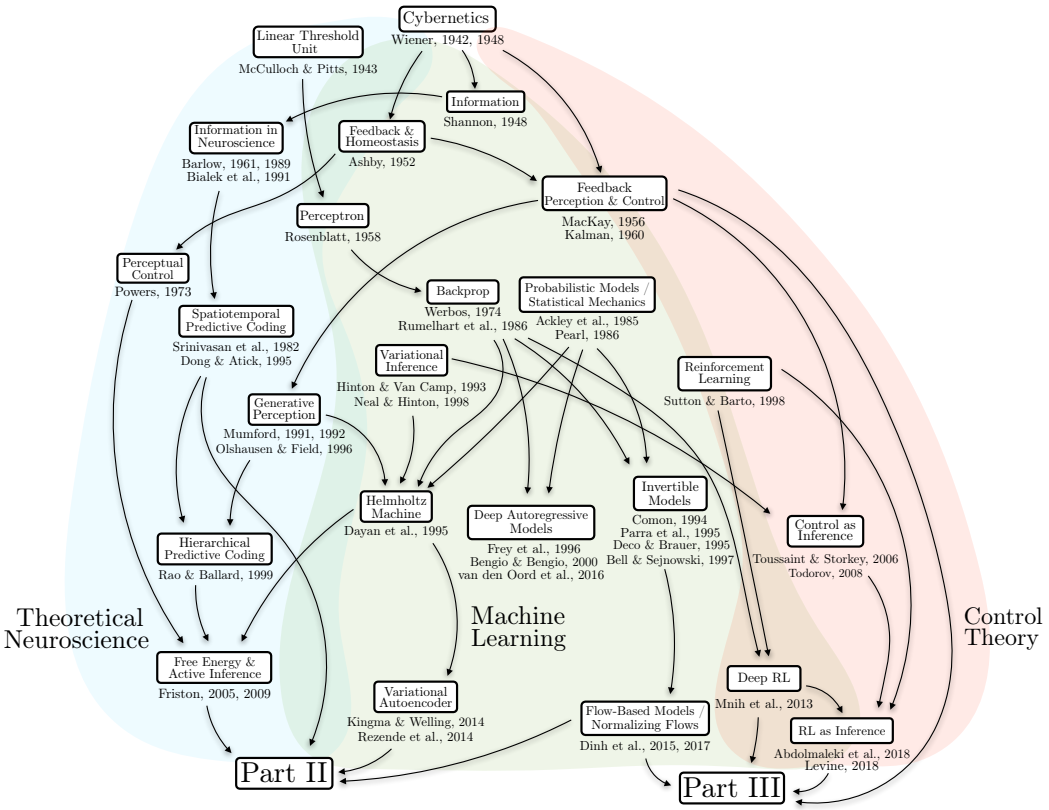


Figure 1.1: **Conceptual Overview**. The concepts put forth by cybernetics permeated the modern fields of theoretical neuroscience (left), machine learning (center), and control theory (right). The graph attempts to identify key concepts and several relevant works in these areas, although the graph can barely begin to capture the full breadth and depth of prior work, as well as the overlap between control theory and theoretical neuroscience. At the bottom of the graph, we have coarsely labeled how Parts II & III of this thesis fit into the conceptual landscape.

feedback). Such models have incorporated many aspects of neuroscience, including local learning rules (Friston, 2005) and attention (Spratling, 2008; Feldman and Friston, 2010; Kanai et al., 2015), and have been coarsely compared with neural circuits (Bastos et al., 2012; Keller and Mrsic-Flogel, 2018; Walsh et al., 2020). While predictive coding and other forms of Bayesian brain theories have become increasingly popular (Doya et al., 2007; Friston, 2009; Clark, 2013), empirically testing these normative models remains challenging. This is partially due to the difficulty of distinguishing between the large number of specific design choices and the more general theoretical claims of probabilistic learning and inference (S. J. Gershman, 2019). Further, because these models have been limited to simplified implementations, often without learned parameters, it has been difficult to bridge the gap to the complexity of biological neural systems.

The AI winter thawed in the early 2010s, brought on by advances in parallel computing as well as standardized datasets (Deng et al., 2009; Krizhevsky and Hinton, 2009) and environments (Todorov, Erez, and Tassa, 2012; Bellemare et al., 2013). In this new era of *deep learning* (LeCun, Y. Bengio, and Hinton, 2015; Schmidhuber, 2015), i.e., artificial neural networks with multiple layers, a flourishing of ideas emerged around probabilistic modeling. Building off of previous work, more expressive classes of deep hierarchical (Gregor et al., 2014; Mnih and Gregor, 2014; Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014), autoregressive (Uria, I. Murray, and Larochelle, 2014; Oord, Kalchbrenner, and Kavukcuoglu, 2016), and invertible (Dinh, Krueger, and Y. Bengio, 2015; Dinh, Sohl-Dickstein, and S. Bengio, 2017) probabilistic models were developed. Of particular importance is a class of models known as *variational autoencoders* (VAEs) (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014), which, as we discuss in Chapter 8, bear a close resemblance to hierarchical predictive coding models. Despite this similarity, the machine learning community remains largely oblivious to the progress in predictive coding that occurred during the AI winter and vice versa. Bridging this divide is one of the primary motivations for this thesis (Section 1.3).

The discussion in this section, both in neuroscience and machine learning, has centered on probabilistic modeling and inference for perception. However, as noted earlier, these techniques can be equally applied to control. This connection was re-established in the various fields that descended from cybernetics, particularly machine learning (Dayan and Hinton, 1997; Attias, 2003), control theory (Toussaint and Storkey, 2006; Todorov, 2008), and theoretical neuroscience (Friston, Daunizeau, and Kiebel, 2009; Botvinick and Toussaint, 2012). While these approaches differ in their exact formulations, they have a similar theme: "observing" desired outcomes, then using a probabilistic model to infer corresponding actions. Friston and colleagues have worked toward connecting predictive coding with control through a framework referred to as *active inference* (Friston, Daunizeau, and Kiebel, 2009; Friston, FitzGerald, et al., 2017). Modern machine learning, in contrast, often formulates control through a framework known as *reinforcement learning (RL) as inference* (Levine, 2018). Although these frameworks are not strictly equivalent (Millidge et al., 2020), they share many mathematical concepts.

Machine learning and neuroscience have diverged in many ways over the past two decades, however, the fundamental ideas emphasized by cybernetics, i.e., probabilistic modeling and inference for perception and control, remain the focus of both fields.

It is unclear whether these fields will independently arrive at similar solutions for building and understanding, respectively, autonomous agents. However, given the similarity of ideas separately housed within each field, it seems likely that bridging these divides will help to once again facilitate cross-pollination and progress.

1.3 Predictive Coding Meets Machine Learning

This thesis is, in large part, an attempt to fortify a bridge between machine learning and neuroscience by translating the ideas of predictive coding into deep probabilistic models. While the present work provides unique contributions, many of the insights were inspired by previous works at this intersection. In particular, Broeke, 2016 outlines the relationship between hierarchical probabilistic models in predictive coding and deep learning. Likewise, Lotter, Kreiman, and Cox, 2017 implement basic predictive coding techniques in deep probabilistic models, later comparing these models with neural phenomena (Lotter, Kreiman, and Cox, 2018). This thesis draws upon these ideas and many others, formalizing and extending them.

Concretely, this thesis takes the negative feedback and feedforward techniques developed in predictive coding and applies them in machine learning using the mathematics of *amortization* (S. Gershman and Goodman, 2014) and *normalizing flows* (Rezende and Mohamed, 2015; Kingma, Salimans, et al., 2016), respectively. Briefly, amortization facilitates efficiently updating estimates to minimize prediction errors (Chapters 3, 4, and 6), i.e., (negative) feedback, whereas normalizing flows, in our formulation, provide a mechanism for improving predictions across time (Chapters 5 and 7), i.e., feedforward. These techniques, which are both learned from data, are demonstrated in the contexts of perception (Part II) and control (Part III). An outline of the thesis content is provided in Table 1.1. A broad conceptual overview of the inspirations for this thesis is also provided in Figure 1.1.

While the construction of this conceptual bridge has proceeded unidirectionally from neuroscience toward machine learning, the insights developed in this process have striking implications in the reverse direction. In Chapter 8, we discuss these implications in depth. In short, traversing the bridge from machine learning, back through predictive coding, to neuroscience provides new perspectives regarding the correspondence between biological and artificial neural networks, as well as computational aspects of normalization in neural circuits (Carandini and Heeger, 2012). Though this discussion is more speculative in nature than the technical contributions of this thesis, it provides multiple lines of inquiry for future cross-

Table 1.1: **Thesis Organization.**

	Perception (Part II)	**Control** (Part III)
Feedback	Chapters 3 & 4	Chapter 6
Feedforward	Chapter 5	Chapter 7

pollination efforts.

1.4 The Future of Feedback & Feedforward

The distance between neuroscience and machine learning is to be expected, given their differing aims and system constraints. A perfect convergence of these fields is neither essential nor necessarily beneficial. Yet, the ideas set forth by cybernetics three quarters of a century ago, namely perception and control through feedback and feedforward processing, are alive and well in both fields. This thesis works toward uniting these areas, developing the ideas of cybernetics with modern computational tools. The result is a set of basic techniques for probabilistic modeling and inference based on feedback and feedforward principles, applicable to both perception and control.

However, scientific research, by its very nature, is constrained to simplified settings, where complex phenomena can be studied in isolation. The real work remains to be done in both integrating these ideas into a cohesive embodied system, as discussed in Chapter 9, as well as pushing these ideas and others to their limits to build and understand bigger, more expressive systems. If we take the correspondences in Chapter 8 seriously, biological systems still contain orders of magnitude more parameters than current machine learning models, likely with many more unforeseen complexities. If we hope to build more capable autonomous systems or understand neural processing in computational terms, a closer collaboration between these fields may prove fruitful. Indeed, these discussions are already bearing fruit: Richards and colleagues, inspired in part by the predictive coding-based machine learning models of Lotter, Kreiman, and Cox, 2017, have identified top-down predictions and bottom-up error signals in separate dendritic compartments of neocortical pyramidal neurons (Gillon et al., 2021). In this way, the exploration of predictive coding ideas in machine learning has helped neuroscientists pose more targeted scientific hypotheses.

Neuroscience and machine learning may never return to the close collaboration envisaged under cybernetics. Nevertheless, a mutually beneficial dialogue can be

maintained by tying these fields together around their unifying concepts, such as feedback and feedforward perception and control. Considering systems in these cybernetic terms could help us move beyond the foggy notions of biological and artificial *intelligence*. Instead, we can understand organisms and autonomous systems from a more firmly-grounded scientific and engineering perspective, in terms of feedback and feedforward computational processing, homeostatic objectives, and ultimately their physical embodiment of complexity and information. This thesis is a step in that direction.

References

Ackley, David H, Geoffrey E Hinton, and Terrence J Sejnowski (1985). "A learning algorithm for Boltzmann machines". In: *Cognitive science* 9.1, pp. 147–169.

Ashby, W Ross (1956). *An Introduction to Cybernetics*. Chapman and Hall.

Astrom, Karl Johan and Richard M Murray (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

Attias, Hagai (2003). "Planning by probabilistic inference." In: *AISTATS*. Citeseer.

Barlow, Horace B (1961). "The coding of sensory messages". In: *Current problems in animal behavior*.

– (1989). "Unsupervised learning". In: *Neural computation* 1.3, pp. 295–311.

Bastos, Andre Moraes et al. (2012). "Canonical microcircuits for predictive coding". In: *Neuron* 76.4, pp. 695–711.

Bell, Anthony J and Terrence J Sejnowski (1997). "The "independent components" of natural scenes are edge filters". In: *Vision research* 37.23, pp. 3327–3338.

Bellemare, Marc G et al. (2013). "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47, pp. 253–279.

Bengio, Yoshua and Samy Bengio (2000). "Modeling high-dimensional discrete data with multi-layer neural networks". In: *Advances in Neural Information Processing Systems*, pp. 400–406.

Bialek, William et al. (1991). "Reading a neural code". In: *Science* 252.5014, pp. 1854–1857.

Botvinick, Matthew and Marc Toussaint (2012). "Planning as inference". In: *Trends in cognitive sciences* 16.10, pp. 485–488.

Broeke, Gerben van den (2016). "What auto-encoders could learn from brains". MA thesis. Aalto University.

Carandini, Matteo and David J Heeger (2012). "Normalization as a canonical neural computation". In: *Nature Reviews Neuroscience* 13.1, pp. 51–62.

Clark, Andy (2013). “Whatever next? Predictive brains, situated agents, and the future of cognitive science”. In: *Behavioral and Brain Sciences* 36.3, pp. 181–204.

Comon, Pierre (1994). “Independent component analysis, a new concept?” In: *Signal processing* 36.3, pp. 287–314.

Conway, Flo and Jim Siegelman (2006). *Dark hero of the information age: In search of Norbert Wiener, the father of cybernetics*. Basic Books.

Dayan, Peter and Geoffrey E Hinton (1996). “Varieties of Helmholtz machine”. In: *Neural Networks* 9.8, pp. 1385–1403.

– (1997). “Using expectation-maximization for reinforcement learning”. In: *Neural Computation* 9.2, pp. 271–278.

Dayan, Peter, Geoffrey E Hinton, et al. (1995). “The helmholtz machine”. In: *Neural computation* 7.5, pp. 889–904.

Deco, Gustavo and Wilfried Brauer (1995). “Higher order statistical decorrelation without information loss”. In: *Advances in Neural Information Processing Systems*, pp. 247–254.

Deng, Jia et al. (2009). “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.

Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). “Nice: Non-linear independent components estimation”. In: *International Conference on Learning Representations*.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). “Density estimation using real nvp”. In: *International Conference on Learning Representations*.

Dong, Dawei W and Joseph J Atick (1995). “Temporal decorrelation: a theory of lagged and nonlagged responses in the lateral geniculate nucleus”. In: *Network: Computation in Neural Systems* 6.2, pp. 159–178.

Doya, Kenji et al. (2007). *Bayesian brain: Probabilistic approaches to neural coding*. MIT press.

Feldman, Harriet and Karl Friston (2010). “Attention, uncertainty, and free-energy”. In: *Frontiers in human neuroscience* 4.

Frey, Brendan J, Geoffrey E Hinton, and Peter Dayan (1996). “Does the wake-sleep algorithm produce good density estimators?” In: *Advances in neural information processing systems*, pp. 661–667.

Friston, Karl (2005). “A theory of cortical responses”. In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 360.1456, pp. 815–836.

– (2008). “Hierarchical models in the brain”. In: *PLoS computational biology* 4.11, e1000211.

Friston, Karl (2009). “The free-energy principle: a rough guide to the brain?” In: *Trends in cognitive sciences* 13.7, pp. 293–301.

Friston, Karl, Jean Daunizeau, and Stefan J Kiebel (2009). “Reinforcement learning or active inference?” In: *PloS one* 4.7, e6421.

Friston, Karl, Thomas FitzGerald, et al. (2017). “Active inference: a process theory”. In: *Neural computation* 29.1, pp. 1–49.

Gershman, Samuel J (2019). “What does the free energy principle tell us about the brain?” In: *arXiv preprint arXiv:1901.07945*.

Gershman, Samuel and Noah Goodman (2014). “Amortized inference in probabilistic reasoning”. In: *Proceedings of the Cognitive Science Society*. Vol. 36. 36.

Gillon, Colleen J et al. (2021). “Learning from unexpected events in the neocortical microcircuit”. In: *bioRxiv*.

Gregor, Karol et al. (2014). “Deep autoregressive networks”. In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1242–1250.

Hinton, Geoffrey E and Drew Van Camp (1993). “Keeping the neural networks simple by minimizing the description length of the weights”. In: *Proceedings of the sixth annual conference on Computational learning theory*. ACM, pp. 5–13.

Hopfield, John J (1982). “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the national academy of sciences* 79.8, pp. 2554–2558.

Kalman, Rudolph Emil (1960). “A new approach to linear filtering and prediction problems”. In: *Journal of Basic Engineering* 82.1, pp. 35–45.

Kanai, Ryota et al. (2015). “Cerebral hierarchies: predictive processing, precision and the pulvinar”. In: *Phil. Trans. R. Soc. B* 370.1668, p. 20140169.

Keller, Georg B and Thomas D Mrsic-Flogel (2018). “Predictive processing: a canonical cortical computation”. In: *Neuron* 100.2, pp. 424–435.

Kingma, Durk P, Tim Salimans, et al. (2016). “Improved variational inference with inverse autoregressive flow”. In: *Advances in neural information processing systems*, pp. 4743–4751.

Kingma, Durk P and Max Welling (2014). “Stochastic gradient VB and the variational auto-encoder”. In: *Proceedings of the International Conference on Learning Representations*.

Krizhevsky, Alex and Geoffrey E Hinton (2009). “Learning multiple layers of features from tiny images”. In:

Laughlin, Simon (1981). “A simple coding procedure enhances a neuron’s information capacity”. In: *Zeitschrift für Naturforschung c* 36.9-10, pp. 910–912.

LeCun, Yann, Yoshua Bengio, and Geoffrey E Hinton (2015). “Deep learning”. In: *nature* 521.7553, pp. 436–444.

Levine, Sergey (2018). “Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review”. In: *arXiv preprint arXiv:1805.00909*.

Lotter, William, Gabriel Kreiman, and David Cox (2017). “Deep predictive coding networks for video prediction and unsupervised learning”. In: *International Conference on Learning Representations*.

– (2018). “A neural network trained to predict future video frames mimics critical properties of biological neuronal responses and perception”. In: *arXiv preprint arXiv:1805.10734*.

MacKay, D M (1956). “The epistemological problem for automata”. In: *Automata studies*, pp. 235–252.

McCulloch, Warren S and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Millidge, Beren et al. (2020). “On the Relationship Between Active Inference and Control as Inference”. In: *arXiv preprint arXiv:2006.12964*.

Mnih, Andriy and Karol Gregor (2014). “Neural Variational Inference and Learning in Belief Networks”. In: *International Conference on Machine Learning*, pp. 1791–1799.

Mumford, David (1991). “On the computational architecture of the neocortex”. In: *Biological cybernetics* 65.2, pp. 135–145.

– (1992). “On the computational architecture of the neocortex”. In: *Biological cybernetics* 66.3, pp. 241–251.

Neal, Radford M and Geoffrey E Hinton (1998). “A view of the EM algorithm that justifies incremental, sparse, and other variants”. In: *Learning in graphical models*. Springer, pp. 355–368.

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). “Pixel Recurrent Neural Networks”. In: *International Conference on Machine Learning*, pp. 1747–1756.

Parra, Lucas, Gustavo Deco, and Stefan Miesbach (1995). “Redundancy reduction with information-preserving nonlinear maps”. In: *Network: Computation in Neural Systems* 6.1, pp. 61–72.

Pearl, Judea (1986). “Fusion, propagation, and structuring in belief networks”. In: *Artificial Intelligence* 29.3, pp. 241–288.

Prieto, Alberto et al. (2016). “Neural networks: An overview of early research, current frameworks and new challenges”. In: *Neurocomputing* 214, pp. 242–268.

Rao, Rajesh PN and Dana H Ballard (1999). “Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects.” In: *Nature neuroscience* 2.1.

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning*, pp. 1530–1538.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models". In: *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Rosenblueth, Arturo, Norbert Wiener, and Julian Bigelow (1943). "Behavior, purpose and teleology". In: *Philosophy of science* 10.1, pp. 18–24.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors." In: *Nature*.

Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview". In: *Neural networks* 61, pp. 85–117.

Shannon, Claude E (1948). "A mathematical theory of communication". In: *The Bell system technical journal* 27.3, pp. 379–423.

Spratling, Michael W (2008). "Reconciling predictive coding and biased competition models of cortical function". In: *Frontiers in computational neuroscience* 2, p. 4.

Srinivasan, Mandyam Veerambudi, Simon Laughlin, and Andreas Dubs (1982). "Predictive coding: a fresh view of inhibition in the retina". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 216.1205, pp. 427–459.

Todorov, Emanuel (2008). "General duality between optimal control and estimation". In: *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, pp. 4286–4292.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "Mujoco: A physics engine for model-based control". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 5026–5033.

Toussaint, Marc and Amos Storkey (2006). "Probabilistic inference for solving discrete and continuous state Markov Decision Processes". In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 945–952.

Uria, Benigno, Iain Murray, and Hugo Larochelle (2014). "A deep and tractable density estimator". In: *International Conference on Machine Learning*, pp. 467–475.

Walsh, Kevin S et al. (2020). "Evaluating the neurophysiological evidence for predictive processing as a model of perception". In: *Annals of the New York Academy of Sciences* 1464.1, p. 242.

Wiener, Norbert (1942). *The Interpolation, Extrapolation and Smoothing of Stationary Time Series. NDRC Report*.

Wiener, Norbert (1948). *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press.

Chapter 2

BACKGROUND

2.1 Introduction

This chapter presents the mathematical concepts underlying the technical contributions developed throughout this thesis. For clarity, the material is presented using the notation for state estimation (Part II), and we delay presenting the concepts more specifically related to action selection until Part III.

2.2 Probabilistic Models

Set-Up: Probability Distributions & Maximum Likelihood

Consider a random variable, $X \in \mathbb{R}^M$, with a corresponding distribution, $p_{\text{data}}(X)$, defining the probability of observing each possible observation, $X = \mathbf{x}$. We will use the shorthand notation $p_{\text{data}}(\mathbf{x})$ to denote the probability $p_{\text{data}}(X = \mathbf{x})$. This distribution is the result of an underlying data generation process, e.g. the emission and scattering of photons. While we do not have direct access to p_{data}, we can sample observations, $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$, yielding an empirical distribution, $\widehat{p}_{\text{data}}(\mathbf{x})$.

Often, we wish to model p_{data}, for instance, to predict or compress observations of X. We refer to this model as $p_\theta(\mathbf{x})$, with parameters θ. A natural approach for estimating the model involves minimizing some divergence measure between $p_{\text{data}}(\mathbf{x})$ and $p_\theta(\mathbf{x})$ with respect to the model parameters, θ. The Kullback Leibler (KL) divergence, denoted D_{KL}, is a common choice of divergence measure:

$$\theta^* \leftarrow \arg\min_\theta D_{\text{KL}}(p_{\text{data}}(\mathbf{x}) || p_\theta(\mathbf{x})), \tag{2.1}$$

$$\theta^* \leftarrow \arg\min_\theta \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})} \left[\log p_{\text{data}}(\mathbf{x}) - \log p_\theta(\mathbf{x})\right], \tag{2.2}$$

$$\theta^* \leftarrow \arg\max_\theta \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})} \left[\log p_\theta(\mathbf{x})\right]. \tag{2.3}$$

Here, $\mathbb{E}$ denotes the expectation operator. We have gone from Eq. 2.1 to Eq. 2.2 using the definition of the KL divergence. Because $p_{\text{data}}(\mathbf{x})$ does not depend on θ, we can omit it from the optimization, expressing the equivalent maximization in Eq. 2.3. This is the standard *maximum log-likelihood* objective, which is found throughout machine learning and probabilistic modeling (Murphy, 2012; Goodfellow, Y. Bengio, and Courville, 2016). In practice, we do not have access to $p_{\text{data}}(\mathbf{x})$ and must instead approximate the objective using data samples, i.e. using $\widehat{p}_{\text{data}}(\mathbf{x})$. With $\mathbf{x}^{(i)}$ as the i^{th}

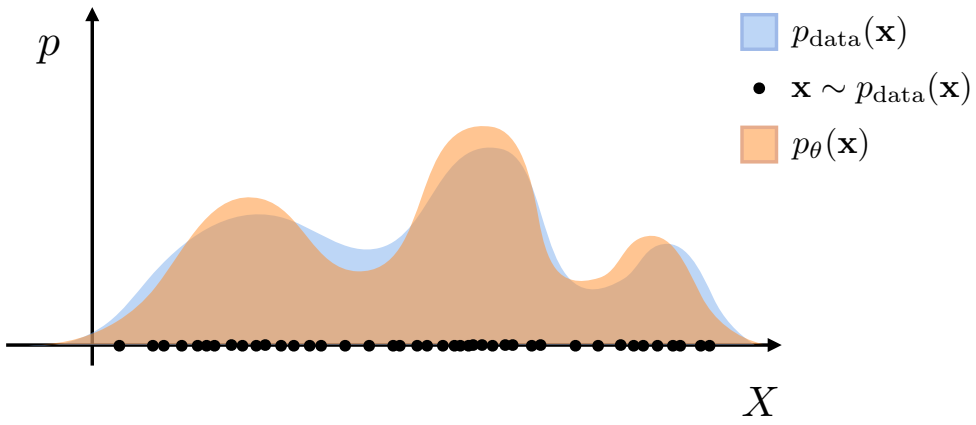


Figure 2.1: **Maximum Likelihood Estimation**. A data distribution, p_{data}, produces samples, $\mathbf{x}$, of a random variable X. Maximum likelihood estimation attempts to fit a model distribution, p_θ, to the empirical distribution of samples by maximizing the log-likelihood of data samples under the model (Eq. 2.3).

observation sample out of N total samples, we can approximate the objective as the following empirical average:

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \approx \mathbb{E}_{\mathbf{x}\sim \widehat{p}_{\text{data}}(\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] = \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)}). \tag{2.4}$$

In summary, we can frame probabilistic modeling as the process of maximizing the log-likelihood, $\log p_\theta(\mathbf{x})$, of data observations under our model. A diagram of this setup is shown in Figure 2.1.

Model Formulation

Formulating a probabilistic model involves considering the dependency structure of the model and the parameterization of these dependencies. As with all models, there is a bias-variance decomposition of modeling error. Without considering a sufficiently flexible dependency structure and parameterization, the model will be inherently limited, i.e. biased, in its capacity to accurately model the data. Yet, with an overly expressive model, containing many dependencies and parameters, there is a risk of overfitting to the training examples, the outcome of a model with high variance.

Dependency Structure

The dependency structure of a probabilistic model is the set of conditional dependencies between variables. One common form of dependency structure is that of *autoregression* (Frey, G. E. Hinton, and Dayan, 1996; Y. Bengio and S. Bengio, 2000), which utilizes the chain rule of probability to model dependencies between variables:

$$p_\theta(\mathbf{x}) = \prod_{j=1}^{M} p_\theta(x_j|x_{<j}). \tag{2.5}$$

Here, we have induced an arbitrary ordering over the M dimensions of $\mathbf{x}$, allowing us to factor the joint distribution over dimensions, $p_\theta(\mathbf{x})$, into a product of M conditional distributions, each conditioned on the previous dimensions, $x_{<j}$. Slightly abusing notation, at $j = 1$, we have $p_\theta(x_j|x_{<j}) = p_\theta(x_1)$. A natural use-case for this dependency structure arises in modeling sequential data, where time provides an ordering over a sequence of T observed variables, $\mathbf{x}_{1:T}$:

$$p_\theta(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}). \tag{2.6}$$

Although such models are conventionally formulated in forward temporal order, this is truly a modeling assumption. Likewise, while the chain rule of probability dictates that we must consider *all* previous variables, there may be cases where it is safe to assume conditional independence outside of some window. In the extreme case, in which we only consider pairwise dependencies, we arrive at a Markov chain: $p_\theta(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$.

Autoregressive models are also referred to as "fully-visible" models (Frey, G. E. Hinton, and Dayan, 1996), as dependencies are only explicitly modeled between observed variables. However, we can also model such dependencies by introducing *latent variables*, denoted as Z. Formally, a latent variable model is defined by the joint distribution

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}), \tag{2.7}$$

where $p_\theta(\mathbf{x}|\mathbf{z})$ is the *conditional likelihood* and $p_\theta(\mathbf{z})$ is the *prior*. Again, we have used the shorthand notation $p_\theta(\mathbf{x}, \mathbf{z})$ to denote $p_\theta(X = \mathbf{x}, Z = \mathbf{z})$. Introducing latent variables is one of, if not, *the* primary technique for increasing the flexibility of a probabilistic model. This is because evaluating the probability of an observation now requires marginalizing over the latent variables. If Z is a continuous variable, this involves integration, $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})d\mathbf{z}$, and if Z is discrete, this involves

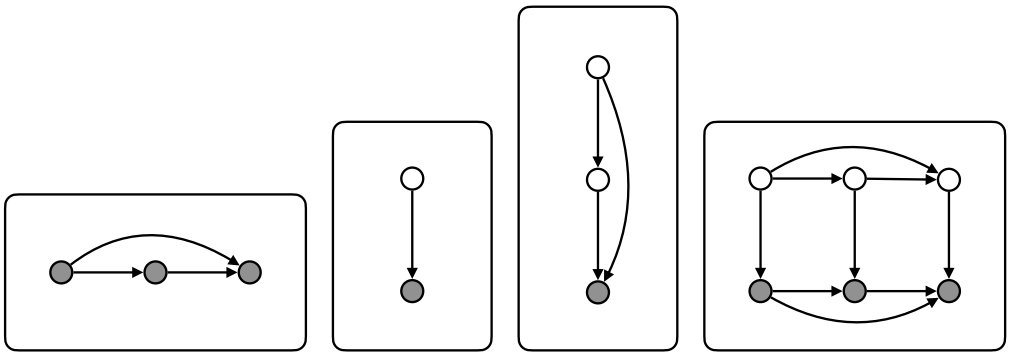


Figure 2.2: **Dependency Structures**. Circles denote random variables, with gray denoting observed variables and white denoting latent variables. Arrows denote probabilistic conditional dependencies. From left to right: autoregressive model (Eqs. 2.5 & 2.6), latent variable model (Eq. 2.7), hierarchical latent variable model (Eq. 2.10), autoregressive or sequential latent variable model (Eq. 2.11). For clarity, we have drawn a subset of the possible dependencies in the final model.

summation, $p_\theta(\mathbf{x}) = \sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z})$. In either case, we have

$$p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim p_\theta(\mathbf{z})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \right], \tag{2.8}$$

which illustrates that $p_\theta(\mathbf{x})$ is a *mixture* distribution, with each mixture component, $p_\theta(\mathbf{x}|\mathbf{z})$, weighted according to $p_\theta(\mathbf{z})$. Thus, even when restricting $p_\theta(\mathbf{x}|\mathbf{z})$ to simple distribution forms, such as Gaussian distributions, $p_\theta(\mathbf{x})$ can take on flexible forms that do not have closed form analytical expressions. In this way, Z can implicitly model dependencies in X, assigning higher probability to particular regions of the observation space.

However, increasing flexibility through latent variables comes with increasing computational overhead. In general, marginalizing over Z is not analytically tractable, particularly with continuous latent variables or complex conditional likelihoods. This requires us to either 1) adopt approximation techniques, which we discuss in Section 2.3, or 2) restrict the form of the model to ensure computationally tractable evaluation of $p_\theta(\mathbf{x})$. This latter approach is the basis of *flow-based* models (Tabak and Turner, 2013; Rippel and Adams, 2013; Dinh, Krueger, and Y. Bengio, 2015), which define the conditional dependency between X and Z in terms of an invertible transform, $\mathbf{x} = f_\theta(\mathbf{z})$ and $\mathbf{z} = f_\theta^{-1}(\mathbf{x})$. We can then express $p_\theta(\mathbf{x})$ using the *change of variables formula*:

$$p_\theta(\mathbf{x}) = p_\theta(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) \right|^{-1}, \tag{2.9}$$

where $\frac{\partial \mathbf{x}}{\partial \mathbf{z}}$ is the Jacobian of the transform and $\det(\cdot)$ denotes matrix determinant. The term $\left|\det\left(\frac{\partial \mathbf{x}}{\partial \mathbf{z}}\right)\right|^{-1}$ can be interpreted as the local scaling of space when moving from Z to X, conserving probability mass in the transform. Flow-based models, also referred to as *normalizing flows* (Rezende and Mohamed, 2015), are the basis of the classical technique of independent components analysis (ICA) (Bell and Sejnowski, 1997; Hyvärinen and Oja, 2000) and non-linear generalizations (S. S. Chen and Gopinath, 2001; Laparra, Camps-Valls, and Malo, 2011). As such, these models can serve as a general-purpose mechanism for adding and removing statistical dependencies between variables. Although flow-based models avoid the intractability of marginalization, their requirement of invertibility may be overly restrictive or undesirable in some contexts (Cornish et al., 2020). And while the change of variables formula can also be applied to non-invertible transforms (Cvitkovic and Koliander, 2019), it raises computational intractabilities. This motivates the use of approximate techniques for training latent variables models (Section 2.3).

While we have presented autoregression and latent variables separately, these techniques can, in fact, be combined in numerous ways to model dependencies. For instance, one can create *hierarchical latent variable models* (Dayan et al., 1995), incorporating autoregressive dependencies between latent variables. Considering L levels of latent variables, $Z^{1:L} = \left[Z^1, \ldots, Z^L\right]$, we can express the joint distribution as

$$p_\theta(\mathbf{x}, \mathbf{z}^{1:L}) = p_\theta(\mathbf{x}|\mathbf{z}^{1:L}) \prod_{\ell=1}^{L} p_\theta(\mathbf{z}^\ell|\mathbf{z}^{\ell+1:L}). \tag{2.10}$$

From this perspective, hierarchical latent variable models are a repeated application of the latent variables technique in order to create increasingly complex *empirical priors* (Efron and Morris, 1973). We can also consider incorporating latent variables within sequential (autoregressive) probabilistic models, giving rise to *sequential latent variable models*. Considering a single level of latent variables in a corresponding sequence, $Z_{1:T}$, we generally have the following joint distribution:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}), \tag{2.11}$$

where we have again assumed a forward sequential ordering. By restricting the dependency structure and distribution forms in sequential latent variable models, we recover familiar special cases, such as hidden Markov models or linear Gaussian state-space models (Murphy, 2012). Beyond hierarchical and sequential latent variable models, there are a variety of other ways to combine autoregression and

latent variables (Gulrajani et al., 2017; Razavi, Aaron van den Oord, and Vinyals, 2019). The remaining chapters focus on hierarchical (Eq. 2.10) and sequential (Eq. 2.11) latent variable models, though models with more flexible hierarchical, sequential, and spatial dependencies will be required to advance the frontier of probabilistic modeling.

Parameterizing the Model

In the previous section, we discussed the dependency structure of probabilistic models. The probability distributions that define these dependencies are ultimately functions. In this section, we discuss possible forms that these functions may take. We restrict our focus here to *parametric* distributions, which are defined by one or more distribution parameters. The canonical example is the Gaussian (or Normal) distribution, $\mathcal{N}(x; \mu, \sigma^2)$, which is defined by a mean, μ, and variance, σ^2. This can be extended to the multivariate setting, where $\mathbf{x} \in \mathbb{R}^M$ is modeled with a mean vector, $\boldsymbol{\mu}$, and covariance matrix, $\boldsymbol{\Sigma}$, with the probability density written as

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{M/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\left(\frac{-1}{2}(\mathbf{x} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right). \tag{2.12}$$

For convenience, we may also consider diagonal covariance matrices, $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2)$, simplifying the parameterization and resulting calculations. In particular, the special case where $\boldsymbol{\Sigma} = \mathbf{I}_M$, the $M \times M$ identity matrix, the log-density, up to a constant, becomes the familiar *mean squared error*,

$$\log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{I}) = -\frac{1}{2}||\mathbf{x} - \boldsymbol{\mu}||_2^2 + \text{const.} \tag{2.13}$$

With a parametric distribution, conditional dependencies are mediated by the distribution parameters, which are functions of the conditioning variables. For example, we can express an autoregressive Gaussian distribution (of the form in Eq. 2.5) through conditional densities, $p_\theta(x_j|x_{<j}) = \mathcal{N}(x_j; \mu_\theta(x_{<j}), \sigma_\theta^2(x_{<j}))$, where μ_θ and σ_θ^2 are functions taking $x_{<j}$ as input. A similar form applies to autoregressive models on sequences of vector inputs (Eq. 2.6), with $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}), \boldsymbol{\Sigma}_\theta(\mathbf{x}_{<t}))$. Likewise, in a latent variable model (Eq. 2.7), we can express a Gaussian conditional likelihood as $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{z}), \boldsymbol{\Sigma}_\theta(\mathbf{z}))$. Note that in the above examples, we have overloaded notation, simply using a subscript θ for all functions. In practice, while it is not uncommon to share parameters across functions, this is ultimately a modeling choice.

The functions supplying each of the distribution parameters can range in complexity, from constant to highly non-linear. Classical modeling techniques often employ linear functions. For instance, in a latent variable model, we could parameterize the mean as a linear function of $\mathbf{z}$:

$$\boldsymbol{\mu}_\theta(\mathbf{z}) = \mathbf{W}\mathbf{z} + \mathbf{b}, \tag{2.14}$$

where $\mathbf{W}$ is a matrix of weights and $\mathbf{b}$ is a bias vector. Models of this form underlie factor analysis, probabilistic principal components analysis (Tipping and Bishop, 1999), independent components analysis (Bell and Sejnowski, 1997; Hyvärinen and Oja, 2000), and sparse coding (Olshausen and Field, 1996). Linear autoregressive models are also the basis of many classical time-series models and are common across fields that use statistical methods. While linear models are relatively computationally efficient, they are often too limited to accurately model complex data distributions, e.g. those found in natural images or audio.

Recent improvements in deep learning (Goodfellow, Y. Bengio, and Courville, 2016) have provided probabilistic models with a more expressive class of non-linear functions, improving their modeling capacity. In these models, the distribution parameters are parameterized with deep networks, which are then trained by backpropagating (Rumelhart, G. E. Hinton, and Williams, 1986) the gradient of the log-likelihood objective, $\nabla_\theta \mathbb{E}_{\mathbf{x}\sim\widehat{p}_{\text{data}}}[\log p_\theta(\mathbf{x})]$, back through the layers of the network. Typically, this is estimated using a mini-batch of data examples, rather than the full empirical distribution as in Eq. 2.4. Deep autoregressive models and deep latent variable models have enabled recent advances across an array of areas, including speech (Graves, 2013; Aäron van den Oord et al., 2016), natural language (Sutskever, Vinyals, and Le, 2014; Radford et al., 2019), images (Razavi, Aaron van den Oord, and Vinyals, 2019), video (Kumar et al., 2020), reinforcement learning (Chua et al., 2018; Ha and Schmidhuber, 2018) and many others.

We visualize a simplified probabilistic computation graph for a deep autoregressive model in Figure 2.3. This diagram translates the autoregressive dependency structure from Figure 2.2, which only depicts the variables and their dependencies in the model, into a more detailed visualization, breaking the variables into their corresponding distributions and terms in the log-likelihood objective. Here, green circles denote the conditional likelihood at each step, containing a Gaussian mean and standard deviation, which are parameterized by a deep network. The log-likelihood, $\log p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})$, evaluated at the data observation, $\mathbf{x}_t \sim p_{\text{data}}(\mathbf{x}_t|\mathbf{x}_{<t})$ (gray circle), provides the

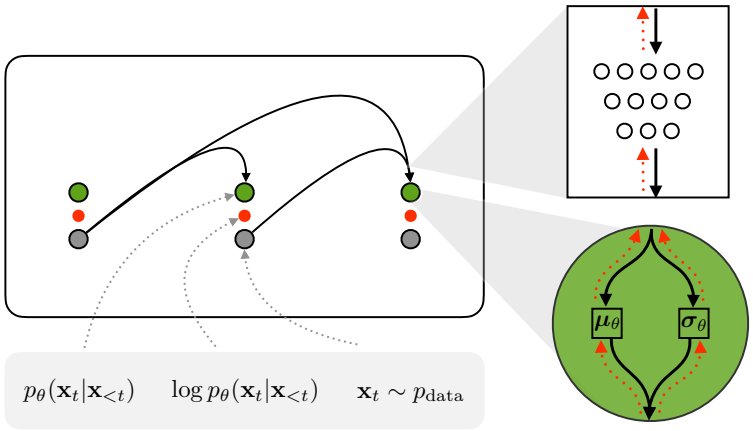


Figure 2.3: **Model Parameterization & Computation Graph**. The diagram depicts a simplified computation graph for a deep autoregressive Gaussian model. Green circles denote the conditional likelihood distribution at each step, while gray circles again denote the (distribution of) data observations. Smaller red circles denote each of the log-likelihood terms in the objective. Gradients w.r.t. these terms are backpropagated through the networks parameterizing the model's distribution parameters (red dotted lines).

objective (red dot). The gradient of this objective w.r.t. the network parameters is calculated through backpropagation (red dotted line). This general depiction of dependencies, distributions, and log-probabilities (or differences of log-probabilities) is used throughout this thesis to describe the various computational setups.

Purely autoregressive models (without latent variables) have proven useful in many domains, often obtaining better log-likelihoods as compared with latent variable models. However, there are a number of reasons to prefer latent variable models in some contexts. First, autoregressive sampling is inherently sequential, and this linear computational scaling becomes costly in high-dimensional domains. Second, latent variables provide a representational space for downstream tasks, compression, and overall data analysis. Finally, latent variables provide added flexibility, which is particularly useful for modeling continuous random variables with relatively simple, e.g. Gaussian, conditional distributions. For these reasons, we require methods for handling the latent marginalization in Eq. 2.8. Variational inference is one such method.

2.3 Variational Inference

Derivation

As we saw in the previous section, training latent variable models through maximum likelihood requires evaluating $\log p_\theta(\mathbf{x})$. However, particularly with continuous latent variables, evaluating $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$ is generally computationally intractable. This problem is only exacerbated in deep latent variable models, where computing $p_\theta(\mathbf{z})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ is more expensive. Thus, we require some technique for tractably estimating $\log p_\theta(\mathbf{x})$ without resorting to exact marginalization.

Variational inference (G. E. Hinton and Van Camp, 1993; Jordan et al., 1998) approaches this problem by introducing an *approximate posterior* distribution, $q(\mathbf{z}|\mathbf{x})$, which provides a tractable lower bound, $\mathcal{L}(\mathbf{x}; q, \theta) \leq \log p_\theta(\mathbf{x})$, on the log-likelihood. This lower bound is variously referred to as the *evidence lower bound (ELBO)*, variational lower bound, and the negative free energy. By tightening and maximizing the ELBO w.r.t. the model parameters, θ, we can approximate maximum likelihood training while avoiding marginalization.

We can interpret variational inference as converting probabilistic inference into an optimization problem. Given a family of distributions, $\mathcal{Q}$, e.g., Gaussian, non-parametric, etc., variational inference attempts to find the distribution, $q \in \mathcal{Q}$, that minimizes $D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$:

$$q(\mathbf{z}|\mathbf{x}) \leftarrow \arg\min_q D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})), \tag{2.15}$$

where $p_\theta(\mathbf{z}|\mathbf{x})$ is the *posterior* distribution,

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})}. \tag{2.16}$$

Because $p_\theta(\mathbf{z}|\mathbf{x})$ includes the intractable $p_\theta(\mathbf{x})$, we cannot minimize the KL divergence in Eq. 2.15 directly. Instead, we can rewrite this as

$$\begin{aligned}
D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{z}|\mathbf{x})\right] && (2.17)\\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x}) - \log\left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})}\right)\right] && (2.18)\\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})\right] + \log p_\theta(\mathbf{x}) && (2.19)\\
&= -\mathcal{L}(\mathbf{x}; q, \theta) + \log p_\theta(\mathbf{x}). && (2.20)
\end{aligned}$$

In Eq. 2.20, we have defined $\mathcal{L}(\mathbf{x}; q, \theta)$, as

$$\begin{aligned}
\mathcal{L}(\mathbf{x}; q, \theta) &\equiv \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] && (2.21)\\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})). && (2.22)
\end{aligned}$$

Algorithm 1 Variational Expectation Maximization (EM)

Input: model $p_\theta(\mathbf{x}, \mathbf{z})$, data examples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)} \sim p_{\text{data}}(\mathbf{x})$
while θ not converged **do**
  for $\mathbf{x} = \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$ **do**
    $q(\mathbf{z}|\mathbf{x}) \leftarrow \arg\max_q \mathcal{L}(\mathbf{x}; q, \theta)$ ▷ inference (E-step)
  end for
  $\theta \leftarrow \arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mathbf{x}^{(i)}; q, \theta)$ ▷ learning (M-step)
end while

Rearranging terms in Eq. 2.20, we have

$$\log p_\theta(\mathbf{x}) = \mathcal{L}(\mathbf{x}; q, \theta) + D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})). \tag{2.23}$$

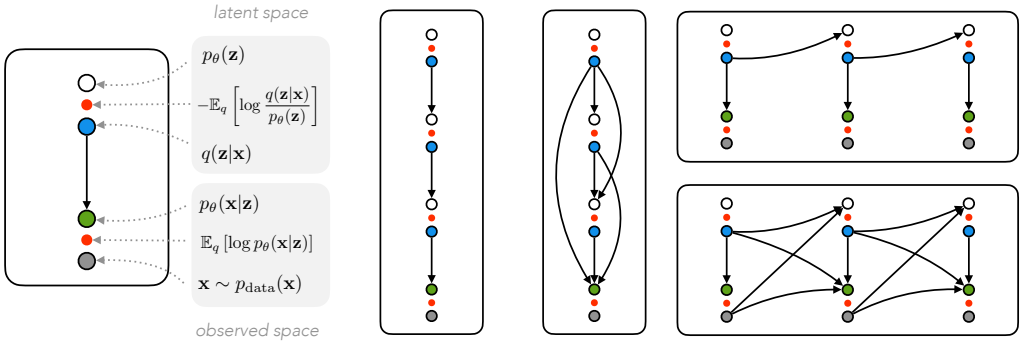
Because KL divergence is non-negative, we see that $\mathcal{L}(\mathbf{x}; q, \theta) \leq \log p_\theta(\mathbf{x})$, with equality when $q(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$. As the LHS of Eq. 2.23 does not depend on $q(\mathbf{z}|\mathbf{x})$, maximizing $\mathcal{L}(\mathbf{x}; q, \theta)$ w.r.t. q implicitly minimizes $D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ w.r.t. q. Together, these statements imply that maximizing $\mathcal{L}(\mathbf{x}; q, \theta)$ w.r.t. q tightens the lower bound on $\log p_\theta(\mathbf{x})$. With this tightened lower bound, we can then maximize $\mathcal{L}(\mathbf{x}; q, \theta)$ w.r.t. θ. This alternating optimization process is referred to as the variational expectation maximization (EM) algorithm (Dempster, Laird, and Rubin, 1977; Neal and G. E. Hinton, 1998) (Algorithm 1), consisting of approximate inference (E-step) and learning (M-step). While Algorithm 1 iterates over the entire data set between parameter updates, one can instead perform inference for a mini-batch of data examples and perform stochastic gradient ascent on θ (Hoffman et al., 2013). This version of the algorithm, referred to as *stochastic variational inference (SVI)*, is used in practice to train deep latent variable models.

Interpreting the ELBO

The primary motivation for variational inference is in providing a tractable lower bound for model training. In the process, however, we are left with an approximate posterior distribution, $q(\mathbf{z}|\mathbf{x})$. To gain insight into this distribution and the approximate inference procedure, we can rewrite the ELBO as

$$\mathcal{L}(\mathbf{x}; q, \theta) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right]. \tag{2.24}$$

Each of the terms in the objective guides the optimization of $q(\mathbf{z}|\mathbf{x})$. The first term quantifies the agreement with the data, i.e., reconstruction: sampling $\mathbf{z}$ from q, we want to assign high log-probability to the observation, $\mathbf{x}$. The second term quantifies the agreement with the prior prediction: sampling $\mathbf{z}$ from q, we want to have high



(a) Computation Graph (b) Hierarchical (c) Sequential

Figure 2.4: **ELBO Computation Graphs**. **(a)** Basic computation graph for variational inference. Outlined circles denote distributions. Smaller red circles denote terms in the ELBO objective. Arrows, again, denote conditional dependencies. This notation can be used to express **(b)** hierarchical and **(c)** sequential models with various model dependencies.

log-probability under the prior. The final term is the *entropy* of $q(\mathbf{z}|\mathbf{x})$, quantifying the spread or uncertainty of the distribution. Maximizing this term encourages $q(\mathbf{z}|\mathbf{x})$ to remain maximally uncertain, all else being equal (Jaynes, 1957). Often, the final two terms are combined, as in Eq. 2.22, with $D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$ interpreted as a form of "regularization," penalizing deviations of $q(\mathbf{z}|\mathbf{x})$ from $p_\theta(\mathbf{z})$.

This KL divergence term has been the focus of recent investigations. In particular, one can interpret the ELBO as a Lagrangian, with Lagrange multiplier $\beta = 1$:

$$\mathcal{L}(\mathbf{x}; q, \theta) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \beta D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})). \tag{2.25}$$

Thus, the ELBO can be seen as expressing a constrained optimization problem, with the regularizing constraint on $D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$ mediated by β. In models with flexible, e.g., autoregressive, conditional likelihoods, the model may not utilize $\mathbf{z}$, in which case the KL term results in a local maximum at $q(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z})$. This is the so-called "dying units" problem (Bowman et al., 2016), in which the latent variables are effectively unused. A common approach is to anneal β from $0 \to 1$ during the course of training (Bowman et al., 2016; Sønderby et al., 2016), though alternative schemes exist (Kingma, Salimans, et al., 2016; X. Chen et al., 2017). We can also dynamically adjust β to maintain a particular KL or conditional likelihood constraint (Rezende and Viola, 2018). And by setting $\beta > 1$ we can over-regularize $q(\mathbf{z}|\mathbf{x})$, and with an independent prior, this can yield more disentangled latent variables (Higgins et al., 2017; Burgess et al., 2018). In these latter cases, where $\beta \neq 1$, $\mathcal{L}(\mathbf{x}; q, \theta)$ is no longer a valid lower bound on $\log p_\theta(\mathbf{x})$. Alemi et al., 2018 provide

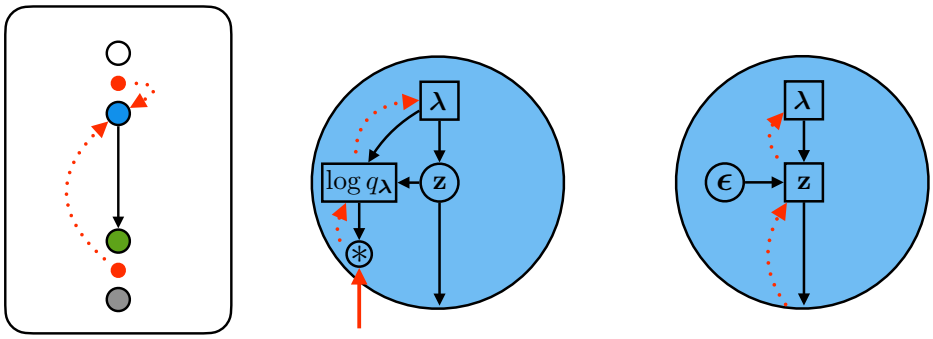
an information-theoretic framework to make sense of these ideas, identifying the connection between the two terms in the ELBO (Eq. 2.25) and the concepts of *distortion* and *rate* respectively. In short, adjusting β allows one to target particular rate-distortion trade-offs, which, in turn, result in bounds on the mutual information between X and Z. This perspective allows one to consider latent variable models that achieve varying compression rates in complex, natural data domains, such as images (Ballé, Laparra, and Simoncelli, 2017) and video (Lombardo et al., 2019).

As with autoregressive models (Figure 2.3), we can represent latent variable models and the ELBO objective as a computation graph, again breaking the dependency structure graphs from Figure 2.2 into separate distributions and terms in the objective. In Figure 2.4, we illustrate examples of these graphs. Each variable contains a red circle, denoting a term in the ELBO objective. In comparison with the fully-observed autoregressive model, we now have an additional objective term for the latent variable, corresponding to the KL divergence. This graphical representation also allows us to visualize the variational objective for more complex hierarchical and sequential models (Figures 2.4b & 2.4c).

Inference Optimization

In deriving variational inference and variational EM, we described inference optimization, the E-step, as the process of maximizing $\mathcal{L}(\mathbf{x}; q, \theta)$ w.r.t. the distribution $q(\mathbf{z}|\mathbf{x})$. The fact that probability distributions are functions makes $\mathcal{L}(\mathbf{x}; q, \theta)$ a *functional* and makes inference optimization a variational calculus problem. One could solve this optimization problem using a variety of techniques, ranging from gradient-free, e.g., evolution strategies, to first-order gradient-based, e.g., black-box variational inference (Ranganath, Gerrish, and Blei, 2014), to second-order gradient-based, e.g., the Laplace approximation (Friston et al., 2007; Park, C. Kim, and G. Kim, 2019). Despite this range, first-order gradient-based techniques are most common in practice; when combined with amortization (see below), they can result in exceedingly efficient optimization. For this reason, we exclusively focus on first-order techniques in the following chapters, which we now describe in more detail.

With a parametric $q(\mathbf{z}|\mathbf{x})$, first-order inference optimization entails calculating the gradients of $\mathcal{L}(\mathbf{x}; q, \theta)$ w.r.t. the distribution parameters of q, which we refer to as $\boldsymbol{\lambda}$. For instance, if $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \text{diag}(\boldsymbol{\sigma}_q^2))$, i.e., $\boldsymbol{\lambda} = \left[\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q\right]$, then we must calculate $\nabla_{\boldsymbol{\mu}_q}\mathcal{L}$ and $\nabla_{\boldsymbol{\sigma}_q}\mathcal{L}$. We can consider the ELBO as an example of a stochastic



(a) Inference (b) Score Function (c) Pathwise Derivative

Figure 2.5: **Stochastic Gradient Estimation**. **(a)** Variational inference with parametric approximate posteriors requires optimizing $\mathcal{L}$ w.r.t. the distribution parameters, $\boldsymbol{\lambda}$. This often requires estimating stochastic gradients (red dotted lines). **(b)** The score function estimator (Eq. 2.26) is applicable to any distribution, but suffers from high variance. Note: the solid red arrow denotes the per-sample objective, $l(\mathbf{x}, \mathbf{z}; \theta)$. **(c)** The pathwise derivative estimator (Eq. 2.27), in contrast, has lower variance, but is less widely applicable.

computation graph (Schulman et al., 2015), containing the stochastic sampling operation $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$. Unfortunately, stochastic sampling is non-differentiable, preventing us from simply differentiating through the chain $\boldsymbol{\lambda} \rightarrow \mathbf{z} \rightarrow \mathcal{L}$. Schulman et al., 2015 highlight two stochastic gradient estimators to tackle this issue: the *score function* estimator (Glynn, 1990; Williams, 1992; Fu, 2006) and the *pathwise derivative* estimator (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014; Titsias and Lázaro-Gredilla, 2014), each of which allows us to calculate stochastic estimates of $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$.

The score function estimator, sometimes referred to as the REINFORCE estimator (Williams, 1992), is generally applicable to any parametric distribution. With $q_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{x})$ denoting the dependence of q on $\boldsymbol{\lambda}$, as well as $\mathcal{L}(\mathbf{x}; q, \theta) = \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{x})} \left[l(\mathbf{x}, \mathbf{z}; \theta) \right]$, the score function estimator allows us to express $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$ as

$$\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{x})} \left[l(\mathbf{x}, \mathbf{z}; \theta) \right] = \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{x}) l(\mathbf{x}, \mathbf{z}; \theta) \right] . \tag{2.26}$$

While the score function estimator is unbiased and can be successfully utilized for variational inference (Gregor et al., 2014; Ranganath, Gerrish, and Blei, 2014; Mnih and Gregor, 2014; Mnih and Rezende, 2016), in practice, it requires considerable tuning and computation, owing to its empirically high variance. Alternatively, the pathwise derivative estimator, sometimes referred to as the reparameterization estimator (Kingma and Welling, 2014), when applicable, allows us to obtain unbiased

gradient estimates with considerably lower variance. This is accomplished by reparameterizing $\mathbf{z}$ in terms of an auxiliary random variable, enabling differentiation through stochastic sampling. The most common example is reparameterizing $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \text{diag}(\boldsymbol{\sigma}_q^2))$ as $\mathbf{z} = \boldsymbol{\mu}_q + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_q$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$ and $\odot$ denotes element-wise multiplication. More generally, for some deterministic function g, if we can express $\mathbf{z} = g(\boldsymbol{\lambda}, \boldsymbol{\epsilon})$ with $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, then the pathwise derivative estimator allows us to express $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$ as

$$\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{x})} \left[l(\mathbf{x}, \mathbf{z}; \theta) \right] = \mathbb{E}_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{\lambda}} l(\mathbf{x}, g(\boldsymbol{\lambda}, \boldsymbol{\epsilon}); \theta) \right]. \tag{2.27}$$

With these stochastic gradient estimators in hand, we can perform stochastic gradient-based optimization of $\mathcal{L}(\mathbf{x}; q, \theta)$ w.r.t. $\boldsymbol{\lambda}$ (Ranganath, Gerrish, and Blei, 2014): estimating $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$ by sampling $\mathbf{z}$ (or $\boldsymbol{\epsilon}$) and updating $\boldsymbol{\lambda}$ using stochastic gradient ascent. Unfortunately, a naïve implementation of this inference optimization (E-step) procedure scales poorly to large models and data sets. This is because many gradient steps must be performed *per example*, just to provide one estimate of $\mathcal{L}(\mathbf{x}; q, \theta)$ for M-step optimization. More sophisticated approaches cache previous estimates of $\boldsymbol{\lambda}$ for each example to initialize future inference optimization. However, beyond the additional memory overhead, this approach still requires tuning the inference optimizer's learning rate and is not applicable to online learning settings. In order to apply variational inference more broadly, we require a simple technique for dramatically improving the efficiency of inference optimization. *Amortization* (Gershman and Goodman, 2014), a form of meta-optimization, offers a viable solution.

Amortized Variational Inference

Amortization, in a generic sense, refers to spreading out costs. In amortized inference, these "costs" are the computational costs of performing inference optimization. Thus, rather than separately optimizing $\boldsymbol{\lambda}$ for each data example, we amortize this optimization cost using a learned optimizer, i.e., an *inference model*. By using this meta-optimization procedure, we can perform inference optimization far more efficiently for each example, with a negligible cost for learning the inference model. The concept of inference models is deeply embedded with deep latent variable models, popularized by the Helmholtz Machine (Dayan et al., 1995), which was formulated as an *autoencoder* (Ballard, 1987). Formally, in such setups, the inference model is a direct mapping from $\mathbf{x}$ to $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda} \leftarrow f_{\phi}(\mathbf{x}), \tag{2.28}$$

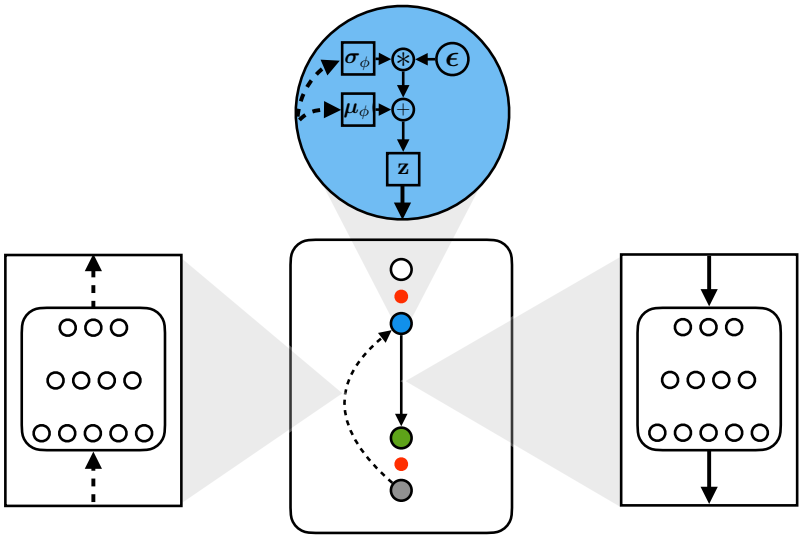


Figure 2.6: **Variational Autoencoder (VAE)**. VAEs combine direct amortization (Eq. 2.28) and the pathwise derivative estimator (Eq. 2.27, top) with Gaussian approximate posteriors to train deep latent variable models. In the model diagram (center), the amortized inference model (left) acts as an *encoder*, with the conditional likelihood (right) acting as a *decoder*. Each are parameterized by deep networks.

where f_ϕ is a model (deep network) with parameters ϕ. Conventionally, we denote the approximate posterior as $q_\phi(\mathbf{z}|\mathbf{x})$ to denote the parameterization by ϕ. Now, rather than optimizing $\boldsymbol{\lambda}$ using gradient-based techniques, we periodically update ϕ using $\nabla_\phi \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} \frac{\partial \boldsymbol{\lambda}}{\partial \phi}$, thereby letting f_ϕ learn to optimize $\boldsymbol{\lambda}$. This procedure is incredibly simple, as we only need to tune the learning rate for ϕ, and efficient, as we have an estimate of $\boldsymbol{\lambda}$ after only one forward pass through f_ϕ. Amortization is also widely applicable: if we can estimate $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$ using stochastic gradient estimation (see above), we can continue differentiating through the chain $\phi \to \boldsymbol{\lambda} \to \mathbf{z} \to \mathcal{L}$.

When direct amortization is combined with the pathwise derivative estimator in deep latent variable models, the resulting setup is referred to as a *variational autoencoder (VAE)* (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014). In this autoencoder interpretation, $q_\phi(\mathbf{z}|\mathbf{x})$ is an *encoder*, $\mathbf{z}$ is the latent *code*, and $p_\theta(\mathbf{x}|\mathbf{z})$ is a *decoder*. A computation graph is shown in Figure 2.6. This direct encoding scheme seems intuitively obvious: in the same way that $p_\theta(\mathbf{x}|\mathbf{z})$ directly maps $\mathbf{z}$ to a distribution over $\mathbf{x}$, $q_\phi(\mathbf{z}|\mathbf{x})$ directly maps $\mathbf{x}$ to a distribution over $\mathbf{z}$. Indeed, with perfect knowledge of $p_\theta(\mathbf{x}, \mathbf{z})$, f_ϕ could act as a lookup table, precisely mapping each $\mathbf{x}$ to the corresponding optimal $\boldsymbol{\lambda}$. However, as we discuss in Chapter 3, there are a number of subtle issues that can arise with direct amortization, forming one of

the primary motivations for the following chapters.

2.4 Discussion

In this chapter, we have reviewed the basic considerations in formulating and training probabilistic models. We started by introducing the maximum log-likelihood (or minimum KL divergence) framework for model training, which involves maximizing $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right]$ w.r.t. the model parameters, θ. The model is formulated in terms of the dependency structure between variables, the distribution families, and the functional form of the distribution parameters. Each of these constitute design choices, which are often selected in light of known structure in the data.

In improving the expressive capacity of the model, we discussed two primary methods for adding dependency structure to the model, autoregression and latent variables, as well as combinations of the two. While we presented these ideas separately, there is, in fact, a link between these two forms of dependency structure. In particular, Kingma, Salimans, et al., 2016 noted that we can express sampling from a Gaussian autoregressive model in both an autoregressive form:

$$x_j \sim \mathcal{N}(x_j; \mu_\theta(x_{<j}), \sigma^2_\theta(x_{<j})), \tag{2.29}$$

as well as in the form of an affine flow-based latent variable model:

$$x_j = \mu_\theta(x_{<j}) + \sigma_\theta(x_{<j}) \cdot z_j, \tag{2.30}$$

where $z_j \sim \mathcal{N}(z_j; 0, 1)$ is an auxiliary latent variable. Note that Eq. 2.30 is the reparameterization trick (from the pathwise derivative estimator) applied to the Gaussian autoregressive sampling in Eq. 2.29. To train an autoregressive model, we can analytically evaluate the log-density directly via Eq. 2.29. Alternatively, we can evaluate the log-density by performing "inference," inverting the affine transform:

$$z_j = \frac{x_j - \mu_\theta(x_{<j})}{\sigma_\theta(x_{<j})}, \tag{2.31}$$

and then evaluating the log-density of z_j under $p_\theta(z) = \mathcal{N}(0, 1)$ and the log-scaling factor of the transform, $\log \sigma_\theta(x_{<j})$. With this scheme, we exactly recover the Gaussian density definition in Eq. 2.12. This case of equivalence between particular classes of autoregressive models and (flow-based) latent variable models suggests that, rather than considering various model classes as entirely distinct, we can consider the general computational principles of probability, with various models making trade-offs in terms of tractability, expressive capacity, computational efficiency, etc.

This chapter has not exhaustively covered the breadth of probabilistic modeling techniques that exist in the literature, instead focusing on the subset of ideas relevant to this thesis. It is worth mentioning that a variety of techniques have been recently developed or have gained renewed interest. For instance, implicit latent variable models (Mohamed and Lakshminarayanan, 2016), such as generative adversarial networks (Goodfellow, Pouget-Abadie, et al., 2014) and generative stochastic networks (Y. Bengio, Laufer, et al., 2014), are formulated in terms of a sampling procedure, avoiding the need to explicitly define $p_\theta(\mathbf{x})$. Energy-based models (LeCun et al., 2006; Salakhutdinov and G. Hinton, 2009; Du and Mordatch, 2019) similarly offer a more flexible form, parameterizing an energy function over the space of $\mathbf{x}$, which is then converted into a Boltzmann distribution. Score function models (Hyvärinen and Dayan, 2005; Vincent, 2011; Song and Ermon, 2019) avoid directly parameterizing the energy function, instead estimating a gradient field to sample from the data distribution. Finally, a variety of contrastive learning procedures have been recently developed (Gutmann and Hyvärinen, 2010; Mnih and Kavukcuoglu, 2013; A. v. d. Oord, Li, and Vinyals, 2018), providing an alternative, "self-supervised" technique for modeling data while avoiding explicit data density estimation.

These ideas may seem hopelessly diverse, however, they are all tied to concepts in probabilistic modeling (and by extension, information theory), as well as learning, inference, and optimization more generally. Indeed, as with the example of flow-based latent variable models and autoregressive models discussed above, many modeling approaches can be interpreted as related or special cases of each other, or even combined in new and interesting ways (Rezende and Mohamed, 2015; Makhzani et al., 2015; Agrawal and Dukkipati, 2016). Thus, the advances in probabilistic modeling and inference presented in this thesis should not be viewed in isolation, as their applicability may cross the blurred boundaries between different approaches.

References

Agrawal, Siddharth and Ambedkar Dukkipati (2016). "Deep Variational Inference Without Pixel-Wise Reconstruction". In: *arXiv preprint arXiv:1611.05209*.

Alemi, Alexander et al. (2018). "Fixing a Broken ELBO". In: *International Conference on Machine Learning*, pp. 159–168.

Ballard, Dana H (1987). "Modular Learning in Neural Networks." In: *AAAI*, pp. 279–284.

Ballé, Johannes, Valero Laparra, and Eero P Simoncelli (2017). "End-to-end optimized image compression". In: *International Conference on Learning Representations*.

Bell, Anthony J and Terrence J Sejnowski (1997). "The "independent components" of natural scenes are edge filters". In: *Vision research* 37.23, pp. 3327–3338.

Bengio, Yoshua and Samy Bengio (2000). "Modeling high-dimensional discrete data with multi-layer neural networks". In: *Advances in Neural Information Processing Systems*, pp. 400–406.

Bengio, Yoshua, Eric Laufer, et al. (2014). "Deep generative stochastic networks trainable by backprop". In: *International Conference on Machine Learning*. PMLR, pp. 226–234.

Bowman, Samuel R et al. (2016). "Generating Sentences from a Continuous Space". In: *CoNLL 2016*, p. 10.

Burgess, Christopher P et al. (2018). "Understanding disentangling in β-VAE". In: *arXiv preprint arXiv:1804.03599*.

Chen, Scott Saobing and Ramesh A Gopinath (2001). "Gaussianization". In: *Advances in neural information processing systems*, pp. 423–429.

Chen, Xi et al. (2017). "Variational Lossy Autoencoder". In: *International Conference on Learning Representations*.

Chua, Kurtland et al. (2018). "Deep reinforcement learning in a handful of trials using probabilistic dynamics models". In: *Advances in Neural Information Processing Systems*, pp. 4754–4765.

Cornish, Rob et al. (2020). "Relaxing bijectivity constraints with continuously indexed normalising flows". In: *International Conference on Machine Learning*.

Cvitkovic, Milan and Günther Koliander (2019). "Minimal Achievable Sufficient Statistic Learning". In: *International Conference on Machine Learning*, pp. 1465–1474.

Dayan, Peter et al. (1995). "The helmholtz machine". In: *Neural computation* 7.5, pp. 889–904.

Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38.

Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). "Nice: Non-linear independent components estimation". In: *International Conference on Learning Representations*.

Du, Yilun and Igor Mordatch (2019). "Implicit Generation and Modeling with Energy Based Models". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Efron, Bradley and Carl Morris (1973). "Stein's estimation rule and its competitors—an empirical Bayes approach". In: *Journal of the American Statistical Association* 68.341, pp. 117–130.

Frey, Brendan J, Geoffrey E Hinton, and Peter Dayan (1996). "Does the wake-sleep algorithm produce good density estimators?" In: *Advances in neural information processing systems*, pp. 661–667.

Friston, Karl et al. (2007). "Variational free energy and the Laplace approximation". In: *Neuroimage* 34.1, pp. 220–234.

Fu, Michael C (2006). "Gradient estimation". In: *Handbooks in operations research and management science* 13, pp. 575–616.

Gershman, Samuel and Noah Goodman (2014). "Amortized inference in probabilistic reasoning". In: *Proceedings of the Cognitive Science Society*. Vol. 36. 36.

Glynn, Peter W (1990). "Likelihood ratio gradient estimation for stochastic systems". In: *Communications of the ACM* 33.10, pp. 75–84.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Goodfellow, Ian, Jean Pouget-Abadie, et al. (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems* 27, pp. 2672–2680.

Graves, Alex (2013). "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850*.

Gregor, Karol et al. (2014). "Deep autoregressive networks". In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1242–1250.

Gulrajani, Ishaan et al. (2017). "Pixelvae: A latent variable model for natural images". In: *International Conference on Learning Representations*.

Gutmann, Michael and Aapo Hyvärinen (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, pp. 297–304.

Ha, David and Jürgen Schmidhuber (2018). "Recurrent world models facilitate policy evolution". In: *Advances in Neural Information Processing Systems*, pp. 2450–2462.

Higgins, Irina et al. (2017). "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hinton, Geoffrey E and Drew Van Camp (1993). "Keeping the neural networks simple by minimizing the description length of the weights". In: *Proceedings of the sixth annual conference on Computational learning theory*. ACM, pp. 5–13.

Hoffman, Matthew D et al. (2013). “Stochastic variational inference”. In: *The Journal of Machine Learning Research* 14.1, pp. 1303–1347.

Hyvärinen, Aapo and Peter Dayan (2005). “Estimation of non-normalized statistical models by score matching.” In: *Journal of Machine Learning Research* 6.4.

Hyvärinen, Aapo and Erkki Oja (2000). “Independent component analysis: algorithms and applications”. In: *Neural networks* 13.4-5, pp. 411–430.

Jaynes, Edwin T (1957). “Information theory and statistical mechanics”. In: *Physical review* 106.4, p. 620.

Jordan, Michael I et al. (1998). “An introduction to variational methods for graphical models”. In: *NATO ASI SERIES D BEHAVIOURAL AND SOCIAL SCIENCES* 89, pp. 105–162.

Kingma, Durk P, Tim Salimans, et al. (2016). “Improved variational inference with inverse autoregressive flow”. In: *Advances in neural information processing systems*, pp. 4743–4751.

Kingma, Durk P and Max Welling (2014). “Stochastic gradient VB and the variational auto-encoder”. In: *Proceedings of the International Conference on Learning Representations*.

Kumar, Manoj et al. (2020). “VideoFlow: A Flow-Based Generative Model for Video”. In: *International Conference on Learning Representations*.

Laparra, Valero, Gustavo Camps-Valls, and Jesús Malo (2011). “Iterative gaussianization: from ICA to random rotations”. In: *IEEE transactions on neural networks* 22.4, pp. 537–549.

LeCun, Yann et al. (2006). “A tutorial on energy-based learning”. In: *Predicting structured data* 1.0.

Lombardo, Salvator et al. (2019). “Deep Generative Video Compression”. In: *Advances in Neural Information Processing Systems*, pp. 9283–9294.

Makhzani, Alireza et al. (2015). “Adversarial autoencoders”. In: *arXiv preprint arXiv:1511.05644*.

Mnih, Andriy and Karol Gregor (2014). “Neural Variational Inference and Learning in Belief Networks”. In: *International Conference on Machine Learning*, pp. 1791–1799.

Mnih, Andriy and Koray Kavukcuoglu (2013). “Learning word embeddings efficiently with noise-contrastive estimation”. In: *Advances in neural information processing systems* 26, pp. 2265–2273.

Mnih, Andriy and Danilo Jimenez Rezende (2016). “Variational Inference for Monte Carlo Objectives”. In: *International Conference on Machine Learning*, pp. 2188–2196.

Mohamed, Shakir and Balaji Lakshminarayanan (2016). "Learning in implicit generative models". In: *arXiv preprint arXiv:1610.03483*.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

Neal, Radford M and Geoffrey E Hinton (1998). "A view of the EM algorithm that justifies incremental, sparse, and other variants". In: *Learning in graphical models*. Springer, pp. 355–368.

Olshausen, Bruno A and David J Field (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583, p. 607.

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748*.

Oord, Aäron van den et al. (2016). "WaveNet: A Generative Model for Raw Audio". In: *9th ISCA Speech Synthesis Workshop*, pp. 125–125.

Park, Yookoon, Chris Kim, and Gunhee Kim (2019). "Variational Laplace Autoencoders". In: *International Conference on Machine Learning*, pp. 5032–5041.

Radford, Alec et al. (2019). "Language Models are Unsupervised Multitask Learners". In:

Ranganath, Rajesh, Sean Gerrish, and David Blei (2014). "Black box variational inference". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 814–822.

Razavi, Ali, Aaron van den Oord, and Oriol Vinyals (2019). "Generating diverse high-fidelity images with vq-vae-2". In: *Advances in Neural Information Processing Systems*, pp. 14866–14876.

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning*, pp. 1530–1538.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models". In: *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286.

Rezende, Danilo Jimenez and Fabio Viola (2018). "Taming VAEs". In: *arXiv preprint arXiv:1810.00597*.

Rippel, Oren and Ryan Prescott Adams (2013). "High-dimensional probability estimation with deep density models". In: *arXiv preprint arXiv:1302.5125*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors." In: *Nature*.

Salakhutdinov, Ruslan and Geoffrey Hinton (2009). "Deep boltzmann machines". In: *Artificial intelligence and statistics*. PMLR, pp. 448–455.

Schulman, John et al. (2015). “Gradient estimation using stochastic computation graphs”. In: *Advances in Neural Information Processing Systems*, pp. 3528–3536.

Sønderby, Casper Kaae et al. (2016). “Ladder variational autoencoders”. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 3738–3746.

Song, Yang and Stefano Ermon (2019). “Generative modeling by estimating gradients of the data distribution”. In: *arXiv preprint arXiv:1907.05600*.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*, pp. 3104–3112.

Tabak, Esteban G and Cristina V Turner (2013). “A family of nonparametric density estimation algorithms”. In: *Communications on Pure and Applied Mathematics* 66.2, pp. 145–164.

Tipping, Michael E and Christopher M Bishop (1999). “Probabilistic principal component analysis”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3, pp. 611–622.

Titsias, Michalis and Miguel Lázaro-Gredilla (2014). “Doubly stochastic variational Bayes for non-conjugate inference”. In: *International conference on machine learning*, pp. 1971–1979.

Vincent, Pascal (2011). “A connection between score matching and denoising autoencoders”. In: *Neural computation* 23.7, pp. 1661–1674.

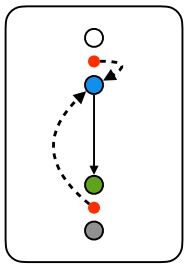
Williams, Ronald J (1992). “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Reinforcement Learning*. Springer, pp. 5–32.

Part II

Perception

Chapter 3

ITERATIVE AMORTIZED INFERENCE



learned negative feedback (static) perception

Marino, Joseph, Yisong Yue, and Stephan Mandt (2018). "Iterative Amortized Inference". In: *International Conference on Machine Learning*, pp. 3403–3412. URL: http://proceedings.mlr.press/v80/marino18a.html.

3.1 Introduction

In Chapter 2, we described latent variable models, $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, defined over observed variables, $\mathbf{x}$, and latent variables, $\mathbf{z}$. If we take the observed variables to be some form of sensory observation, e.g., visual, audio, or tactile input, then we can reasonably describe the compressed representation in $\mathbf{z}$ as a *state estimate*. That is, if we treat $p_\theta(\mathbf{x}, \mathbf{z})$ as an internal "forward model" of the environment generating the observations, i.e., the generative process, then $\mathbf{z}$ corresponds to the underlying state of the environment. Note that this does not necessarily correspond to the *actual* state of the environment; indeed, $\mathbf{z}$ may not be interpretable at all. Given an observation, we will refer to the process of state estimation, i.e., inferring $p_\theta(\mathbf{z}|\mathbf{x})$ or some approximation $q(\mathbf{z}|\mathbf{x})$, as *perception*. As noted in Chapter 2, this is generally computationally intractable, as it requires marginalizing over $\mathbf{z}$. For this reason, we described the technique of variational inference (Section 2.3), formulating inference as optimization. Finally, we noted that the concept of amortization has proven essential in efficiently performing variational inference in deep generative models, most notably in variational autoencoders (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014).

However, while amortization has dramatically improved the efficiency of variational inference, the direct inference models (or "encoders") typically employed raise their own issues (see Section 3.2). This chapter attempts to overcome these issues by introducing a more flexible and general amortization technique referred to as *iterative amortized inference*. Broadly, rather than framing variational inference as a *direct* mapping from observations to approximate posterior estimates, we use gradients or errors to *iteratively* update these estimates. In this way, inference once again becomes an iterative optimization algorithm, using negative feedback to minimize errors or gradient magnitudes, but now with the efficiency of amortization. This idea was inspired by (hierarchical) predictive coding (Mumford, 1992; Rao and Ballard, 1999; Friston, 2005), which uses weighted errors to perform inference. In turn, these works borrowed heavily from classical Bayesian inference techniques, such as Kalman filtering (Kalman, 1960), which performs exact probabilistic inference using prediction errors. More recent works, discussed below, have pursued similar ideas.

3.2 Issues with Direct Inference Models

The Amortization Gap

Variational inference reformulates inference as the maximization of $\mathcal{L}$ w.r.t. $q(\mathbf{z}|\mathbf{x})$, constituting the expectation step of the variational EM algorithm (Algorithm 1). In general, this is a difficult non-convex optimization problem, typically requiring a lengthy iterative estimation procedure (Ranganath, Gerrish, and Blei, 2014). Yet, direct inference models attempt to perform this optimization through a direct, discriminative mapping from data observations to approximate posterior parameters. Of course, generative models can adapt to accommodate sub-optimal approximate posteriors. Nevertheless, the possible limitations of a direct inference mapping applied to this difficult optimization procedure may result in sub-optimal estimates, limiting model performance.

We demonstrate this concept in Figure 3.1 by visualizing the optimization surface of $\mathcal{L}$ defined by a 2D latent Gaussian model and a particular binarized MNIST (LeCun et al., 1998) data example. To visualize the approximate posterior, we use a point estimate, $q(\mathbf{z}|\mathbf{x}) = \delta(\boldsymbol{\mu}_q)$, where $\boldsymbol{\mu}_q = (\mu_1, \mu_2)$ is the estimate and δ is the Dirac delta function. See Appendix C.1 for details. Shown on the plot are the optimal (maximum a posteriori or MAP) estimate, the estimate from a direct inference model, and an optimization trajectory of gradient ascent. The inference model is unable to achieve the optimum, but manages to output a reasonable estimate in one forward pass. Gradient ascent requires many iterations and is sensitive to step-size, but

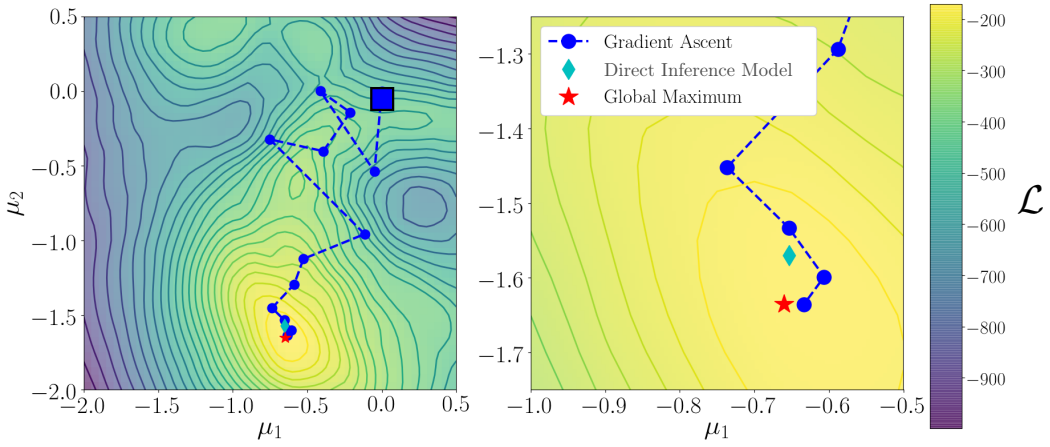


Figure 3.1: **The Amortization Gap.** Optimization surface of $\mathcal{L}$ (in *nats*) for a 2-D latent Gaussian model and an MNIST data example. Shown on the plots are the optimal estimate (★), the output of a direct inference model (♦), and an optimization trajectory of gradient ascent (●), initialized at the blue square at (0,0). The plot on the right shows an enlarged view near the optimum. Gradient-based optimization outperforms the direct inference model, exhibiting an amortization gap in performance: $\mathcal{L}(★) > \mathcal{L}(●) > \mathcal{L}(♦)$.

through the iterative estimation procedure, ultimately arrives at a better final estimate. The inability of inference models to reach optimal approximate posterior estimates, as typically compared with gradient-based methods, creates an *amortization gap* (Krishnan, Liang, and Hoffman, 2018; Cremer, Li, and Duvenaud, 2018), which impairs model performance. This is because, within the same parametric distribution family, a sub-optimal amortized approximate posterior, q_ϕ, creates a looser ELBO than the optimal approximate posterior, denoted q_*:

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}(\mathbf{x}; q_*, \theta) \geq \mathcal{L}(\mathbf{x}; q_\phi, \theta).$$

The difference between $\mathcal{L}(\mathbf{x}; q_*, \theta)$ and $\mathcal{L}(\mathbf{x}; q_\phi, \theta)$ is precisely the amortization gap (Cremer, Li, and Duvenaud, 2018). Additional latent dimensions and more complex data or models could further exacerbate this gap.

Lack of Prior Information

An altogether distinct issue with direct inference models concerns their lack of prior information. Naïve formulations of direct inference models consider a direct mapping from observations, $\mathbf{x}$, to the parameters of the approximate posterior (Eq. 2.28). With a constant prior, e.g., $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ in conventional VAEs (Kingma and Welling, 2014), such a direct mapping is sufficient to estimate the optimal approximate posterior within the class of diagonal distributions. However, in structured models,

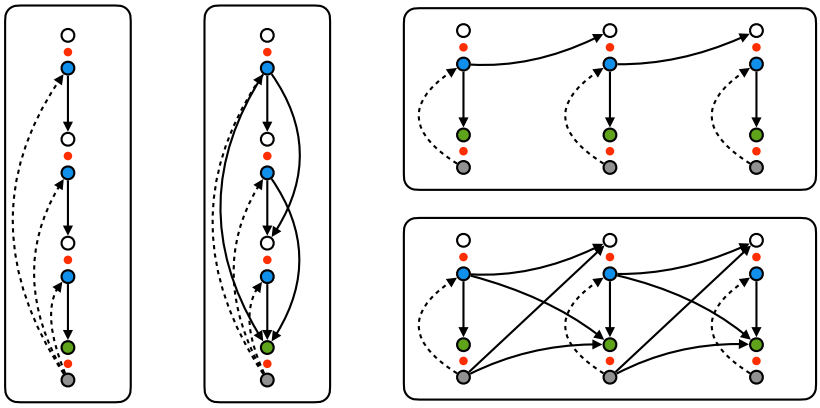


Figure 3.2: **Lack of Prior Information.** Naïve direct inference models, encoding only observations, cannot account for conditional priors in structured latent variable models. These arise in hierarchical (left) and sequential (right) latent variable models. For instance, in hierarchical models, bottom-up approximate posteriors at intermediate levels cannot account for conditional "top-down" priors which result from sampling higher-level latent variables.

e.g., hierarchical or autoregressive, the situation is more complicated. This is because the prior is now conditional, dependent on samples from upstream (parent) latent variables. If the inference model is only conditioned on the observation, it does not have access to the previously sampled latent variables and therefore cannot account for the conditional prior.

For instance, consider a hierarchical latent variable model with following prior:

$$p_\theta(\mathbf{z}^{1:L}) = \prod_{\ell=1}^{L} p_\theta(\mathbf{z}^\ell|\mathbf{z}^{\ell+1:L}).$$

A naïve direct inference model, i.e., only encoding observations, corresponds to the following approximate posterior factorization:

$$q_\phi(\mathbf{z}^{1:L}|\mathbf{x}) = \prod_{\ell=1}^{L} q_\phi(\mathbf{z}^\ell|\mathbf{x}).$$

Plugging these factorizations into the ELBO, we have

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,q) &= \mathbb{E}_{q_\phi(\mathbf{z}^{1:L}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}^{1:L})\right] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}^{1:L}|\mathbf{x})||p_\theta(\mathbf{z}^{1:L})) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}^{1:L}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}^{1:L})\right] - \sum_{\ell=1}^{L} \mathbb{E}_{q_\phi(\mathbf{z}^{\ell:L}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}^\ell|\mathbf{x})}{p_\theta(\mathbf{z}^\ell|\mathbf{z}^{\ell+1:L})}\right].\end{aligned}$$

The prior at each level depends on $\mathbf{z}^{\ell+1:L}$ while the approximate posterior does not. Thus, at best, a naïve direct inference model can only model the marginal prior,

$p_\theta(\mathbf{z}^\ell)$, unable to capture the structured latent dependencies. This limitation was hinted at by Dayan et al., 1995, noting that "*the top-down generative model plays no direct role*" in inference, but that "*such effects are important in real perception.*" A solution to this issue was proposed by Sønderby et al., 2016 and Salimans, 2016, reusing parts of the generative model during inference. Note that the same issue arises in sequential latent variable models, where naïve direct inference models do not have access to latent prior estimates. A similar solution is used in these situations, often using recurrent networks to condition on past latent variables and observations (Chung et al., 2015). While these solutions work empirically, they require hand-crafting the inference procedure.

3.3 Iterative Amortized Inference

Learning to Iteratively Optimize

While direct inference networks have provided significant benefits in computational efficiency for performing approximate inference, such models can be inaccurate and require additional considerations in structured models (Section 3.2). Dayan et al., 1995 put forth one possible solution: "*using iterative recognition, in which the generative and recognition activations interact to produce the final activity of a unit*." Thus, to improve upon the direct inference model paradigm, we pose the following question: *can we retain the computational efficiency of inference models while incorporating more powerful and general iterative estimation capabilities?* Our proposed solution is a new class of inference models, capable of learning how to update approximate posterior estimates by encoding gradients or errors. Due to the iterative nature of these models, we refer to them as *iterative inference models*. Through an analysis with latent Gaussian models, we show that iterative inference models generalize direct inference models (Section 3.4) and naturally are capable of performing structured variational inference (Section 3.4 & Chapter 4).

Our approach relates to learning to learn (Andrychowicz et al., 2016), where an *optimizer* model learns to optimize the parameters of an *optimizee* model. The optimizer receives the optimizee's parameter gradients and outputs updates to these parameters to improve the optimizee's loss. The optimizer itself can be learned due to the differentiable computation graph. Such models can adaptively adjust step sizes, potentially outperforming conventional optimizers. For inference optimization, previous works have combined direct inference models with gradient updates (Hjelm et al., 2016; Krishnan, Liang, and Hoffman, 2018; Kim et al., 2018), however, these works do not *learn* to iteratively optimize. Putzky and Welling, 2017 use recurrent

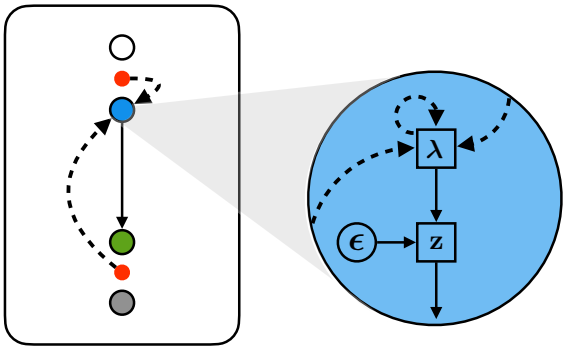


Figure 3.3: **Iterative Amortized Inference**. Computation graph for a latent variable model with iterative amortized inference. Red dots, denoting errors or gradients from the ELBO objective, are used to update the current estimate of the approximate posterior parameters, $\boldsymbol{\lambda}$. Using the pathwise derivative estimator, one can update the inference model parameters, ϕ, learning to efficiently perform iterative inference.

inference models for MAP estimation of denoised images in linear models. We propose a single method for learning to perform variational inference, generally applicable to latent variable models. Our work extends techniques for learning to optimize along several directions, discussed in Section 3.4.

Iterative Inference Models

Following the notation from Chapter 2, we denote an iterative amortized inference model as f_ϕ with parameters ϕ. If we consider the case of a single data example, $\mathbf{x}$, and corresponding approximate posterior distribution parameter estimate, $\boldsymbol{\lambda}$, the basic form of an iterative amortized inference model is given as:

$$\boldsymbol{\lambda} \leftarrow f_\phi(\nabla_{\boldsymbol{\lambda}} \mathcal{L}, \boldsymbol{\lambda}), \tag{3.1}$$

where we have used the shorthand $\mathcal{L} \equiv \mathcal{L}(\mathbf{x}; \theta, q)$. Iterative inference models take in the current estimate of $\boldsymbol{\lambda}$, as well as the inference gradient, $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$, and output an updated estimate of $\boldsymbol{\lambda}$. As with direct inference models, iterative inference model parameters are updated using estimates of $\nabla_\phi \mathcal{L}$, obtained through the pathwise derivative estimator (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014) or the score function estimator (Gregor, Danihelka, Mnih, et al., 2014; Ranganath, Gerrish, and Blei, 2014). While we consider other forms of iterative inference models in Section 3.4, note that Eq. 3.1 is a generalization of basic stochastic gradient-based optimization. For instance, one special case is

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \alpha \nabla_{\boldsymbol{\lambda}} \mathcal{L},$$

however, Eq. 3.1 also contains more general non-linear updates (Andrychowicz et al., 2016). Figure 3.3 displays a simplified computation graph of the inference procedure. We discuss more specific design choices in Section 3.5.

3.4 Iterative Inference in Latent Gaussian Models

Latent Gaussian models are used in both VAEs (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014) and hierarchical predictive coding (Rao and Ballard, 1999; Friston, 2005), so we focus on iterative amortized inference in these models. Latent Gaussian models have Gaussian prior densities over latent variables: $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_p, \operatorname{diag} \boldsymbol{\sigma}_p^2)$.[1] While the approximate posterior can be any probability density, it is typically also chosen as Gaussian: $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)$. With this choice, $\boldsymbol{\lambda} \equiv \left[\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q\right]$. We can express Eq. 3.1 for this setup as:

$$\left[\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q\right] \leftarrow f_\phi(\nabla_{\boldsymbol{\mu}_q} \mathcal{L}, \nabla_{\boldsymbol{\sigma}_q} \mathcal{L}, \boldsymbol{\mu}_q, \boldsymbol{\sigma}_q), \tag{3.2}$$

In Marino, Yue, and Mandt, 2018, we derive the gradients $\nabla_{\boldsymbol{\mu}_q} \mathcal{L}$ and $\nabla_{\boldsymbol{\sigma}_q} \mathcal{L}$ for the cases where $p_\theta(\mathbf{x}|\mathbf{z})$ takes a Gaussian and Bernoulli form, though *any* output distribution can be used. Generally, these gradients are composed of 1) errors, expressing the mismatch in distributions, and 2) Jacobian matrices, which invert the generative mappings. For instance, assuming a Gaussian output density, $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\mathbf{x}, \operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2)$, the gradient for $\boldsymbol{\mu}_q$ is

$$\nabla_{\boldsymbol{\mu}_q} \mathcal{L} = \mathbf{J}^\intercal \boldsymbol{\varepsilon}_\mathbf{x} - \boldsymbol{\varepsilon}_\mathbf{z}, \tag{3.3}$$

where the Jacobian, $\mathbf{J}$, and observed and latent errors, $\boldsymbol{\varepsilon}_\mathbf{x}$ and $\boldsymbol{\varepsilon}_\mathbf{z}$, are defined as

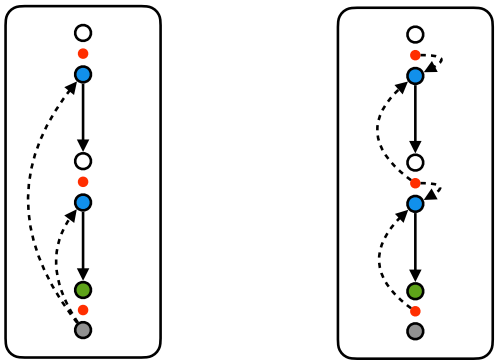
$$\mathbf{J} \equiv \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\frac{\partial \boldsymbol{\mu}_\mathbf{x}}{\partial \boldsymbol{\mu}_q} \right], \tag{3.4}$$

$$\boldsymbol{\varepsilon}_\mathbf{x} \equiv \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\frac{\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}}{\boldsymbol{\sigma}_\mathbf{x}^2} \right], \tag{3.5}$$

$$\boldsymbol{\varepsilon}_\mathbf{z} \equiv \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\frac{\mathbf{z} - \boldsymbol{\mu}_p}{\boldsymbol{\sigma}_p^2} \right]. \tag{3.6}$$

Here, we have assumed $\boldsymbol{\mu}_\mathbf{x}$ is a function of $\mathbf{z}$ and $\boldsymbol{\sigma}_\mathbf{x}^2$ is a global parameter. The gradient $\nabla_{\boldsymbol{\sigma}_q} \mathcal{L}$ is composed of similar terms as well as an additional term penalizing approximate posterior entropy. Inspecting and understanding the composition of the

[1] Although the prior is typically a standard Normal density, we use this prior form for generality.



(a) Direct (b) Iterative

Figure 3.4: **Automatic Top-Down Inference.** Bottom-up direct amortized inference cannot account for conditional priors, whereas iterative amortized inference automatically accounts for these priors through gradients or errors.

gradients reveals the forces pushing the approximate posterior toward agreement with the data, through $\varepsilon_{\mathbf{x}}$, and agreement with the prior, through $\varepsilon_{\mathbf{z}}$. In other words, *inference is as much a "top-down" process as it is a "bottom-up" process*, and the optimal combination of these terms is given by the approximate posterior gradients.

Interpreting Top-Down Inference

Naïve direct inference models, which only encode the data, lack the top-down prior information encapsulated in $\varepsilon_{\mathbf{z}}$. For instance, in a chain-structured hierarchical latent variable model (Figure 3.2, left), the gradient of $\boldsymbol{\mu}_q^\ell$, the approximate posterior mean at layer ℓ, is

$$\nabla_{\boldsymbol{\mu}_q^\ell} \mathcal{L} = \mathbf{J}^{\ell\intercal} \varepsilon_{\mathbf{z}}^{\ell-1} - \varepsilon_{\mathbf{z}}^{\ell}, \tag{3.7}$$

where $\mathbf{J}^\ell$ is the Jacobian of the generative mapping at layer ℓ and $\varepsilon_{\mathbf{z}}^\ell$ is defined similarly to Eq. 3.6. The error $\varepsilon_{\mathbf{z}}^\ell$ depends on the top-down prior at layer ℓ, which, unlike the single-level case, varies across data examples. Thus, a purely bottom-up inference procedure will struggle to optimize, and therefore utilize, the conditional prior. Iterative inference models, which rely on approximate posterior gradients, naturally account for both bottom-up and top-down influences (Figure 3.4).

Approximating Approximate Posterior Derivatives

In the formulation of iterative inference models given in Eq. 3.1, inference optimization is restricted to first-order approximate posterior derivatives. Thus, it may require many inference iterations to reach reasonable approximate posterior estimates. Rather than calculate costly higher-order derivatives, we can take a different approach.

Approximate posterior derivatives (e.g., Eq. 3.3 and higher-order derivatives) are essentially defined by the errors at the current estimate, as the other factors, such as the Jacobian matrices, are internal to the model. Thus, the errors provide more general information about the curvature beyond the gradient. As iterative inference models already learn to perform approximate posterior updates, it is natural to ask whether the errors provide a sufficient signal for faster inference optimization. In other words, we may be able to offload approximate posterior derivative calculation onto the inference model, yielding a model that requires fewer inference iterations while maintaining or possibly improving computational efficiency.

Comparing with Eq. 3.2, the form of this new iterative inference model is

$$\left[\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q\right] \leftarrow f_\phi(\boldsymbol{\varepsilon}_\mathbf{x}, \boldsymbol{\varepsilon}_\mathbf{z}, \boldsymbol{\mu}_q, \boldsymbol{\sigma}_q). \tag{3.8}$$

In Section 3.5, we empirically find that models of this form converge to better solutions than gradient-encoding models when given fewer inference iterations. It is also worth noting that this error encoding scheme is similar to DRAW (Gregor, Danihelka, Graves, et al., 2015). However, in addition to architectural differences in the generative model, DRAW and later extensions (Gregor, Besse, et al., 2016) do not include top-down errors nor error weighting. This encoding form is also similar to PredNet (Lotter, Kreiman, and Cox, 2017), which is also inspired by predictive coding, but is not strictly formulated in terms of variational inference.

Generalizing Direct Inference Models

Under certain assumptions on single-level latent Gaussian models, iterative inference models of the form in Section 3.4 generalize direct inference models. First, note that $\boldsymbol{\varepsilon}_\mathbf{x}$ (Eq. 3.5) is an affine transformation of $\mathbf{x}$:

$$\boldsymbol{\varepsilon}_\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{3.9}$$

where

$$\mathbf{A} \equiv \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[(\operatorname{diag}\boldsymbol{\sigma}_\mathbf{x}^2)^{-1}\right], \tag{3.10}$$

$$\mathbf{b} \equiv -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\frac{\boldsymbol{\mu}_\mathbf{x}}{\boldsymbol{\sigma}_\mathbf{x}^2}\right]. \tag{3.11}$$

Making the reasonable assumption that the initial approximate posterior and prior are both constant, then in expectation, $\mathbf{A}$, $\mathbf{b}$, and $\boldsymbol{\varepsilon}_\mathbf{z}$ are constant across all data examples at the initial inference iteration. Using proper weight initialization and input normalization, it is equivalent to input $\mathbf{x}$ or an affine transformation of $\mathbf{x}$ into a

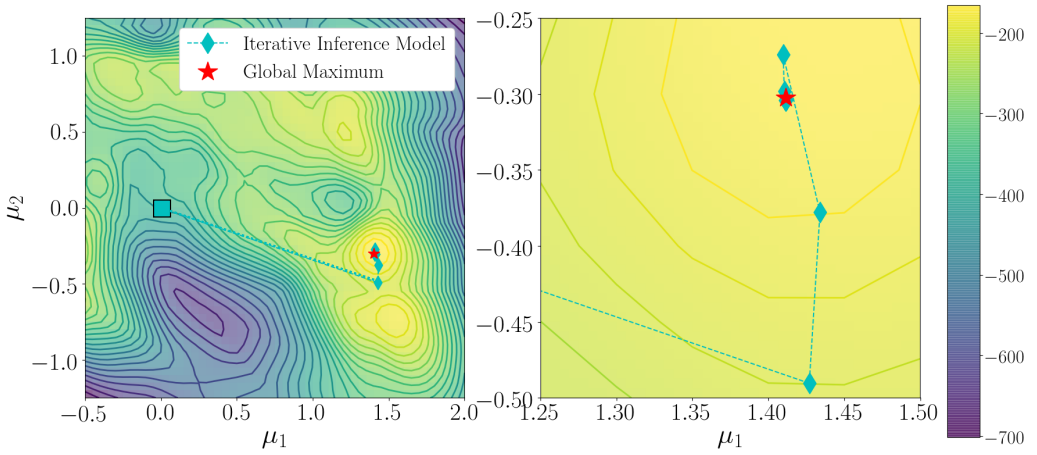


Figure 3.5: **Iterative Amortized Inference Optimization.** Optimization trajectory on $\mathcal{L}$ (in *nats*) for an iterative inference model with a 2D latent Gaussian model for a particular MNIST example. The estimate is initialized at $(0, 0)$ (cyan square), and the iterative inference model adaptively adjusts inference update step sizes to iteratively refine the approximate posterior estimate (♦) to arrive at the optimum (★).

fully-connected neural network. Therefore, in this case, *direct inference models are equivalent to the special case of a one-step iterative inference model*. Thus, we can interpret standard inference models as learning a map of local curvature around a fixed approximate posterior estimate. Iterative inference models, in contrast, learn to traverse the optimization landscape more generally.

3.5 Experiments

Setup

Data Using latent Gaussian models, we empirically evaluate iterative amortized inference on image and text data. For images, we use **MNIST** (LeCun et al., 1998), **Omniglot** (Lake, Salakhutdinov, and Tenenbaum, 2013), **Street View House Numbers (SVHN)** (Netzer et al., 2011), and **CIFAR-10** (Krizhevsky and Hinton, 2009). MNIST and Omniglot are dynamically binarized and modeled with Bernoulli conditional likelihoods, and SVHN and CIFAR-10 are modeled with Gaussian conditional likelihoods, using the procedure from (Gregor, Besse, et al., 2016). For text, we use **RCV1** (Lewis et al., 2004), with word count data modeled with a multinomial output.

Models Unlike direct inference models, which deal with a fixed input domain, approximate posterior parameters, errors, and inference gradients change throughout inference and learning. We found it beneficial to separately normalize each input

using layer normalization (Ba, Kiros, and Hinton, 2016). We also found it useful, though not necessary, to input the data observation (Figure 3.9a). For comparison with direct amortization, all experiments use feedforward networks, though recurrent networks may yield improved performance (Andrychowicz et al., 2016). For the output, we use a gated ("highway" (Srivastava, Greff, and Schmidhuber, 2015)) update:

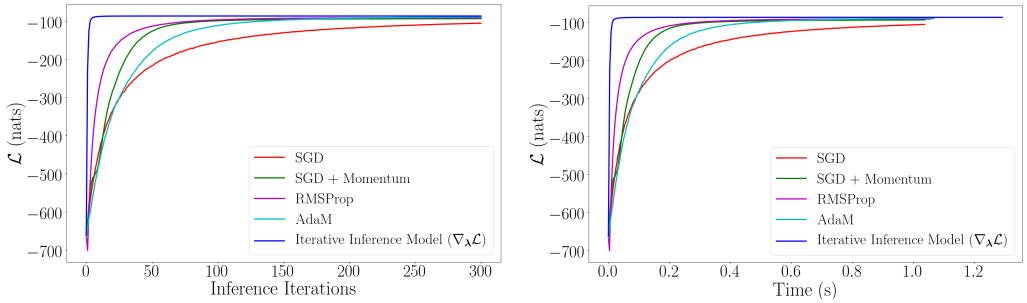
$$\boldsymbol{\lambda} \leftarrow \mathbf{g}_\phi(\cdot) \odot \boldsymbol{\lambda} + (\mathbf{1} - \mathbf{g}_\phi(\cdot)) \odot \boldsymbol{\delta}_\phi(\cdot), \tag{3.12}$$

where $\mathbf{g}_\phi(\cdot) \in [0, 1]$ is the gate and $\boldsymbol{\delta}_\phi(\cdot)$ is the update, both of which are output by the iterative inference model, and $\odot$ denotes element-wise multiplication. We estimate $\nabla_\phi \mathcal{L}$ using the pathwise derivative estimator, averaging over inference iterations. Reported values of $\mathcal{L}$ are estimated using 1 latent sample, and reported values of $\log p(\mathbf{x})$ and perplexity (Tables 3.1, 3.2, & 3.3) are estimated using $5,000$ importance-weighted samples. Additional experiment details can be found in Marino, Yue, and Mandt, 2018, and accompanying code can be found at `github.com/joelouismarino/iterative_inference`.

Approximate Inference Optimization

We start by demonstrating the inference optimization capabilities of iterative inference models. These models indeed learn to perform inference optimization through an adaptive iterative estimation procedure, with the results highlighting the unique aspects of iterative amortization as compared with direct amortization. We show that iterative inference models utilize multiple inference iterations rather than collapsing to static, one-step encoders.

2D Visualization As in Section 3.2, we directly visualize iterative amortized inference optimization in a 2D latent Gaussian model trained on MNIST with a point estimate approximate posterior. Model architectures are identical to those used in Section 3.2, with additional details found in Marino, Yue, and Mandt, 2018. Figure 3.5 shows a 16-step inference optimization trajectory taken by the iterative inference model for a particular example. The model adaptively adjusts inference update step sizes to navigate the optimization surface, quickly arriving and remaining at a near-optimal estimate. This is in contrast of other iterative inference optimization schemes, e.g., Krishnan, Liang, and Hoffman, 2018 and Kim et al., 2018, which must tune the step size for gradient-based optimizers.



(a) Iterations (b) Wall Clock Time

Figure 3.6: **Amortized vs. Gradient-Based Optimization.** Average ELBO over MNIST validation set during inference optimization as a function of (**a**) inference iterations and (**b**) inference wall-clock time. Iterative amortized inference converges *faster* than gradient-based optimizers to *better* estimates, remaining stable over hundreds of iterations, despite only being trained with 16 inference iterations.

ELBO During Inference To compare iterative amortization with gradient-based optimizers, we can quantify and compare optimization performance through the ELBO. In Figure 3.6, we plot the average ELBO on the MNIST validation set during inference for each optimizer, comparing in terms of iterations (Figure 3.6a) and wall clock time (Figure 3.6b). On average, iterative amortized inference converges significantly *faster* to *better* estimates than gradient-based optimizers. Note that, in this case, the inference model has *less* information than the optimizers; it only has access to the local gradient, whereas the optimizers use momentum and similar terms. Also note that the model's final estimates are empirically stable over 100s of iterations, despite only being trained using 16 inference iterations.

Reconstructions Iterative amortized inference optimization can also be visualized through image reconstructions. We demonstrate this for each dataset in Figure 3.7. As the reconstruction term is typically the dominant term in $\mathcal{L}$, the output reconstructions improve in terms of visual quality during inference optimization (left to right), more closely resembling $\mathbf{x}$ (far right of each figure).

Gradient Magnitudes During inference optimization, iterative inference models attempt to find local maxima. The gradient magnitudes of the ELBO w.r.t. the approximate posterior parameters, i.e., $\nabla_{\boldsymbol{\lambda}}\mathcal{L}$, should thus decrease during inference. In Figure 3.8, we plot these gradient magnitudes over inference iterations (each curve) throughout training for a model trained on RCV1. We find that the gradient magnitudes do, indeed, decrease during inference, with diminishing reduction at

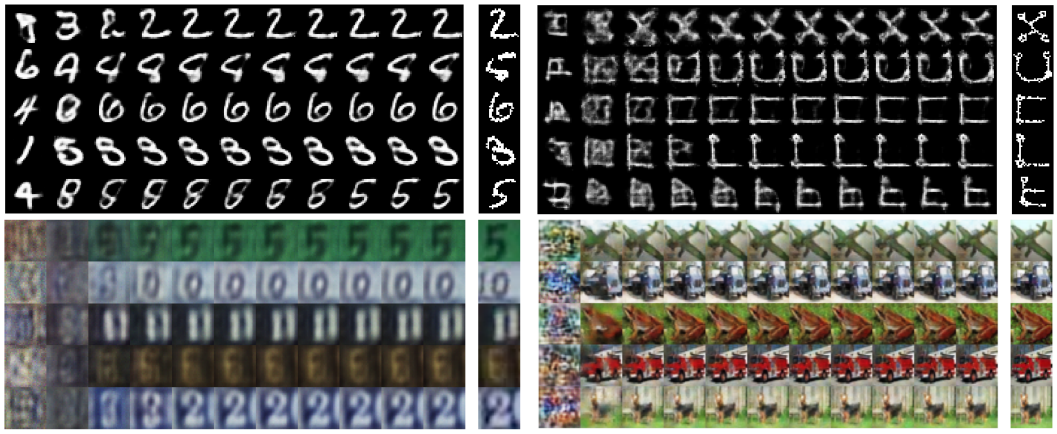


Figure 3.7: **Reconstructions Over Inference Iterations**. During inference (left to right), reconstruction means become gradually sharper, more closely resembling data examples (right).

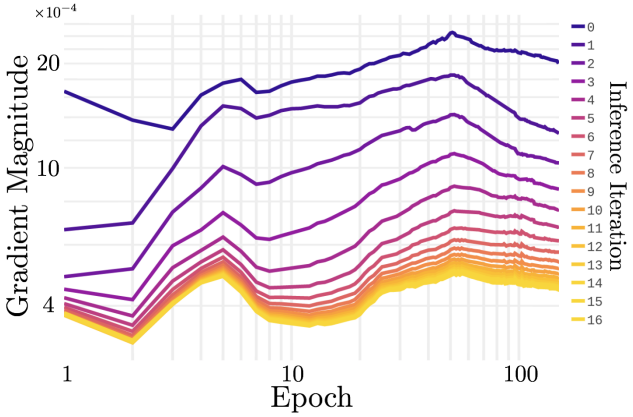
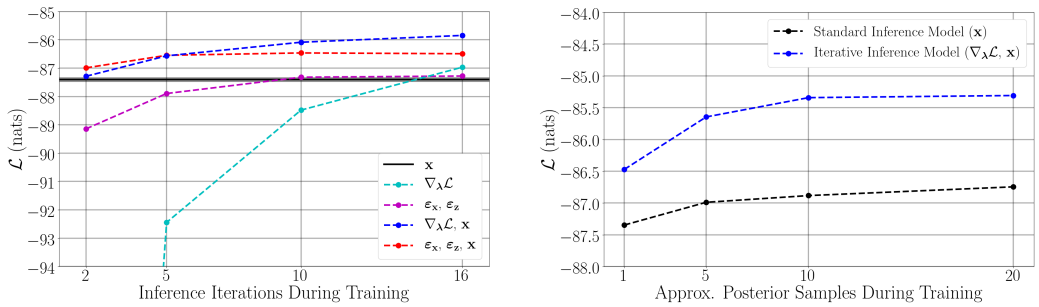


Figure 3.8: **Reduced Gradient Magnitudes**. Gradient magnitudes (*vertical axis*) over inference iterations (indexed by color on right) during training (*horizontal axis*) on RCV1. Approximate posterior mean gradient magnitudes decrease over inference iterations as estimates approach local maxima.

later iterations.

Additional Inference Iterations & Latent Samples

We highlight two sources that enable iterative inference models to further improve performance: additional **1**) inference iterations and **2**) latent samples. Additional inference iterations allow the inference model to further refine approximate posterior estimates. We demonstrate this on binarized MNIST, using inference models that encode approximate posterior gradients ($\nabla_{\boldsymbol{\lambda}}\mathcal{L}$) or errors ($\boldsymbol{\varepsilon}_{\mathbf{x}}, \boldsymbol{\varepsilon}_{\mathbf{z}}$), with or without the data ($\mathbf{x}$). These inference models are trained with 2, 5, 10, and 16 inference iterations. The model architecture is identical in each case, the only difference being the number



(a) (b)

Figure 3.9: **Hyperparameter Comparison**. ELBO for direct and iterative inference models on binarized MNIST for **(a)** additional inference iterations during training and **(b)** additional samples. Iterative inference models improve significantly with both quantities.

of input weights. Note that the small size of $\mathbf{z}$ relative to $\mathbf{x}$ gives the gradient encoding model ($\nabla_{\boldsymbol{\lambda}}$) *fewer* input parameters than the direct inference model (solid black line). The other inference models have more input parameters. Results are shown in Figure 3.9a, where we observe improved performance with increasing inference iterations. With enough iterations, each iterative inference model outperforms the direct inference model. As discussed in Section 3.4, encoding errors approximates higher-order derivatives, which helps when training with fewer inference iterations.

Additional approximate posterior samples provide more precise (lower variance) gradient and error estimates, potentially allowing an iterative inference model to output improved updates. To verify this, we train direct and iterative inference models on binarized MNIST using 1, 5, 10, and 20 approximate posterior samples. Here, iterative inference models encode the data, $\mathbf{x}$, and approximate posterior gradients, $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$, for 5 iterations. Results are shown in Figure 3.9b. Iterative inference models improve by more than 1 *nat* with additional samples, further widening the improvement over the comparable direct inference model.

Comparison with Direct Amortized Inference

We now compare the resulting log-likelihood performance between direct and iterative amortized inference on binarized MNIST, CIFAR-10, and RCV1. Inference model architectures are identical across each comparison, again, with the exception of input parameters. Further details are found in Marino, Yue, and Mandt, 2018. Tables 3.1 & 3.2 report estimated negative log-likelihood performance on binarized MNIST and CIFAR-10 respectively. Table 3.3 reports estimated perplexity on RCV1.

Table 3.1: **Negative Log-Likelihood** on binarized MNIST (in *nats*) for direct and iterative amortized inference.

	$-\log p(\mathbf{x})$
Single-Level	
Direct	84.14 ± 0.02
Iterative	$\mathbf{83.84 \pm 0.05}$
Hierarchical	
Direct	82.63 ± 0.01
Iterative	$\mathbf{82.457 \pm 0.001}$

Table 3.2: **Negative Log-Likelihood** on CIFAR-10 (in *bits/dim.*) for direct and iterative amortized inference.

	$-\log p(\mathbf{x})$
Single-Level	
Direct	5.823 ± 0.001
Iterative	$\mathbf{5.64 \pm 0.03}$
Hierarchical	
Direct	5.565 ± 0.002
Iterative	$\mathbf{5.456 \pm 0.005}$

Table 3.3: **Perplexity** on RCV1 for direct and iterative amortized inference.

	Perplexity	$\leq$
Krishnan, Liang, and Hoffman (2018)		331
Direct	323 ± 3	377.4 ± 0.5
Iterative	$\mathbf{285.0 \pm 0.1}$	$\mathbf{314 \pm 1}$

Briefly, Perplexity, P, is defined as

$$P \equiv \exp\left(-\frac{1}{N}\sum_i \frac{1}{n_i}\log p(\mathbf{x}^{(i)})\right),$$

where N is the number of data examples and n_i is the total number of word counts in document i. On each dataset and model, iterative amortized inference outperforms direct amortization. This holds for both single-level and hierarchical (2 level) models. We observe larger improvements on the high-dimensional RCV1 dataset, consistent with Krishnan, Liang, and Hoffman, 2018. Because the generative model architectures are kept fixed, performance improvements demonstrate improvements in the inference procedure.

3.6 Discussion

In this chapter, we introduced iterative amortized inference, which learns to refine inference estimates by encoding approximate posterior gradients or errors, i.e., learned negative feedback perception. This inference procedure is inspired by hierarchical predictive coding (Rao and Ballard, 1999; Friston, 2005), which uses Gaussian errors to perform gradient-based variational inference. On several benchmark datasets of images and text, we have demonstrated that iterative amortized inference empirically outperforms direct amortization, while retaining the efficiency benefits of amortization over gradient-based optimization. In latent Gaussian models, iterative

amortized inference generalizes and extends direct amortization, and by naturally accounting for priors through gradients or errors, these models provide insight and justification for top-down inference. This chapter has focused exclusively on performing inference in static latent variable models. In the following chapter, we apply this inference technique to sequential latent variable models, where we devise a general-purpose filtering procedure centered around prediction and updating.

References

Andrychowicz, Marcin et al. (2016). "Learning to learn by gradient descent by gradient descent". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 3981–3989.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer normalization". In: *arXiv preprint arXiv:1607.06450*.

Chung, Junyoung et al. (2015). "A recurrent latent variable model for sequential data". In: *Advances in neural information processing systems*, pp. 2980–2988.

Cremer, Chris, Xuechen Li, and David Duvenaud (2018). "Inference Suboptimality in Variational Autoencoders". In: *International Conference on Machine Learning*, pp. 1078–1086.

Dayan, Peter et al. (1995). "The helmholtz machine". In: *Neural computation* 7.5, pp. 889–904.

Friston, Karl (2005). "A theory of cortical responses". In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 360.1456, pp. 815–836.

Gregor, Karol, Frederic Besse, et al. (2016). "Towards conceptual compression". In: *Advances In Neural Information Processing Systems (NIPS)*, pp. 3549–3557.

Gregor, Karol, Ivo Danihelka, Alex Graves, et al. (2015). "DRAW: A recurrent neural network for image generation". In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1462–1471.

Gregor, Karol, Ivo Danihelka, Andriy Mnih, et al. (2014). "Deep autoregressive networks". In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1242–1250.

Hjelm, Devon et al. (2016). "Iterative refinement of the approximate posterior for directed belief networks". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 4691–4699.

Kalman, Rudolph Emil (1960). "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering* 82.1, pp. 35–45.

Kim, Yoon et al. (2018). "Semi-Amortized Variational Autoencoders". In: *Proceedings of the International Conference on Machine Learning (ICML)*.

Kingma, Durk P and Max Welling (2014). "Stochastic gradient VB and the variational auto-encoder". In: *Proceedings of the International Conference on Learning Representations*.

Krishnan, Rahul G, Dawen Liang, and Matthew Hoffman (2018). "On the challenges of learning with inference networks on sparse, high-dimensional data". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 143–151.

Krizhevsky, Alex and Geoffrey E Hinton (2009). "Learning multiple layers of features from tiny images". In:

Lake, Brenden M, Ruslan R Salakhutdinov, and Josh Tenenbaum (2013). "One-shot learning by inverting a compositional causal process". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 2526–2534.

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lewis, David D et al. (2004). "Rcv1: A new benchmark collection for text categorization research". In: *The Journal of Machine Learning Research* 5.Apr, pp. 361–397.

Lotter, William, Gabriel Kreiman, and David Cox (2017). "Deep predictive coding networks for video prediction and unsupervised learning". In: *International Conference on Learning Representations*.

Marino, Joseph, Yisong Yue, and Stephan Mandt (2018). "Iterative Amortized Inference". In: *International Conference on Machine Learning*, pp. 3403–3412. URL: http://proceedings.mlr.press/v80/marino18a.html.

Mumford, David (1992). "On the computational architecture of the neocortex". In: *Biological cybernetics* 66.3, pp. 241–251.

Netzer, Yuval et al. (2011). "Reading digits in natural images with unsupervised feature learning". In: *NIPS workshop on deep learning and unsupervised feature learning*.

Putzky, Patrick and Max Welling (2017). "Recurrent inference machines for solving inverse problems". In: *arXiv preprint arXiv:1706.04008*.

Ranganath, Rajesh, Sean Gerrish, and David Blei (2014). "Black box variational inference". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 814–822.

Rao, Rajesh PN and Dana H Ballard (1999). "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects." In: *Nature neuroscience* 2.1.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backpropagation and approximate inference in deep generative models”. In: *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286.

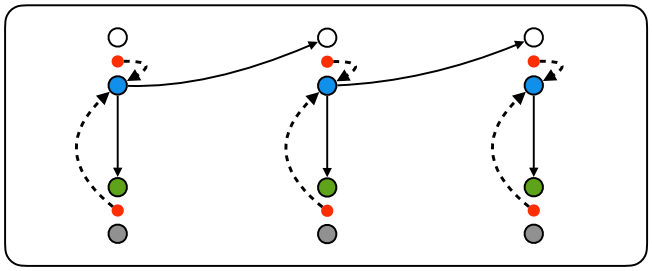
Salimans, Tim (2016). “A structured variational auto-encoder for learning deep hierarchies of sparse features”. In: *arXiv preprint arXiv:1602.08734*.

Sønderby, Casper Kaae et al. (2016). “Ladder variational autoencoders”. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 3738–3746.

Srivastava, Rupesh K, Klaus Greff, and Jürgen Schmidhuber (2015). “Training very deep networks”. In: *Advances in neural information processing systems (NIPS)*, pp. 2377–2385.

Chapter 4

AMORTIZED VARIATIONAL FILTERING



learned negative feedback (dynamic) perception

Marino, Joseph, Milan Cvitkovic, and Yisong Yue (2018). "A general method for amortizing variational filtering". In: *Advances in Neural Information Processing Systems*, pp. 7857–7868. URL: http://papers.nips.cc/paper/8011-a-general-method-for-amortizing-variational-filtering.

4.1 Introduction

Chapter 3 introduced iterative amortized inference, a technique for accurately and efficiently performing inference optimization using gradients or errors. While inference in this setup is iterative, the models themselves were formulated in static settings, e.g., images. Many data sources, in contrast, contain sequential structure, e.g., video, audio, or proprioceptive kinematics. This structure can be exploited to improve modeling performance and enable prediction of future observations. Thus, we require techniques for formulating deep *sequential* latent variable models, as well as inference techniques that enable their training.

In this chapter, we extend iterative amortized inference to perform filtering in deep sequential latent variable models, providing a general-purpose algorithm for performing approximate Bayesian filtering. We first formulate variational EM in the filtering setting, i.e., conditioning only on past and current variables. We then describe an instantiation of this algorithm, implementing inference optimization at each step with iterative amortized inference. The resulting inference method, *amortized variational filtering* (AVF), is generally applicable to sequential latent

variable models, while retaining the efficiency of amortization. We demonstrate this generality by applying AVF to three previously proposed deep sequential latent variable models across three data domains: audio, video, and MIDI music. In comparison with the custom inference models proposed for each generative model, we show that our single method, AVF, matches or outperforms log-likelihood performance in each case, with further improvement resulting from additional inference iterations. Thus, AVF provides a conceptually simple, generally applicable, and powerful filtering inference procedure.

4.2 Background

Sequential Latent Variable Models

Here, we briefly review sequential latent variable models (SLVMs). A sequence of T observations, $\mathbf{x}_{1:T}$, can be modeled using a SLVM, $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$, which models the joint distribution between $\mathbf{x}_{1:T}$ and a sequence of latent variables, $\mathbf{z}_{1:T}$, with parameters θ. It is typically assumed that $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$ can be factorized into conditional joint distributions at each step, $p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$, which are conditioned on preceding variables. This results in the following autoregressive form:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}). \quad (4.1)$$

$p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})$ is the *observation* model, and $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$ is the *latent dynamics* model, both of which can be arbitrary functions of their conditioning variables. However, while Eq. 4.1 provides the general form of a SLVM, further assumptions about the dependency structure, e.g., Markov, or functional forms, e.g., linear, are often necessary for tractable inference and learning.

Variational Inference

Given a model and a set of observations, we typically want to *infer* the posterior for each sequence, $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$, and *learn* the model parameters, θ. Inference can be performed online or offline through Bayesian filtering or smoothing respectively (Särkkä, 2013), and learning can be performed through maximum likelihood estimation. Unfortunately, inference and learning are intractable for all but the simplest, e.g., linear, model classes. For non-linear functions, which are present in *deep* latent variable models, we must resort to approximate inference. As introduced in Chapter 2, variational inference (Jordan et al., 1998), framed in the sequential setting, reformulates inference as optimization by introducing an approximate posterior, $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$, then minimizing the KL divergence to the true posterior, $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$.

To avoid evaluating $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$, one can express the KL divergence as

$$D_{\mathrm{KL}}(q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})||p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})) = \log p_\theta(\mathbf{x}_{1:T}) - \mathcal{L}(\mathbf{x}_{1:T}; \theta, q), \tag{4.2}$$

where, again, $\mathcal{L}$ is the *evidence lower bound* (ELBO), defined as

$$\mathcal{L}(\mathbf{x}_{1:T}; \theta, q) \equiv \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\log \frac{p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\right]. \tag{4.3}$$

In Eq. 4.2, $\log p_\theta(\mathbf{x}_{1:T})$ is independent of $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$, so one can minimize the KL divergence to the true posterior, thereby performing approximate inference, by maximizing $\mathcal{L}$ w.r.t. $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$. Further, as KL divergence is non-negative, Eq. 4.2 implies that the ELBO lower bounds the log-likelihood. Therefore, upon maximizing $\mathcal{L}$ w.r.t. $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$, one can use the gradient $\nabla_\theta \mathcal{L}$ to learn the model parameters. These two optimization procedures are respectively the expectation and maximization steps of the variational EM algorithm (Neal and Hinton, 1998), which alternate until convergence. To scale this algorithm, stochastic gradients can be used for both inference (Ranganath, Gerrish, and Blei, 2014) and learning (Hoffman et al., 2013).

As we have noted, performing inference using gradient-based optimization can be computationally inefficient, potentially requiring many inference iterations. To increase efficiency, we can use amortization (Gershman and Goodman, 2014). However, direct inference models (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014), only receiving the data as input, are unable to account for conditional (empirical) priors, which occur with structured latent variables. Such priors arise in the latent dynamics of SLVMs, forming priors across time steps. As with "top-down" inference in hierarchical latent variable models (C. K. Sønderby et al., 2016), many previous works use recurrent inference models in SLVMs, e.g., Chung, Kastner, et al., 2015, to encode previous observations and latent samples. However, in this case, the inference model must effectively *re-learn* the latent dynamics, possibly introducing inaccuracies into the inference scheme.

Related Work

Many *deterministic* deep sequential models have been proposed for sequence data (Chung, Gulcehre, et al., 2014; N. Srivastava, Mansimov, and Salakhudinov, 2015; Lotter, Kreiman, and Cox, 2017; Finn, Goodfellow, and Levine, 2016). While these models often capture many aspects of the data, they cannot account for the uncertainty inherent in many domains, typically arising from partial observability of the environment. By averaging over multi-modal distributions, these models

often produce samples in regions of low probability, e.g., blurry video frames. This inadequacy necessitates moving to models with stochastic latent variables, which can better model uncertainty to accurately capture the distribution of possible sequences.

Amortized variational inference (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014) has enabled many recently proposed probabilistic deep sequential latent variable models, with applications to video (Walker et al., 2016; Karl et al., 2017; Xue et al., 2016; Johnson et al., 2016; Gemici et al., 2017; Fraccaro, Kamronn, et al., 2017; Babaeizadeh et al., 2018; Denton and Fergus, 2018; Li and Mandt, 2018; He et al., 2018), speech (Chung, Kastner, et al., 2015; Fraccaro, S. K. Sønderby, et al., 2016; Goyal et al., 2017; Hsu, Zhang, and Glass, 2017; Li and Mandt, 2018), handwriting (Chung, Kastner, et al., 2015), music (Fraccaro, S. K. Sønderby, et al., 2016), etc. While these models differ in their functional mappings, most fall within the general form of Eq. 4.1. Crucially, simply encoding the observation at each step is insufficient to accurately perform approximate inference, as the prior can vary across steps. Thus, with each model, a hand-crafted amortized inference procedure has been proposed. For instance, many filtering inference methods re-use various components of the generative model (Chung, Kastner, et al., 2015; Fraccaro, S. K. Sønderby, et al., 2016; Gemici et al., 2017; Denton and Fergus, 2018), while some methods introduce separate recurrent neural networks into the filtering procedure (Bayer and Osendorfer, 2014; Denton and Fergus, 2018) or encode the previous latent sample (Karl et al., 2017). Specifying an amortized filtering method has been an engineering effort, as we have lacked a general-purpose method.

The variational filtering EM algorithm (Section 4.3) precisely specifies the inference optimization procedure implied by performing filtering inference on the sequential variational objective (Eq. 4.3). The main insight from this analysis is that, having drawn approximate posterior samples at previous steps, inference becomes a temporally localized optimization, depending only on the current prior and observation. This suggests a general approach that explicitly performs inference optimization at each step, replacing the current collection of custom filtering methods. When the approximate posterior at each step is initialized at the corresponding prior, this approach entails a Bayesian prediction-update loop, with the update composed of a gradient (or error) signal.

Perhaps the closest technique in the probabilistic modeling literature is the "residual" inference method from Fraccaro, S. K. Sønderby, et al., 2016, which updates the approximate posterior mean from the prior. Similar ideas have been proposed

on an empirical basis for deterministic models (Lotter, Kreiman, and Cox, 2017; Henaff, Zhao, and LeCun, 2017). PredNet (Lotter, Kreiman, and Cox, 2017) is a deterministic model that encodes prediction errors to perform inference. Like our method, this approach is inspired by hierarchical predictive coding (Rao and Ballard, 1999; Friston, 2005). Predictive coding, in turn, is an approximation to classical Bayesian filtering (Särkkä, 2013), which updates the posterior from the prior using the log-likelihood (error) of the prediction. For linear Gaussian models, this manifests as the Kalman filter (Kalman, 1960), which uses prediction errors to perform exact inference.

Finally, several recent works have used particle filtering in conjunction with amortized inference to provide a tighter bound on the log-likelihood for sequential models (Maddison et al., 2017; Naesseth et al., 2018; Le et al., 2018). The techniques developed here can also be applied to this setting, providing an improved method for estimating the proposal distribution.

4.3 Variational Filtering

Variational Filtering Expectation Maximization (EM)

In the filtering setting, the approximate posterior at each step is conditioned only on information from past and present variables, enabling *online* approximate inference. This implies a structured approximate posterior, in which $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ factorizes across steps as

$$q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}). \tag{4.4}$$

Note that the conditioning variables in each term of q denote an *indirect* dependence that arises through inference optimization and does not necessarily constitute a *direct* functional mapping. Under a filtering approximate posterior, the ELBO can be expressed as

$$\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{\prod_{\tau=1}^{t-1} q(\mathbf{z}_\tau|\mathbf{x}_{\leq\tau},\mathbf{z}_{<\tau})} [\mathcal{L}_t] = \sum_{t=1}^{T} \widetilde{\mathcal{L}}_t, \tag{4.5}$$

(derived in Appendix 4.6) where $\mathcal{L}_t$ is the *step ELBO*, defined as

$$\mathcal{L}_t \equiv \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t},\mathbf{z}_{<t})} \left[\log \frac{p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})}{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})} \right], \tag{4.6}$$

and we have also defined $\widetilde{\mathcal{L}}_t$ as the t^{th} term in the summation. Note that with a single step, the filtering ELBO reduces to the first step ELBO, thereby recovering the case of a static latent variable model. As in this setting, the step ELBO can be

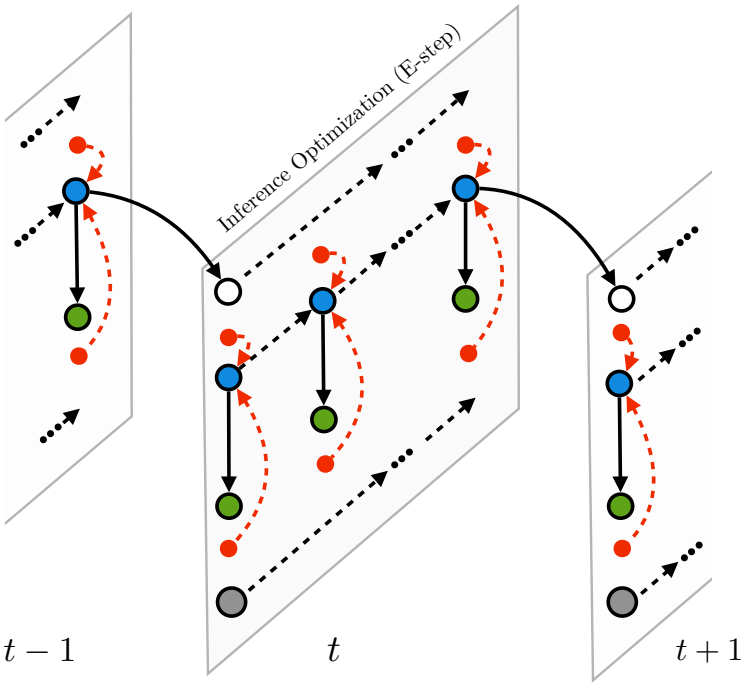


Figure 4.1: **Variational Filtering Inference.** The diagram shows filtering inference within a sequential latent variable model, as outlined in Algorithm 2. The central gray region depicts inference optimization at step t, initialized at or near the corresponding prior (white), $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$. Sampling from the approximate posterior (blue) generates the conditional likelihood (green), $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})$, which is evaluated at the observation (gray), $\mathbf{x}_t$, to calculate the prediction error. This term is combined with the KL divergence between the approximate posterior and prior, yielding the step ELBO (red), $\mathcal{L}_t$ (Eq. 4.7). Inference optimization (E-step) involves finding the approximate posterior that maximizes the step ELBO terms.

re-expressed as a reconstruction term and a KL divergence term:

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}\left[\log p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})\right] - D_{\mathrm{KL}}(q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})||p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})). \tag{4.7}$$

The filtering ELBO in Eq. 4.5 is the sum of these step ELBO terms, each of which is evaluated according to expectations over past latent sequences. To perform filtering variational inference, we must find the set of T terms in $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ that maximize the filtering ELBO summation.

We now describe the variational filtering EM algorithm, given in Algorithm 2 and depicted in Figure 4.1, which optimizes Eq. 4.5. This algorithm sequentially optimizes each of the approximate posterior terms to perform filtering inference. Consider the approximate posterior at step t, $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$. This term appears in $\mathcal{L}$,

either directly or in expectations, in terms t through T of the summation:

$$\mathcal{L} = \underbrace{\widetilde{\mathcal{L}}_1 + \widetilde{\mathcal{L}}_2 + \cdots + \widetilde{\mathcal{L}}_{t-1} + \overbrace{\widetilde{\mathcal{L}}_t}^{\text{terms in which } q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) \text{ appears}}}_{\text{steps on which } q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) \text{ depends}} + \widetilde{\mathcal{L}}_{t+1} + \cdots + \widetilde{\mathcal{L}}_{T-1} + \widetilde{\mathcal{L}}_T \,. \tag{4.8}$$

However, the filtering setting dictates that the optimization of the approximate posterior at each step can only condition on past and present variables, i.e., steps 1 through t. Therefore, of the T terms in $\mathcal{L}$, the *only* term through which we can optimize $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ is the t^{th} term:

$$q^*(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) = \underset{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}{\arg\max} \; \widetilde{\mathcal{L}}_t. \tag{4.9}$$

Optimizing $\widetilde{\mathcal{L}}_t$ requires evaluating expectations over previous approximate posteriors. Again, because approximate posterior estimates cannot be influenced by future variables, these past expectations remain *fixed* through the future. Thus, variational filtering (the variational E-step) can be performed by sequentially maximizing each $\mathcal{L}_t$ w.r.t. $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$, holding the expectations over past variables fixed. Conveniently, once the past expectations have been evaluated, inference optimization is entirely defined by the ELBO at that step.

For simple models, such as linear Gaussian models, these expectations may be computed exactly. However, in general, the expectations must be estimated through Monte Carlo samples from q, with inference optimization carried out using stochastic gradients (Ranganath, Gerrish, and Blei, 2014). As in the static setting, we can initialize $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ at (or near) the prior, $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$. This yields a simple interpretation: starting with q at the prior, we generate a *prediction* of the data through the likelihood, $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})$, to evaluate the current step ELBO. Using the approximate posterior gradient, we then perform an inference *update* to the estimate of q. This resembles classical Bayesian filtering, where the posterior is updated from the prior prediction according to the likelihood of observations. Unlike the classical setting, reconstruction and update steps are repeated until inference convergence. This resembles the extended Kalman filter (Kalman, 1960; Murphy, 2012), but is applicable to non-linear models with non-Gaussian distributions.

After inferring an optimal approximate posterior, learning (the variational M-step) can be performed by maximizing the total filtering ELBO w.r.t. the model parameters, θ. As Eq. 4.5 is a summation and differentiation is a linear operation, $\nabla_\theta \mathcal{L}$ is the

Algorithm 2 Variational Filtering Expectation Maximization

1: **Input:** observation sequence $\mathbf{x}_{1:T}$, model $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$
2: $\nabla_\theta \mathcal{L} = 0$ ▷ parameter gradient
3: **for** $t = 1$ **to** T **do**
4: initialize $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ ▷ at/near $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$
5: $\widetilde{\mathcal{L}}_t := \mathbb{E}_{q(\mathbf{z}_{<t}|\mathbf{x}_{<t}, \mathbf{z}_{<t-1})}\left[\mathcal{L}_t\right]$
6: $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) = \arg\max_q \widetilde{\mathcal{L}}_t$ ▷ inference (E-Step)
7: $\nabla_\theta \mathcal{L} \leftarrow \nabla_\theta \mathcal{L} + \nabla_\theta \widetilde{\mathcal{L}}_t$
8: **end for**
9: $\theta = \theta + \alpha \nabla_\theta \mathcal{L}$ ▷ learning (M-Step)

sum of contributions from each of these terms:

$$\nabla_\theta \mathcal{L} = \sum_{t=1}^{T} \nabla_\theta \left[\mathbb{E}_{\prod_{\tau=1}^{t-1} q(\mathbf{z}_\tau|\mathbf{x}_{\leq \tau}, \mathbf{z}_{<\tau})} \left[\mathcal{L}_t\right] \right]. \tag{4.10}$$

Parameter gradients can be estimated online by accumulating the result from each term in the filtering ELBO. The parameters are then updated at the end of the sequence. For large datasets, stochastic estimates of parameter gradients can be obtained from a mini-batch of data examples (Hoffman et al., 2013).

Amortized Variational Filtering

Performing approximate inference optimization (Algorithm 2, Line 6) with traditional techniques can be computationally costly, requiring many iterations of gradient updates and hand-tuning of optimizer hyper-parameters. In online settings, with large models and data sets, this may be impractical. An alternative approach is to employ an amortized inference model, which can learn to maximize $\mathcal{L}_t$ w.r.t. $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ more efficiently at each step. Note that $\mathcal{L}_t$ (Eq. 4.6) contains $p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) = p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$. The prior, $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$, varies across steps, constituting the latent dynamics. Direct inference models, which only encode $\mathbf{x}_t$, do not have access to the prior and therefore cannot properly optimize $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$. Many inference models in the sequential setting attempt to account for this information by including hidden states (Chung, Kastner, et al., 2015; Fraccaro, S. K. Sønderby, et al., 2016; Denton and Fergus, 2018). However, given the complexities of many generative models, it can be difficult to determine how to properly route the necessary prior information into the inference model. As a result, each SLVM has been proposed with an accompanying custom inference model.

We propose a simple and general alternative method for amortizing filtering inference that is agnostic to the particular form of the generative model. Iterative

amortized inference (Chapter 3) naturally accounts for the changing prior through the approximate posterior gradients or errors. These models are thus a natural candidate for performing inference at each step. Similar to Eq. 3.1, when $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ is a parametric distribution with parameters $\boldsymbol{\lambda}_t^q$, the inference update takes the form:

$$\boldsymbol{\lambda}_t^q \leftarrow f_\phi(\boldsymbol{\lambda}_t^q, \nabla_{\boldsymbol{\lambda}_t^q} \widetilde{\mathcal{L}}_t). \tag{4.11}$$

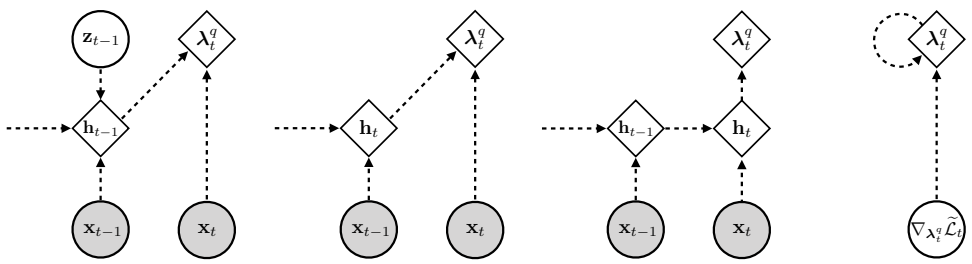
We refer to this setup as *amortized variational filtering* (AVF). Again, Eq. 4.11 offers just one particular encoding form for an iterative inference model. For instance, $\mathbf{x}_t$ could be additionally encoded at each step. As noted in Chapter 3, in latent Gaussian models, precision-weighted errors provide an alternative inference optimization signal. There are two main benefits to using iterative inference models in the filtering setting:

- The approximate posterior is updated from the prior, so model capacity is utilized for inference *corrections* rather than re-estimating the latent dynamics at each step.
- These inference models contain all of the terms necessary to perform inference optimization, providing a simple model form that does not require any additional hidden states or inputs.

In practice, these advantages permit the use of relatively simple iterative inference models that can perform filtering inference efficiently and accurately. We demonstrate this in the following section.

4.4 Experiments

We empirically evaluate amortized variational filtering using multiple deep sequential latent Gaussian model architectures on a variety of sequence datasets. Specifically, we use AVF to train VRNN (Chung, Kastner, et al., 2015), SRNN (Fraccaro, S. K. Sønderby, et al., 2016), and SVG (Denton and Fergus, 2018) on speech (Garofolo et al., 1993), music (Boulanger-Lewandowski, Bengio, and Vincent, 2012), and video (Schuldt, Laptev, and Caputo, 2004) data. In each setting, we compare AVF against the originally proposed filtering method for the model. Diagrams of the filtering methods are shown in Figure 4.2. Implementations of the models are based on code provided by the respective au-



(a) VRNN (b) SRNN (c) SVG (d) AVF

Figure 4.2: **Filtering Inference Models for VRNN, SRNN, SVG, and AVF**. Each diagram shows the computational graph for inferring the approximate posterior parameters, $\boldsymbol{\lambda}^q$, at step t. Previously proposed methods rely on hand-crafted architectures of observations, hidden states, and latent variables. AVF is a simple, general filtering procedure that only requires the local inference gradient.

thors of VRNN[1], SRNN[2], and SVG[3]. Accompanying code can be found online at `github.com/joelouismarino/amortized-variational-filtering`.

Setup

Iterative inference models are implemented as specified in Eq. 4.11, encoding the approximate posterior parameters and their gradients at each inference iteration at each step. As in the static setting (Chapter 3), we separately normalize each of the inputs to the inference model using layer normalization (Ba, Kiros, and Hinton, 2016) and use highway gating (Eq. 3.12) for for each output (R. K. Srivastava, Greff, and Schmidhuber, 2015). The generative models that we evaluate contain non-spatial latent variables, thus, we use fully-connected layers to parameterize the inference models. Specifically, we use two-layer fully-connected networks with $1,024$ units per layer, internal highway connections R. K. Srivastava, Greff, and Schmidhuber, 2015, and ELU non-linearities Clevert, Unterthiner, and Hochreiter, 2015. Importantly, minimal effort went into engineering the inference model architectures: across all models and datasets, we utilize the *same* inference model architecture for AVF. Further model and experiment details can be found in Marino, Cvitkovic, and Yue, 2018. Finally, we note that AVF is comparable in runtime to the baseline filtering methods. For example, with our implementation of SRNN on TIMIT, AVF requires $13.1ms$ per time step, whereas the baseline method requires $15.6ms$ per time step. AVF requires an additional decoding per step, but inference is local to the current

[1] `https://github.com/jych/nips2015_vrnn`
[2] `https://github.com/marcofraccaro/srnn`
[3] `https://github.com/edenton/svg`

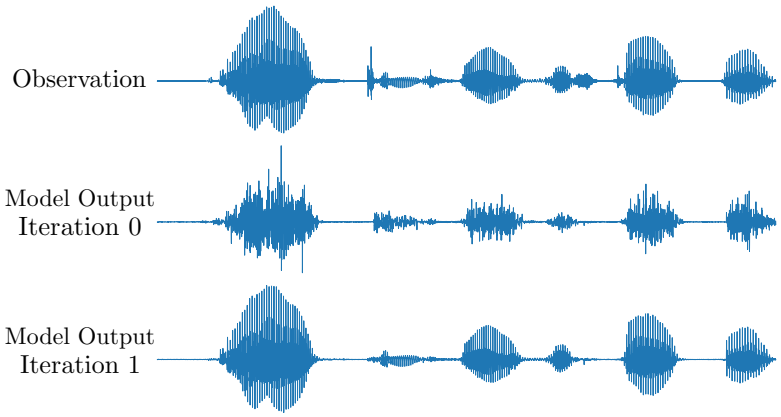


Figure 4.3: **Prediction-Update Visualization**. Test data (top), output predictions (middle), and updated reconstructions (bottom) for TIMIT using SRNN with AVF. Sequences run from left to right. The predictions made by the model already contain the general structure of the data. AVF explicitly updates the approximate posterior from the prior prediction, focusing on inference *corrections* rather than re-estimation.

time step, making backpropagation more efficient.

Speech Modeling

Models For speech modeling, we use VRNN and SRNN, attempting to keep the model architectures consistent with the original implementations. The most notable difference in our implementation occurs in SRNN, where we use an LSTM rather than a GRU as the recurrent module. As in Fraccaro, S. K. Sønderby, et al., 2016, we anneal the KL divergence initially during training. In both models, we use a Gaussian output density. Unlike Chung, Kastner, et al., 2015; Fraccaro, S. K. Sønderby, et al., 2016; Goyal et al., 2017, which evaluate log *densities*, we evaluate and report log *probabilities* by integrating the output density over the data discretization window, as in modeling image pixels.

Data We train and evaluate on TIMIT (Garofolo et al., 1993), which consists of audio recordings of 6, 300 sentences spoken by 630 individuals. As performed in Chung, Kastner, et al., 2015, we sample the audio waveforms at 16 kHz, split the training and validation sets into half second clips, and group each sequence into bins of 200 consecutive samples. Thus, each training and validation sequence consists of 40 model steps. Evaluation is performed on the full duration of each test sequence, averaging roughly 3 seconds.

Music Modeling

Model We model polyphonic music using SRNN. The generative model architecture is the same as in the speech modeling experiments, with changes in the number of layers and units to match Fraccaro, S. K. Sønderby, et al., 2016. To model the binary music notes, we use a Bernoulli output distribution. Again, we anneal the KL divergence initially during training.

Data We use four datasets of polyphonic (MIDI) music (Boulanger-Lewandowski, Bengio, and Vincent, 2012): Piano-midi.de, MuseData, JSB Chorales, and Nottingham. Each dataset contains between 100 and $1,000$ songs, with each song between 100 to $4,000$ steps. For training and validation, we break the sequences into clips of length 25, and we test on the entire test sequences.

Video Modeling

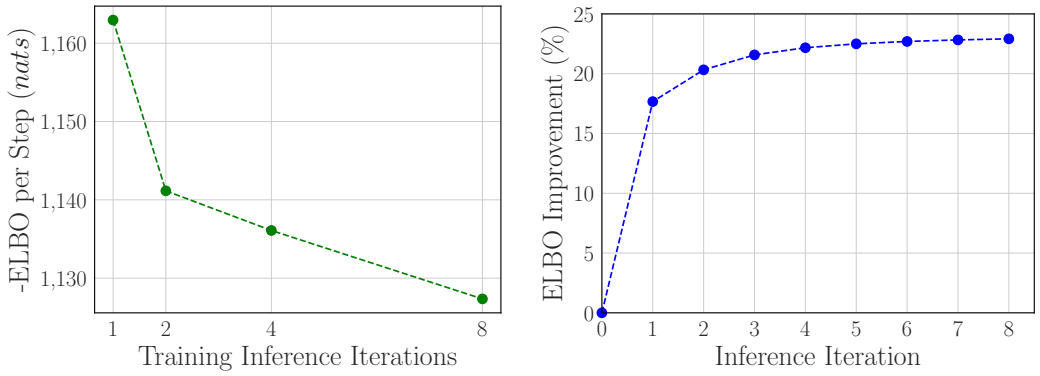
Model Our implementation of SVG differs from the original model in that we evaluate the conditional log-likelihood under a Gaussian output density rather than mean squared output error. All other architecture details are identical to the original model. However, Denton and Fergus, 2018 down-weight the KL-divergence by a factor of 10^{-6} at all steps. We instead remove this factor to use the ELBO during training and evaluation. As to be expected, this results in the model using the latent variables to a lesser extent. We train and evaluate SVG using filtering inference at all steps, rather than predicting multiple steps into the future, as in Denton and Fergus, 2018.

Data We train and evaluate SVG on KTH Actions (Schuldt, Laptev, and Caputo, 2004), which contains 760 train / 768 val / 863 test videos of people performing various actions, each of which is between roughly 50 to 150 frames. Frames are re-sized to 64×64. For training and validation, we split the data into clips of 20 frames.

Results

Additional Inference Iterations

The variational filtering EM algorithm involves inference optimization at each step (Algorithm 2, Line 6). AVF optimizes each approximate posterior through a model that learns to perform iterative updates (Eq. 4.11). Additional inference iterations



(a) (b)

Figure 4.4: **Improvement with Inference Iterations.** Results are shown on the TIMIT validation set using VRNN with AVF. **(a)** Average negative ELBO per step with varying numbers of inference iterations during training. Additional iterations result in improved performance. **(b)** Average relative ELBO improvement from the initial (prior) estimate at each inference iteration for a single model. Empirically, each successive iteration provides diminishing additional improvement.

may lead to further improvement in performance (Marino, Yue, and Mandt, 2018). We explore this aspect on TIMIT using VRNN. In Figure 4.4a, we plot the average negative ELBO per step on validation sequences for models trained with varying numbers of inference iterations. Figure 4.4b shows average relative improvement over the prior estimate for a single model trained with 8 inference iterations. We observe that training with additional inference iterations empirically leads to improved performance (Figure 4.4a), with each iteration providing diminishing improvement during inference (Figure 4.4b). This aspect is distinct from many baseline filtering methods, which directly output the approximate posterior at each step.

We can also directly visualize inference improvement through the model output. Figure 4.3 illustrates example reconstructions over inference iterations, using SRNN on TIMIT. At the initial inference iteration, the approximate posterior is initialized from the prior, resulting in an output prediction. The iterative inference model then uses the approximate posterior gradients to update the estimate, improving the output reconstruction.

Quantitative Comparison

Tables 4.1, 4.2, and 4.3 present quantitative comparisons of average negative filtering ELBO per step (lower is better) between AVF and baseline filtering methods for TIMIT, KTH Actions, and the polyphonic music datasets respectively. On TIMIT,

Table 4.1: **Average negative ELBO per step** (in *nats*) on the TIMIT speech dataset for SRNN and VRNN with the respective originally proposed filtering procedures (baselines) and with AVF.

	TIMIT
VRNN	
baseline	1,082
AVF (1 Iter.)	1,105
AVF (2 Iter.)	**1,071**
SRNN	
baseline	1,026
AVF (1 Iter.)	**1,024**

Table 4.2: **Average negative ELBO per step** (in *nats per dimension*) on the KTH Actions video dataset for SVG with the originally proposed filtering procedure (baseline) and with AVF.

	KTH Actions
SVG	
baseline	3.69
AVF (1 Iter.)	**2.86**

Table 4.3: **Average negative ELBO per step** (in *nats*) on polyphonic music datasets for SRNN with and without AVF. Results from Fraccaro, S. K. Sønderby, et al., 2016 are provided for comparison, however, our model implementation differs in several aspects (see Marino, Cvitkovic, and Yue, 2018).

	Piano-midi.de	MuseData	JSB Chorales	Nottingham
SRNN				
baseline (Fraccaro et al., 2016)	8.20	6.28	4.74	2.94
baseline	8.19	6.27	6.92	3.19
AVF (1 Iter.)	**8.12**	**5.99**	6.97	**3.13**
AVF (5 Iter.)	–	–	**6.77**	–

training with AVF performs comparably to the baseline methods for both VRNN and SRNN. We note that VRNN with AVF using 2 inference iterations resulted in a final test performance of $1,071$ *nats* per step, outperforming the baseline method. Similar results are also observed on each of the polyphonic music datasets. Again, increasing the number of inference iterations to 5 for AVF on JSB Chorales resulted in a final test performance of 6.77 *nats* per step. AVF significantly improves the performance of SVG on KTH Actions. We attribute this, likely, to the absence of the KL down-weighting factor in our training objective as compared with (Denton and Fergus, 2018). This agrees with Kumar et al., 2020, who observe that many recently proposed video models trained with proper log-likelihood objectives tend to perform poorly. The baseline filtering procedure seems to struggle to a greater degree than AVF. From comparing the results above, we see that AVF is a general filtering procedure that performs well across multiple models and datasets, despite using a relatively simple inference model structure. In the cases where single-

step AVF did not outperform the baseline inference procedure, we were able to improve performance by simply increasing the number of inference iterations, with no additional parameter overhead.

4.5 Discussion

This chapter introduced the variational filtering EM algorithm for performing filtering inference in sequential latent variable models. Filtering variational inference can be expressed as a sequence of optimization objectives, linked across steps through previous latent samples. Using iterative amortized inference (Chapter 3) to perform inference optimization, we arrived at an efficient implementation of the algorithm: *amortized variational filtering*. This general filtering method scales to large models and datasets. Numerous methods have been proposed for filtering in deep sequential latent variable models, with each method hand–designed for each model. The variational filtering EM algorithm provides a single framework for analyzing and constructing these methods. Amortized variational filtering is a simple, theoretically-motivated, and general filtering method that we have shown performs on-par with or better than multiple existing state-of-the-art methods across various data domains.

Unlike previous amortized filtering methods, AVF is formulated in terms of iterative optimization at each time step. That is, AVF is initialized at each time step from the prior and only optimizes the relevant objective terms at the current time step. As a result, AVF does not need to re-estimate the latent dynamics, as in previous amortized filtering methods. This prediction-update scheme is reminiscent of classical Bayesian filtering techniques, such as Kalman filtering (Kalman, 1960), using prediction errors to perform exact probabilistic inference in linear-Gaussian models. From this perspective, AVF is thus a modern instantiation of the same negative feedback principles for perception, harnessing the efficiency of amortization, combined with the generality of variational inference.

4.6 Appendix: Filtering ELBO Derivation

This derivation largely follows that of Gemici et al., 2017 and is valid for any filtering approximate posterior. From Eq. 4.3, we have the definition of the ELBO:

$$\mathcal{L} \equiv \mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\log \frac{p_\theta(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T})}{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\right]. \tag{4.12}$$

Plugging in the forms of the joint distribution (Eq. 4.1) and approximate posterior (Eq. 4.4), we can write the term within the expectation as a sum:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\log\left(\prod_{t=1}^{T}\frac{p(\mathbf{x}_t,\mathbf{z}_t|\mathbf{x}_{<t},\mathbf{z}_{<t})}{q(\mathbf{z}_t|\mathbf{x}_{\leq t},\mathbf{z}_{<t})}\right)\right] \tag{4.13}$$

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\sum_{t=1}^{T}\log\frac{p(\mathbf{x}_t,\mathbf{z}_t|\mathbf{x}_{<t},\mathbf{z}_{<t})}{q(\mathbf{z}_t|\mathbf{x}_{\leq t},\mathbf{z}_{<t})}\right] \tag{4.14}$$

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\sum_{t=1}^{T}C_t\right], \tag{4.15}$$

where the term C_t is defined to simplify notation. We then expand the expectation:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\ldots\mathbb{E}_{q(\mathbf{z}_T|\mathbf{x}_{\leq T},\mathbf{z}_{<T})}\left[\sum_{t=1}^{T}C_t\right] \tag{4.16}$$

There are T terms within the sum, but each C_t only depends on the expectations up to time t because we only condition on past and present variables. This allows us to write:

$$\begin{aligned}\mathcal{L} =& \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\left[C_1\right] \\ &+ \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\mathbb{E}_{q(\mathbf{z}_2|\mathbf{x}_{\leq 2},\mathbf{z}_1)}\left[C_2\right] \\ &+ \ldots \\ &+ \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\mathbb{E}_{q(\mathbf{z}_2|\mathbf{x}_{\leq 2},\mathbf{z}_1)}\ldots\mathbb{E}_{q(\mathbf{z}_T|,\mathbf{x}_{\leq T},\mathbf{z}_{<T})}\left[C_T\right]\end{aligned} \tag{4.17}$$

$$\mathcal{L} = \sum_{t=1}^{T}\mathbb{E}_{q(\mathbf{z}_{\leq t}|\mathbf{x}_{\leq t})}\left[C_t\right] \tag{4.18}$$

$$\mathcal{L} = \sum_{t=1}^{T}\mathbb{E}_{\prod_{\tau=1}^{t}q(\mathbf{z}_\tau|\mathbf{x}_{\leq\tau},\mathbf{z}_{<\tau})}\left[C_t\right] \tag{4.19}$$

$$\mathcal{L} = \sum_{t=1}^{T}\mathbb{E}_{\prod_{\tau=1}^{t-1}q(\mathbf{z}_\tau|\mathbf{x}_{\leq\tau},\mathbf{z}_{<\tau})}\left[\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t},\mathbf{z}_{<t})}\left[C_t\right]\right]. \tag{4.20}$$

As in Section 4.2, we define $\mathcal{L}_t$ as

$$\mathcal{L}_t \equiv \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t},\mathbf{z}_{<t})}\left[C_t\right] \tag{4.21}$$

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t},\mathbf{z}_{<t})}\left[\log\frac{p_\theta(\mathbf{x}_t,\mathbf{z}_t|\mathbf{x}_{<t},\mathbf{z}_{<t})}{q(\mathbf{z}_t|\mathbf{x}_{\leq t},\mathbf{z}_{<t})}\right]. \tag{4.22}$$

This allows us to write Eq. 4.20 as

$$\mathcal{L} = \sum_{t=1}^{T}\mathbb{E}_{\prod_{\tau=1}^{t-1}q(\mathbf{z}_\tau|\mathbf{x}_{\leq\tau},\mathbf{z}_{<\tau})}\left[\mathcal{L}_t\right], \tag{4.23}$$

which agrees with Eq. 4.5.

References

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer normalization". In: *arXiv preprint arXiv:1607.06450*.

Babaeizadeh, Mohammad et al. (2018). "Stochastic Variational Video Prediction". In: *International Conference on Learning Representations*.

Bayer, Justin and Christian Osendorfer (2014). "Learning Stochastic Recurrent Networks". In: *NeurIPS 2014 Workshop on Advances in Variational Inference*.

Boulanger-Lewandowski, Nicolas, Yoshua Bengio, and Pascal Vincent (2012). "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription". In: *International Conference on Machine Learning*.

Chung, Junyoung, Caglar Gulcehre, et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555*.

Chung, Junyoung, Kyle Kastner, et al. (2015). "A recurrent latent variable model for sequential data". In: *Advances in neural information processing systems*, pp. 2980–2988.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289*.

Denton, Emily and Rob Fergus (2018). "Stochastic Video Generation with a Learned Prior". In: *International Conference on Machine Learning*, pp. 1182–1191.

Finn, Chelsea, Ian Goodfellow, and Sergey Levine (2016). "Unsupervised learning for physical interaction through video prediction". In: *Advances in Neural Information Processing Systems*.

Fraccaro, Marco, Simon Kamronn, et al. (2017). "A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning". In: *Advances in Neural Information Processing Systems*.

Fraccaro, Marco, Søren Kaae Sønderby, et al. (2016). "Sequential neural models with stochastic layers". In: *Advances in neural information processing systems*, pp. 2199–2207.

Friston, Karl (2005). "A theory of cortical responses". In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 360.1456, pp. 815–836.

Garofolo, J. S. et al. (1993). *DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus*.

Gemici, Mevlana et al. (2017). "Generative Temporal Models with Memory". In: *arXiv preprint arXiv:1702.04649*.

Gershman, Samuel and Noah Goodman (2014). "Amortized inference in probabilistic reasoning". In: *Proceedings of the Cognitive Science Society*. Vol. 36. 36.

Goyal, Anirudh et al. (2017). “Z-Forcing: Training Stochastic Recurrent Networks”. In: *Advances in Neural Information Processing Systems*.

He, Jiawei, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal (2018). “Probabilistic video generation using holistic attribute control”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 452–467. URL: http://openaccess.thecvf.com/content_ECCV_2018/html/Jiawei_He_Probabilistic_Video_Generation_ECCV_2018_paper.html.

Henaff, Mikael, Junbo Zhao, and Yann LeCun (2017). “Prediction Under Uncertainty with Error-Encoding Networks”. In: *arXiv preprint arXiv:1711.04994*.

Hoffman, Matthew D et al. (2013). “Stochastic variational inference”. In: *The Journal of Machine Learning Research* 14.1, pp. 1303–1347.

Hsu, Wei-Ning, Yu Zhang, and James Glass (2017). “Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data”. In: *Advances in Neural Information Processing Systems*.

Johnson, Matthew et al. (2016). “Composing graphical models with neural networks for structured representations and fast inference”. In: *Advances in Neural Information Processing Systems*.

Jordan, Michael I et al. (1998). “An introduction to variational methods for graphical models”. In: *NATO ASI SERIES D BEHAVIOURAL AND SOCIAL SCIENCES* 89, pp. 105–162.

Kalman, Rudolph Emil (1960). “A new approach to linear filtering and prediction problems”. In: *Journal of Basic Engineering* 82.1, pp. 35–45.

Karl, Maximilian et al. (2017). “Deep variational Bayes filters: Unsupervised learning of state space models from raw data”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kingma, Durk P and Max Welling (2014). “Stochastic gradient VB and the variational auto-encoder”. In: *Proceedings of the International Conference on Learning Representations*.

Kumar, Manoj et al. (2020). “VideoFlow: A Flow-Based Generative Model for Video”. In: *International Conference on Learning Representations*.

Le, Tuan Anh et al. (2018). “Auto-Encoding Sequential Monte Carlo”. In: *International Conference on Learning Representations*.

Li, Yingzhen and Stephan Mandt (2018). “A Deep Generative Model for Disentangled Representations of Sequential Data”. In: *International Conference on Machine Learning*.

Lotter, William, Gabriel Kreiman, and David Cox (2017). “Deep predictive coding networks for video prediction and unsupervised learning”. In: *International Conference on Learning Representations*.

Maddison, Chris J et al. (2017). “Filtering Variational Objectives”. In: *Advances in Neural Information Processing Systems*.

Marino, Joseph, Milan Cvitkovic, and Yisong Yue (2018). “A general method for amortizing variational filtering”. In: *Advances in Neural Information Processing Systems*, pp. 7857–7868. URL: http://papers.nips.cc/paper/8011-a-general-method-for-amortizing-variational-filtering.

Marino, Joseph, Yisong Yue, and Stephan Mandt (2018). “Iterative Amortized Inference”. In: *International Conference on Machine Learning*, pp. 3403–3412. URL: http://proceedings.mlr.press/v80/marino18a.html.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

Naesseth, Christian et al. (2018). “Variational Sequential Monte Carlo”. In: *International Conference on Artificial Intelligence and Statistics*.

Neal, Radford M and Geoffrey E Hinton (1998). “A view of the EM algorithm that justifies incremental, sparse, and other variants”. In: *Learning in graphical models*. Springer, pp. 355–368.

Ranganath, Rajesh, Sean Gerrish, and David Blei (2014). “Black box variational inference”. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 814–822.

Rao, Rajesh PN and Dana H Ballard (1999). “Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects.” In: *Nature neuroscience* 2.1.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backpropagation and approximate inference in deep generative models”. In: *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286.

Särkkä, Simo (2013). *Bayesian filtering and smoothing*. Vol. 3. Cambridge University Press.

Schuldt, Christian, Ivan Laptev, and Barbara Caputo (2004). “Recognizing human actions: a local SVM approach”. In: *International Conference on Pattern Recognition*.

Sønderby, Casper Kaae et al. (2016). “Ladder variational autoencoders”. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 3738–3746.

Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov (2015). “Unsupervised learning of video representations using lstms”. In: *International conference on machine learning*, pp. 843–852.

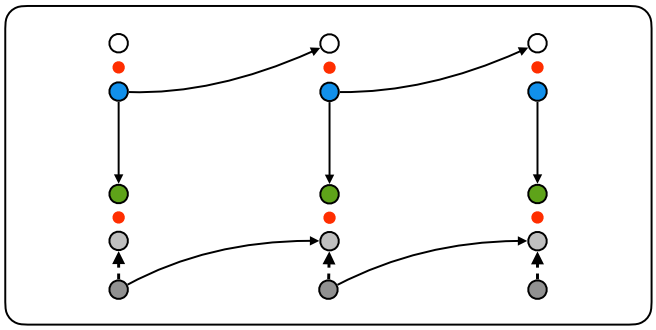
Srivastava, Rupesh K, Klaus Greff, and Jürgen Schmidhuber (2015). “Training very deep networks”. In: *Advances in neural information processing systems (NIPS)*, pp. 2377–2385.

Walker, Jacob et al. (2016). “An uncertain future: Forecasting from static images using variational autoencoders”. In: *European Conference on Computer Vision*.

Xue, Tianfan et al. (2016). “Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks”. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 91–99.

Chapter 5

IMPROVING SEQUENTIAL LATENT VARIABLE MODELS WITH AUTOREGRESSIVE FLOWS



learned feedforward perception

Marino, Joseph, Lei Chen, Jiawei He, and Stephan Mandt (2020). "Improving Sequential Latent Variable Models with Autoregressive Flows". In: *Symposium on Advances in Approximate Bayesian Inference*, pp. 1–16. URL: `http://proceedings.mlr.press/v118/marino20a.html`.

Yang, Ruihan, Yibo Yang, Joseph Marino, and Stephan Mandt (2021). "Hierarchical Autoregressive Modeling for Neural Video Compression". In: *International Conference on Learning Representations*. URL: `https://openreview.net/pdf?id=TK_6nNb_C7q`.

5.1 Introduction

The previous chapters explored the use of learned negative feedback to perform inference in deep latent variable models. These methods attempt to reduce errors, via variational inference optimization, *as they arise*. As we saw in Chapter 4, inference optimization can be augmented with prior predictions, providing a useful initialization that *preemptively* reduces errors. However, these priors occur within the latent space, which must be transformed into data-level predictions at each time step. That is, the entire raw data input is constantly being newly predicted.

This chapter explores a complementary technique, incorporating predictions directly at the data level, via the conditional likelihood, thereby completely bypassing inference

optimization. In this way, low-level predictions remove temporal redundancy from the data, with latent variables, and therefore inference optimization, only reserved for any remaining errors. We formulate this process as an autoregressive flow across time, yielding a form of learned feedforward perception. Feedforward processing can be seen as learning a *frame of reference* to assist in modeling the data by pre-processing the input at each time step. This not only simplifies downstream dynamics modeling, but, by modeling errors rather than the data itself, also has the effect of improving generalization to unseen sequences. We demonstrate these improvements across multiple benchmark video datasets.

5.2 Method

We start by motivating the use of autoregressive flows to reduce temporal dependencies, thereby simplifying dynamics. We then show how this simple technique can be incorporated within sequential latent variable models.

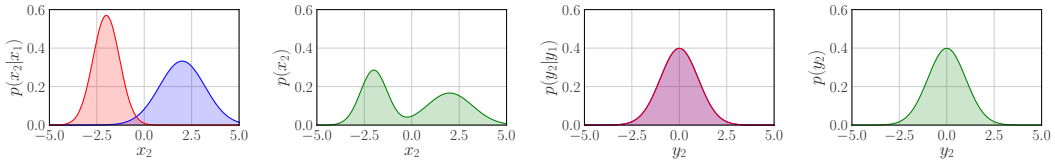
Motivation: Temporal Redundancy Reduction

Normalizing flows, while often utilized for density estimation, originated from data pre-processing techniques (Friedman, 1987; Hyvärinen and Oja, 2000; S. S. Chen and Gopinath, 2001), which remove dependencies between dimensions, i.e., *redundancy reduction* (Barlow et al., 1961). Removing dependencies simplifies the resulting probability distribution by restricting variation to individual dimensions, generally simplifying downstream tasks (Laparra, Camps-Valls, and Malo, 2011). Normalizing flows improve upon these procedures using flexible, non-linear functions (Deco and Brauer, 1995; Dinh, Krueger, and Bengio, 2015). While flows have been used for spatial decorrelation (Agrawal and Dukkipati, 2016; Winkler et al., 2019) and with other models (Huang et al., 2017), this capability remains under-explored.

Data sequences contain dependencies in time, for example, in the redundancy of video pixels (Figure 5.3), which are often highly predictable. These dependencies define the *dynamics* of the data, with the degree of dependence quantified by the multi-information,

$$\mathcal{I}(\mathbf{x}_{1:T}) = \sum_t \mathcal{H}(\mathbf{x}_t) - \mathcal{H}(\mathbf{x}_{1:T}), \tag{5.1}$$

where $\mathcal{H}$ denotes entropy. Normalizing flows are capable of reducing redundancy, arriving at a new sequence, $\mathbf{y}_{1:T}$, with $\mathcal{I}(\mathbf{y}_{1:T}) \leq \mathcal{I}(\mathbf{x}_{1:T})$, thereby reducing temporal dependencies and simplifying dynamics. Thus, rather than fit the data distribution directly, we can first simplify the dynamics by pre-processing sequences with a



(a) (b) (c) (d)

Figure 5.1: **Redundancy Reduction**. **(a)** Conditional densities for $p(x_2|x_1)$. **(b)** The marginal, $p(x_2)$ differs from the conditional densities, thus, $\mathcal{I}(x_1; x_2) > 0$. **(c)** In the normalized space of y, the corresponding densities $p(y_2|y_1)$ are identical. **(d)** The marginal $p(y_2)$ is identical to the conditionals, so $\mathcal{I}(y_1; y_2) = 0$. Thus, in this case, a conditional affine transform removed the dependencies.

normalizing flow, then fit the resulting sequence. Through training, the flow will attempt to remove redundancies to meet the modeling capacity of the higher-level dynamics model, $p_\theta(\mathbf{y}_{1:T})$.

Example To visualize this procedure for an affine autoregressive flow, consider a one-dimensional input over two time steps, x_1 and x_2. For each value of x_1, there is a conditional density, $p(x_2|x_1)$. Assume that these densities take one of two forms, which are identical but shifted and scaled, shown in Figure 5.1. Transforming these densities through their conditional means, $\mu_2 = \mathbb{E}[x_2|x_1]$, and standard deviations, $\sigma_2 = \mathbb{E}\left[(x_2 - \mu_2)^2|x_1\right]^{1/2}$, creates a normalized space, $y_2 = (x_2 - \mu_2)/\sigma_2$, where the conditional densities are identical. In this space, the multi-information is

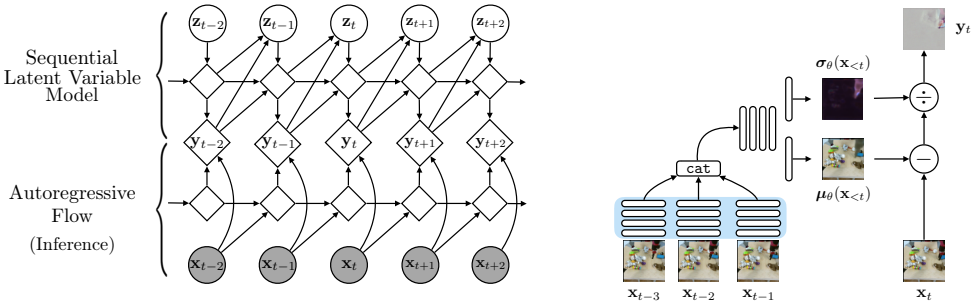
$$\mathcal{I}(y_1; y_2) = \mathbb{E}_{p(y_1,y_2)}\left[\log p(y_2|y_1) - \log p(y_2)\right] = 0,$$

whereas $\mathcal{I}(x_1; x_2) > 0$. Indeed, if $p(x_t|x_{<t})$ is linear-Gaussian, inverting an affine autoregressive flow exactly corresponds to Cholesky whitening (Pourahmadi, 2011; Kingma, Salimans, et al., 2016), removing all linear dependencies.

In the example above, μ_2 and σ_2 act as a *frame of reference* for estimating x_2. More generally, in the special case where $\boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) = \mathbf{x}_{t-1}$ and $\boldsymbol{\sigma}(\mathbf{x}_{<t}) = \mathbf{1}$, we recover $\mathbf{y}_t = \mathbf{x}_t - \mathbf{x}_{t-1} = \Delta\mathbf{x}_t$. Modeling finite differences (or *generalized coordinates* (Friston, 2008)) is a well-established technique, (see, e.g., (Chua et al., 2018; Kumar et al., 2020)), which is generalized by affine autoregressive flows.

Modeling Dynamics with Autoregressive Flows

We now discuss utilizing autoregressive flows to improve sequence modeling, highlighting use cases for modeling dynamics in the data and latent spaces.



(a) (b)

Figure 5.2: **Model Diagrams**. **(a)** An autoregressive flow pre-processes a data sequence, $\mathbf{x}_{1:T}$, to produce a new sequence, $\mathbf{y}_{1:T}$, with reduced temporal dependencies. This simplifies dynamics modeling for a higher-level sequential latent variable model, $p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T})$. Empty diamond nodes represent deterministic dependencies, not recurrent states. **(b)** Diagram of the autoregressive flow architecture. Blank white rectangles represent convolutional layers (see Appendix). The three stacks of convolutional layers within the blue region are shared. `cat` denotes channel-wise concatenation.

Data Dynamics

The form of an affine autoregressive flow across sequences is given by

$$\mathbf{x}_t = \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) + \boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \odot \mathbf{y}_t, \tag{5.2}$$

and its inverse

$$\mathbf{y}_t = \frac{\mathbf{x}_t - \boldsymbol{\mu}_\theta(\mathbf{x}_{<t})}{\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})}, \tag{5.3}$$

which, again, are equivalent to a Gaussian autoregressive model when $\mathbf{y}_t \sim \mathcal{N}(\mathbf{y}_t; \mathbf{0}, \mathbf{I})$. We can stack hierarchical chains of flows to improve the model capacity. Denoting the shift and scale functions at the m^{th} transform as $\boldsymbol{\mu}_\theta^m(\cdot)$ and $\boldsymbol{\sigma}_\theta^m(\cdot)$ respectively, we then calculate $\mathbf{y}^m$ using the inverse transform:

$$\mathbf{y}_t^m = \frac{\mathbf{y}_t^{m-1} - \boldsymbol{\mu}_\theta^m(\mathbf{y}_{<t}^{m-1})}{\boldsymbol{\sigma}_\theta^m(\mathbf{y}_{<t}^{m-1})}. \tag{5.4}$$

After the final (M^{th}) transform, we can choose the form of the base distribution, $p_\theta(\mathbf{y}_{1:T}^M)$, e.g., Gaussian. While we could attempt to model $\mathbf{x}_{1:T}$ completely using stacked autoregressive flows, these models are limited to affine element-wise transforms that maintain the data dimensionality. Due to this limited capacity, purely flow-based models often require many transforms to be effective (Kingma and Dhariwal, 2018).

Instead, we can model the base distribution using an expressive sequential latent variable model (SLVM), or, equivalently, we can augment the conditional likelihood of a SLVM using autoregressive flows (Fig. 5.2a). Following the motivation from Section 5.2, the flow can remove temporal dependencies, simplifying the modeling task for the SLVM. With a single flow, the joint probability is

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) \left| \det \left(\frac{\partial \mathbf{x}_{1:T}}{\partial \mathbf{y}_{1:T}} \right) \right|^{-1}, \tag{5.5}$$

where the SLVM distribution is given by

$$p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{y}_t | \mathbf{y}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t | \mathbf{y}_{<t}, \mathbf{z}_{<t}). \tag{5.6}$$

If the SLVM is itself a flow-based model, we can use maximum log-likelihood training. If not, we can resort to variational inference (Chung et al., 2015; Fraccaro et al., 2016). We derive and discuss this procedure in Section 5.5.

Latent Dynamics

We can also consider simplifying latent dynamics modeling using autoregressive flows in the latent prior. This is relevant in hierarchical SLVMs, such as VideoFlow (Kumar et al., 2020), where each latent variable is modeled as a function of past and higher-level latent variables. Using $\mathbf{z}_t^{(\ell)}$ to denote the latent variable at the ℓ^{th} level at time t, we can parameterize the prior as

$$p_\theta(\mathbf{z}_t^{(\ell)} | \mathbf{z}_{<t}^{(\ell)}, \mathbf{z}_t^{(>\ell)}) = p_\theta(\mathbf{u}_t^{(\ell)} | \mathbf{u}_{<t}^{(\ell)}, \mathbf{z}_t^{(>\ell)}) \left| \det \left(\frac{\partial \mathbf{z}_t^{(\ell)}}{\partial \mathbf{u}_t^{(\ell)}} \right) \right|^{-1}, \tag{5.7}$$

where we convert $\mathbf{z}_t^{(\ell)}$ into $\mathbf{u}_t^{(\ell)}$ using the inverse transform

$$\mathbf{u}_t^{(\ell)} = (\mathbf{z}_t^{(\ell)} - \boldsymbol{\alpha}_\theta(\mathbf{z}_{<t}^{(\ell)}))/\boldsymbol{\beta}_\theta(\mathbf{z}_{<t}^{(\ell)}).$$

As noted previously, VideoFlow uses a special case of this procedure, setting $\boldsymbol{\alpha}_\theta(\mathbf{z}_{<t}^{(\ell)}) = \mathbf{z}_{t-1}^{(\ell)}$ and $\boldsymbol{\beta}_\theta(\mathbf{z}_{<t}^{(\ell)}) = \mathbf{1}$. Generalizing this procedure further simplifies dynamics throughout the model.

5.3 Experiments

We demonstrate and evaluate the proposed technique on three benchmark video datasets: Moving MNIST (Srivastava, Mansimov, and Salakhudinov, 2015), KTH Actions (Schuldt, Laptev, and Caputo, 2004), and BAIR Robot Pushing (Ebert

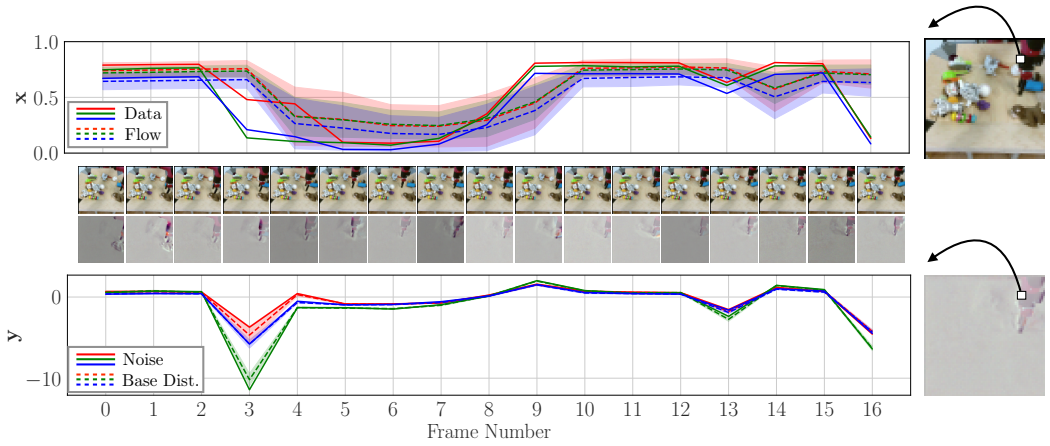


Figure 5.3: **Sequence Modeling with Autoregressive Flows**. **Top**: Pixel values (solid) for a particular pixel location in a video sequence. An autoregressive flow models the pixel sequence using an affine shift (dashed) and scale (shaded), acting as a frame of reference. **Middle**: Frames of the data sequence (top) and the resulting "noise" (bottom) from applying the shift and scale. The redundant, static background has been largely removed. **Bottom**: The noise values (solid) are modeled using a base distribution (dashed and shaded) provided by a higher-level model. By removing temporal redundancy from the data sequence, the autoregressive flow simplifies dynamics modeling.

et al., 2017). Experimental setups are first described, followed by a set of empirical analyses. Further details and additional results can be found in Marino, L. Chen, et al., 2020.

Experimental Setup

For data space modeling, we compare four model classes: **1)** standalone affine autoregressive flows with one (1-AF) and **2)** two (2-AF) transforms, **3)** a sequential latent variable model (SLVM), and **4)** SLVM with flow-based pre-processing (SLVM + AF). As we are not proposing a specific architecture, but rather a general modeling technique, the SLVM architecture is representative of recurrent convolutional video models with a single latent level (Denton and Fergus, 2018; Ha and Schmidhuber, 2018; Hafner et al., 2019). Flows are implemented with convolutional networks, taking in a fixed window of previous frames (Fig. 5.2b). These models allow us to evaluate the benefits of temporal pre-processing (SLVM vs. SLVM + AF) and the benefits of more expressive higher-level dynamics models (2-AF vs. SLVM + 1-AF).

To evaluate latent dynamics modeling with autoregressive flows, we use the

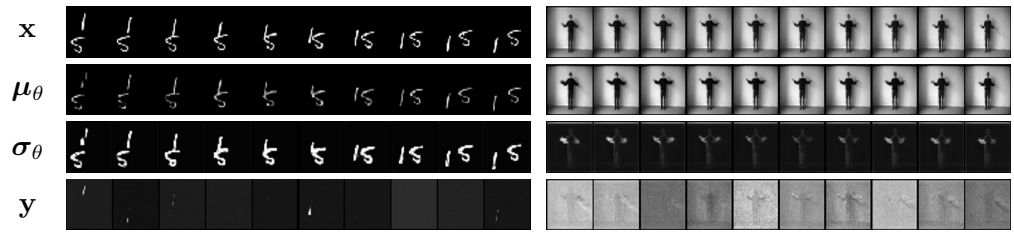


Figure 5.4: **Flow Visualization** for SLVM + 1-AF on Moving MNIST (left) and KTH Actions (right).

`tensor2tensor` library (Vaswani et al., 2018) to compare **1)** VideoFlow[1] and **2)** the same model with affine autoregressive flow latent dynamics (VideoFlow + AF). VideoFlow is significantly larger (3× more parameters) than the one-level SLVM, allowing us to evaluate whether autoregressive flows are beneficial in this high-capacity regime.

To enable a fairer comparison in our experiments, models with autoregressive flow dynamics have comparable or fewer parameters than baseline counterparts. We note that autoregressive dynamics adds only a constant computational cost per time-step, and this computation can be parallelized for training and evaluation. Full architecture, training, and analysis details can be found in Marino, L. Chen, et al., 2020. Finally, as noted by Kumar et al. (Kumar et al., 2020), many previous works do not train SLVMs with proper log-likelihood objectives. Our SLVM results are consistent with previously reported log-likelihood values (Marino, Cvitkovic, and Yue, 2018) for the Stochastic Video Generation model (Denton and Fergus, 2018) trained with a log-likelihood bound objective.

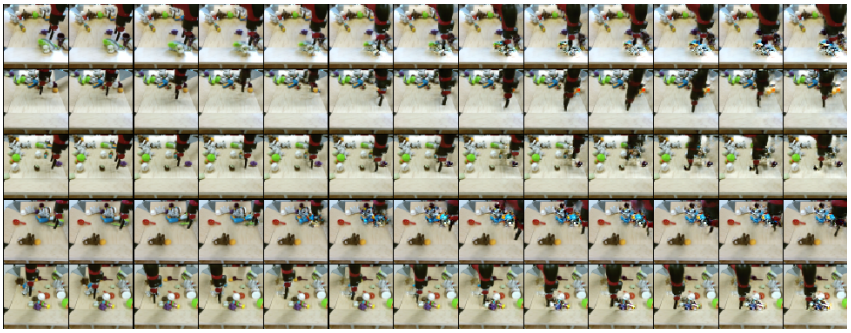
Analyses

Visualization In Figure 5.3, we visualize the pre-processing procedure for SLVM + 1-AF on BAIR Robot Pushing. The plots show the RGB values for a pixel before (top) and after (bottom) the transform. The noise sequence is nearly zero throughout, despite large changes in the pixel value. We also see that the noise sequence (center, lower) is invariant to the static background, capturing the moving robotic arm. At some time steps (e.g., fourth frame), the autoregressive flow incorrectly predicts the next frame, however, the higher-level SLVM compensates for this prediction error.

We also visualize each component of the flow. Figure 5.2b illustrates this for SLVM

[1]We used a smaller version of the original model architecture, with half of the flow depth, due to GPU memory constraints.

(a) VideoFlow



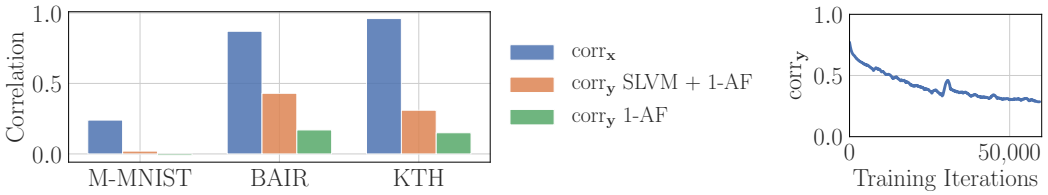
(b) VideoFlow + AF

Figure 5.5: **Improved Generated Samples**. Random samples generated from (**a**) VideoFlow and (**b**) VideoFlow + AF, each conditioned on the first 3 frames. Using AF produces more coherent samples. The robot arm blurs for VideoFlow in samples 1 and 4 (red), but does not blur for VideoFlow + AF.

+ 1-AF on an input from BAIR Robot Pushing. We see that $\boldsymbol{\mu}_\theta$ captures the static background, while $\boldsymbol{\sigma}_\theta$ highlights regions of uncertainty. In Figure 5.4 and the Appendix, we present visualizations on full sequences, where we see that different models remove varying degrees of temporal structure.

Temporal Redundancy Reduction To quantify temporal redundancy reduction, we estimate the empirical correlation (linear dependence) between frames, denoted as corr, for the data and noise variables. This is an average normalized version of the *auto-covariance* of each signal with a time delay of 1 time step. Specifically, we estimate the temporal correlation as

$$\text{corr}_{\mathbf{x}} \equiv \frac{1}{HWC} \cdot \sum_{i,j,k}^{H,W,C} \mathbb{E}_{x_t^{(i,j,k)}, x_{t+1}^{(i,j,k)} \sim \mathcal{D}} \left[\xi_{t,t+1}(i,j,k) \right], \tag{5.8}$$



(a) (b)

Figure 5.6: **Decreased Temporal Correlation**. **(a)** Affine autoregressive flows result in sequences, $\mathbf{y}_{1:T}$, with decreased temporal correlation, $\text{corr}_\mathbf{y}$, as compared with that of the original data, $\text{corr}_\mathbf{x}$. **(b)** For SLVM + 1-AF, $\text{corr}_\mathbf{y}$ decreases during training on KTH Actions.

where the term inside the expectation is

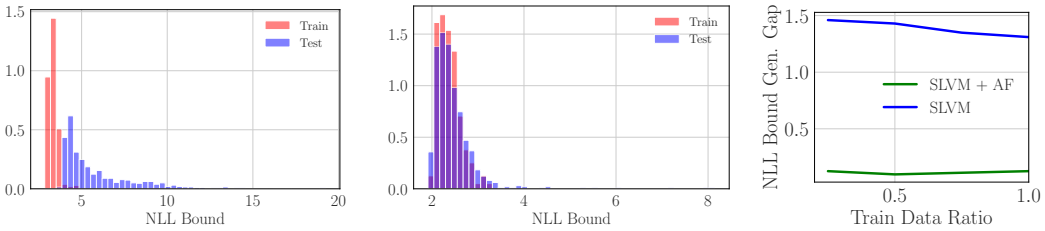
$$\xi_{t,t+1}(i,j,k) \equiv \frac{(x_t^{(i,j,k)} - \mu^{(i,j,k)})(x_{t+1}^{(i,j,k)} - \mu^{(i,j,k)})}{\left(\sigma^{(i,j,k)}\right)^2}. \tag{5.9}$$

Here, $x_t^{(i,j,k)}$ denotes the image at location (i, j) and channel k at time t, $\mu^{(i,j,k)}$ is the empirical mean of this dimension, and $\sigma^{(i,j,k)}$ is the empirical standard deviation. H, W, and C respectively denote the height, width, and number of channels of the observations, and $\mathcal{D}$ denotes the dataset. We define an analogous expression for $\mathbf{y}$, denoted as $\text{corr}_\mathbf{y}$. We evaluate $\text{corr}_\mathbf{x}$ and $\text{corr}_\mathbf{y}$ for SLVM + 1-AF and 1-AF, with results shown in Figure 5.6a. In Figure 5.6b, we plot $\text{corr}_\mathbf{y}$ for SLVM + 1-AF during training on KTH Actions. Flows decrease temporal correlation, and base distributions without temporal structure (1-AF) yield comparatively more decorrelation. Temporal redundancy is progressively removed throughout training. However, due to the limited capacity of the flows, they are not capable of completely removing temporal correlations on the more complex datasets, necessitating the use of higher-level dynamics models.

Performance Comparison Table 5.1 reports average negative log-likelihood results. Standalone flow-based models perform surprisingly well. Increasing flow depth from AF-1 to AF-2 generally results in improvement. SLVM + 1-AF outperforms the baseline SLVM despite having *fewer* parameters. Incorporating autoregressive flows into VideoFlow results in a modest but noticeable improvement, demonstrating that removing spatial dependencies, through VideoFlow, and temporal dependencies, through autoregressive flows, are complementary techniques.

Table 5.1: **Quantitative Comparison.** Average test and (train) negative log-likelihood in *nats per dimension* for Moving MNIST, BAIR Robot Pushing, and KTH Actions. Lower values are better.

	M-MNIST	BAIR	KTH
1-AF	2.15 (2.06)	3.05 (2.98)	3.34 (2.95)
2-AF	2.13 (2.04)	2.90 (2.76)	3.35 (2.95)
SLVM	$\leq$ 1.92 ($\leq$ 1.93)	$\leq$ 3.57 ($\leq$ 3.46)	$\leq$ 4.63 ($\leq$ 3.05)
SLVM + 1-AF	$\leq$ **1.86** ($\leq$ 1.85)	$\leq$ **2.35** ($\leq$ 2.31)	$\leq$ **2.39** ($\leq$ 2.21)
VideoFlow	–	1.53 (1.50)	–
VideoFlow + AF	–	**1.50** (1.49)	–



(a) SLVM (b) SLVM + 1-AF (c)

Figure 5.7: **Improved Generalization**. The low-level reference frame improves generalization to unseen sequences. Train and test negative log-likelihood bound histograms for **(a)** SLVM and **(b)** SLVM + 1-AF on KTH Actions. **(c)** The generalization gap for SLVM + 1-AF remains small for varying amounts of KTH training data, while it becomes worse in the low-data regime for SLVM.

Improved Samples The quantitative improvement over VideoFlow is less dramatic, as this is already a high-capacity model. However, qualitatively, we observe that incorporating autoregressive flow dynamics improves sample quality (Figure 5.5). In these randomly selected samples, the robot arm occasionally becomes blurry for VideoFlow (red boxes) but remains clear for VideoFlow + AF.

Improved Generalization Our temporal normalization technique also improves generalization to unseen examples, a key benefit of normalization schemes, e.g., batch norm (Ioffe and Szegedy, 2015). Intuitively, higher-level dynamics are often preserved, whereas lower-level appearance is not. This is apparent on KTH Actions, which contains a substantial degree of train-test mismatch (different identities and activities). NLL histograms on KTH are shown in Figure 5.7, with greater overlap for SLVM + 1-AF. We also train SLVM and SLVM + 1-AF on subsets of the KTH Actions dataset. In Figure 5.7c, we see that autoregressive flows enable generalization

even in the low-data regime, whereas SLVM becomes worse.

5.4 Discussion

This chapter introduced a technique for improving sequence modeling using autoregressive flows, yielding a learned form of feedforward processing. Learning a frame of reference, parameterized by autoregressive transforms, reduces temporal redundancy in input sequences, thereby simplifying downstream dynamics estimation. Thus, rather than expanding the model, we can simplify the input to meet the capacity of the model. We have analyzed and empirically shown how autoregressive pre-processing in both the data and latent spaces can improve sequence modeling performance and lead to improved sample quality and generalization.

This approach is distinct from previous works with normalizing flows on sequences, yet contains connections to classical modeling and compression. Indeed, video compression schemes use ("inter-frame") predictions as a frame of reference to remove temporal redundancy, enabling more efficient encoding (Oliver, 1952; Wiegand et al., 2003). Inspired by these techniques, follow-up work by Yang et al., 2021 applied sequential autoregressive flows to high-resolution video compression, demonstrating that this not only generalizes existing methods, but also yields improved compression performance.

5.5 Appendix: ELBO Derivation

Consider the model defined in Section 3.3, with the conditional likelihood parameterized with autoregressive flows. That is, we parameterize

$$\mathbf{x}_t = \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) + \boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \odot \mathbf{y}_t, \tag{5.10}$$

yielding

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) = p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) \left|\det \left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|^{-1}. \tag{5.11}$$

The joint distribution over all time steps is then given as

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) \tag{5.12}$$

$$= \prod_{t=1}^{T} p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) \left|\det \left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|^{-1} p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}). \tag{5.13}$$

To perform variational inference, we consider a filtering approximate posterior of the form

$$q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}). \tag{5.14}$$

We can then plug these expressions into the evidence lower bound:

$$\mathcal{L} \equiv \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\log p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) - \log q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})\right] \tag{5.15}$$

$$= \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\log\left(\prod_{t=1}^{T} p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t})\left|\det\left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|^{-1} p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})\right)\right.$$
$$\left. - \log\left(\prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})\right)\right] \tag{5.16}$$

$$= \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\sum_{t=1}^{T} \log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) - \log\frac{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})} - \log\left|\det\left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|\right]. \tag{5.17}$$

Finally, in the filtering setting, we can rewrite the expectation, bringing it inside of the sum (see Gemici et al., 2017; Marino, Cvitkovic, and Yue, 2018):

$$\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{\leq t}|\mathbf{x}_{\leq t})}\left[\log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) - \log\frac{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})} - \log\left|\det\left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|\right]. \tag{5.18}$$

Because there exists a one-to-one mapping between $\mathbf{x}_{1:T}$ and $\mathbf{y}_{1:T}$, we can equivalently condition the approximate posterior and the prior on $\mathbf{y}$, i.e.

$$\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{\leq t}|\mathbf{y}_{\leq t})}\left[\log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) - \log\frac{q(\mathbf{z}_t|\mathbf{y}_{\leq t}, \mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})} - \log\left|\det\left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|\right]. \tag{5.19}$$

References

Agrawal, Siddharth and Ambedkar Dukkipati (2016). "Deep Variational Inference Without Pixel-Wise Reconstruction". In: *arXiv preprint arXiv:1611.05209*.

Barlow, Horace B et al. (1961). "Possible principles underlying the transformation of sensory messages". In: *Sensory communication* 1, pp. 217–234.

Chen, Scott Saobing and Ramesh A Gopinath (2001). "Gaussianization". In: *Advances in neural information processing systems*, pp. 423–429.

Chua, Kurtland et al. (2018). "Deep reinforcement learning in a handful of trials using probabilistic dynamics models". In: *Advances in Neural Information Processing Systems*, pp. 4754–4765.

Chung, Junyoung et al. (2015). “A recurrent latent variable model for sequential data”. In: *Advances in neural information processing systems*, pp. 2980–2988.

Deco, Gustavo and Wilfried Brauer (1995). “Higher order statistical decorrelation without information loss”. In: *Advances in Neural Information Processing Systems*, pp. 247–254.

Denton, Emily and Rob Fergus (2018). “Stochastic Video Generation with a Learned Prior”. In: *International Conference on Machine Learning*, pp. 1182–1191.

Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). “Nice: Non-linear independent components estimation”. In: *International Conference on Learning Representations*.

Ebert, Frederik et al. (2017). “Self-Supervised Visual Planning with Temporal Skip Connections”. In: *Conference on Robot Learning*.

Fraccaro, Marco et al. (2016). “Sequential neural models with stochastic layers”. In: *Advances in neural information processing systems*, pp. 2199–2207.

Friedman, Jerome H (1987). “Exploratory projection pursuit”. In: *Journal of the American statistical association* 82.397, pp. 249–266.

Friston, Karl (2008). “Hierarchical models in the brain”. In: *PLoS computational biology* 4.11, e1000211.

Gemici, Mevlana et al. (2017). “Generative Temporal Models with Memory”. In: *arXiv preprint arXiv:1702.04649*.

Ha, David and Jürgen Schmidhuber (2018). “Recurrent world models facilitate policy evolution”. In: *Advances in Neural Information Processing Systems*, pp. 2450–2462.

Hafner, Danijar et al. (2019). “Learning Latent Dynamics for Planning from Pixels”. In: *International Conference on Machine Learning*, pp. 2555–2565.

Huang, Chin-Wei et al. (2017). “Learnable explicit density for continuous latent space and variational inference”. In: *arXiv preprint arXiv:1710.02248*.

Hyvärinen, Aapo and Erkki Oja (2000). “Independent component analysis: algorithms and applications”. In: *Neural networks* 13.4-5, pp. 411–430.

Ioffe, Sergey and Christian Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *International Conference on Machine Learning*, pp. 448–456.

Kingma, Durk P and Prafulla Dhariwal (2018). “Glow: Generative flow with invertible 1x1 convolutions”. In: *Advances in Neural Information Processing Systems*, pp. 10215–10224.

Kingma, Durk P, Tim Salimans, et al. (2016). “Improved variational inference with inverse autoregressive flow”. In: *Advances in neural information processing systems*, pp. 4743–4751.

Kumar, Manoj et al. (2020). "VideoFlow: A Flow-Based Generative Model for Video". In: *International Conference on Learning Representations*.

Laparra, Valero, Gustavo Camps-Valls, and Jesús Malo (2011). "Iterative gaussianization: from ICA to random rotations". In: *IEEE transactions on neural networks* 22.4, pp. 537–549.

Marino, Joseph, Lei Chen, Jiawei He, and Stephan Mandt (2020). "Improving Sequential Latent Variable Models with Autoregressive Flows". In: *Symposium on Advances in Approximate Bayesian Inference*, pp. 1–16. URL: http://proceedings.mlr.press/v118/marino20a.html.

Marino, Joseph, Milan Cvitkovic, and Yisong Yue (2018). "A general method for amortizing variational filtering". In: *Advances in Neural Information Processing Systems*, pp. 7857–7868. URL: http://papers.nips.cc/paper/8011-a-general-method-for-amortizing-variational-filtering.

Oliver, BM (1952). "Efficient coding". In: *The Bell System Technical Journal* 31.4, pp. 724–750.

Pourahmadi, Mohsen (2011). "Covariance estimation: The GLM and regularization perspectives". In: *Statistical Science*, pp. 369–387.

Schuldt, Christian, Ivan Laptev, and Barbara Caputo (2004). "Recognizing human actions: a local SVM approach". In: *International Conference on Pattern Recognition*.

Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov (2015). "Unsupervised learning of video representations using lstms". In: *International conference on machine learning*, pp. 843–852.

Vaswani, Ashish et al. (2018). "Tensor2Tensor for Neural Machine Translation". In: *CoRR* abs/1803.07416. URL: http://arxiv.org/abs/1803.07416.

Wiegand, Thomas et al. (2003). "Overview of the H. 264/AVC video coding standard". In: *IEEE Transactions on circuits and systems for video technology* 13.7, pp. 560–576.

Winkler, Christina et al. (2019). "Learning Likelihoods with Conditional Normalizing Flows". In: *arXiv preprint arXiv:1912.00042*.

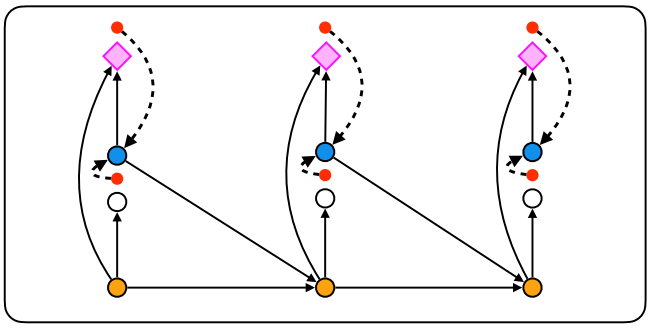
Yang, Ruihan, Yibo Yang, Joseph Marino, and Stephan Mandt (2021). "Hierarchical Autoregressive Modeling for Neural Video Compression". In: *International Conference on Learning Representations*. URL: https://openreview.net/pdf?id=TK_6nNb_C7q.

Part III

Control

Chapter 6

ITERATIVE AMORTIZED POLICY OPTIMIZATION



learned negative feedback control

Marino, Joseph, Alexandre Piché, and Yisong Yue (2019). "On the Design of Variational RL Algorithms". In: *NeurIPS Workshop on Deep Reinforcement Learning*. URL: https://drive.google.com/file/d/10hB0AS_naGNgSNG8p1kqElnc9HRmMYde/view?usp=drivesdk.

Marino, Joseph, Alexandre Piché, Alessandro Davide Ialongo, and Yisong Yue (2020). "Iterative Amortized Policy Optimization". In: *Preprint*. URL: https://arxiv.org/abs/2010.10670.

6.1 Introduction

Chapters 3 and 4 investigated improved methods for perceptual inference using learned negative feedback. In this chapter, we apply the same approach to control. On the surface, perception and control may appear to be disparate settings; perception, as we have framed it, involves an internal generative model, whereas control is typically formulated with an agent in a pre-defined environment along with a reward (or cost) function. Yet, by specifying the probabilistic computation graph implied by the agent-environment interaction, and with an appropriate mathematical framing, we can apply the techniques of probabilistic graphical models and variational inference to control (Levine, 2018). While the notion of framing control in terms of variational calculus dates back to cybernetics (Wiener, 1948), it is only recently that this connection with probabilistic inference has garnered renewed attention in the control

and reinforcement learning communities (Toussaint and Storkey, 2006; Todorov, 2008; Levine, 2018).

In general, reinforcement learning (RL) algorithms involve policy evaluation and policy optimization (Sutton and Barto, 2018). Given a policy, one can estimate the value for each state or state-action pair following that policy, and given a value estimate, one can improve the policy to maximize the value. This latter procedure, policy optimization, can be challenging in continuous control due to instability and poor asymptotic performance. In deep RL, where policies over continuous actions are often parameterized by deep networks, such issues are typically tackled using regularization from previous policies (Schulman, Levine, et al., 2015; Schulman, Wolski, et al., 2017) or by maximizing policy entropy (V. Mnih, Badia, et al., 2016; Fox, Pakman, and Tishby, 2016). These techniques can be interpreted as variational inference (Levine, 2018), using optimization to infer a policy that yields high expected return while satisfying prior policy constraints. This smooths the optimization landscape, improving stability and performance (Ahmed et al., 2019).

However, one subtlety arises: when used with entropy or KL regularization, policy networks perform *amortized* optimization (Gershman and Goodman, 2014). That is, rather than optimizing the action distribution, e.g., mean and variance, many deep RL algorithms, such as soft actor-critic (SAC) (Haarnoja, Zhou, Abbeel, et al., 2018; Haarnoja, Zhou, Hartikainen, et al., 2018), instead optimize a network to output these parameters, *learning* to optimize the policy. Typically, this is implemented as a direct mapping from states to action distribution parameters. While such *direct* amortization schemes have improved the efficiency of variational inference as "encoder" networks (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014; A. Mnih and Gregor, 2014), they also suffer from several drawbacks: ***1)*** they tend to provide suboptimal estimates (Cremer, Li, and Duvenaud, 2018; Kim et al., 2018; Marino, Yue, and Mandt, 2018), yielding a so-called "amortization gap" in performance (Cremer, Li, and Duvenaud, 2018), ***2)*** they are restricted to a single estimate (Greff et al., 2019), thereby limiting exploration, and ***3)*** they cannot generalize to new objectives, unlike, e.g., gradient-based (Henaff, Whitney, and LeCun, 2017) or gradient-free optimizers (Rubinstein and Kroese, 2013).

Inspired by techniques and improvements from variational inference, we investigate *iterative* amortized policy optimization. Iterative amortization (Chapter 3) uses gradients or errors to iteratively update the parameters of a distribution. Unlike direct amortization, which receives gradients only *after* outputting the distribution, iterative

amortization uses these gradients *online*, thereby learning to iteratively optimize. In generative modeling settings, iterative amortization empirically outperforms direct amortization (Chapters 3 & 4) and can find multiple modes of the optimization landscape (Greff et al., 2019). We evaluate iterative amortized policy optimization on the suite of OpenAI gym MuJoCo environments (Todorov, Erez, and Tassa, 2012; Brockman et al., 2016), with performance improvements over direct amortized policies, as well as more complex flow-based policies. We also demonstrate novel benefits of this amortization technique: improved accuracy, providing multiple policy estimates, and generalizing to new objectives.

6.2 Background

Preliminaries

We consider Markov decision processes (MDPs), where $\mathbf{s}_t \in \mathcal{S}$ and $\mathbf{a}_t \in \mathcal{A}$ are the state and action at time t, resulting in reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$. Environment state transitions are given by $\mathbf{s}_{t+1} \sim p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, and the agent is defined by a parametric distribution, $p_\theta(\mathbf{a}_t|\mathbf{s}_t)$, with parameters θ. The discounted sum of rewards is denoted as $\mathcal{R}(\tau) = \sum_t \gamma^t r_t$, where $\gamma \in (0, 1]$ is the discount factor, and $\tau = (\mathbf{s}_1, \mathbf{a}_1, \dots)$ is a trajectory. The distribution over trajectories is

$$p(\tau) = \rho(\mathbf{s}_1) \prod_{t=1}^{T} p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) p_\theta(\mathbf{a}_t|\mathbf{s}_t), \tag{6.1}$$

where the initial state is drawn from the distribution $\rho(\mathbf{s}_1)$. The standard RL objective consists of maximizing the expected discounted return, $\mathbb{E}_{p(\tau)}[\mathcal{R}(\tau)]$. For convenience of presentation, we use the undiscounted setting ($\gamma = 1$), though the formulation can be applied with any valid γ.

KL-Regularized Reinforcement Learning

Various works have formulated RL, planning, and control problems in terms of probabilistic inference (Dayan and Hinton, 1997; Attias, 2003; Verma and Rao, 2006; Toussaint and Storkey, 2006; Todorov, 2008; Botvinick and Toussaint, 2012; Levine, 2018). These approaches consider the agent-environment interaction as a graphical model, then convert reward maximization into maximum marginal likelihood estimation, learning and inferring a policy that results in maximal reward. This conversion is accomplished by introducing one or more binary observed variables (Cooper, 1988), denoted as $\mathcal{O}$, with

$$p(\mathcal{O} = 1|\tau) \propto \exp\big(\mathcal{R}(\tau)/\alpha\big),$$

where α is a temperature hyper-parameter. These new variables are often referred to as "optimality" variables (Levine, 2018). We would like to infer latent variables, τ, and learn parameters, θ, that yield the maximum log-likelihood of optimality, i.e., $\log p(\mathcal{O} = 1)$. Evaluating this likelihood requires marginalizing the joint distribution, $p(\mathcal{O} = 1) = \int p(\tau, \mathcal{O} = 1) d\tau$. This involves averaging over all trajectories, which is intractable in high-dimensional spaces. Instead, we can use variational inference to lower bound this objective, introducing a structured approximate posterior distribution:

$$\pi(\tau|\mathcal{O}) = \prod_{t=1}^{T} p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}). \tag{6.2}$$

This provides the following lower bound on the objective, $\log p(\mathcal{O} = 1)$:

$$\log \int p(\mathcal{O} = 1|\tau)p(\tau)d\tau \geq \int \pi(\tau|\mathcal{O}) \left[\log p(\mathcal{O} = 1|\tau) + \log \frac{p(\tau)}{\pi(\tau|\mathcal{O})} \right] d\tau \tag{6.3}$$

$$= \mathbb{E}_\pi[\mathcal{R}(\tau)/\alpha] - D_{\text{KL}}(\pi(\tau|\mathcal{O})\|p(\tau)). \tag{6.4}$$

Equivalently, we can multiply by α, defining the variational RL objective as

$$\mathcal{J}(\pi, \theta) \equiv \mathbb{E}_\pi[\mathcal{R}(\tau)] - \alpha D_{\text{KL}}(\pi(\tau|\mathcal{O})\|p(\tau)). \tag{6.5}$$

This objective consists of the expected return (i.e., the standard RL objective) and a KL divergence between $\pi(\tau|\mathcal{O})$ and $p(\tau)$. In terms of states and actions, this objective is written as

$$\mathcal{J}(\pi, \theta) = \mathbb{E}_{\substack{\mathbf{s}_t, r_t \sim p_{\text{env}} \\ \mathbf{a}_t \sim \pi}} \left[\sum_{t=1}^{T} r_t - \alpha \log \frac{\pi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O})}{p_\theta(\mathbf{a}_t|\mathbf{s}_t)} \right]. \tag{6.6}$$

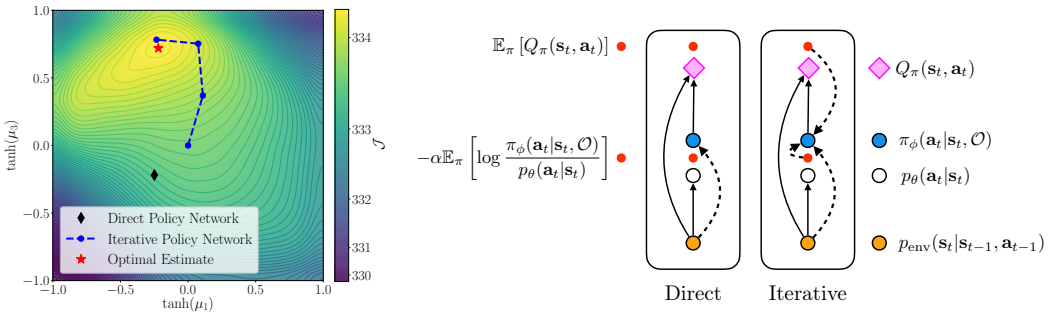
At a given timestep, t, one can optimize this objective by estimating the future terms in the summation using a "soft" action-value (Q_π) network (Haarnoja, H. Tang, et al., 2017) or model (Piché et al., 2019). For instance, sampling $\mathbf{s}_t \sim p_{\text{env}}$, slightly abusing notation, we can write the objective at time t as

$$\mathcal{J}(\pi, \theta) = \mathbb{E}_\pi \left[Q_\pi(\mathbf{s}_t, \mathbf{a}_t) \right] - \alpha D_{\text{KL}}(\pi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O})||p_\theta(\mathbf{a}_t|\mathbf{s}_t)). \tag{6.7}$$

Policy optimization in the KL-regularized setting corresponds to maximizing $\mathcal{J}$ w.r.t. π. We often consider parametric policies, in which π is defined by distribution parameters, $\boldsymbol{\lambda}$, e.g., Gaussian mean, $\boldsymbol{\mu}$, and variance, $\boldsymbol{\sigma}^2$. In this case, policy optimization corresponds to maximizing:

$$\boldsymbol{\lambda} \leftarrow \arg\max_{\boldsymbol{\lambda}} \mathcal{J}(\pi, \theta). \tag{6.8}$$

Optionally, we can then also learn the policy prior parameters, θ (Abdolmaleki et al., 2018).



(a) (b)

Figure 6.1: **Amortized Policy Optimization**. **(a)** 2D visualization of policy optimization. A direct amortized policy network fails to output an optimal estimate, resulting in an *amortization gap* in performance. An iterative amortized policy network finds an improved estimate. **(b)** Diagrams of direct and iterative amortized policy optimization. Larger circles denote distributions, and smaller red circles denote terms in the objective, $\mathcal{J}$. Dashed arrows denote amortized optimizers. Iterative amortization uses gradient-based feedback during optimization, whereas direct amortization does not.

Entropy & KL Regularized Policy Networks Perform Direct Amortization

Policy-based approaches to RL typically do not directly optimize the action distribution parameters, e.g., through gradient-based optimization. Instead, the action distribution parameters are output by a function approximator (deep network), f_ϕ, which is trained using deterministic (Silver et al., 2014; Lillicrap et al., 2016) or stochastic gradients (Williams, 1992; Heess et al., 2015). When combined with entropy or KL regularization, this policy network is a form of *amortized* optimization (Gershman and Goodman, 2014), learning to estimate policies. Again, denoting the action distribution parameters, e.g., mean and variance, as $\boldsymbol{\lambda}$, for a given state, $\mathbf{s}$, we can express this direct mapping as

$$\boldsymbol{\lambda} \leftarrow f_\phi(\mathbf{s}), \qquad \text{(direct amortization)} \qquad (6.9)$$

denoting the corresponding policy as $\pi_\phi(\mathbf{a}|\mathbf{s}, \mathcal{O}; \boldsymbol{\lambda})$. Thus, f_ϕ attempts to *learn* to optimize Eq. 6.8. This setup is shown in Figure 6.1 (Right). Without entropy or KL regularization, i.e., $\pi_\phi(\mathbf{a}|\mathbf{s}) = p_\theta(\mathbf{a}|\mathbf{s})$, we can instead interpret the network as directly integrating the LHS of Eq. 6.3, which is less efficient and more challenging. Regularization smooths the optimization landscape, yielding more stable improvement and higher asymptotic performance (Ahmed et al., 2019).

Viewing policy networks as a form of direct amortized variational optimizer (Eq. 6.9) allows us to see that they are similar to "encoder" networks in variational autoencoders

Algorithm 3 Direct Amortization

Initialize ϕ
for each environment step **do**
  $\boldsymbol{\lambda} \leftarrow f_\phi(\mathbf{s}_t)$
  $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t, O; \boldsymbol{\lambda})$
  $\mathbf{s}_{t+1} \sim p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
end for
for each training step **do**
  $\phi \leftarrow \phi + \eta \nabla_\phi \mathcal{J}$
end for

Algorithm 4 Iterative Amortization

Initialize ϕ
for each environment step **do**
  Initialize $\boldsymbol{\lambda}$
  for each policy optimization iteration **do**
    $\boldsymbol{\lambda} \leftarrow f_\phi(\mathbf{s}_t, \boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}} \mathcal{J})$
  end for
  $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t, O; \boldsymbol{\lambda})$
  $\mathbf{s}_{t+1} \sim p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
end for
for each training step **do**
  $\phi \leftarrow \phi + \eta \nabla_\phi \mathcal{J}$
end for

(VAEs) (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014). However, there are several drawbacks to direct amortization.

Amortization Gap. Direct amortization results in suboptimal approximate posterior estimates, with the resulting gap in the variational bound referred to as the *amortization gap* (Cremer, Li, and Duvenaud, 2018). Thus, in the RL setting, an amortized policy, π_ϕ, results in worse performance than the optimal policy within the parametric policy class, denoted as $\widehat{\pi}$. The amortization gap is the gap in following inequality:

$$\mathcal{J}(\pi_\phi, \theta) \leq \mathcal{J}(\widehat{\pi}, \theta).$$

Because $\mathcal{J}$ is a variational bound on the RL objective, i.e., expected return, a looser bound, due to amortization, prevents one from more completely optimizing this objective.

This is shown in Figure 6.1 (Left),[1] where $\mathcal{J}$ is plotted over two dimensions of the policy mean at a particular state in the MuJoCo environment `Hopper-v2`. The estimate of a direct amortized policy (⧫) is suboptimal, far from the optimal estimate (★). While the relative difference in the objective is relatively small, suboptimal estimates prevent sampling and exploring high-value regions of the action-space. That is, suboptimal estimates have only a *minor* impact on evaluation performance (see Appendix B.4) but hinder effective data collection.

[1] Additional 2D plots are shown in Appendix Figure B.3.

Single Estimate. Direct amortization is limited to a single, static estimate. In other words, if there are multiple high-value regions of the action-space, a uni-modal (e.g., Gaussian) direct amortized policy is restricted to only one region, thereby limiting exploration. Note that this is an additional restriction beyond simply considering uni-modal distributions, as a generic optimization procedure may arrive at multiple uni-modal estimates depending on initialization and stochastic sampling (see Section 6.3). While multi-modal distributions reduce the severity of this restriction (Y. Tang and Agrawal, 2018; Haarnoja, Hartikainen, et al., 2018), the other limitations of direct amortization still persist.

Inability to Generalize Across Objectives. Direct amortization is a feedforward procedure, receiving gradients from the objective only *after* estimation. This is contrast to other forms of optimization, which receive gradients (feedback) *during* estimation. Thus, unlike other optimizers, direct amortization is incapable of generalizing to new objectives, e.g., if $Q_\pi(\mathbf{s}, \mathbf{a})$ or $p_\theta(\mathbf{a}|\mathbf{s})$ change, which is a desirable capability for adapting to new tasks or environments.

To improve upon this scheme and overcome these drawbacks, in Section 6.3, we turn to *iterative amortization* (Chapter 3), retaining the efficiency of amortization while employing a more flexible iterative estimation procedure.

Related Work

Previous works have investigated methods for improving policy optimization. QT-Opt (Kalashnikov et al., 2018) uses the cross-entropy method (CEM) (Rubinstein and Kroese, 2013), an iterative derivative-free optimizer, to optimize a Q-value estimator for robotic grasping. CEM and related methods are also used in model-based RL for performing model-predictive control (Nagabandi et al., 2018; Chua et al., 2018; Piché et al., 2019; Hafner et al., 2019). Gradient-based policy optimization (Henaff, Whitney, and LeCun, 2017; Srinivas et al., 2018; Bharadhwaj, Xie, and Shkurti, 2020), in contrast, is less common, however, gradient-based optimization can also be combined with CEM (Amos and Yarats, 2020). Most policy-based methods use direct amortization, either using a feedforward (Haarnoja, Zhou, Abbeel, et al., 2018) or recurrent (Guez et al., 2019) network. Similar approaches have also been applied to model-based value estimates (Byravan et al., 2020; Clavera, Y. Fu, and Abbeel, 2020; Amos, Stanton, et al., 2020), as well as combining direct amortization with model predictive control (Lee, Saigol, and Theodorou, 2019) and planning (Rivière et al., 2020). A separate line of work has explored improving the policy distribution,

using normalizing flows (Haarnoja, Hartikainen, et al., 2018; Y. Tang and Agrawal, 2018) and latent variables (Tirumala et al., 2019). In principle, iterative amortization can perform policy optimization in each of these settings.

Iterative amortized policy optimization is conceptually similar to negative feedback control (Astrom and Murray, 2008), using errors to update policy estimates. However, while conventional feedback control methods are often restricted in their applicability, e.g., linear systems and quadratic cost, iterative amortization is generally applicable to any differentiable control objective. This is analogous to the generalization of Kalman filtering (Kalman, 1960) to amortized variational filtering (Chapter 4) for state estimation.

6.3 Iterative Amortized Policy Optimization

Formulation

Iterative amortized optimizers (Chapter 3) utilize some form of error or gradient to update the approximate posterior distribution parameters. While various forms exist, we consider gradient-encoding models (Andrychowicz et al., 2016) due to their generality. Compared with direct amortization in Eq. 6.9, we use iterative amortized optimizers of the general form

$$\boldsymbol{\lambda} \leftarrow f_\phi(\mathbf{s}, \boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}} \mathcal{J}), \tag{6.10}$$

also shown in Figure 6.1 (Right), where f_ϕ is a deep network and $\boldsymbol{\lambda}$ are the policy distribution parameters. For example, if $\pi = \mathcal{N}(\mathbf{a}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, then $\boldsymbol{\lambda} \equiv [\boldsymbol{\mu}, \boldsymbol{\sigma}]$. Technically, $\mathbf{s}$ is redundant, as the state dependence is already captured in $\mathcal{J}$, but this can empirically improve performance (Chapter 3). In practice, the update is carried out using a "highway" gating operation (Hochreiter and Schmidhuber, 1997; Srivastava, Greff, and Schmidhuber, 2015). Denoting $\boldsymbol{\omega}_\phi \in [0, 1]$ as the gate and $\boldsymbol{\delta}_\phi$ as the update, both of which are output by f_ϕ, the gating operation is expressed as

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\omega}_\phi \odot \boldsymbol{\lambda} + (\mathbf{1} - \boldsymbol{\omega}_\phi) \odot \boldsymbol{\delta}_\phi, \tag{6.11}$$

where $\odot$ denotes elementwise multiplication. This update is typically run for a fixed number of steps, and, as with a direct policy, the iterative optimizer is trained using stochastic gradient estimates of $\nabla_\phi \mathcal{J}$, obtained through the pathwise derivative estimator (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014; Heess et al., 2015). Because the gradients $\nabla_{\boldsymbol{\lambda}} \mathcal{J}$ must be estimated online, i.e., during policy optimization, this scheme requires some way of estimating $\mathcal{J}$ online, e.g., through a parameterized Q-value network (V. Mnih, Kavukcuoglu, et al., 2013; Lillicrap et al., 2016) or a differentiable model (Heess et al., 2015).

Benefits of Iterative Amortization

Reduced Amortization Gap. Iterative amortized optimizers are more flexible than their direct counterparts, incorporating feedback from the objective *during* policy optimization (Algorithm 4), rather than only *after* optimization (Algorithm 3). Increased flexibility improves the accuracy of optimization, thereby tightening the variational bound (Chapters 3 & 4). We see this flexibility in Figure 6.1 (Left), where an iterative amortized policy network iteratively refines the policy estimate (•), quickly arriving near the optimal estimate.

Multiple Estimates. Iterative amortization, by using stochastic gradients and random initialization, can traverse the optimization landscape. As with any iterative optimization scheme, this allows iterative amortization to obtain multiple valid estimates, referred to as "multi-stability" in the perception literature (Greff et al., 2019). We illustrate this capability across two action dimensions in Figure 6.2 for a state in the `Ant-v2` MuJoCo environment. Over multiple policy optimization runs, iterative amortization finds multiple modes, sampling from two high-value regions of the action space. This provides increased flexibility in action exploration, despite only using a uni-modal policy distribution.

Generalization Across Objectives. Iterative amortization uses the gradients of the objective *during* optimization, i.e., feedback, allowing it to potentially generalize to new or updated objectives. We see this in Figure 6.1 (Left), where iterative amortization, despite being trained with a *different* value estimator, is capable of generalizing to this new objective. We demonstrate this capability further in Section 6.4. This opens the possibility of accurately and efficiently performing policy optimization in new settings, e.g., a rapidly changing model or new tasks.

Consideration: Mitigating Value Overestimation

Why are more powerful policy optimizers typically not used in practice? As we now describe, part of the issue stems from value overestimation. Model-free approaches generally estimate Q_π using function approximation and temporal difference learning. However, this has the pitfall of value overestimation, i.e., positive bias in the estimate, $\widehat{Q}_\pi$ (Thrun and Schwartz, 1993). This issue is tied to uncertainty in the value estimate, though it is distinct from optimism under uncertainty. If the policy can exploit regions of high uncertainty, the resulting target values will introduce positive bias into the estimate. More flexible policy optimizers exacerbate the problem, exploiting

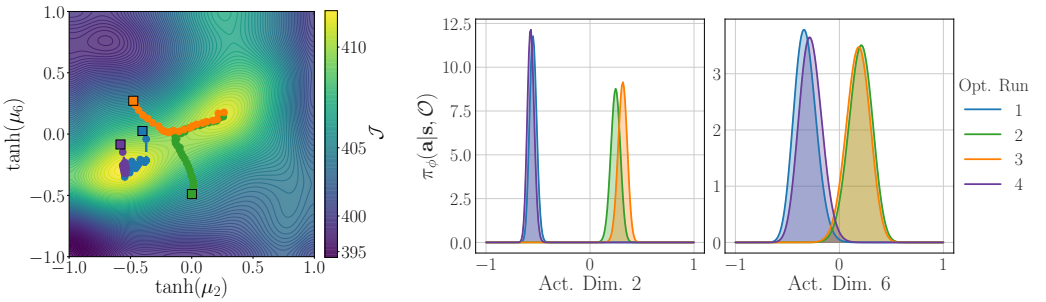


Figure 6.2: **Estimating Multiple Policy Modes**. Unlike direct amortization, which is restricted to a single estimate, the stochasticity of iterative optimization allows iterative amortization to effectively sample from multiple high-value modes in the action space. This capability is shown for a particular state in `Ant-v2`, showing multiple optimization runs across two action dimensions (**Left**). Each colored square denotes an initialization. The optimizer finds both modes, with the assigned density plotted on the **Right**. This capability provides increased flexibility in action exploration.

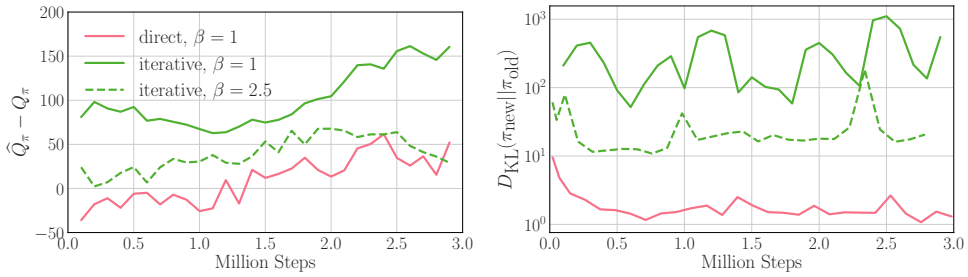
this uncertainty to a greater degree. Further, a rapidly changing policy increases the difficulty of value estimation (Rajeswaran, Mordatch, and V. Kumar, 2020).

Various techniques have been proposed for mitigating value overestimation in deep RL. The most prominent technique, double deep Q-network (Van Hasselt, Guez, and Silver, 2016) maintains two Q-value estimates (Van Hasselt, 2010), attempting to decouple policy optimization from value estimation. Fujimoto, Hoof, and Meger, 2018 apply and improve upon this technique for actor-critic settings, estimating the target Q-value as the minimum of two Q-networks, Q_{ψ_1} and Q_{ψ_2}:

$$\widehat{Q}_\pi(\mathbf{s},\mathbf{a}) = \min_{i=1,2} Q_{\psi_i'}(\mathbf{s},\mathbf{a}),$$

where ψ_i' denotes the "target" network parameters. As noted by Fujimoto, Hoof, and Meger, 2018, this not only counteracts value overestimation, but also penalizes high-variance value estimates, because the minimum decreases with the variance of the estimate. Ciosek et al., 2019 noted that, for a bootstrapped ensemble of two Q-networks, the minimum operation can be interpreted as estimating

$$\widehat{Q}_\pi(\mathbf{s},\mathbf{a}) = \mu_Q(\mathbf{s},\mathbf{a}) - \beta\sigma_Q(\mathbf{s},\mathbf{a}),$$



(a) (b)

Figure 6.3: **Mitigating Value Overestimation**. Using the same value estimation setup ($\beta = 1$), shown on `Ant-v2`, iterative policy optimization results in (**a**) higher value overestimation bias and (**b**) a more rapidly changing policy as compared with direct policy optimization. Increasing β helps to mitigate these issues by further penalizing variance in the value estimate.

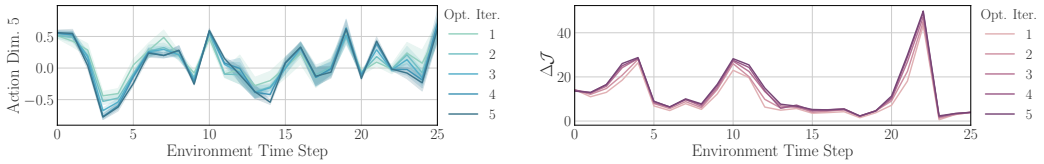
with mean and standard deviation

$$\mu_Q(\mathbf{s}, \mathbf{a}) \equiv \frac{1}{2} \sum_{i=1,2} Q_{\psi_i'}(\mathbf{s}, \mathbf{a}),$$

$$\sigma_Q(\mathbf{s}, \mathbf{a}) \equiv \left[\frac{1}{2} \sum_{i=1,2} \left(Q_{\psi_i'}(\mathbf{s}, \mathbf{a}) - \mu_Q(\mathbf{s}, \mathbf{a}) \right)^2 \right]^{1/2},$$

and $\beta = 1$. Thus, to further penalize high-variance value estimates, preventing value overestimation, we can increase β. For large β, however, value estimates become overly pessimistic, negatively impacting training. Thus, β reduces target value variance at the cost of increased bias.

Due to the flexibility of iterative amortization, the default $\beta = 1$ results in increased value bias (Figure 6.3a) and a more rapidly changing policy (Figure 6.3b) as compared with direct amortization. Further penalizing high-variance target values with $\beta = 2.5$ reduces value overestimation and improves policy stability. For details, see Marino, Piché, et al., 2020. Recent techniques for mitigating overestimation have been proposed, such as adjusting the temperature, α (Fox, 2019). In offline RL, this issue has been tackled through the action prior (Scott Fujimoto, David Meger, and Precup, 2019; A. Kumar, J. Fu, et al., 2019; Wu, Tucker, and Nachum, 2019) or by altering Q-network training (Agarwal, Schuurmans, and Norouzi, 2019; A. Kumar, Zhou, et al., 2020). While such techniques could be used here, increasing β provides a simple solution with no additional computational overhead.



(a) Policy (b) Improvement

Figure 6.4: **Policy Optimization**. Visualization over time steps of **(a)** one dimension of the policy distribution and **(b)** the improvement in the objective, $\Delta\mathcal{J}$, across policy optimization iterations.

6.4 Experiments

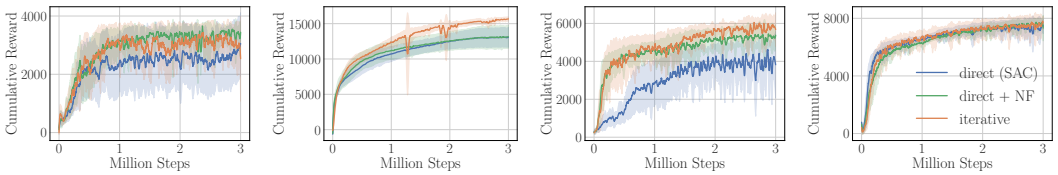
Setup

To focus specifically on policy optimization, we implement iterative amortized policy optimization using the setup of soft actor-critic (SAC) described by Haarnoja, Zhou, Hartikainen, et al., 2018. This involves using two Q-networks, uniform action prior, $p_\theta(\mathbf{a}|\mathbf{s}) = \mathcal{U}(-1, 1)$, and an automatic tuning scheme for the temperature, α. In our experiments, "direct" refers to direct amortized policy optimization employed in SAC, i.e., a direct policy network, and "iterative" refers to iterative amortized policy optimization. Both approaches use the *same* network architecture, adjusting only the number of inputs and outputs to accommodate gradients, current policy estimates, and gated updates (Sec. 6.3). Unless otherwise stated, we use 5 iterations per time step for iterative amortization, as in Chapter 3. For additional details, we refer to Marino, Piché, et al., 2020 and Haarnoja, Zhou, Abbeel, et al., 2018; Haarnoja, Zhou, Hartikainen, et al., 2018. Accompanying code is available at `github.com/joelouismarino/variational_rl`.

Analysis

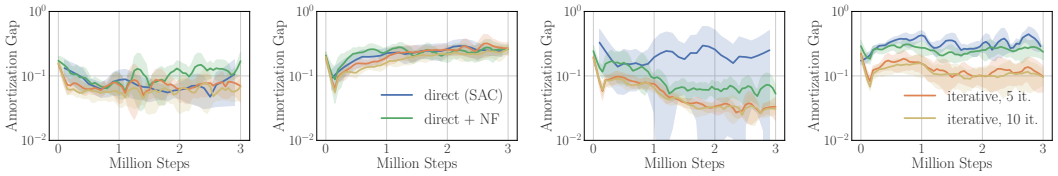
Visualizing Policy Optimization

In Figure 6.1 (Left), we visualize the trajectory of iterative amortized policy optimization along two dimensions of the policy mean on a state from `Hopper-v2`. Through iterative optimization, the network arrives near the optimum. Notably, this is performed with a value function trained using a *different*, direct policy, demonstrating generalization to other optimization landscapes. In Figure 6.4, we visualize iterative refinement using a single action dimension from `Ant-v2` across time steps. The refinements in Figure 6.4a give rise to the objective improvements in Figure 6.4b.



(a) `Hopper-v2` (b) `HalfCheetah-v2` (c) `Walker2d-v2` (d) `Ant-v2`

Figure 6.5: **Performance Comparison**. Iterative amortized policy optimization performs comparably with or better than direct amortized policies across a range of MuJoCo environments. Performance curves show the mean and ± standard deviation over 5 random seeds.



(a) `Hopper-v2` (b) `HalfCheetah-v2` (c) `Walker2d-v2` (d) `Ant-v2`

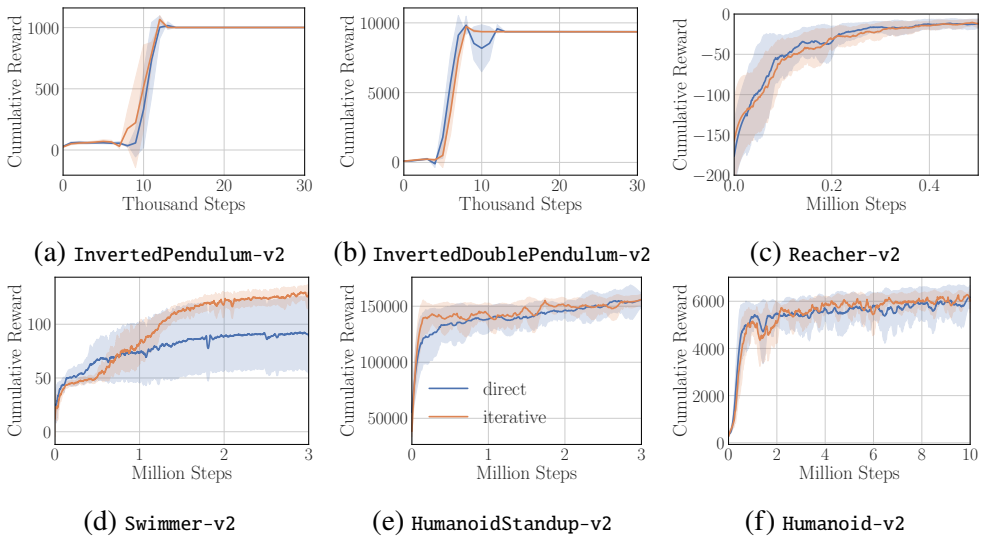
Figure 6.6: **Amortization Gap**. Estimated amortization gaps per step for direct and iterative amortized policy optimization. Iterative amortization achieves comparable or lower gaps across environments. Gaps are estimated using stochastic gradient-based optimization over 100 random states. Curves show the mean and ± standard deviation over 5 random seeds.

Performance Comparison

We evaluate iterative amortized policy optimization on the suite of MuJoCo (Todorov, Erez, and Tassa, 2012) continuous control tasks from OpenAI gym (Brockman et al., 2016). In Figures 6.5 & 6.7, we compare the cumulative reward, i.e., return, of direct and iterative amortized policy optimization across environments. Each curve shows the mean and ± standard deviation of 5 random seeds. In all cases, iterative amortized policy optimization matches or outperforms the baseline direct amortized method, both in sample efficiency and final performance. Across environments, iterative amortization also yields more consistent, i.e., lower variance, performance. This suggests that iterative amortization is able to flexibly explore the optimization landscape, consistently arriving at improved policy estimates.

Multiple Policy Modes

To better understand the exploration benefits of iterative amortization, we also compare with direct amortization with a multi-modal policy distribution, formed using inverse autoregressive flows (Kingma, Salimans, et al., 2016), a type of normalizing flow



(a) `InvertedPendulum-v2` (b) `InvertedDoublePendulum-v2` (c) `Reacher-v2`

(d) `Swimmer-v2` (e) `HumanoidStandup-v2` (f) `Humanoid-v2`

Figure 6.7: **Additional Performance Comparison**. Results are shown on the remaining MuJoCo environments from OpenAI gym. Performance curves show the mean and ± standard deviation over 5 random seeds.

(NF). Using a multi-modal policy reduces the performance deficiencies of direct amortization on `Hopper-v2` and `Walker2d-v2`, indicating that much of the benefit of iterative amortization is due to lifting direct amortization's restriction to a single, uni-modal policy estimate. Yet, direct + NF still struggles on `HalfCheetah-v2` compared with iterative amortization (Fig. 6.5b), suggesting that more complex, multi-modal distributions are not the *only* consideration.

To confirm that iterative amortization has captured multiple policy modes, at the end of training, we take `Walker2d-v2` and histogram the distances between policy means across separate runs of policy optimization per state (Fig. 6.8a). For the state with the largest distance, we plot 2D projections of the optimization objective across action dimensions in Figure 6.8b, as well as the policy density across 10 optimization runs (Fig. 6.8c). We see that a subset of states indeed still retain multiple policy modes.

Amortization Gap

To evaluate policy optimization accuracy, we estimate per-step amortization gaps, performing additional iterations of gradient ascent on $\mathcal{J}$ w.r.t. the policy parameters, $\boldsymbol{\lambda} \equiv [\boldsymbol{\mu}, \boldsymbol{\sigma}]$ (see Appendix A.3). To analyze generalization, we also evaluate the iterative agents trained with 5 iterations for an additional 5 amortized iterations. Results are shown in Figure 6.6. We emphasize that it is challenging to *directly*

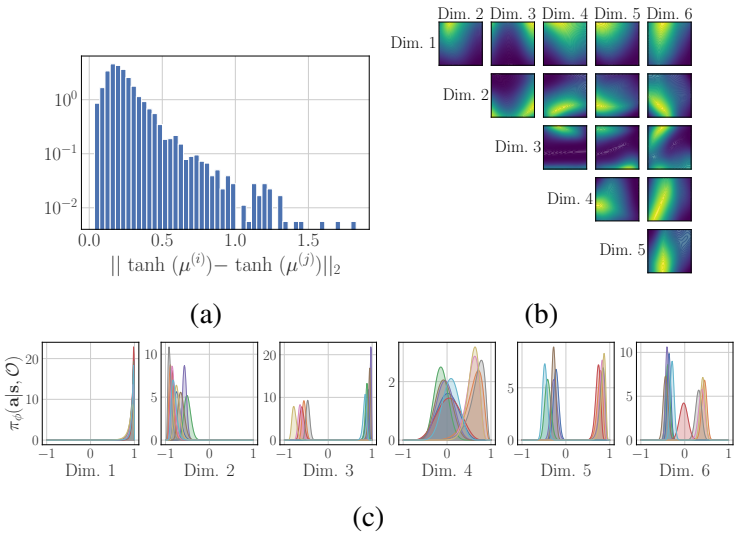
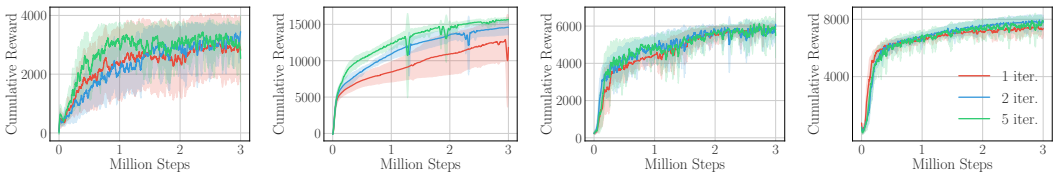


Figure 6.8: **Multiple Policy Modes**. **(a)** Histogram of distances between policy means ($\boldsymbol{\mu}$) across optimization runs (i and j) over seeds and states on `Walker2d-v2` at 3 million environment steps. For the state with the largest distance, **(b)** shows the projected optimization surface on each pair of action dimensions, and **(c)** shows the policy density for 10 optimization runs.

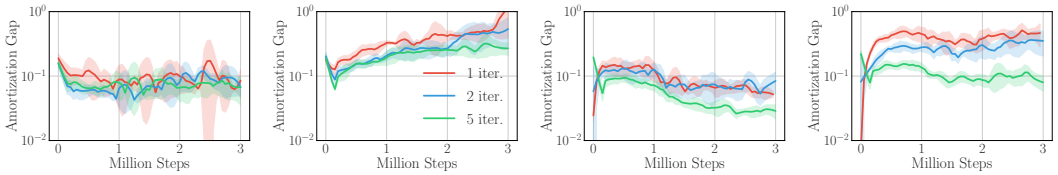


(a) `Hopper-v2` (b) `HalfCheetah-v2` (c) `Walker2d-v2` (d) `Ant-v2`

Figure 6.9: **Varying Iterations During Training**. Performance of iterative amortized policy optimization for varying numbers of iterations during training. Increasing the number of iterations generally results in improvements. Curves show the mean and ± standard deviation over 5 random seeds.

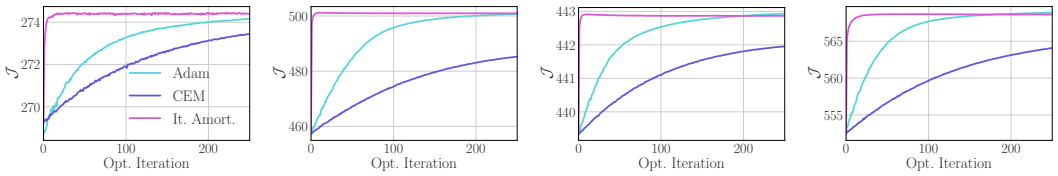
compare amortization gaps across optimization schemes, as these involve different value functions, and therefore different objectives. Likewise, we estimate the amortization gap using the learned Q-networks, which may be biased (Figure 6.3). Nevertheless, we find that iterative amortized policy optimization achieves, on average, lower amortization gaps than direct amortization across all environments. Additional amortized iterations at evaluation yield further estimated improvement, demonstrating generalization beyond the optimization horizon used during training.

The amortization gaps are small relative to the objective, playing a negligible role in *evaluation* performance. Rather, improved policy optimization is helpful for *training*, allowing the agent to explore states where value estimates are highest. To



(a) `Hopper-v2` (b) `HalfCheetah-v2` (c) `Walker2d-v2` (d) `Ant-v2`

Figure 6.10: **Amortization Gap of Varying Iterations During Training**. Corresponding amortization gaps for varying numbers of iterations during training. We generally see that increasing the number of iterations generally reduces the amortization gap.



(a) `Hopper-v2` (b) `HalfCheetah-v2` (c) `Walker2d-v2` (d) `Ant-v2`

Figure 6.11: **Comparison with Iterative Optimizers**. Average estimated objective over policy optimization iterations, comparing with Adam (Kingma and Ba, 2014) and CEM (Rubinstein and Kroese, 2013). These iterative optimizers require over an order of magnitude more iterations to reach comparable performance with iterative amortization, making them impractical in many applications.

probe this further, we train iterative amortized policy optimization while varying the number of iterations per step in $\{1, 2, 5\}$, yielding optimizers with varying degrees of accuracy. Note that each optimizer is, in theory, capable of finding multiple modes. In Figure 6.9, we see that training with additional iterations improves performance and optimization accuracy. `Walker2d-v2` provides an interesting example. Even with a single iteration, we see that iterative amortization outperforms direct amortization, suggesting that multi-modality is the dominant factor for improved performance here. Yet, 1 iteration is slightly worse compared with 2 and 5 iterations early in training, both in terms of performance and optimization. As the amortization gap decreases later in training, we see that the performance gap ultimately decreases. Further work could help to analyze this process in even more detail. We stress that the exact form of this relationship depends on the Q-value estimator and other factors.

Comparison with Iterative Optimizers

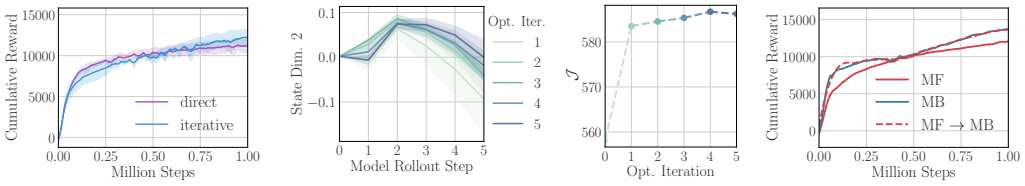
Iterative amortized policy optimization obtains the *accuracy* benefits of iterative optimization while retaining the *efficiency* benefits of amortization. We now compare iterative amortization with two popular iterative optimizers: Adam (Kingma and Ba,

2014), a gradient-based optimizer, and cross-entropy method (CEM) (Rubinstein and Kroese, 2013), a gradient-free optimizer. We collect 100 states for each seed in each environment from the model-free experiments. For each optimizer, we optimize the variational objective, $\mathcal{J}$, starting from the same initialization. Tuning the step size, we found that 0.01 yielded the steepest improvement without diverging for both Adam and CEM. Gradients are evaluated with 10 action samples. For CEM, we sample 100 actions and fit a Gaussian mean and variance to the top 10 samples. This is comparable with QT-Opt (Kalashnikov et al., 2018), which draws 64 samples and retains the top 6 samples.

The results, averaged across states and random seeds, are shown in Figure 6.11. CEM (gradient-free) is less efficient than Adam (gradient-based), which is unsurprising, especially considering that Adam effectively approximates higher-order curvature through momentum terms. However, Adam and CEM both require over *an order of magnitude* more iterations to reach comparable performance with iterative amortization. While iterative amortized policy optimization does not always obtain asymptotically optimal estimates, we note that these networks were trained with only 5 iterations, yet continue to improve and remain stable far beyond this limit. Finally, comparing wall clock time for each optimizer, iterative amortization is only roughly 1.25× slower than CEM and 1.15× slower than Adam, making iterative amortization still substantially more efficient.

Generalizing to Model-Based Values

Direct amortization is a purely feedforward process and is therefore incapable of generalizing to new objectives. In contrast, because iterative amortization is formulated through gradient-based feedback, such optimizers may be capable of generalizing to new objective estimators. To demonstrate this capability further, we apply iterative amortization with model-based value estimators, using a learned deterministic model on `HalfCheetah-v2` (see Appendix A.5). In Figure 6.12a, we see that iterative amortization slightly outperforms direct amortization in this setting. The resulting policy optimization procedure refines planned trajectories, shown for a single state dimension in Figure 6.12b, yielding corresponding improvements (Fig. 6.12c). We evaluate the generalizing capabilities in Figure 6.12d by transferring the policy optimizer from a model-free agent to a model-based agent. Iterative amortization generalizes to these new value estimates, *instantly* recovering the performance of the model-based agent. This highlights the opportunity for instantly



(a) (b) (c) (d)

Figure 6.12: **Optimizing Model-Based Value Estimates**. **(a)** Performance comparison of direct and iterative amortization using model-based value estimates. **(b)** Planned trajectories over policy optimization iterations. **(c)** The corresponding estimated objective increases over iterations. **(d)** Zero-shot transfer of iterative amortization from model-free (MF) to model-based (MB) estimates.

incorporating new tasks, goals, or model estimates into policy optimization.

6.5 Discussion

In this chapter, we applied the technique of learned negative feedback to control, arriving at iterative amortized policy optimization, a flexible and powerful policy optimization technique. In so doing, we have highlighted several limitations of direct amortization: 1) limited accuracy, as quantified by the amortization gap, 2) restriction to a single estimate, limiting exploration, and 3) inability to generalize to new objectives, limiting the transfer of these policy optimizers. As confirmed through our empirical analysis on benchmark continuous control environments, iterative amortization provides a step toward improving each of these restrictions, with accompanying improvements in performance over current direct amortization methods. Thus, iterative amortization provides a drop-in replacement and improvement over direct policy networks in deep reinforcement learning.

Although we have discussed three separate limitations of direct amortization, these factors are highly interconnected. By broadening policy optimization to an iterative procedure, we automatically obtain a potentially more accurate and general policy optimizer, with the capability of obtaining multiple modes. While our analysis suggests that the improved exploration resulting from multiple modes is the primary factor affecting performance, future work could tease out these effects further and assess the relative contributions of these improvements in additional environments. We are hopeful that iterative amortized policy optimization, by providing a more powerful, exploratory, and general optimizer, will enable a range of improved reinforcement learning algorithms.

References

Abdolmaleki, Abbas et al. (2018). "Maximum a Posteriori Policy Optimisation". In: *International Conference on Learning Representations*.

Agarwal, Rishabh, Dale Schuurmans, and Mohammad Norouzi (2019). "An Optimistic Perspective on Offline Reinforcement Learning". In: *arXiv preprint arXiv:1907.04543*.

Ahmed, Zafarali et al. (2019). "Understanding the impact of entropy on policy optimization". In: *International Conference on Machine Learning*, pp. 151–160.

Amos, Brandon, Samuel Stanton, et al. (2020). "On the model-based stochastic value gradient for continuous reinforcement learning". In: *arXiv preprint arXiv:2008.12775*.

Amos, Brandon and Denis Yarats (2020). "The differentiable cross-entropy method". In: *International Conference on Machine Learning*.

Andrychowicz, Marcin et al. (2016). "Learning to learn by gradient descent by gradient descent". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 3981–3989.

Astrom, Karl Johan and Richard M Murray (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

Attias, Hagai (2003). "Planning by probabilistic inference." In: *AISTATS*. Citeseer.

Bharadhwaj, Homanga, Kevin Xie, and Florian Shkurti (2020). "Model-Predictive Planning via Cross-Entropy and Gradient-Based Optimization". In: *Learning for Dynamics and Control*, pp. 277–286.

Botvinick, Matthew and Marc Toussaint (2012). "Planning as inference". In: *Trends in cognitive sciences* 16.10, pp. 485–488.

Brockman, Greg et al. (2016). "Openai gym". In: *arXiv preprint arXiv:1606.01540*.

Byravan, Arunkumar et al. (2020). "Imagined Value Gradients: Model-Based Policy Optimization with Tranferable Latent Dynamics Models". In: *Conference on Robot Learning*, pp. 566–589.

Chua, Kurtland et al. (2018). "Deep reinforcement learning in a handful of trials using probabilistic dynamics models". In: *Advances in Neural Information Processing Systems*, pp. 4754–4765.

Ciosek, Kamil et al. (2019). "Better exploration with optimistic actor critic". In: *Advances in Neural Information Processing Systems*, pp. 1787–1798.

Clavera, Ignasi, Yao Fu, and Pieter Abbeel (2020). "Model-Augmented Actor-Critic: Backpropagating through Paths". In: *International Conference on Learning Representations*.

Cooper, Gregory F (1988). "A method for using belief networks as influence diagrams". In: *Fourth Workshop on Uncertainty in Artificial Intelligence*.

Cremer, Chris, Xuechen Li, and David Duvenaud (2018). "Inference Suboptimality in Variational Autoencoders". In: *International Conference on Machine Learning*, pp. 1078–1086.

Dayan, Peter and Geoffrey E Hinton (1997). "Using expectation-maximization for reinforcement learning". In: *Neural Computation* 9.2, pp. 271–278.

Fox, Roy (2019). "Toward Provably Unbiased Temporal-Difference Value Estimation". In: *Optimization Foundations for Reinforcement Learning Workshop at NeurIPS*.

Fox, Roy, Ari Pakman, and Naftali Tishby (2016). "Taming the noise in reinforcement learning via soft updates". In: *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*. AUAI Press, pp. 202–211.

Fujimoto, S, H van Hoof, and D Meger (2018). "Addressing function approximation error in actor-critic methods". In: *Proceedings of Machine Learning Research* 80, pp. 1587–1596.

Fujimoto, Scott, David Meger, and Doina Precup (2019). "Off-policy deep reinforcement learning without exploration". In: *International Conference on Machine Learning*, pp. 2052–2062.

Gershman, Samuel and Noah Goodman (2014). "Amortized inference in probabilistic reasoning". In: *Proceedings of the Cognitive Science Society*. Vol. 36. 36.

Greff, Klaus et al. (2019). "Multi-Object Representation Learning with Iterative Variational Inference". In: *International Conference on Machine Learning*, pp. 2424–2433.

Guez, Arthur et al. (2019). "An Investigation of Model-Free Planning". In: *International Conference on Machine Learning*, pp. 2464–2473.

Haarnoja, Tuomas, Kristian Hartikainen, et al. (2018). "Latent Space Policies for Hierarchical Reinforcement Learning". In: *International Conference on Machine Learning*, pp. 1846–1855.

Haarnoja, Tuomas, Haoran Tang, et al. (2017). "Reinforcement learning with deep energy-based policies". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1352–1361.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, et al. (2018). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning*, pp. 1856–1865.

Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen, et al. (2018). "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905*.

Hafner, Danijar et al. (2019). "Learning Latent Dynamics for Planning from Pixels". In: *International Conference on Machine Learning*, pp. 2555–2565.

Heess, Nicolas et al. (2015). "Learning continuous control policies by stochastic value gradients". In: *Advances in Neural Information Processing Systems*, pp. 2944–2952.

Henaff, Mikael, William F Whitney, and Yann LeCun (2017). "Model-based planning with discrete and continuous actions". In: *arXiv preprint arXiv:1705.07177*.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Kalashnikov, Dmitry et al. (2018). "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation". In: *arXiv preprint arXiv:1806.10293*.

Kalman, Rudolph Emil (1960). "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering* 82.1, pp. 35–45.

Kim, Yoon et al. (2018). "Semi-Amortized Variational Autoencoders". In: *Proceedings of the International Conference on Machine Learning (ICML)*.

Kingma, Durk P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kingma, Durk P, Tim Salimans, et al. (2016). "Improved variational inference with inverse autoregressive flow". In: *Advances in neural information processing systems*, pp. 4743–4751.

Kingma, Durk P and Max Welling (2014). "Stochastic gradient VB and the variational auto-encoder". In: *Proceedings of the International Conference on Learning Representations*.

Kumar, Aviral, Justin Fu, et al. (2019). "Stabilizing off-policy q-learning via bootstrapping error reduction". In: *Advances in Neural Information Processing Systems*, pp. 11784–11794.

Kumar, Aviral, Aurick Zhou, et al. (2020). "Conservative Q-Learning for Offline Reinforcement Learning". In: *arXiv preprint arXiv:2006.04779*.

Lee, Keuntaek, Kamil Saigol, and Evangelos A Theodorou (2019). "Safe end-to-end imitation learning for model predictive control". In: *International Conference on Robotics and Automation*.

Levine, Sergey (2018). "Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review". In: *arXiv preprint arXiv:1805.00909*.

Lillicrap, Timothy P et al. (2016). "Continuous control with deep reinforcement learning". In: *International Conference on Learning Representations*.

Marino, Joseph, Alexandre Piché, Alessandro Davide Ialongo, and Yisong Yue (2020). "Iterative Amortized Policy Optimization". In: *Preprint*. URL: `https://arxiv.org/abs/2010.10670`.

Marino, Joseph, Yisong Yue, and Stephan Mandt (2018). "Iterative Amortized Inference". In: *International Conference on Machine Learning*, pp. 3403–3412. URL: `http://proceedings.mlr.press/v80/marino18a.html`.

Mnih, Andriy and Karol Gregor (2014). "Neural Variational Inference and Learning in Belief Networks". In: *International Conference on Machine Learning*, pp. 1791–1799.

Mnih, Volodymyr, Adria Puigdomenech Badia, et al. (2016). "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*, pp. 1928–1937.

Mnih, Volodymyr, Koray Kavukcuoglu, et al. (2013). "Playing Atari With Deep Reinforcement Learning". In: *NIPS Deep Learning Workshop*.

Nagabandi, Anusha et al. (2018). "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7559–7566.

Piché, Alexandre et al. (2019). "Probabilistic Planning with Sequential Monte Carlo methods". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=ByetGn0cYX.

Rajeswaran, Aravind, Igor Mordatch, and Vikash Kumar (2020). "A Game Theoretic Framework for Model Based Reinforcement Learning". In: *arXiv preprint arXiv:2004.07804*.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models". In: *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286.

Rivière, Benjamin et al. (2020). "GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning with End-to-End Learning". In: *IEEE Robotics and Automation Letters* 5.3, pp. 4249–4256.

Rubinstein, Reuven Y and Dirk P Kroese (2013). *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media.

Schulman, John, Sergey Levine, et al. (2015). "Trust region policy optimization". In: *International Conference on Machine Learning*, pp. 1889–1897.

Schulman, John, Filip Wolski, et al. (2017). "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347*.

Silver, David et al. (2014). "Deterministic Policy Gradient Algorithms". In: *International Conference on Machine Learning*, pp. 387–395.

Srinivas, Aravind et al. (2018). "Universal Planning Networks: Learning Generalizable Representations for Visuomotor Control". In: *International Conference on Machine Learning*, pp. 4732–4741.

Srivastava, Rupesh K, Klaus Greff, and Jürgen Schmidhuber (2015). "Training very deep networks". In: *Advances in neural information processing systems (NIPS)*, pp. 2377–2385.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Tang, Yunhao and Shipra Agrawal (2018). “Boosting Trust Region Policy Optimization by Normalizing Flows Policy”. In: *arXiv preprint arXiv:1809.10326*.

Thrun, Sebastian and Anton Schwartz (1993). “Issues in using function approximation for reinforcement learning”. In: *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*.

Tirumala, Dhruva et al. (2019). “Exploiting Hierarchy for Learning and Transfer in KL-regularized RL”. In: *arXiv preprint arXiv:1903.07438*.

Todorov, Emanuel (2008). “General duality between optimal control and estimation”. In: *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, pp. 4286–4292.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). “Mujoco: A physics engine for model-based control”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 5026–5033.

Toussaint, Marc and Amos Storkey (2006). “Probabilistic inference for solving discrete and continuous state Markov Decision Processes”. In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 945–952.

Van Hasselt, Hado (2010). “Double Q-learning”. In: *Advances in neural information processing systems*, pp. 2613–2621.

Van Hasselt, Hado, Arthur Guez, and David Silver (2016). “Deep reinforcement learning with double q-learning”. In: *Thirtieth AAAI conference on artificial intelligence*.

Verma, Deepak and Rajesh PN Rao (2006). “Goal-based imitation as probabilistic inference over graphical models”. In: *Advances in neural information processing systems*. Citeseer, pp. 1393–1400.

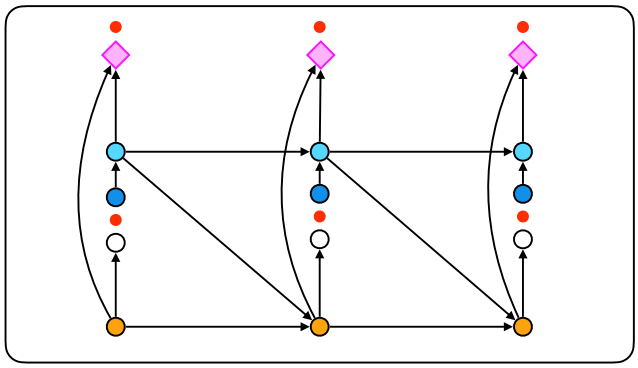
Wiener, Norbert (1948). *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press.

Williams, Ronald J (1992). “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Reinforcement Learning*. Springer, pp. 5–32.

Wu, Yifan, George Tucker, and Ofir Nachum (2019). “Behavior regularized offline reinforcement learning”. In: *arXiv preprint arXiv:1911.11361*.

Chapter 7

SEQUENTIAL AUTOREGRESSIVE FLOW-BASED POLICIES



learned feedforward control

Guerra, Alex and Joseph Marino (2020). "Sequential Autoregressive Flow-Based Policies". In: *ICML workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

7.1 Introduction

In this chapter, we complete the table outlined in Chapter 1, applying learned feedforward processing to control. As opposed to purely feedback policies (Chapter 6), which must optimize the control objective from scratch at each time step, we consider policies with a feedforward component, predicting useful actions at future time steps. This is loosely inspired by the hierarchical organization of animal motor control (Merel, Botvinick, and Wayne, 2019), in which central pattern generator circuits (Marder and Bucher, 2001) provide low-level dynamical motor primitives, receiving transient top-down signals from higher-level motor areas (Shalit et al., 2012). This improves computational efficiency and speed, as feedback optimization (inference) is relied upon less and less as feedforward behavioral routines are learned.

In formulating learned feedforward control policies, we again consider sequential autoregressive flows, as introduced in Chapter 5. Thus, the overall policy is composed of a dynamical feedforward component, conditioned on previous actions, and a feedback component, conditioned on the current state. We provide an initial set of

experiments exploring the use of these policies in continuous control environments, demonstrating improvements in performance as well as the ability to *distill* purely feedforward locomotion policies.

7.2 Autoregressive Flow-Based Policies

Basic Setup

We again consider the variational RL setup, as introduced in Chapter 6. As a brief review, we consider a Markov decision process (MDP), in which, at time t, an agent receives a state observation $\mathbf{s}_t \in \mathcal{S}$ and takes action $\mathbf{a}_t \in \mathcal{A}$ by sampling from a policy distribution, π. The agent then receives reward $r(\mathbf{s}_t, \mathbf{a}_t)$ and the environment transitions to the next state $\mathbf{s}_{t+1} \sim p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$.

By interpreting RL as variational inference (Ziebart, 2010; Levine, 2018), we arrive at a KL-regularized lower bound on the standard RL objective:

$$\mathcal{J}(\pi) = \mathbb{E}_{p_{\text{env}},\pi}\left[\sum_{t=1}^{T} \gamma^t \left(r(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \frac{\pi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O})}{p(\mathbf{a}_t|\mathbf{s}_t)}\right)\right], \tag{7.1}$$

where $\gamma \in [0, 1)$ is the discount factor, α is a Lagrange multiplier controlling the regularization weight, $\mathcal{O}$ is *optimality* (as introduced in the previous chapter), and $p(\mathbf{a}_t|\mathbf{s}_t)$ is an action prior. For simplicity, we again consider a uniform action prior in this chapter (see Section 7.4 for further discussion).

Soft actor-critic (SAC) (Haarnoja, Zhou, Abbeel, et al., 2018) provides one instantiation of this setup, learning a Q-network (critic) to estimate future terms in the objective, and a direct amortized policy network (actor) to optimize the objective. The critic is learned using temporal difference learning, with an ensemble of deep networks (Fujimoto, Hoof, and Meger, 2018), target value networks (Mnih et al., 2015), and an experience replay buffer (Lin, 1992). SAC estimates the policy using a deep network, denoted π_ϕ, which typically takes the form of a conditional Gaussian:

$$\pi_\phi(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}) = \mathcal{N}(\mathbf{a}_t; \boldsymbol{\mu}_\phi(\mathbf{s}_t), \text{diag}(\boldsymbol{\sigma}^2_\phi(\mathbf{s}_t))).$$

In bounded action spaces, `tanh` is typically applied to the policy samples (Haarnoja, Zhou, Abbeel, et al., 2018). The policy network parameters, ϕ, are optimized by differentiating through the objective using the reparameterization gradient estimator (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014).

Motivation

As noted in the previous chapter, performing policy optimization (inference) at the current time step only requires access to an estimate of the objective (Eq. 7.1).

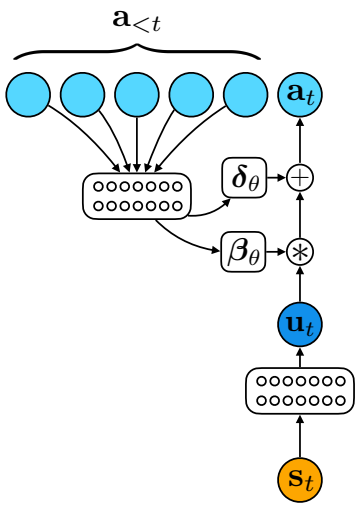


Figure 7.1: **Autoregressive Flow-Based Policy**. An autoregressive flow-based policy converts samples from a base distribution, $\pi_\phi(\mathbf{u}_t|\mathbf{s}_t)$, using an affine transform with parameters $\boldsymbol{\beta}_\theta(\mathbf{a}_{<t})$ and $\boldsymbol{\delta}_\theta(\mathbf{a}_{<t})$, into actions, $\mathbf{a}_t$. The affine transform (top) acts as a feedforward policy, purely conditioned on previous actions, whereas the base distribution (bottom) acts as a feedback policy. Each are parameterized by a separate deep network, with parameters θ and ϕ, respectively.

However, in fully-observable MDPs with continuous action spaces, it is common to use direct amortized policy networks, typically directly mapping the current state to a distribution over actions, i.e., $\pi_\phi(\mathbf{a}_t|\mathbf{s}_t, O)$. This is because $\mathbf{s}_t$ provides a compact description of the information necessary to perform policy optimization. Yet, this is ultimately a design choice, and we are free to condition direct amortization on *any* current or past variables: $\pi_\phi(\mathbf{a}_t|\mathbf{s}_{\leq t}, \mathbf{a}_{<t}, O)$.

In this chapter, we specifically focus on conditioning on previous actions, parameterizing policies of the general form $\pi_\phi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{a}_{<t}, O)$. Although this may, at first, appear to be an odd design choice, there are several possible benefits to parameterizing policies with a feedforward component conditioned on previous actions. First, policy optimization can be computationally costly; having a feedforward policy component may reduce the computational burden on feedback optimization. Second, there may be communication constraints or delays from high-level state information; a feedforward policy may provide a reasonable control estimate quickly. Finally, learning to perform feedback optimization can be challenging; a feedforward policy provides a basis of dynamical motor primitives (Ijspeert, Nakanishi, and Schaal, 2002), which may simplify upstream control learning.

Formulation

We now describe adapting sequential autoregressive flows (Chapter 5) to the control setting. To simplify notation, we omit the dependency on optimality, O, in the description. We consider a policy composed of a state-dependent base distribution, $\pi_\phi(\mathbf{u}_t|\mathbf{s}_t)$, over a latent variable, $\mathbf{u}$, and a dynamical affine autoregressive (feedforward) component, defined by a shift, $\boldsymbol{\delta}_\theta(\mathbf{a}_{<t})$, and scale, $\boldsymbol{\beta}_\theta(\mathbf{a}_{<t})$. To generate action $\mathbf{a}_t$, we sample the latent variable, $\mathbf{u}_t \sim \pi_\theta(\mathbf{u}_t|\mathbf{s}_t)$, then apply the affine transform,

$$\mathbf{a}_t = \boldsymbol{\beta}_\theta(\mathbf{a}_{<t}) \odot \mathbf{u}_t + \boldsymbol{\delta}_\theta(\mathbf{a}_{<t}). \tag{7.2}$$

With the change of variables, the action probability is then

$$\pi_{\phi,\theta}(\mathbf{a}_t|\mathbf{s}_t, \mathbf{a}_{<t}) = \pi_\phi(\mathbf{u}_t|\mathbf{s}_t) \left| \prod_i \beta_{\theta,i}(\mathbf{a}_{<t}) \right|^{-1}, \tag{7.3}$$

where $\beta_{\theta,i}$ is the i^{th} dimension of $\boldsymbol{\beta}_\theta$. As in SAC, we can also apply a final `tanh` transform if the action space is bounded in $[-1, 1]$. In the implementation presented here, the base distribution is a Gaussian, output by a deep network with parameters ϕ, and, similarly, the affine transform parameters are output by a deep network with parameters θ. The overall setup is shown in Figure 7.1.

As we have noted multiple times throughout this thesis, if $\mathbf{u}_t \sim \mathcal{N}(\mathbf{u}_t; \mathbf{0}, \mathbf{I})$, then the affine flow is equivalent to an autoregressive Gaussian model, as Eq. 7.2 is the Gaussian reparameterization trick (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014). For more general base distributions, this serves as a technique for adding temporal dependencies to output sequences, $\mathbf{a}_{1:T}$. Conversely, given a sequence, $\mathbf{a}_{1:T}$, the affine transform provides a mechanism for removing temporal dependencies, i.e., $\mathcal{I}(\mathbf{u}_{1:T}) \leq \mathcal{I}(\mathbf{a}_{1:T})$, as seen in Chapter 5, thereby simplifying estimation in the space of $\mathbf{u}_{1:T}$.

7.3 Experiments

We incorporate the proposed autoregressive feedforward component within soft actor-critic (SAC), which we refer to as autoregressive SAC (ARSAC). The affine transform parameters in ARSAC are conditioned on a varying number of previous actions, which we denote as, e.g., ARSAC-5 for 5 previous actions. Unless otherwise stated, we adopt the default hyperparameters from Haarnoja, Zhou, Abbeel, et al., 2018, as well as the automatic entropy-tuning scheme proposed in Haarnoja, Zhou, Hartikainen, et al., 2018. We evaluate ARSAC across a range of locomotion tasks in environments from the `dm_control` suite (Tassa et al., 2020).

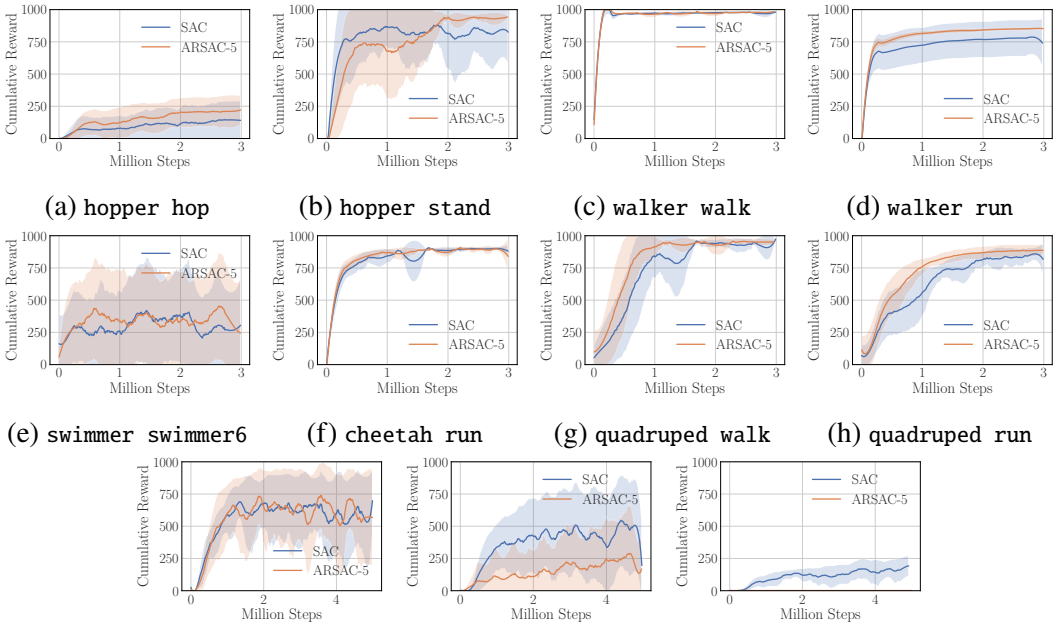
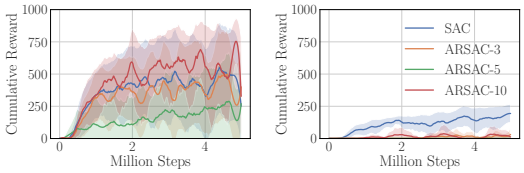


Figure 7.2: **Performance Comparison,** 2×256 **(Default) Policy Network**. Comparison between SAC and ARSAC with an action window of 5 time steps across `dm_control` suite environments. Each curve shows the mean and standard deviation across 5 random seeds.

Performance Comparison

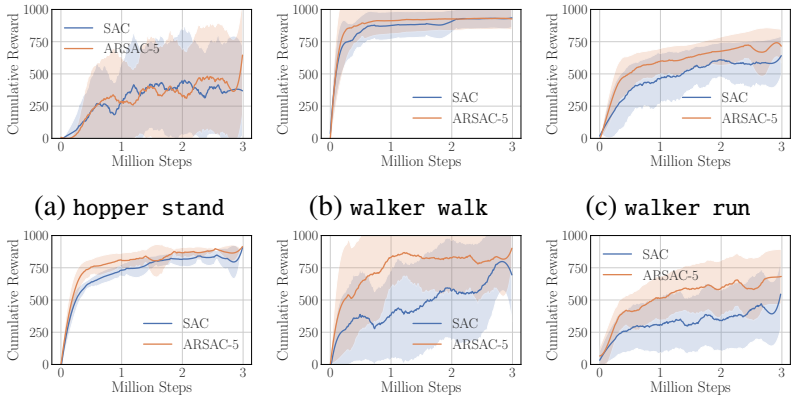
In Figure 7.2, we compare the cumulative reward of SAC and ARSAC-5 throughout training, with 5 random seeds for each setup on each environment. On most environments, ARSAC-5 performs roughly as well or better than SAC, with improved sample efficiency and/or final performance. However, ARSAC-5 struggles on the more difficult tasks, `humanoid walk` and `humanoid run`. In Figure 7.3, we present performance results on these latter environments with varying autoregressive window sizes. We see that changing the size improves performance, however, this is still not able to bridge the performance gap on `humanoid run`.

We hypothesize that the autoregressive transform will have a larger impact when the (state-dependent) base distribution is constrained in some way, either computationally or through delays. To probe this aspect, we decrease the size of the base distribution network, restricting the network to a single hidden layer with 32 units (the default size is 2 hidden layers, each with 256 units). The results, on a subset of environments, are shown in Figure 7.4, where we see that, indeed, the gap in performance slightly increases.



(a) `humanoid walk` (b) `humanoid run`

Figure 7.3: **Humanoid Performance Comparison,** 2×256 **(Default) Policy Network**. Comparison between SAC and ARSAC with varying action window sizes on `humanoid` tasks from `dm_control` suite. Each curve shows the mean and standard deviation across 5 random seeds.



(a) `hopper stand` (b) `walker walk` (c) `walker run`

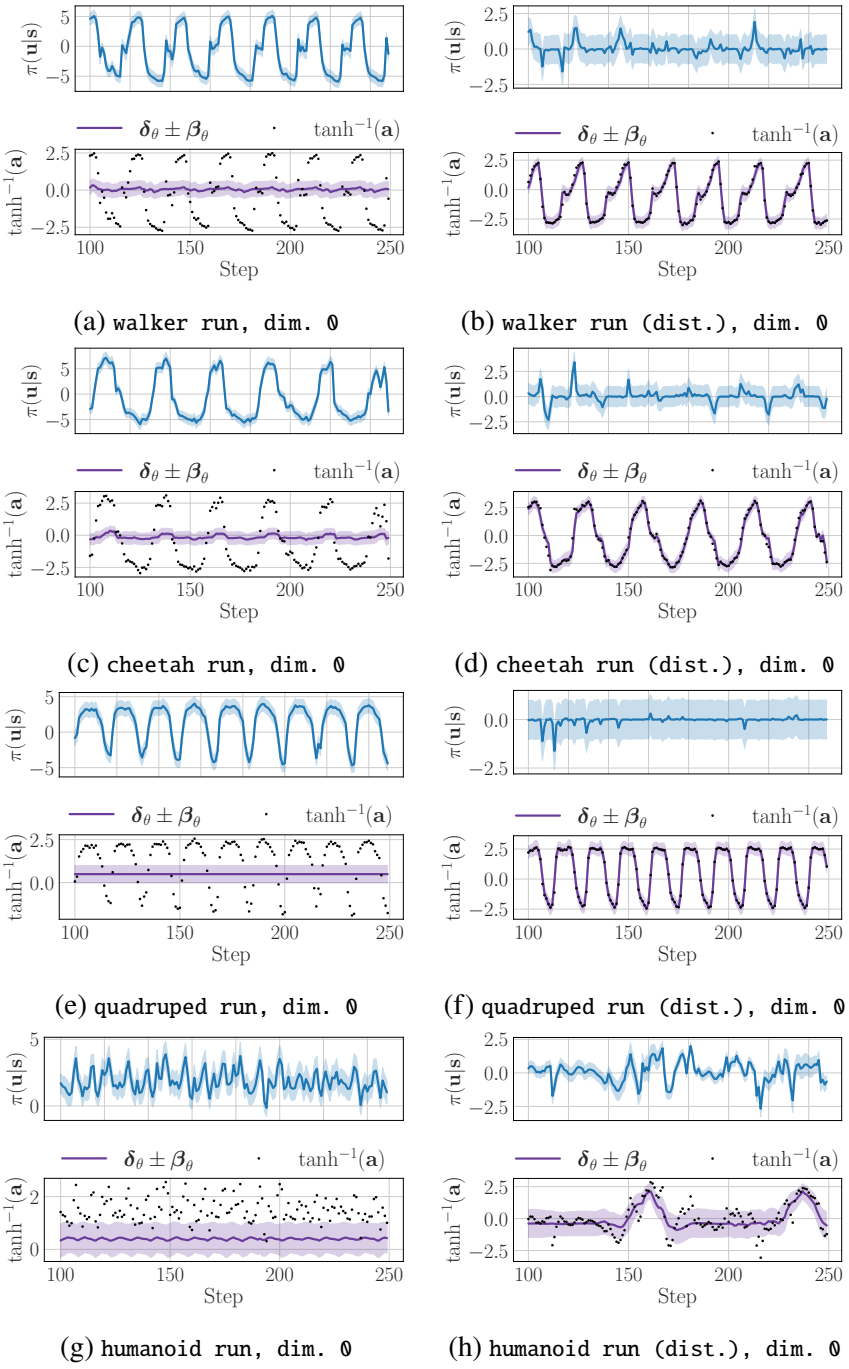
(d) `cheetah run` (e) `quadruped walk` (f) `quadruped run`

Figure 7.4: **Performance Comparison,** 1×32 **Policy Network**. Comparison between SAC and ARSAC with an action window of 5 time steps across `dm_control` suite environments. Each curve shows the mean and standard deviation across 5 random seeds.

Policy Visualization & Feedforward Distillation

We now analyze the policy distributions learned by ARSAC. On the left side of Figure 7.5, we plot the base distribution, $\pi(\mathbf{u}|\mathbf{s})$, as well as the autoregressive affine transform, $\boldsymbol{\delta}_\theta \pm \boldsymbol{\beta}_\theta$, and the actions (before applying the tanh transform). Arbitrarily, we only select the first action dimension for purposes of visualization. We see that the autoregressive flow contains relatively little temporal structure at the end of training. This is not entirely surprising, as there is nothing in the objective that explicitly requires the policy to be distilled into the autoregressive flow. Thus, while the flow is beneficial for training, it may not necessarily automatically provide dynamical "motor primitives," part of the motivation for this approach.

To explore this capability, we attempt to distill the policy entirely into the affine transform, arriving at a purely feedforward policy. We do so by restricting the base



(a) `walker run, dim. 0` (b) `walker run (dist.), dim. 0`

(c) `cheetah run, dim. 0` (d) `cheetah run (dist.), dim. 0`

(e) `quadruped run, dim. 0` (f) `quadruped run (dist.), dim. 0`

(g) `humanoid run, dim. 0` (h) `humanoid run (dist.), dim. 0`

Figure 7.5: **Policy Visualization & Distillation**. **Left**: Visualization of the base distribution and autoregressive flow for various action dimensions across various environments. **Right**: Visualization after policy distillation.

distribution, progressively penalizing the L2 norm of the mean ($\boldsymbol{\mu}_\phi$) and log-standard deviation ($\log \boldsymbol{\sigma}_\phi$), thereby bringing the base distribution toward a standard Gaussian, $\mathcal{N}(\mathbf{u}; \mathbf{0}, \mathbf{I})$. The results are shown on the right side of Figure 7.5, where we see that

the flow now contains nearly all of the temporal structure. In other words, these policies are effectively autoregressive Gaussian densities:

$$\pi_\theta(\mathbf{a}_t|\mathbf{a}_{<t}) = \mathcal{N}(\mathbf{a}_t; \boldsymbol{\delta}_\theta(\mathbf{a}_{<t}), \mathrm{diag}(\boldsymbol{\beta}_\theta^2(\mathbf{a}_{<t}))), \tag{7.4}$$

almost entirely independent of the state, $\mathbf{s}$. Comparing the left and right columns, we see that the feedforward policies are more consistent overall, resulting from the limited temporal window of the affine flow. Thus, the autoregressive nature of the policy provides an inductive bias, resulting in policies with more regular rhythmic structure. As will be described in Chapter 8, this may hold some connections with central pattern generator circuits in biological neural systems, which produce rhythmic patterns in the absence of any state input (Marder and Bucher, 2001).

7.4 Discussion

This chapter has presented a formulation of learned feedforward control based on sequential autoregressive flows. This approach decomposes the policy into a feedforward component, purely conditioned on previous actions, and a feedback component, conditioned on the current state, which are combined through an affine (linear) transform. In the initial set of experiment presented here on locomotion tasks, we have shown that such policies can outperform purely feedback-based counterparts. This is somewhat unsurprising, considering that these flow-based policies condition on additional variables and contain additional parameters. However, it is also somewhat surprising, considering that, in these environments, the current state is sufficient for estimating the optimal policy. This suggests that incorporating a feedforward policy component is a useful inductive bias for these tasks, which is reasonable, as they require periodic policies. Through qualitative analyses, we have observed that this periodic structure is not ultimately preserved in the flow. However, we have shown that it is feasible to distill the policy almost entirely into the flow. This may be advantageous when computational costs or temporal delays prohibit excessive amounts of feedback control. We also expect this approach to be more useful in environments with relatively minimal amounts of stochasticity (like `dm_control`), where state information can eventually be ignored. Future works may wish to consider other forms of normalizing flows, e.g., non-affine flows, flows with multiple transforms, or combining both spatial and temporal normalizing flows. We also recommend exploring the use of the prior, $p(\mathbf{a}_t|\cdot)$, as the feedforward component, as this will naturally preserve the dynamical structure of the policy. However, there are added challenges associated with training the prior, which will need to be resolved

(Abdolmaleki et al., 2018). At the very least, our results show that feedforward policies are practically possible, and exploring additional computational architectures for policies is a fruitful direction for future research.

References

Abdolmaleki, Abbas et al. (2018). "Relative entropy regularized policy iteration". In: *arXiv preprint arXiv:1812.02256*.

Fujimoto, S, H van Hoof, and D Meger (2018). "Addressing function approximation error in actor-critic methods". In: *Proceedings of Machine Learning Research* 80, pp. 1587–1596.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, et al. (2018). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning*, pp. 1856–1865.

Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen, et al. (2018). "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905*.

Ijspeert, Auke Jan, Jun Nakanishi, and Stefan Schaal (2002). "Movement imitation with nonlinear dynamical systems in humanoid robots". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 2. IEEE, pp. 1398–1403.

Kingma, Durk P and Max Welling (2014). "Stochastic gradient VB and the variational auto-encoder". In: *Proceedings of the International Conference on Learning Representations*.

Levine, Sergey (2018). "Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review". In: *arXiv preprint arXiv:1805.00909*.

Lin, Long-Ji (1992). "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine learning* 8.3-4, pp. 293–321.

Marder, Eve and Dirk Bucher (2001). "Central pattern generators and the control of rhythmic movements". In: *Current biology*.

Merel, Josh, Matthew Botvinick, and Greg Wayne (2019). "Hierarchical motor control in mammals and machines". In: *Nature Communications* 10.1, pp. 1–12.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, p. 529.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models". In: *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286.

Shalit, Uri et al. (2012). "Descending systems translate transient cortical commands into a sustained muscle activation signal". In: *Cerebral cortex* 22.8, pp. 1904–1914.

Tassa, Yuval et al. (2020). “dm_control: Software and tasks for continuous control”. In: *arXiv preprint arXiv:2006.12983*.

Ziebart, Brian D (2010). “Modeling purposeful adaptive behavior with the principle of maximum causal entropy”. PhD thesis. CMU.

Part IV

Discussion

Chapter 8

CONNECTIONS TO PREDICTIVE CODING & NEUROSCIENCE

Marino, Joseph (2019). "Predictive Coding, Variational Autoencoders, and Biological Connections". In: *NeurIPS Workshop on Real Neurons and Hidden Units*. URL: https://openreview.net/forum?id=SyeumQYUUH.

8.1 Introduction

The previous chapters brought feedback and feedforward ideas from predictive coding into the realm of machine learning. In this chapter, we more explicitly identify these connections, implying surprising analogies between machine learning and neuroscience. We start in Section 8.2 by describing predictive coding, its hypothesized connections to neuroscience, and the empirical evidence for this overall theory. In Section 8.3, we then connect the ideas developed in this thesis back to their origins in predictive coding. By connecting these areas through this conceptual bridge, we arrive at new perspectives on the possible correspondences between machine learning and neuroscience (Section 8.4). These correspondences draw into question fundamental assumptions on the analogy of biological and artificial neurons (McCulloch and Pitts, 1943), with a host of implications for learning, inference, inhibition, etc. Unlike the technical contributions in Parts II & III, this chapter is more speculative, meant to serve as a starting point for further cross-pollination between machine learning and neuroscience. Like the works of Broeke, 2016 and Lotter, Kreiman, and Cox, 2018, we hope that these ideas will inspire future research in exploring this promising paradigm.

8.2 Predictive Coding

Predictive coding, as described within neuroscience, can be divided into two separate settings, spatiotemporal and hierarchical, corresponding to the two main forms of structured probabilistic dependencies (Chapter 2). We review each of these settings, discussing previously hypothesized correspondences with neural anatomy. Finally, we outline the current empirical support for predictive coding in neural systems, highlighting the need for large-scale, testable models.

Spatiotemporal Predictive Coding

Spatiotemporal predictive coding (Srinivasan, Laughlin, and Dubs, 1982), as the name implies, involves forming predictions across spatial dimensions and temporal sequences. These predictions then produce the resulting "code" as the prediction error. Concretely, in the temporal setting, we can consider a Gaussian autoregressive model, p_θ, defined over observation sequences, $\mathbf{x}_{1:T}$. The conditional probability at time t can be written as

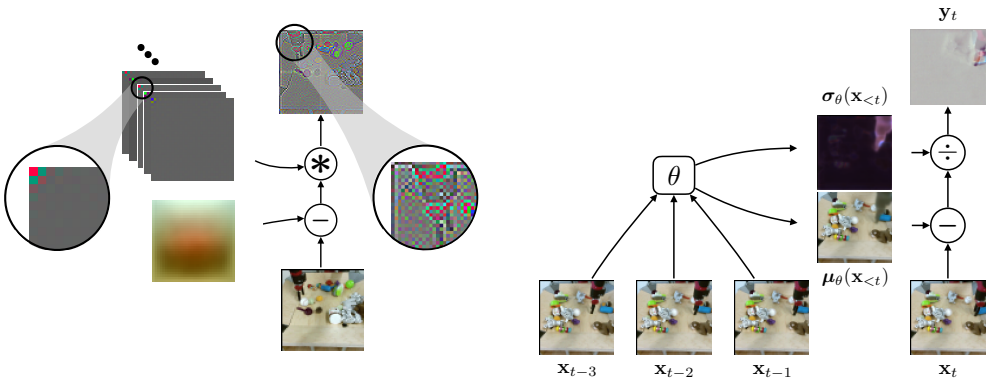
$$p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}), \text{diag}(\boldsymbol{\sigma}^2_\theta(\mathbf{x}_{<t}))).$$

Introducing auxiliary variables, $\mathbf{y}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we can use the reparameterization trick to express $\mathbf{x}_t = \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) + \boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \odot \mathbf{y}_t$, where $\odot$ denotes element-wise multiplication. Conversely, we can express the inverse, normalization or *whitening* transform as

$$\mathbf{y}_t = \frac{\mathbf{x}_t - \boldsymbol{\mu}_\theta(\mathbf{x}_{<t})}{\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})}. \tag{8.1}$$

An example of temporal normalization with video, adapted from Chapter 5, is shown in Figure 8.1b. Note that one special case of this transform involves setting $\boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) \equiv \mathbf{x}_{t-1}$ and $\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \equiv \mathbf{1}$, in which case, $\mathbf{y}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$, i.e. temporal differences. For sequences that change slowly relative to the temporal step-size, this is a reasonable assumption. As noted in Chapter 5, this inverse transform can remove temporal redundancy in the input sequence. Thus, by Shannon's source coding theorem (Shannon, 1948), we can encode or compress $\mathbf{y}_{1:T}$ more efficiently than $\mathbf{x}_{1:T}$. The benefit of this sequential predictive coding scheme was recognized in the early days of information theory (Harrison, 1952; Oliver, 1952), forming the basis of modern video (Wiegand et al., 2003) and audio (Atal and Schroeder, 1979) compression.

A similar process can also be applied within $\mathbf{x}_t$ to remove spatial dependencies. For instance, we could also apply an autoregressive affine transform over spatial dimensions, predicting the i^{th} dimension, $x_{i,t}$, as a function of previous spatial dimensions, $\mathbf{x}_{1:i,t}$. With linear functions, this corresponds to Cholesky whitening (Pourahmadi, 2011; Kingma, Salimans, et al., 2016). However, this requires imposing an arbitrary ordering over spatial dimensions. Perhaps a more reasonable approach in the spatial setting is to learn a set of *symmetric* dependencies between dimensions. Here, the linear case corresponds to ZCA whitening (Kessy, Lewin, and Strimmer, 2018), shown in Figure 8.1a. In the natural image domain, both of these whitening schemes generally result in center-surround spatial filters, extracting edges from the input.



(a) Spatial (b) Temporal

Figure 8.1: **Spatiotemporal Predictive Coding**. **(a)** Spatial predictive coding models and removes spatial dependencies. In the domain of natural images, one version of linear predictive coding is ZCA whitening, which yields center-surround filters (left). As a result, the whitened image contains highlighted edges (right). **(b)** Temporal predictive coding models and removes temporal dependencies. In the domain of natural video, this tends to remove static backgrounds.

Srinivasan, Laughlin, and Dubs, 1982 investigated the principles of spatiotemporal predictive coding in the retina, where compression is essential for transmission through the optic nerve. Estimating the auto-correlation function of input sensory signals, i.e. a linear prediction, they showed that spatiotemporal predictive coding provides a reasonable fit to retinal ganglion cell recordings from flies, allowing the retina's output neurons to more fully utilize their dynamic range. It is now generally accepted that retina, in part, performs stages of spatial and temporal normalization through center-surround receptive fields and on-off responses (Hosoya, Baccus, and Meister, 2005; Graham, Chandler, and Field, 2006; Pitkow and Meister, 2012; Palmer et al., 2015). Dong and Atick, 1995 applied similar predictive coding ideas to the thalamus, proposing an additional stage of temporal normalization. Likewise, Friston's use of generalized coordinates (K. Friston, 2008a), i.e. modeling multiple orders of temporal derivatives, can be approximated using finite temporal differences through repeated application of predictive coding. That is, $\frac{d\mathbf{x}}{dt} \approx \Delta\mathbf{x}_t \equiv \mathbf{x}_t - \mathbf{x}_{t-1}$. Thus, spatiotemporal predictive coding may be utilized at multiple stages of sensory processing to remove redundancy (Y. Huang and Rao, 2011).

In neural circuits, spatiotemporal normalization often involves inhibitory interneurons (Carandini and Heeger, 2012), carrying out operations similar to those in Eq. 8.1 (though other mechanisms are also possible). For instance, retinal inhibitory interactions take place between photoreceptors, via horizontal cells, and between

bipolar cells, via amacrine cells. This enables unpredicted motion to be computed, e.g. an object moving relative to the background, using inhibitory interactions from amacrine cells (Ölveczky, Baccus, and Meister, 2003; Baccus et al., 2008). Similar inhibitory interactions are present in the lateral geniculate nucleus (LGN) in thalamus, with interneurons inhibiting relay cells originating from retina (Sherman and Guillery, 2002). As mentioned above, this is thought to implement a form of temporal normalization (Dong and Atick, 1995), removing, at least, linear dependencies (Dan, Atick, and Reid, 1996). Inhibition via lateral inhibitory interactions is also a prominent feature of neocortex, with distinct classes of local interneurons playing a significant role in shaping the responses of principal pyramidal neurons (Isaacson and Scanziani, 2011). While these distinct classes may serve separate computational roles, part of this purpose appears to be for spatiotemporal normalization (Carandini and Heeger, 2012). Finally, while we have focused largely on early stages of sensory processing, inhibitory interneurons are also prevalent in other areas of neocortex, as well as in central pattern generator (CPG) circuits (Marder and Bucher, 2001), found in the spinal cord. These circuits are repsonsible for the rhythmic generation of movement, such as locomotion. Thus, just as inhibitory interactions *remove* spatiotemporal dependencies in early sensory areas, similar computational operations can *add* spatiotemporal dependencies in motor activation.

Hierarchical Predictive Coding

The other main form of predictive coding, mathematically formulated by Rao and Ballard, 1999; K. Friston, 2005, involves hierarchies of latent variables and, as such, has been postulated as a model of hierarchical cortical processing. The neocortex (Figure 8.2) is a sheet-like structure involved in many aspects of sensory and motor processing. It is composed of six layers (I–VI), containing particular classes of neurons and connections. Across layers, neurons are arranged into columns, which are engaged in related computations (Mountcastle, Berman, and Davies, 1955). Columns interact locally via inhibitory interactions from interneurons while also forming processing hierarchies through longer-range excitatory interactions from pyramidal neurons. Particularly in earlier sensory areas, longer-range connections are generally grouped into forward (up the hierarchy) and backward (down the hierarchy) directions. Forward connections are traditionally thought to be driving (evoking neural activity) (Girard and Bullier, 1989; Girard, Salin, and Bullier, 1991). Backward connections are traditionally thought to be modulatory, however, they have also been shown be driving (Covic and Sherman, 2011; De Pasquale and Sherman,

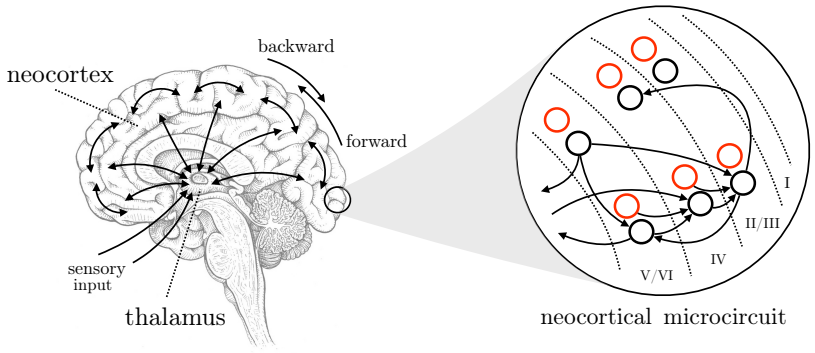


Figure 8.2: **Brain Anatomy & Cortical Circuitry**. **Left**: Sensory inputs enter first-order relays in thalamus from sensory organs. Thalamus forms reciprocal connections with neocortex. Neocortex consists of hierarchies of cortical areas, with both forward and backward connections. **Right**: Neocortex is composed of six layers (I–VI), with specific neuron classes and connections at each layer. The simplified schematic depicts two cortical columns. Black and red circles represent excitatory and inhibitory neurons respectively, with arrows denoting major connections. This basic circuit motif is repeated with slight variations throughout neocortex.

2011), which can be inverted through inhibition (H. S. Meyer et al., 2011). These sets of connections, repeated with slight variations throughout neocortex, constitute a canonical *neocortical microcircuit* (Douglas, Martin, and Whitteridge, 1989), which could suggest a single processing algorithm (Hawkins and Blakeslee, 2004), capable of adapting to a variety of inputs (Sharma, Angelucci, and Sur, 2000).

In formulating a theory of neocortex, Mumford, 1992 proposed that thalamus acts as an 'active blackboard,' with the cortical hierarchy attempting to reconstruct or predict the thalamic input and activity in areas throughout the hierarchy. Backward (top-down) projections would convey predictions, while forward (bottom-up) projections would use prediction errors to update the estimates throughout the hierarchy. Through a dynamic process of activation, the entire system would settle to a consistent pattern of activity, minimizing prediction error. Over longer periods of time, the model parameters would be adjusted to yield improved predictions. In this way, cortex would use negative feedback, both in inference and learning, to use and construct a generative model of its inputs. This notion of generative state estimation dates back (at least) to Helmholtz (Von Helmholtz, 1867), and the notion of correcting predictions based on prediction errors is inline with concepts from cybernetics (Wiener, 1948; MacKay, 1956), which influenced techniques like Kalman filtering (Kalman, 1960), a ubiquitous Bayesian filtering algorithm.

A more complete mathematical formulation of this hierarchical predictive coding

model, with many similarities to Kalman filtering (see Rao, 1998), was provided by Rao and Ballard, 1999, with the generalization to variational inference provided by K. Friston, 2005. To illustrate this setup, consider a simple model consisting of a single level of continuous latent variables, $\mathbf{z}$, modeling continuous data observations, $\mathbf{x}$. We will use Gaussian densities for each distribution and assume we have

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f(\mathbf{W}\mathbf{z}), \text{diag}(\boldsymbol{\sigma}_\mathbf{x}^2)), \tag{8.2}$$

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\mathbf{z}, \text{diag}(\boldsymbol{\sigma}_\mathbf{z}^2)), \tag{8.3}$$

where f is an element-wise function (e.g. logistic sigmoid, tanh, or the identity), $\mathbf{W}$ is a weight matrix, $\boldsymbol{\mu}_\mathbf{z}$ is the constant prior mean, and $\boldsymbol{\sigma}_\mathbf{x}^2$ and $\boldsymbol{\sigma}_\mathbf{z}^2$ are constant vectors of variances.

In the simplest approach to inference, we can find the maximum-a-posteriori (MAP) estimate, i.e. estimate the $\mathbf{z}^*$ which maximizes $p_\theta(\mathbf{z}|\mathbf{x})$. While we cannot tractably evaluate $p_\theta(\mathbf{z}|\mathbf{x})$ directly, we can use Bayes' rule to write

$$\begin{aligned}
\mathbf{z}^* &= \arg\max_\mathbf{z} p_\theta(\mathbf{z}|\mathbf{x}) \\
&= \arg\max_\mathbf{z} \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})} \\
&= \arg\max_\mathbf{z} p_\theta(\mathbf{x}, \mathbf{z}).
\end{aligned}$$

Thus, rather than evaluating the posterior distribution, $p_\theta(\mathbf{z}|\mathbf{x})$, we can perform this maximization using the joint distribution, $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, which we can tractably evaluate. We can also replace the optimization over the probability distribution with an optimization over the log probability, since $\log(\cdot)$ is a monotonically increasing function and will not affect the optimization. We then have

$$\begin{aligned}
\mathbf{z}^* &= \arg\max_\mathbf{z} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z})\right]. \\
&= \arg\max_\mathbf{z} \left[\log \mathcal{N}(\mathbf{x}; f(\mathbf{W}\mathbf{z}), \text{diag}(\boldsymbol{\sigma}_\mathbf{x}^2)) + \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\mathbf{z}, \text{diag}(\boldsymbol{\sigma}_\mathbf{z}^2))\right].
\end{aligned}$$

Each of the terms in this objective is a weighted squared error. For instance, the first term is the weighted squared error in reconstructing the data observation:

$$\log \mathcal{N}(\mathbf{x}; f(\mathbf{W}\mathbf{z}), \text{diag}(\boldsymbol{\sigma}_\mathbf{x}^2)) = \frac{-n_\mathbf{x}}{2}\log(2\pi) - \frac{1}{2}\log|\text{diag}(\boldsymbol{\sigma}_\mathbf{x}^2)| - \frac{1}{2}\left\|\frac{\mathbf{x} - f(\mathbf{W}\mathbf{z})}{\boldsymbol{\sigma}_\mathbf{x}}\right\|_2^2,$$

where $n_\mathbf{x}$ is the dimensionality of $\mathbf{x}$ and $||\cdot||_2^2$ denotes the squared L2 norm. Plugging

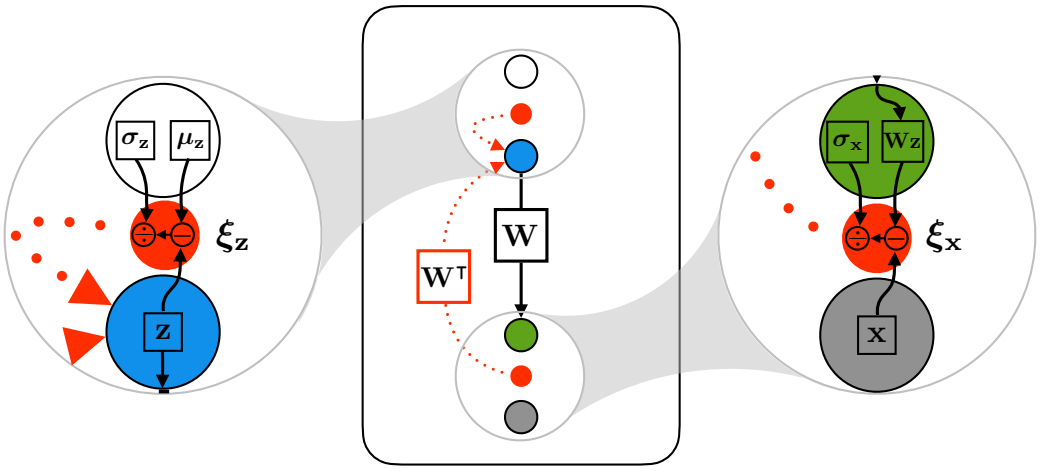


Figure 8.3: **Hierarchical Predictive Coding**. The diagram shows the basic computation graph for a Gaussian latent variable model with MAP inference. The insets show the weighted error calculation for the latent (left) and observed (right) variables.

these terms into the objective and dropping terms that do not depend on $\mathbf{z}$ yields

$$\begin{aligned}\mathbf{z}^* &= \arg\max_{\mathbf{z}} \left[\frac{-1}{2} \left\| \frac{\mathbf{x} - f(\mathbf{W}\mathbf{z})}{\boldsymbol{\sigma}_\mathbf{x}} \right\|_2^2 - \frac{1}{2} \left\| \frac{\mathbf{z} - \boldsymbol{\mu}_\mathbf{z}}{\boldsymbol{\sigma}_\mathbf{z}} \right\|_2^2 \right], \\ &= \arg\max_{\mathbf{z}} \mathscr{L}(\mathbf{z}; \theta),\end{aligned} \tag{8.4}$$

where we have defined the objective as $\mathscr{L}(\mathbf{z}; \theta)$. For purposes of illustration, let us assume that $f(\cdot)$ is the identity function, i.e. $f(\mathbf{W}\mathbf{z}) = \mathbf{W}\mathbf{z}$. We can then evaluate the gradient of $\mathscr{L}(\mathbf{z}; \theta)$ w.r.t. $\mathbf{z}$, yielding

$$\nabla_\mathbf{z} \mathscr{L}(\mathbf{z}; \theta) = \mathbf{W}^\intercal \left(\frac{\mathbf{x} - \mathbf{W}\mathbf{z}}{\boldsymbol{\sigma}_\mathbf{x}} \right) - \frac{\mathbf{z} - \boldsymbol{\mu}_\mathbf{z}}{\boldsymbol{\sigma}_\mathbf{z}}.$$

The transposed weight matrix, $\mathbf{W}^\intercal$, comes from differentiating $\mathbf{W}\mathbf{z}$, and translates the error in reconstruction into an update in $\mathbf{z}$. If we define the following terms as weighted errors:

$$\boldsymbol{\xi}_\mathbf{x} \equiv \frac{\mathbf{x} - \mathbf{W}\mathbf{z}}{\boldsymbol{\sigma}_\mathbf{x}}, \qquad\qquad \boldsymbol{\xi}_\mathbf{z} \equiv \frac{\mathbf{z} - \boldsymbol{\mu}_\mathbf{z}}{\boldsymbol{\sigma}_\mathbf{z}},$$

then we can re-write the gradient using these terms:

$$\nabla_\mathbf{z} \mathscr{L}(\mathbf{z}; \theta) = \mathbf{W}^\intercal \boldsymbol{\xi}_\mathbf{x} - \boldsymbol{\xi}_\mathbf{z}. \tag{8.5}$$

Thus, if we want to perform inference using gradient-based optimization, e.g. $\mathbf{z} \leftarrow \mathbf{z} + \alpha \nabla_{\mathbf{z}} \mathscr{L}(\mathbf{z}; \theta)$, we need 1) the weighted errors, $\boldsymbol{\xi}_{\mathbf{x}}$ and $\boldsymbol{\xi}_{\mathbf{z}}$, and 2) the transposed weights, $\mathbf{W}^\intercal$, or more generally, the Jacobian of the conditional likelihood mean. This overall scheme is depicted in Figure 8.3, using the computational graph format from previous chapters.

To learn the weight parameters, we can differentiate $\mathscr{L}(\mathbf{z}; \theta)$ (Eq. 8.4) w.r.t. $\mathbf{W}$:

$$\begin{aligned} \nabla_{\mathbf{W}} \mathscr{L}(\mathbf{z}; \theta) &= \left(\frac{\mathbf{x} - \mathbf{W}\mathbf{z}}{\boldsymbol{\sigma}_{\mathbf{x}}} \right) \mathbf{z}^\intercal \\ &= \boldsymbol{\xi}_{\mathbf{x}} \mathbf{z}^\intercal . \end{aligned}$$

This gradient is the product of a local error term, $\boldsymbol{\xi}_{\mathbf{x}}$, and the latent variable, $\mathbf{z}$, possibly suggesting a biologically-plausible learning rule (Whittington and Bogacz, 2017).

Predictive coding identifies the conditional likelihood (Eq. 8.2) with backward (or top-down) cortical projections, whereas inference updating (Eq. 8.5) is identified with forward (or bottom-up) cortical projections (K. Friston, 2005). Such connections are thought to be mediated by pyramidal neurons. Scaling this model up in size and structure, each cortical column could contain the necessary computational elements involved in predicting and estimating a latent variable. Interneurons within columns could be involved in error calculation ($\boldsymbol{\xi}_{\mathbf{x}}$ and $\boldsymbol{\xi}_{\mathbf{z}}$). Although we only discussed diagonal covariance matrices ($\boldsymbol{\sigma}_{\mathbf{x}}^2$ and $\boldsymbol{\sigma}_{\mathbf{z}}^2$), interneurons involved in local lateral inhibition could parameterize (the inverse of) full covariance matrices, i.e. $\boldsymbol{\Sigma}_{\mathbf{x}}$ and $\boldsymbol{\Sigma}_{\mathbf{z}}$. This is an instance of *spatial* predictive coding, which we discussed in the previous section. Note that these factors, which weight $\boldsymbol{\xi}_{\mathbf{x}}$ and $\boldsymbol{\xi}_{\mathbf{z}}$, effectively modulate the *gain* of each error term, possibly providing a form of "attention" (Feldman and K. Friston, 2010). Possible neural correspondences are summarized in Table 8.1.

Here, we have discussed a simplified model of hierarchical predictive coding, with a single latent level and no dynamics. However, a full theory of hierarchical predictive coding would include these additional aspects and others. Karl Friston has explored various design choices throughout multiple papers (K. Friston et al., 2007; K. Friston, 2008a; K. Friston, 2008b), yet the core aspects of probabilistic generative modeling and variational inference remain largely the same. Further elaborating and comparing these design choices will be essential for empirically validating the details of hierarchical predictive coding.

Table 8.1: Proposed Neural Correspondences of Hierarchical Predictive Coding.

Neuroscience	Predictive Coding
Top-Down Cortical Projections	Generative Model Conditional Mapping
Bottom-Up Cortical Projections	Inference Updating
Lateral Inhibition	Covariance Matrices
(Pyramidal) Neuron Activity	Latent Variable Estimates & Errors
Cortical Column	Corresponding Estimate & Error

Empirical Support

Empirically validating predictive coding in neural circuits is an active area of research. This remains challenging, as it is difficult to disentangle the theory itself from the wide array of possible design choices, e.g. distributions, parameterizations, etc. (Gershman, 2019). Nevertheless, many of the core aspects of predictive coding do appear to have *some* empirical support. We briefly outline some of these studies here, but we refer the reader to the multiple review papers on the topic (Y. Huang and Rao, 2011; Bastos, Usrey, et al., 2012; Clark, 2013; Keller and Mrsic-Flogel, 2018; Walsh et al., 2020).

Spatiotemporal As discussed above, various works have investigated spatiotemporal predictive coding in early sensory areas, primarily retina (Srinivasan, Laughlin, and Dubs, 1982; Atick and Redlich, 1992). This typically involves fitting retinal ganglion cell responses to a spatial whitening (or decorrelation) process (Graham, Chandler, and Field, 2006; Pitkow and Meister, 2012), which is dynamically adjusted based on lighting conditions (Hosoya, Baccus, and Meister, 2005). Similar analyses suggest that retina employs temporal predictive coding as well (Srinivasan, Laughlin, and Dubs, 1982; Palmer et al., 2015). While the exact mathematical details of these neural computations have not been fully characterized, the corresponding models contain stages of linear decorrelating filters (e.g. center-surround) followed by non-linearities. Importantly, non-linearities have been shown to be an essential aspect in explaining retinal ganglion cell responses (Pitkow and Meister, 2012), possibly inducing an added degree of sparsity (Graham, Chandler, and Field, 2006). As previously noted, similar spatiotemporal predictive coding computations may be found in thalamus (Dong and Atick, 1995) and cortex. While Dan, Atick, and Reid, 1996 provide some supporting evidence, such investigations are complicated by the presence of backward and modulatory interactions.

Hierarchical Early work toward empirically validating hierarchical predictive coding came from explaining extra-classical receptive field effects (Rao and Ballard, 1999; Rao and Sejnowski, 2002), whereby top-down processing in cortex can alter classical visual receptive fields, suggesting that top-down influences play an important role in sensory processing (Gilbert and Sigman, 2007). Likewise, temporal influences have been demonstrated in the form of repetition suppression (Summerfield et al., 2006), in which cortical activity diminishes in response to repeated, i.e. predictable, stimuli. This effect may reflect the suppression of errors through improved predictions. Predictive coding has also been postulated as an explanation of biphasic responses in LGN (Jehee and Ballard, 2009), in which reversing the visual input with an anti-correlated image results in a large neural response, presumably due to prediction errors. Predictive signals have been documented in auditory (Wacongne et al., 2011) and visual (T. Meyer and Olson, 2011) processing. Activity seemingly corresponding to prediction errors has also been observed in a variety of areas and contexts, including visual flow in primary visual cortex in mice (Keller, Bonhoeffer, and Hübener, 2012; Zmarz and Keller, 2016), auditory cortex in monkeys (Eliades and Wang, 2008) and rodents (Parras et al., 2017), and visual cortex in humans (S. O. Murray et al., 2002; Alink et al., 2010; Egner, Monti, and Summerfield, 2010). While further studies are needed, it appears that sensory cortex is engaged in some form of hierarchical and temporal prediction, with prediction error signals playing a key role in driving the perceptual process.

The empirical evidence for spatiotemporal and hierarchical predictive coding is suggestive, but given the complexity of neural systems, some aspects of the theory are undoubtedly incorrect, incomplete, or under-specified. In particular, the complexity of biological systems makes it is difficult to isolate and assess detailed aspects of predictive coding. For instance, it appears that cortex calculates some form of prediction error, but without access to fine-grained recordings of all relevant signals, e.g. dendritic currents, neuromodulators, etc., it is difficult to determine the exact computational form of the circuit. Thus, while general aspects of predictive coding appear supported, we are unable to probe into the details of such models, making predictive coding a largely *normative* theory. One of the purposes of this chapter, and this thesis more broadly, has been to establish connections between predictive coding and machine learning. Ideally, by building larger-scale models and training them on similar sensory data, we can form more fine-grained empirical predictions for biological neural systems. Building off of the example of Rao and Ballard, 1999, Lotter, Kreiman, and Cox, 2018 provided another step in this direction, comparing

the responses of neural systems and their hierarchical predictive coding model. In the current thesis, we have attempted to help further build the foundation for this collaborative effort.

8.3 Connections

We now discuss the connections between the ideas developed in this thesis and their inspirations from predictive coding.

Iterative Amortization

Iterative amortized inference (Chapter 3) was inspired by the inference scheme proposed by Rao and Ballard, 1999 and K. Friston, 2005. In these early formulations of hierarchical predictive coding, approximate inference is performed using gradient-based optimization of a point estimate of the latent variables. These works made it clear that prediction (or reconstruction) errors drive both inference and learning optimization, and this procedure can be readily extended to sequential settings (K. Friston, 2008b). However, such procedures typically assume that the inference gradients, supplied by forward connections, can be easily calculated, but the weights of these forward connections are, in fact, the Jacobian of the backward connections (Rao and Ballard, 1999) (Section 8.2). This is an example of the *weight transport* problem (Grossberg, 1987), i.e. the weights of one set of connections (forward) depends on the weights from another set of connections (backward). This is generally regarded as not being biologically-plausible.

Amortization (Dayan et al., 1995) provides a simple solution to this problem: learn to perform inference optimization. That is, rather than transporting the generative weights to the inference connections, amortization learns a separate set of inference weights, potentially using similar local learning rules (Y. Bengio, 2014; Lee et al., 2015). Thus, despite criticism from K. Friston, 2018, amortization may offer a more biologically-plausible account of inference. Further, as demonstrated in Chapters 3, 4, and 6, by using non-linear functions, amortization is capable of automatically adjusting update step sizes, yielding accurate estimates with exceedingly few inference iterations. These substantial benefits in computational efficiency provide another argument for amortization over the gradient-based schemes often employed in predictive coding.

Iterative amortization is an example of the more general approach of negative feedback. As noted at the beginning of this thesis, negative feedback was the core concept of cybernetics, which went on to inspire predictive coding. While hierarchical

predictive coding has largely focused on perceptual inference in cortex, the principles of negative feedback appear to apply more broadly to neural systems. Indeed, even at the outset of cybernetics, it was clear that cerebellum plays a central role in negative feedback control (Wiener, 1948). From more recent studies of cerebellum and other cerebellum-like structures (Ito, 1998; Bell, 2001; Kennedy et al., 2014), we are beginning to understand how such circuits correct sensorimotor prediction errors. One prominent example is given by the Purkinje cells of the cerebellum, which appear to take in error signals as inputs and output motor corrections. This follows the general paradigm of iterative amortization, mapping errors to updates. Casting these neural circuits in terms of amortization, i.e. learned negative feedback, may provide insights into how such error-correcting mechanisms are learned from experience.

Sequential Autoregressive Flows

The technique of sequential autoregressive flows (Chapter 5) was, in part, inspired by the temporal normalization schemes from Srinivasan, Laughlin, and Dubs, 1982 and Dong and Atick, 1995, which are thought to occur in retina and first-order relays of thalamus. Unlike these earlier works, which were limited to linear functions of previous inputs, sequential autoregressive flows can utilize non-linear functions to parameterize the normalizing affine transform. Likewise, because these feedforward transforms are learned using the prediction errors on the normalized variables, they can adapt to meet the demands or limitations of higher-level models (Figure 5.6a).

A related technique is that of generalized coordinates (K. Friston, 2008a), decomposing a sequence into its temporal derivatives. Friston has suggested that this may be a general modeling technique employed by neural circuits. As we have seen, a simplified version of sequential autoregressive flows, using the previous variable as the affine shift, extracts an approximation of temporal derivatives (Section 5.2). Thus, given a short enough time step, $\Delta t = t_1 - t_0$, sequential autoregressive flows provide a technique for automatically learning an approximation of generalized coordinates.

In Chapter 7, we saw that the *same* technique can be applied to control, serving as a low-level dynamical policy, i.e. a dynamical motor basis for control. While early sensory processing and motor processing are often considered separately, we see that spatiotemporal dependencies are central to both areas. Normalization (and its inverse) plays a singular role in both cases, simplifying estimation for upstream models. If normalization is truly a canonical neural computation (Carandini and Heeger,

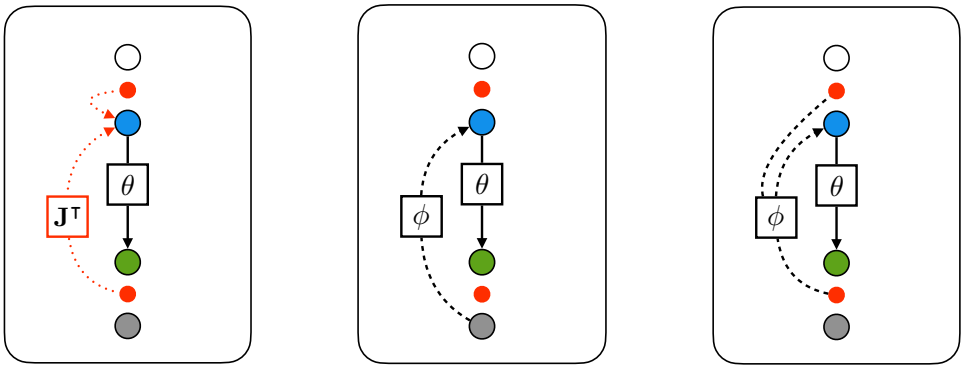
2012), then similar temporal normalization operations may parameterize dynamics estimation throughout cortex, operating in conjunction with spatial normalization. We saw this in Chapter 5, where the modified VideoFlow model (Kumar et al., 2020) utilized *spatial* normalization within time steps and *temporal* normalization across time steps, demonstrating the utility of these complementary procedures.

We have exclusively investigated normalizing flows with affine transforms (Dinh, Sohl-Dickstein, and S. Bengio, 2017), due to their simplicity and their similarity to proposed neural computations. Given their connection with Gaussian densities and Gaussianization (Chen and Gopinath, 2001) (*normal*ization is, ultimately, the process of converting a data distribution into a standard *Normal* (Gaussian) density), transforms of this form are prominent in the statistics literature (Friedman, 1987; Kessy, Lewin, and Strimmer, 2018). However, the change of variables formula readily applies to all invertible transforms, including non-affine transforms. This more general perspective, afforded by normalizing flows, offers a method toward improving spatiotemporal normalization in current models. Similarly, it suggests that neural circuits may implement multiple computational forms of normalization transforms, ranging from simple, affine or constant transforms to more complex, non-affine transforms (Durkan et al., 2019). These may rely on computational mechanisms within individual neurons as well as spatiotemporal interactions between neurons.

8.4 Correspondences

Having drawn connections between the machine learning approaches developed in this thesis and their inspirations in predictive coding, we can now traverse this bridge from machine learning, through predictive coding, to neuroscience. In this section, we identify correspondences implied by this bridge. In particular, we explore the consequences of two implied correspondences: **1**) pyramidal neurons and deep networks and **2**) lateral inhibition and normalizing flows. These correspondences should be interpreted at a *functional* level, potentially shedding new light on the computational and learning mechanisms employed in biological neural circuits. In certain aspects, these correspondences offer a substantial departure from the current paradigm linking biological and artificial neural networks, providing an alternative approach toward connecting these areas.

Hierarchical predictive coding and deep latent variable models, particularly variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014), are highly related in both their model formulations and inference approaches



(a) Predictive Coding (b) VAE (Direct) (c) VAE (Iterative)

Figure 8.4: **Hierarchical Predictive Coding & VAEs**. Computation diagrams for **(a)** hierarchical predictive coding, **(b)** VAE with direct amortized inference, and **(c)** VAE with iterative amortized inference (Chapter 3). $\mathbf{J}^\intercal$ denotes the transposed Jacobian matrix of the generative model's conditional likelihood. Red dotted lines denote gradients, and black dashed lines denote amortized inference. Hierarchical predictive coding and VAEs are highly similar in both their model formulation and inference approach.

(Figure 8.4). Specifically,

- **Model Formulation**: Both areas focus on hierarchical latent Gaussian models with non-linear dependencies between latent levels, as well as dependencies within levels via covariance matrices (predictive coding) or normalizing flows (VAEs). Note that a covariance matrix is computationally equivalent to an affine normalizing flow with linear dependencies (Kingma, Salimans, et al., 2016).
- **Inference**: Both areas use variational inference, often with Gaussian approximate posteriors. While predictive coding employs gradient-based optimization and VAEs employ amortized optimization, these are just different design choices in solving the same inference optimization problem.

While previous works within predictive coding and deep latent variable models have explored distinct design choices, e.g. in parameterizing dynamics, the two areas share a common mathematical foundation, inherited from cybernetics and descendant areas.

With this connection explicitly established, we can now identify the biological correspondences implied by the connection from VAEs to predictive coding to neuroscience. Looking at Table 8.1, we see that top-down and bottom-up cortical

projections, each mediated by pyramidal neurons, respectively parameterize the generative model and inference updates. Mapping this onto VAEs implies that deep (artificial) neural networks are in correspondence with pyramidal neuron dendrites (Figure 8.5). This analogy is not perfect, as each output dimension of a deep network shares parameters with the other outputs through previous layers. In this regard, the analogy to pyramidal dendrites in cortex would specifically imply a separate deep network *per variable* in a VAE. Or, conversely, a deep network corresponds to a collection of pyramidal dendrites operating in parallel. Lateral inhibitory interneurons, which parameterize the inverse covariance matrices at each latent level in predictive coding, map onto normalizing flows. As mentioned above, normalizing flows are a non-linear generalization of linear covariance matrices, suggesting the possibility of non-linear normalization computations in cortex and elsewhere. These correspondences are obviously quite coarse-grained, and many details are left to be filled-in. However, they may provide a useful starting point for shifting the current analogies between machine learning and neuroscience. Below, we explore some of the consequences of these correspondences.

Pyramidal Neurons & Deep Networks

Non-linear Dendritic Computation Placing deep networks in correspondence with pyramidal dendrites departs from the traditional one-to-one correspondence of biological and artificial neurons (McCulloch and Pitts, 1943). This suggests that (some) individual biological neurons may be better computationally described as non-linear functions. Evidence from neuroscience supports this analogy. Early work in simulations proposed that individual pyramidal neurons, through dendritic processing, could operate as multi-layer artificial networks (Zador, Claiborne, and Brown, 1992; Mel, 1992). This was later supported by empirical findings that hippocampal pyramidal dendrites act as computational 'subunits,' yielding the equivalent of a two-layer artificial network (Poirazi, Brannon, and Mel, 2003; Polsky, Mel, and Schiller, 2004). More recently, Gidon et al., 2020 demonstrated that individual L2/3 pyramidal neurons are capable of computing the XOR operation, known for requiring non-linear processing (Minsky and Papert, 1969). This potential correspondence between deep networks and pyramidal dendrites posits a substantial role for dendritic computation (London and Häusser, 2005), moving beyond the overly simplistic comparison of biological and artificial neurons. Further, rather than assuming every neuron is equivalent, i.e. linear summation with non-linearity, separate classes of neurons would represent distinct function classes, likely derived

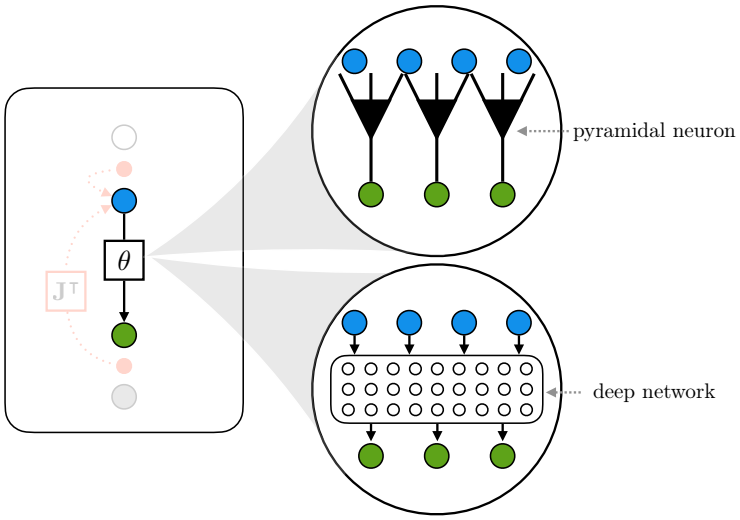


Figure 8.5: **Pyramidal Neurons & Deep Networks**. Connecting deep latent variable models with predictive coding places deep networks (bottom) in correspondence with the dendrites of pyramidal neurons (top). This is in contrast with conventional one-to-one analogies of biological and artificial neurons, suggesting a larger role for non-linear dendritic computation and alternative correspondences for backpropagation.

from their morphology. This places a greater emphasis on understanding neural circuits instead of assuming a uniform network of identical computational elements.

Amortization It is feasible that similar hardware and mechanisms underlying generative predictions could also perform inference updating, i.e. amortization. This is the insight of deep latent variable models: deep networks can parameterize conditional probabilities in *both* directions. The computational components are identical, with different inputs and output targets. Building off of the correspondence of pyramidal dendrites and deep networks, in cortex, we see particular classes of pyramidal neurons with separate apical and basal dendrites. These segregated dendritic compartments selectively take inputs from top-down and bottom-up pathways respectively (Bekkers, 2011; Guergiuev, Lillicrap, and Richards, 2016; Richards, 2019), thought to perform separate computations. These pyramidal neurons could implement a form of iterative amortized inference model (Chapter 3), separately processing top-down and bottom-up error signals to update inference estimates (Figure 8.6). This agrees with the conjecture from predictive coding that separate neurons in the forward pathway perform inference updating. Amortization also resolves the weight-transport issue from predictive coding, as separate inference weights are *learned*. While some empirical evidence appears to support amortization (Yildirim et al., 2015; Dasgupta et

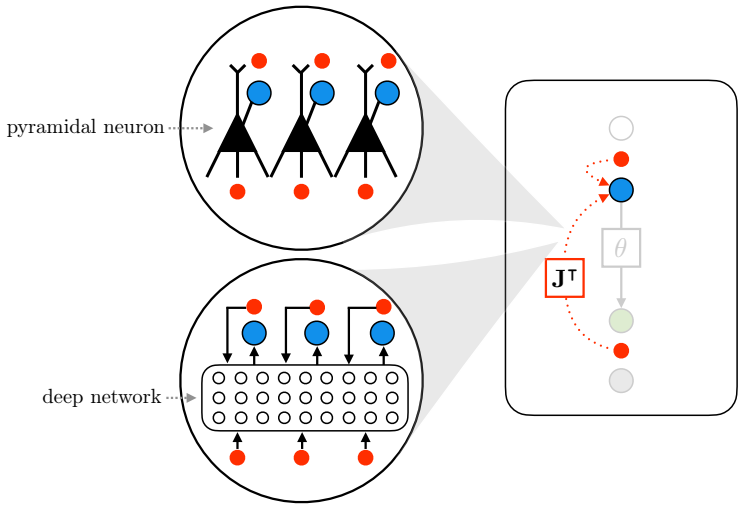


Figure 8.6: **Pyramidal Neurons & Amortization**. In predictive coding, inference updating is implemented using forward pyramidal neurons in cortex, taking prediction errors as input. In deep latent variable models, iterative amortized inference plays a similar role, continuing the analogy of pyramidal neurons and deep networks. Interestingly, this suggests a separation of processing in apical and basal dendrites, incorporating errors from the current and lower latent level.

al., 2018), we note that K. Friston, 2018 remains skeptical of its biological-plausibility. Further experiments, particularly at the cortical circuit level, are needed to resolve this question. For instance, amortization would require some form of stochastic gradient estimation, e.g. reparameterization gradients (Chapter 2). Although all of the necessary error signals are local to the amortized optimizer, the details remain unclear.

Neural Oscillations Oscillations are a common feature of neural circuits, giving rise to various frequency bands in the local field potential (LFP). These frequencies are thought to arise from the synchronous activity of populations of neurons, such as recurrent activity in hippocampus resulting in the theta frequency band (4–10 Hz). Bastos, Vezoli, et al., 2015 identify distinct frequency bands associated with forward (gamma, 30–80 Hz) and backward (beta, 10–30 Hz) activity in neocortex. Likewise, Walsh et al., 2020, reviewing the literature, note that violations of expectations, i.e. large prediction errors, are associated with increased gamma amplitude, whereas beta amplitude tends to increase in preparation for a predicted stimulus (Fujioka et al., 2009). In agreement with hierarchical predictive coding, this supports the conjecture that backward projections convey predictions, while forward projections perform inference updating using prediction errors. If we consider the gamma and beta

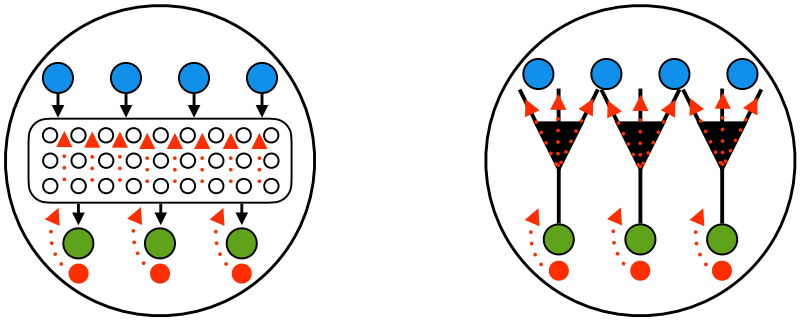


Figure 8.7: **Backpropagation**. Placing deep networks in correspondence with pyramidal neuron dendrites suggests an alternative perspective on the biological-plausibility of backpropagation. In deep latent variable models, backpropagation is only performed across variables that are directly connected through a conditional probability (left). From the perspective presented here, this corresponds to learning *within* pyramidal neurons. One possible implementation may be through backpropagating action potentials, perhaps combined with other neuromodulatory inputs (right).

frequency bands as distinct network-wide "clock rates" associated with inference and dynamics estimation, respectively, this suggests that cortex performs anywhere from 1–8 inference iterations per time step. Each time step would constitute 30–100 ms. As seen with amortized variational filtering in Chapter 4, as well iterative amortized policy optimization in Chapter 6, even with a *single* inference iteration per time step, amortization can yield reasonably accurate inference estimates. Note that direct amortization (Dayan et al., 1995) would suggest that these frequencies should be identical, in disagreement with empirical observations. Gradient-based optimization, in contrast, as suggested and employed by Rao and Ballard, 1999 and K. Friston, 2005, may not be sufficient to provide accurate inference estimates with so few iterations. This may provide further support for the claim that forward pyramidal neurons in cortex implement a form of iterative amortized inference.

Backpropagation Training deep networks at scale appears to require gradient-based parameter optimization, i.e. backpropagation (Werbos, 1974; Rumelhart, Geoffrey E Hinton, and R. J. Williams, 1986). However, the biological plausibility of backpropagation remains an open question (Lillicrap, Santoro, et al., 2020). Critics argue that backpropagation requires non-local learning signals and other techniques (Grossberg, 1987; Crick, 1989), whereas the brain relies largely on local learning rules (Hebb, 1949; Markram et al., 1997; Bi and Poo, 1998). A number

of "biologically-plausible" formulations of backpropagation have been proposed (Stork, 1989; Körding and König, 2001; Xie and Seung, 2003; Geoffrey E. Hinton, 2007; Lillicrap, Cownden, et al., 2016), attempting to reconcile this disparity and others. Yet, consensus is still lacking regarding the biological implementations of these proposed techniques. From another perspective, the apparent biological implausibility of backpropagation may instead be the result of incorrectly assuming a one-to-one correspondence between biological and artificial neurons.

If deep networks are in correspondence with pyramidal dendrites, this suggests a different perspective on the biological-plausibility of backpropagation. In hierarchical latent variable models, prediction errors at each level of the latent hierarchy provide a local learning signal (K. Friston, 2005; Y. Bengio, 2014; Lee et al., 2015; Whittington and Bogacz, 2017). Thus, the global objective is decomposed into local errors, with learning within each latent level performed through gradient-based optimization. This is exemplified by deep latent variable models, which utilize backpropagation *within* each latent level, but not (necessarily) *across* latent levels. Again, considering the correspondence of pyramidal dendrites and deep networks, this suggests that learning *within* pyramidal neurons may be more analogous to backpropagation (Figure 8.7). Not surprisingly, one possible candidate is backpropagating action potentials (G. J. Stuart and Sakmann, 1994; S. R. Williams and G. J. Stuart, 2000). These occur in the dendrites of pyramidal neurons, actively propagating a signal of neural activity back to synaptic inputs (G. Stuart et al., 1997; Brunner and Szabadics, 2016). This results in the location-dependent influx of calcium, leading to a variety of synaptic changes throughout the dendritic tree (Johenning et al., 2015). Indeed, Schiess, Urbanczik, and Senn, 2016 recently investigated a computational model of gradient backpropagation within dendritic trees. While many details remain unclear, this overall perspective of backpropagation *within* neurons, rather than across networks of neurons, offers a more biologically-plausible alternative; all signals are local to the pyramidal neurons/dendrites within the cortical circuit. Given the proposed theoretical role and empirical observations, this possible correspondence between backpropagation and backpropagating action potentials warrants further investigation.

Lateral Inhibition & Normalizing Flows

Sensory Input Normalization Sensory stimuli are highly redundant in both space and time. Examples include luminance at neighboring photoreceptors or pressure on adjacent mechanoreceptors, each of which tend to persist over time intervals. One

of the key computational roles of early sensory areas, e.g. retina, appears to be in reducing these redundancies through normalization. In retina, such normalization operations are carried out through lateral inhibition via horizontal and amacrine cells. As a result, the transmitted output signals are less correlated (Graham, Chandler, and Field, 2006; Pitkow and Meister, 2012). As we have discussed in this chapter and elsewhere, normalization and prediction are inseparable, i.e. one must form a prediction in order to normalize. Accordingly, previous works have framed early sensory processing in terms of (spatiotemporal) predictive coding (Srinivasan, Laughlin, and Dubs, 1982; Hosoya, Baccus, and Meister, 2005; Palmer et al., 2015). This is often motivated in terms of increased sensitivity or efficiency (Srinivasan, Laughlin, and Dubs, 1982; Atick and Redlich, 1990) due to redundancy reduction (Barlow et al., 1961; Barlow, Kaushal, and Mitchison, 1989), i.e. compression.

If we consider cortex as a hierarchical latent variable model of sensory inputs, then early sensory areas are implicated in parameterizing the conditional likelihood. The ubiquity of normalization operations in early sensory areas is suggestive of normalization in a flow-based model. That is, early sensory areas may implement the "inference" direction of a flow-based conditional likelihood (Siddharth Agrawal and Dukkipati, 2016; Winkler et al., 2019). This would create a learned, normalized space in which cortex makes predictions. In addition to the sensitivity and efficiency arguments above, this could simplify downstream generative modeling and improve generalization, as demonstrated in Chapter 5. Interestingly, Rao and Ballard, 1999 employ a similar whitening scheme on image inputs, presumably to imitate retina. Normalizing flows offers a generalization of this idea to non-affine parameterizations and multiple stages of normalization. Further, framing early sensory areas in terms of normalizing flows connects these computations conceptually with the rest of the cortical generative model, i.e. truly evaluating the data-level predictions of cortex would require inverting the normalization of early sensory areas. While normalizing flows may not provide a perfect description of early sensory processing (e.g. these operations may not be completely invertible), this framework may help to unify many disparate input normalization circuits.

Normalization in Thalamus & Cortex Normalization is also thought to be a key aspect of first-order relays in thalamus, such as the lateral geniculate nucleus (LGN). Dong and Atick, 1995 proposed that inhibition across time in LGN could provide a mechanism for temporally decorrelating the input from retina, with some supporting evidence provided by Dan, Atick, and Reid, 1996. Again, this can

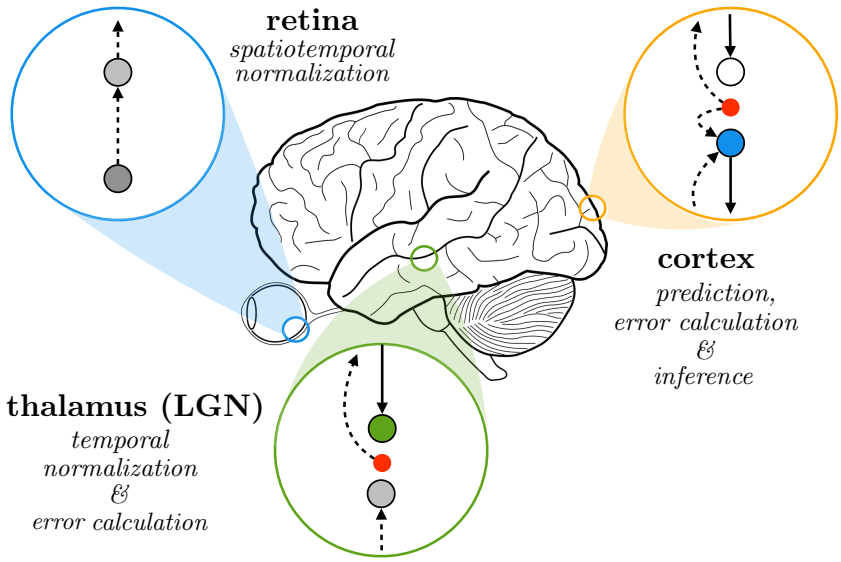


Figure 8.8: **Computational Schematic of the Visual Pathway**. Interpreting the early visual pathway from the perspective of a latent variable model, we can assign computational functions to retina, LGN, and cortex. Retina and LGN are interpreted as implementing normalizing flows, i.e. spatiotemporal predictive coding, reducing spatial and temporal redundancy in the visual input. LGN also serves as the lowest level for hierarchical predictions, which are computed through backward connections in cortex. Using prediction errors throughout the hierarchy, forward cortical connections update latent estimates.

be considered as a form of temporal predictive coding (Srinivasan, Laughlin, and Dubs, 1982), removing easily predictable temporal information. Following the interpretation above, normalization in thalamus may provide a second stage of normalizing flow, further removing redundancy. In Chapter 5, we implemented this general technique using autoregressive flows in deep latent variable models. There, we saw that low-level temporal normalization removes static backgrounds, improving modeling and generalization. Further work is needed to assess the functional form (affine/non-affine) and types of dependencies (linear/non-linear) implemented in first-order thalamic relays, though suggestions are given by Dong and Atick, 1995.

Normalization, via local lateral inhibition, is also found throughout cortex (King, Zylberberg, and DeWeese, 2013). K. Friston, 2005 suggested that lateral inhibition plays the computational role of inverse covariance (precision) matrices, modeling dependencies between dimensions within the same latent level of the hierarchy. This corresponds to parameterizing approximate posteriors (Rezende and Mohamed, 2015; Kingma, Salimans, et al., 2016) and/or conditional priors (C.-W. Huang et al.,

2017) with (linear affine) normalizing flows.[1] Again, note that normalizing flows offers a more general mathematical framework for describing these normalization computations. Further, predictive coding often assumes that these dependencies are modeled using *symmetric* lateral weights (K. Friston, 2005). In contrast, normalizing flows also permits *non-symmetric* schemes, e.g. using spatially autoregressive models (Kingma, Salimans, et al., 2016) or ensembles of such models (Uria, I. Murray, and Larochelle, 2014). In Chapter 5, we also applied normalizing flows across time in hierarchical models to assist in parameterizing dynamics, discussing connections to Friston's notion of generalized coordinates (K. Friston, 2008a). Thus, multiple forms of spatial and temporal normalization may occur within cortex, allowing cortical columns to add and remove dependencies across space and time. Finally, multiple works within predictive coding have explored the use of prediction precision as a form of attention (Spratling, 2008; Feldman and K. Friston, 2010). Increasing the precision of predictions modulates the gain of prediction errors in driving inference, leading to more precise inferred estimates. This may prove to be a useful technique in machine learning. The overall computational scheme, ignoring spatiotemporal normalization in cortex for simplicity, is shown in Figure 8.8.

Motor Dependencies Much like sensory input areas, there is a striking degree of low-level interneuron circuitry in motor output areas. A canonical example is central pattern generator (CPG) circuits, which, through local excitation and inhibition, give rise to coordinated muscle activation (Marder and Bucher, 2001). These circuits provide a basis of "motor primitives," allowing muscle activations to be carried out in a lower-dimensional manifold rather than the entire combinatorial space. Given the anatomical similarity of these circuits with normalization circuitry in sensory areas, as well as the close relationship between adding (generation) and removing (inference) dependencies in normalizing flows, it is possible that such low-level motor circuits are implementing flow-based distributions. Under this scheme, spinal circuits convert compressed, uncorrelated signals from higher-level motor areas into correlated low-level muscle activation. Similar efficiency arguments from sensory input areas (Atick and Redlich, 1990) are applicable to motor outputs, treating the spinal cord as a communication channel. Further, spatiotemporal dependencies in muscle activation may also improve the expressive capacity and generalization of motor routines. Indeed, multiple works have investigated improving control policies

[1] Specifically, Friston's proposal corresponds to ZCA whitening (K. Friston, 2005), whereas those within machine learning have explored Cholesky whitening (Kingma, Salimans, et al., 2016).

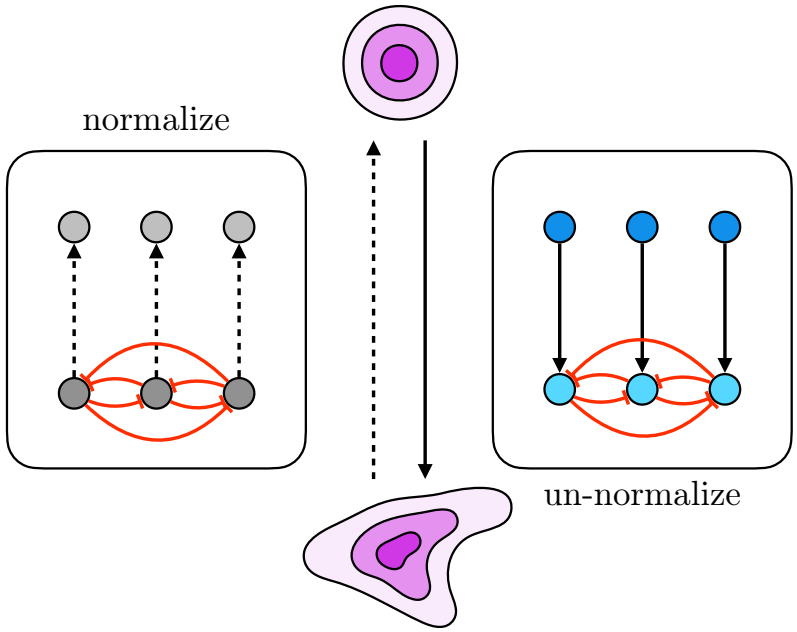


Figure 8.9: **Adding & Removing Dependencies with Normalizing Flows**. Normalizing flows provides a general mathematical framework for removing (left) or adding (right) probabilistic dependencies. Using lateral interactions, one can move between a normalized (top) or un-normalized (bottom) space. Normalized spaces have benefits for compression, whereas un-normalized spaces are more expressive. Neural systems may employ these transforms for sensory and motor processing.

with normalizing flows across motor dimensions (Haarnoja et al., 2018; Tang and Shipra Agrawal, 2018; Ward, Smofsky, and Bose, 2019), and in Chapter 7, we investigated incorporating flow-based motor dependencies across time. Consistent with this setup, it has been empirically observed in neural systems that higher-level motor areas output transient update signals, which are converted to lower-level motor trajectories (Shalit et al., 2012). While these findings are encouraging, more apparent benefits are likely to arise in high-dimensional action spaces, where the *curse of dimensionality* (Chen and Gopinath, 2001) makes it essential to model dependencies for effective exploration. For instance, current MuJoCo environments from OpenAI gym (Todorov, Erez, and Tassa, 2012; Brockman et al., 2016) contain <20 action dimensions, whereas it has been estimated that there are roughly 800 independent dimensions to human motor control (Powers, 1973). Operating and learning efficiently in such large action spaces may require modeling spatiotemporal dependencies through some form of hierarchical or flow-based decomposition.

8.5 Discussion

In this chapter, we connected the ideas presented in this thesis back to their origins in predictive coding. The two core techniques developed in this thesis, iterative amortized inference and sequential autoregressive flows, map onto aspects of hierarchical and spatiotemporal predictive coding, respectively. By connecting these techniques back to predictive coding, we arrived at a variety of possible implied correspondences between machine learning and neuroscience. In particular,

- we identified the dendrites of pyramidal neurons as functionally analogous to (nonlinear) deep networks, and
- we identified lateral inhibition as implementing normalizing flows.

Placing pyramidal neuron dendrites in correspondence with deep networks departs from the traditional one-to-one analogy of biological and artificial neurons, raising a host of questions regarding dendritic computation and learning via backpropagation. Likewise, normalizing flows offers a more general framework for considering the normalization computations carried out by lateral inhibitory interactions found within multiple brain regions. We are hopeful that connecting these areas will provide new insights for both machine learning and neuroscience.

For practical reasons, we primarily focused on perception in discussing predictive coding and the ideas in this thesis. This is a result of the fact that predictive coding was initially developed and studied in the context of generative models of sensory inputs (Srinivasan, Laughlin, and Dubs, 1982; Rao and Ballard, 1999; K. Friston, 2005). However, if the cortical microcircuit implements a general-purpose modeling and inference algorithm, we should expect similar computations to be applicable to motor and prefrontal cortices. Over the past decade, Friston and colleagues have developed a range of exciting ideas, interpreting motor control as a process of proprioceptive prediction (Adams, Shipp, and K. J. Friston, 2013) and prefrontal cortex as performing hierarchical goal inference (Pezzulo, Rigoli, and K. J. Friston, 2018). Such ideas can be seen as modern extensions of early ideas in cybernetics (Wiener, 1948; MacKay, 1956; Powers, 1973), using motor control to correct for discrepancies (errors) between desired and actual outcomes. Indeed, as shown in Part III, the *same* perceptual modeling and inference techniques can be applied to control. Nevertheless, as we discuss in the final chapter, further work is needed within machine learning to unify perception and control under a single formulation.

References

Adams, Rick A, Stewart Shipp, and Karl J Friston (2013). "Predictions not commands: active inference in the motor system". In: *Brain Structure and Function* 218.3, pp. 611–643.

Agrawal, Siddharth and Ambedkar Dukkipati (2016). "Deep Variational Inference Without Pixel-Wise Reconstruction". In: *arXiv preprint arXiv:1611.05209*.

Alink, Arjen et al. (2010). "Stimulus predictability reduces responses in primary visual cortex". In: *Journal of Neuroscience* 30.8, pp. 2960–2966.

Atal, B and M Schroeder (1979). "Predictive coding of speech signals and subjective error criteria". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 27.3, pp. 247–254.

Atick, Joseph J and A Norman Redlich (1990). "Towards a theory of early visual processing". In: *Neural computation* 2.3, pp. 308–320.

– (1992). "What does the retina know about natural scenes?" In: *Neural computation* 4.2, pp. 196–210.

Baccus, Stephen A et al. (2008). "A retinal circuit that computes object motion". In: *Journal of Neuroscience* 28.27, pp. 6807–6817.

Barlow, Horace B et al. (1961). "Possible principles underlying the transformation of sensory messages". In: *Sensory communication* 1, pp. 217–234.

Barlow, Horace B, Tej P Kaushal, and Graeme J Mitchison (1989). "Finding minimum entropy codes". In: *Neural Computation* 1.3, pp. 412–423.

Bastos, Andre Moraes, W Martin Usrey, et al. (2012). "Canonical microcircuits for predictive coding". In: *Neuron* 76.4, pp. 695–711.

Bastos, Andre Moraes, Julien Vezoli, et al. (2015). "Visual areas exert feedforward and feedback influences through distinct frequency channels". In: *Neuron* 85.2, pp. 390–401.

Bekkers, John M (2011). "Pyramidal neurons". In: *Current Biology* 21.24, R975.

Bell, Curtis C (2001). "Memory-based expectations in electrosensory systems". In: *Current opinion in neurobiology* 11.4, pp. 481–487.

Bengio, Yoshua (2014). "How auto-encoders could provide credit assignment in deep networks via target propagation". In: *arXiv preprint arXiv:1407.7906*.

Bi, Guo-qiang and Mu-ming Poo (1998). "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type". In: *Journal of neuroscience* 18.24, pp. 10464–10472.

Brockman, Greg et al. (2016). "Openai gym". In: *arXiv preprint arXiv:1606.01540*.

Broeke, Gerben van den (2016). "What auto-encoders could learn from brains". MA thesis. Aalto University.

Brunner, János and János Szabadics (2016). "Analogue modulation of back-propagating action potentials enables dendritic hybrid signalling". In: *Nature communications* 7, p. 13033.

Carandini, Matteo and David J Heeger (2012). "Normalization as a canonical neural computation". In: *Nature Reviews Neuroscience* 13.1, pp. 51–62.

Chen, Scott Saobing and Ramesh A Gopinath (2001). "Gaussianization". In: *Advances in neural information processing systems*, pp. 423–429.

Clark, Andy (2013). "Whatever next? Predictive brains, situated agents, and the future of cognitive science". In: *Behavioral and Brain Sciences* 36.3, pp. 181–204.

Covic, Elise N and S Murray Sherman (2011). "Synaptic properties of connections between the primary and secondary auditory cortices in mice". In: *Cerebral Cortex* 21.11, pp. 2425–2441.

Crick, Francis (1989). "The recent excitement about neural networks". In: *Nature* 337.6203, pp. 129–132.

Dan, Yang, Joseph J Atick, and R Clay Reid (1996). "Efficient coding of natural scenes in the lateral geniculate nucleus: experimental test of a computational theory". In: *Journal of Neuroscience* 16.10, pp. 3351–3362.

Dasgupta, Ishita et al. (2018). "Remembrance of inferences past: Amortization in human hypothesis generation". In: *Cognition* 178, pp. 67–81.

Dayan, Peter et al. (1995). "The helmholtz machine". In: *Neural computation* 7.5, pp. 889–904.

De Pasquale, Roberto and S Murray Sherman (2011). "Synaptic properties of corticocortical connections between the primary and secondary visual cortical areas in the mouse". In: *Journal of Neuroscience* 31.46, pp. 16494–16506.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density estimation using real nvp". In: *International Conference on Learning Representations*.

Dong, Dawei W and Joseph J Atick (1995). "Temporal decorrelation: a theory of lagged and nonlagged responses in the lateral geniculate nucleus". In: *Network: Computation in Neural Systems* 6.2, pp. 159–178.

Douglas, Rodney J, Kevan AC Martin, and David Whitteridge (1989). "A canonical microcircuit for neocortex". In: *Neural computation* 1.4, pp. 480–488.

Durkan, Conor et al. (2019). "Neural Spline Flows". In: *arXiv preprint arXiv:1906.04032*.

Egner, Tobias, Jim M Monti, and Christopher Summerfield (2010). "Expectation and surprise determine neural population responses in the ventral visual stream". In: *Journal of Neuroscience* 30.49, pp. 16601–16608.

Eliades, Steven J and Xiaoqin Wang (2008). "Neural substrates of vocalization feedback monitoring in primate auditory cortex". In: *Nature* 453.7198, p. 1102.

Feldman, Harriet and Karl Friston (2010). "Attention, uncertainty, and free-energy". In: *Frontiers in human neuroscience* 4.

Friedman, Jerome H (1987). "Exploratory projection pursuit". In: *Journal of the American statistical association* 82.397, pp. 249–266.

Friston, Karl (2005). "A theory of cortical responses". In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 360.1456, pp. 815–836.

– (2008a). "Hierarchical models in the brain". In: *PLoS computational biology* 4.11, e1000211.

– (2008b). "Variational filtering". In: *NeuroImage* 41.3, pp. 747–766.

– (2018). "Does predictive coding have a future?" In: *Nature neuroscience* 21.8, p. 1019.

Friston, Karl et al. (2007). "Variational free energy and the Laplace approximation". In: *Neuroimage* 34.1, pp. 220–234.

Fujioka, Takako et al. (2009). "Beta and gamma rhythms in human auditory cortex during musical beat processing". In: *Annals of the New York Academy of Sciences* 1169.1, pp. 89–92.

Gershman, Samuel J (2019). "What does the free energy principle tell us about the brain?" In: *arXiv preprint arXiv:1901.07945*.

Gidon, Albert et al. (2020). "Dendritic action potentials and computation in human layer 2/3 cortical neurons". In: *Science*.

Gilbert, Charles D and Mariano Sigman (2007). "Brain states: top-down influences in sensory processing". In: *Neuron* 54.5, pp. 677–696.

Girard, Pascal and Jean Bullier (1989). "Visual activity in area V2 during reversible inactivation of area 17 in the macaque monkey". In: *Journal of neurophysiology* 62.6, pp. 1287–1302.

Girard, Pascal, PA Salin, and Jean Bullier (1991). "Visual activity in areas V3a and V3 during reversible inactivation of area V1 in the macaque monkey". In: *Journal of Neurophysiology* 66.5, pp. 1493–1503.

Graham, Daniel J, Damon M Chandler, and David J Field (2006). "Can the theory of "whitening" explain the center-surround properties of retinal ganglion cell receptive fields?" In: *Vision research* 46.18, pp. 2901–2913.

Grossberg, Stephen (1987). "Competitive learning: From interactive activation to adaptive resonance". In: *Cognitive science* 11.1, pp. 23–63.

Guergiuev, Jordan, Timothy P Lillicrap, and Blake A Richards (2016). "Biologically feasible deep learning with segregated dendrites". In: *arXiv preprint arXiv:1610.00161*.

Haarnoja, Tuomas et al. (2018). "Latent Space Policies for Hierarchical Reinforcement Learning". In: *International Conference on Machine Learning*, pp. 1846–1855.

Harrison, CW (1952). “Experiments with linear prediction in television”. In: *Bell System Technical Journal* 31.4, pp. 764–783.

Hawkins, Jeff and Sandra Blakeslee (2004). *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan.

Hebb, Donald O (1949). “The organization of behavior; a neuropsycholocigal theory.” In: *A Wiley Book in Clinical Psychology.*, pp. 62–78.

Hinton, Geoffrey E. (2007). “How to do backpropagation in a brain”. In: *NeurIPS Deep Learning Workshop*.

Hosoya, Toshihiko, Stephen A Baccus, and Markus Meister (2005). “Dynamic predictive coding by the retina”. In: *Nature* 436.7047, p. 71.

Huang, Chin-Wei et al. (2017). “Learnable explicit density for continuous latent space and variational inference”. In: *arXiv preprint arXiv:1710.02248*.

Huang, Yanping and Rajesh PN Rao (2011). “Predictive coding”. In: *Wiley Interdisciplinary Reviews: Cognitive Science* 2.5, pp. 580–593.

Isaacson, Jeffry S and Massimo Scanziani (2011). “How inhibition shapes cortical activity”. In: *Neuron* 72.2, pp. 231–243.

Ito, Masao (1998). “Cerebellar learning in the vestibulo–ocular reflex”. In: *Trends in cognitive sciences* 2.9, pp. 313–321.

Jehee, Janneke FM and Dana H Ballard (2009). “Predictive feedback can account for biphasic responses in the lateral geniculate nucleus”. In: *PLoS Comput Biol* 5.5, e1000373.

Johenning, Friedrich W et al. (2015). “Ryanodine receptor activation induces long-term plasticity of spine calcium dynamics”. In: *PLoS biology* 13.6, e1002181.

Kalman, Rudolph Emil (1960). “A new approach to linear filtering and prediction problems”. In: *Journal of Basic Engineering* 82.1, pp. 35–45.

Keller, Georg B, Tobias Bonhoeffer, and Mark Hübener (2012). “Sensorimotor mismatch signals in primary visual cortex of the behaving mouse”. In: *Neuron* 74.5, pp. 809–815.

Keller, Georg B and Thomas D Mrsic-Flogel (2018). “Predictive processing: a canonical cortical computation”. In: *Neuron* 100.2, pp. 424–435.

Kennedy, Ann et al. (2014). “A temporal basis for predicting the sensory consequences of motor commands in an electric fish”. In: *Nature neuroscience* 17.3, pp. 416–422.

Kessy, Agnan, Alex Lewin, and Korbinian Strimmer (2018). “Optimal whitening and decorrelation”. In: *The American Statistician* 72.4, pp. 309–314.

King, Paul D, Joel Zylberberg, and Michael R DeWeese (2013). “Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of V1”. In: *Journal of Neuroscience*.

Kingma, Durk P, Tim Salimans, et al. (2016). “Improved variational inference with inverse autoregressive flow”. In: *Advances in neural information processing systems*, pp. 4743–4751.

Kingma, Durk P and Max Welling (2014). “Stochastic gradient VB and the variational auto-encoder”. In: *Proceedings of the International Conference on Learning Representations*.

Körding, Konrad P and Peter König (2001). “Supervised and unsupervised learning with two sites of synaptic integration”. In: *Journal of computational neuroscience* 11.3, pp. 207–215.

Kumar, Manoj et al. (2020). “VideoFlow: A Flow-Based Generative Model for Video”. In: *International Conference on Learning Representations*.

Lee, Dong-Hyun et al. (2015). “Difference target propagation”. In: *Joint european conference on machine learning and knowledge discovery in databases*. Springer, pp. 498–515.

Lillicrap, Timothy P, Daniel Cownden, et al. (2016). “Random synaptic feedback weights support error backpropagation for deep learning”. In: *Nature communications* 7, p. 13276.

Lillicrap, Timothy P, Adam Santoro, et al. (2020). “Backpropagation and the brain”. In: *Nature Reviews Neuroscience*, pp. 1–12.

London, Michael and Michael Häusser (2005). “Dendritic computation”. In: *Annu. Rev. Neurosci.* 28, pp. 503–532.

Lotter, William, Gabriel Kreiman, and David Cox (2018). “A neural network trained to predict future video frames mimics critical properties of biological neuronal responses and perception”. In: *arXiv preprint arXiv:1805.10734*.

MacKay, D M (1956). “The epistemological problem for automata”. In: *Automata studies*, pp. 235–252.

Marder, Eve and Dirk Bucher (2001). “Central pattern generators and the control of rhythmic movements”. In: *Current biology*.

Markram, Henry et al. (1997). “Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs”. In: *Science* 275.5297, pp. 213–215.

McCulloch, Warren S and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Mel, Bartlett W (1992). “The clusteron: toward a simple abstraction for a complex neuron”. In: *Advances in neural information processing systems*.

Meyer, Hanno S et al. (2011). “Inhibitory interneurons in a cortical column form hot zones of inhibition in layers 2 and 5A”. In: *Proceedings of the National Academy of Sciences* 108.40, pp. 16807–16812.

Meyer, Travis and Carl R Olson (2011). "Statistical learning of visual transitions in monkey inferotemporal cortex". In: *Proceedings of the National Academy of Sciences* 108.48, pp. 19401–19406.

Minsky, Marvin and Seymour Papert (1969). *Perceptrons*.

Mountcastle, VB, AL Berman, and PW Davies (1955). "Topographic organization and modality representation in first somatic area of cat's cerebral cortex by method of single unit analysis". In: *Am J Physiol* 183.464, p. 10.

Mumford, David (1992). "On the computational architecture of the neocortex". In: *Biological cybernetics* 66.3, pp. 241–251.

Murray, Scott O et al. (2002). "Shape perception reduces activity in human primary visual cortex". In: *Proceedings of the National Academy of Sciences* 99.23, pp. 15164–15169.

Oliver, BM (1952). "Efficient coding". In: *The Bell System Technical Journal* 31.4, pp. 724–750.

Ölveczky, Bence P, Stephen A Baccus, and Markus Meister (2003). "Segregation of object and background motion in the retina". In: *Nature* 423.6938, pp. 401–408.

Palmer, Stephanie E et al. (2015). "Predictive information in a sensory population". In: *Proceedings of the National Academy of Sciences* 112.22, pp. 6908–6913.

Parras, Gloria G et al. (2017). "Neurons along the auditory pathway exhibit a hierarchical organization of prediction error". In: *Nature communications* 8.1, pp. 1–17.

Pezzulo, Giovanni, Francesco Rigoli, and Karl J Friston (2018). "Hierarchical active inference: A theory of motivated control". In: *Trends in cognitive sciences* 22.4, pp. 294–306.

Pitkow, Xaq and Markus Meister (2012). "Decorrelation and efficient coding by retinal ganglion cells". In: *Nature neuroscience* 15.4, p. 628.

Poirazi, Panayiota, Terrence Brannon, and Bartlett W Mel (2003). "Pyramidal neuron as two-layer neural network". In: *Neuron* 37.6, pp. 989–999.

Polsky, Alon, Bartlett W Mel, and Jackie Schiller (2004). "Computational subunits in thin dendrites of pyramidal cells". In: *Nature neuroscience* 7.6, p. 621.

Pourahmadi, Mohsen (2011). "Covariance estimation: The GLM and regularization perspectives". In: *Statistical Science*, pp. 369–387.

Powers, William T (1973). *Behavior: The control of perception*. Aldine Chicago.

Rao, Rajesh PN (1998). "Correlates of attention in a model of dynamic visual recognition". In: *Advances in neural information processing systems*, pp. 80–86.

Rao, Rajesh PN and Dana H Ballard (1999). "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects." In: *Nature neuroscience* 2.1.

Rao, Rajesh PN and Terrence J Sejnowski (2002). "Predictive Coding, Cortical Feedback, and Spike-Timing Dependent Plasticity". In: *Probabilistic models of the brain*, p. 297.

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning*, pp. 1530–1538.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models". In: *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286.

Richards, Blake A (Sept. 6, 2019). personal communication.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors." In: *Nature*.

Schiess, Mathieu, Robert Urbanczik, and Walter Senn (2016). "Somato-dendritic synaptic plasticity and error-backpropagation in active dendrites". In: *PLoS computational biology* 12.2, e1004638.

Shalit, Uri et al. (2012). "Descending systems translate transient cortical commands into a sustained muscle activation signal". In: *Cerebral cortex* 22.8, pp. 1904–1914.

Shannon, Claude E (1948). "A mathematical theory of communication". In: *The Bell system technical journal* 27.3, pp. 379–423.

Sharma, Jitendra, Alessandra Angelucci, and Mriganka Sur (2000). "Induction of visual orientation modules in auditory cortex". In: *Nature* 404.6780, p. 841.

Sherman, S Murray and RW Guillery (2002). "The role of the thalamus in the flow of information to the cortex". In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 357.1428, pp. 1695–1708.

Spratling, Michael W (2008). "Reconciling predictive coding and biased competition models of cortical function". In: *Frontiers in computational neuroscience* 2, p. 4.

Srinivasan, Mandyam Veerambudi, Simon Laughlin, and Andreas Dubs (1982). "Predictive coding: a fresh view of inhibition in the retina". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 216.1205, pp. 427–459.

Stork, David G (1989). "Is backpropagation biologically plausible". In: *International Joint Conference on Neural Networks*. Vol. 2. IEEE Washington, DC, pp. 241–246.

Stuart, Greg J and Bert Sakmann (1994). "Active propagation of somatic action potentials into neocortical pyramidal cell dendrites". In: *Nature* 367.6458, p. 69.

Stuart, Greg et al. (1997). "Action potential initiation and backpropagation in neurons of the mammalian CNS". In: *Trends in neurosciences* 20.3, pp. 125–131.

Summerfield, Christopher et al. (2006). "Predictive codes for forthcoming perception in the frontal cortex". In: *Science* 314.5803, pp. 1311–1314.

Tang, Yunhao and Shipra Agrawal (2018). “Boosting Trust Region Policy Optimization by Normalizing Flows Policy”. In: *arXiv preprint arXiv:1809.10326*.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). “Mujoco: A physics engine for model-based control”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 5026–5033.

Uria, Benigno, Iain Murray, and Hugo Larochelle (2014). “A deep and tractable density estimator”. In: *International Conference on Machine Learning*, pp. 467–475.

Von Helmholtz, Hermann (1867). *Handbuch der physiologischen Optik*. Vol. 9. Voss.

Wacongne, Catherine et al. (2011). “Evidence for a hierarchy of predictions and prediction errors in human cortex”. In: *Proceedings of the National Academy of Sciences* 108.51, pp. 20754–20759.

Walsh, Kevin S et al. (2020). “Evaluating the neurophysiological evidence for predictive processing as a model of perception”. In: *Annals of the New York Academy of Sciences* 1464.1, p. 242.

Ward, Patrick Nadeem, Ariella Smofsky, and Avishek Joey Bose (2019). “Improving Exploration in Soft-Actor-Critic with Normalizing Flows Policies”. In: *ICML Workshop on Invertible Neural Nets and Normalizing Flows*.

Werbos, Paul (1974). “Beyond regression:" new tools for prediction and analysis in the behavioral sciences”. In: *Ph. D. dissertation, Harvard University*.

Whittington, James CR and Rafal Bogacz (2017). “An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity”. In: *Neural computation*.

Wiegand, Thomas et al. (2003). “Overview of the H. 264/AVC video coding standard”. In: *IEEE Transactions on circuits and systems for video technology* 13.7, pp. 560–576.

Wiener, Norbert (1948). *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press.

Williams, Stephen R and Greg J Stuart (2000). “Backpropagation of physiological spike trains in neocortical pyramidal neurons: implications for temporal coding in dendrites”. In: *Journal of Neuroscience*.

Winkler, Christina et al. (2019). “Learning Likelihoods with Conditional Normalizing Flows”. In: *arXiv preprint arXiv:1912.00042*.

Xie, Xiaohui and H Sebastian Seung (2003). “Equivalence of backpropagation and contrastive Hebbian learning in a layered network”. In: *Neural computation* 15.2, pp. 441–454.

Yildirim, Ilker et al. (2015). “Efficient and robust analysis-by-synthesis in vision: A computational framework, behavioral tests, and modeling neuronal representations”. In: *Thirty-Seventh Annual Conference of the Cognitive Science Society*. Vol. 4.

Zador, Anthony M, Brenda J Claiborne, and Thomas H Brown (1992). “Nonlinear pattern separation in single hippocampal neurons with active dendritic membrane”. In: *Advances in neural information processing systems*, pp. 51–58.

Zmarz, Pawel and Georg B Keller (2016). “Mismatch receptive fields in mouse visual cortex”. In: *Neuron* 92.4, pp. 766–772.

Chapter 9

CONCLUSION

Marino, Joseph and Yisong Yue (2019). "An Inference Perspective on Model-Based Reinforcement Learning". In: *ICML Workshop on Generative Modeling and Model-Based Reasoning for Robotics and AI*. URL: https://drive.google.com/open?id=1uc1FdjHVkkEAaU6jz7aVrP3khGyqkdKi.

This thesis has presented two core techniques for probabilistic modeling and inference, iterative amortization and sequential autoregressive flows, applied to both perception (Part II) and control (Part III). These techniques are modern incarnations of classical feedback and feedforward ideas popularized and developed under cybernetics (Wiener, 1948; Ashby, 1956). Importantly, these techniques are both *learned*, providing an additional layer of optimization, i.e., negative feedback, for added efficiency and efficacy. Thus, these ideas are both generally applicable to a range of perception and control settings, as well as powerful, enabling performance improvements. Figure 9.1 provides an overview of the contributions. In this chapter, we conclude this thesis by examining the general themes developed in the preceding chapters and broadening the discussion toward a synthesis of these ideas and the path ahead.

9.1 Iterative Estimation

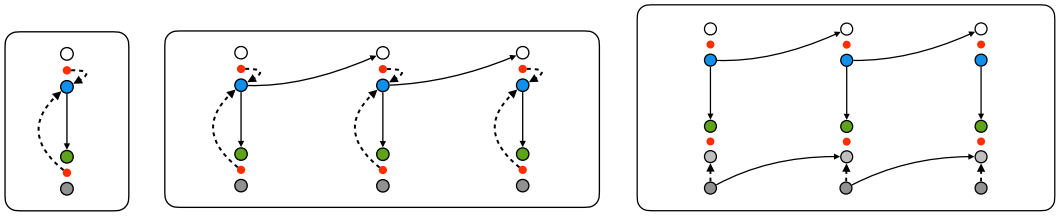
The technical contributions of this thesis demonstrate the benefits of iterative estimation, applied to both probabilistic modeling of dynamics (sequential autoregressive flows; Chapters 5 & 7) and variational inference (iterative amortization; Chapters 3, 4, & 6). We have already seen how iterative amortization provides updates to inferred estimates of distribution parameters, $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda} \leftarrow f_\phi(\boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}} \mathcal{L}).$$

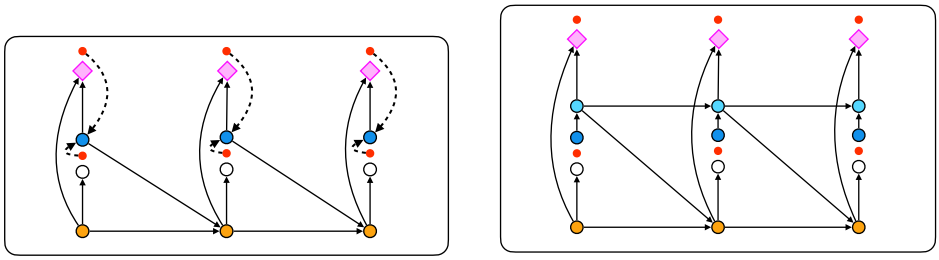
In each case, we used a gated transform to parameterize this update, which is a specialized version of a residual update:

$$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}. \tag{9.1}$$

This had numerous benefits, including the ability to improve performance by simply increasing the number of iterations, i.e., amount of computation, and the



(a) Ch. 3 (b) Ch. 4 (c) Ch. 5



(d) Ch. 6 (e) Ch. 7

Figure 9.1: **Thesis Summary**. Graphical representations of perception (**a–c**) and control (**d–e**) using feedback (**a, b, d**) and feedforward (**c, e**) processes.

ability to estimate multiple solutions (Greff et al., 2019). Iterative estimation also allowed us to automatically adapt to new priors, as seen in sequential latent variable models (Chapter 4), and new conditional likelihoods, as seen in policy optimization (Chapter 6). Thus, this procedure is both more powerful *and* more general.

Switching settings, with sequential autoregressive flows on observations, $\mathbf{x}_{1:T}$, we can consider the special case of parameterizing the shift at time t as $\mathbf{x}_{t-1}$ and the scale as the vector $\mathbf{1}$. The normalized base distribution variable then becomes

$$\begin{aligned} \mathbf{y}_t &= \frac{\mathbf{x}_t - \mathbf{x}_{t-1}}{\mathbf{1}}, \\ &\equiv \Delta \mathbf{x}_{t-1}. \end{aligned}$$

Equivalently, if we are modeling $\mathbf{y}_t = \Delta \mathbf{x}_{t-1}$ using a higher-level model for the base distribution, the autoregressive generative model can be expressed as

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \Delta \mathbf{x}_{t-1}. \tag{9.2}$$

In Chapters 5 & 7, we saw that this scheme simplified estimation, improving performance in both cases. With conditional likelihood estimation (Chapter 5), we also saw accompanying generalization improvements, resulting from the learned normalization of inputs. We can interpret $\mathbf{x}_{t-1}$ in Eq. 9.2 as a type of frame of reference, providing an initial estimate of $\mathbf{x}_t$, which is then refined through $\Delta \mathbf{x}_{t-1}$.

While formulated from the differing perspectives of feedback and feedforward processing, we see in Eqs. 9.1 & 9.2 that iterative amortization and sequential autoregressive flows ultimately involve similar iterative estimation schemes, estimating changes from previous estimates. Iterative estimation is arguably the most generic scheme for both optimization and prediction, as it simply asks, "*should this variable be larger or smaller?*" In cases where variables change relatively smoothly, due to the continuity of time or functional constraints, this is a useful inductive bias for performing estimation. As we have shown in this thesis, the *same* mathematical principles can be applied to both perception (Part II) and control (Part III), highlighting the generality of these iterative estimation schemes. This suggests a unified set of relatively simple computational principles for performing a host of processes, an alluring possibility given the seeming standardization of biological neural "hardware." Thus, considering the biological connections discussed in Chapter 8, top-down and bottom-up cortical projections (mediated by pyramidal neurons) may use similar computational principles, respectively computing changes in predictions and latent estimates.

9.2 Combining Generative Perception & Control

As we have laid out each of these ideas separately, there is still work to be done in combining these techniques to form a unified agent, combining both perception *and* control. Importantly, this is not as simple as merely stitching together the computation graphs for perception and control, e.g., Figures 9.1b & 9.1d. We can see this by starting from the control-as-inference perspective (Levine, 2018), presented in Chapter 6, and considering a hierarchical policy, with an internal latent variable, $\mathbf{z}$. Then, we can express the agent-environment interaction[1] as

$$p_\theta(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{1:T}) = \rho(\mathbf{s}_1) \prod_{t=1}^{T} p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) p_\theta(\mathbf{z}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{<t}, \mathbf{a}_{<t}) p_\theta(\mathbf{a}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{\leq t}, \mathbf{a}_{<t}).$$

Note that the distribution over $\mathbf{z}$, i.e., $p_\theta(\mathbf{z}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{<t}, \mathbf{a}_{<t})$, is expressed as a direct conditional mapping, rather than as an inverse generative mapping (as in Part II). This is a result of the mathematical formulation of policy distributions and not a design choice. However, we can use Bayes' rule to re-express the posterior in terms of a generative mapping

$$p_\theta(\mathbf{z}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{<t}, \mathbf{a}_{<t}) = \frac{p_\theta(\mathbf{s}_t|\mathbf{s}_{<t}, \mathbf{z}_{\leq t}, \mathbf{a}_{<t}) p_\theta(\mathbf{z}_t|\mathbf{s}_{<t}, \mathbf{z}_{<t}, \mathbf{a}_{<t})}{p_\theta(\mathbf{s}_t|\mathbf{s}_{<t}, \mathbf{z}_{<t}, \mathbf{a}_{<t})}. \tag{9.3}$$

[1] To simplify notation, we express the environment in terms of states, i.e., an MDP. However, an additional mapping from states to *observations* could be used, yielding a POMDP, as in Marino and Yue, 2019.

Much like the perception generative models from Part II, the numerator in Eq. 9.3 is the agent's action-conditioned generative model, a joint distribution over internal latent variables and state observations. However, we see that the denominator is the marginal likelihood of the current observation, conditioned only on the past variables. To see the implications of this term, we can continue with the control-as-inference approach, introducing an approximate posterior, $\pi(\tau|O)$, now over $\tau \equiv [\mathbf{s}_{1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{1:T}]$. At time step t, we have

$$p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi(\mathbf{z}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{<t}, \mathbf{a}_{<t}, O)\pi(\mathbf{a}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{\leq t}, \mathbf{a}_{<t}, O),$$

where, as before, O denotes *optimality*. Plugging this into the variational objective (and ignoring Lagrange factors for simplicity), we then have

$$\mathcal{J} = \mathbb{E}_{p_{\text{env}},\pi}\Bigg[\sum_{t=1}^{T} \overbrace{r_t}^{\text{reward}} + \overbrace{\log \frac{p_\theta(\mathbf{s}_t|\mathbf{s}_{<t}, \mathbf{z}_{\leq t}, \mathbf{a}_{<t})}{p_\theta(\mathbf{s}_t|\mathbf{s}_{<t}, \mathbf{z}_{<t}, \mathbf{a}_{<t})}}^{\text{Task Information Gain}} - \overbrace{\log \frac{\pi(\mathbf{z}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{<t}, \mathbf{a}_{<t}, O)}{p_\theta(\mathbf{z}_t|\mathbf{s}_{<t}, \mathbf{z}_{<t}, \mathbf{a}_{<t})}}^{\text{Internal KL}} \tag{9.4}$$
$$\underbrace{- \log \frac{\pi(\mathbf{a}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{\leq t}, \mathbf{a}_{<t}, O)}{p_\theta(\mathbf{a}_t|\mathbf{s}_{\leq t}, \mathbf{z}_{\leq t}, \mathbf{a}_{<t})}}_{\text{Action KL}}\Bigg].$$

The first and last term are the typical reward and action KL from the control-as-inference formulation, modified for the hierarchical setting. The "Internal KL" term resembles the latent dynamics found within the generative models from Part II.

However, instead of just the conditional log-likelihood of the current state observation, $\mathbf{s}_t$, we have a "Task Information Gain" term. This is the crucial difference from simply combining the perception and control techniques from this thesis, which would only yield the numerator in this term. Briefly, because $\mathbf{z}$ is sampled from π, which is optimized for the task (i.e., conditioned on O), the information gain can be thought of as the amount of task-relevant information extracted from the current state observation. In other words, *the agent is optimized to learn an internal model that maximally extracts task-relevant information from the environment.* This contrasts with current approaches to model-based reinforcement learning with latent variable models (Ha and Schmidhuber, 2018; Hafner et al., 2019), which learn to model observations as well as possible, i.e., the simple combination of perception and control techniques. Intuitively, by modeling *everything*, the agent's model may neglect less salient task-relevant information, resulting in poor performance. Similarly, from an evolutionary perspective, organisms obtain no utility in modeling their environment unless it can extract information relevant for survival and reproduction.

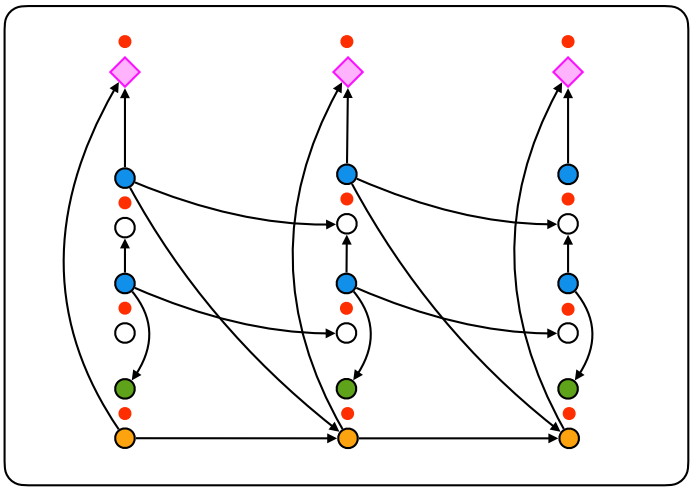


Figure 9.2: **Combining Generative Perception & Control**. Graphical representation combining generative perception (Part II) and control (Part III). Internal perceptual latent variables are a hierarchical variable in the control policy. Importantly, the agent maximizes the task-relevant *information gain* from the environment's state observation, rather than the conditional log-likelihood.

Stepping back, this exercise also demonstrates that internal perceptual variables may be reasonably considered as hierarchical variables in the control policy. This echoes the "motor chauvinist" view from neuroscience (Wolpert, Ghahramani, and Flanagan, 2001): the brain is ultimately concerned with control, with perception simply a means of improving control. Interestingly, if we consider deeper hierarchies of variables, *there may not be a clear line between perception and control*, supporting the application of similar computational principles put forth in this thesis. For a more in-depth derivation of this formulation, we refer to Marino and Yue, 2019.

In this section, we have attempted to combine perception and control by treating perception as an initial stage of processing, providing an input to the control policy. While we considered systems with only a single internal latent variable for perception, one could, in theory, construct policies with many hierarchical levels. However, even with many variables, this scheme is ultimately just a method for parameterizing more complex policy distributions, interpreting intermediate variables as "perceptual variables." In this sense, we have a hierarchy of *actions*. In the next section, we discuss an alternative approach toward combining perception and control, instead using a hierarchy of *tasks*.

9.3 Controlling Perception

In a task hierarchy, each level receives a task (e.g., a goal) from the level above, defined in the space of the perceptual representation at the current level. The level's "policy" attempts to satisfy this task by assigning tasks to the level below. In this way, each level attempts to *control its perception* (Powers, 1973), bringing the perceptual state estimate toward the goal. The top level assigns sub-tasks to solve the overall task, e.g., survival, and the lowest level outputs primitive actions to the environment. By estimating and controlling perceptions at each level, the agent can control abstract aspects of the environment, simplifying the process of solving the overall task over larger spatiotemporal scales.

To a greater degree than in an action hierarchy, perception and control are intimately linked in this formulation. That is, perception does not merely precede control. Rather, the two processes are concurrent, with perception providing the basis for estimating and evaluating control. In the RL literature, this setup is sometimes referred to as feudal RL (Dayan and Hinton, 1993), with several recent incarnations (Vezhnevets et al., 2017; Nachum et al., 2018a). In the control theory literature, it is sometimes referred to as hierarchical or cascaded control (Albus, Barbera, and Nagel, 1980). And in the cognitive science and neuroscience literature, this relates to ideas of hierarchical cognitive control (Badre, 2008; Yin, 2016). We now discuss several aspects of this approach, as well as connections to concepts across various fields.

Task Specification with State Distributions

We require some method for defining the tasks throughout the hierarchy. To define these tasks, we first step back and reconsider the general framing of tasks. In Part III, we utilized the control-as-inference formulation of reinforcement learning, largely following the notation of Levine, 2018. This formulation, born out of an effort to frame reinforcement learning (Sutton and Barto, 2018) in terms of probabilistic inference, makes the somewhat unnatural move of introducing auxiliary "optimality" variables (Cooper, 1988), O, which are set to 1 by construction. While this mathematical trick provides a convenient framing of the policy optimization problem in RL, one is still left with the difficult and opaque task of defining and optimizing a reward function. Further, because the reward function is considered as a fundamental element of the task formulation, one does not have direct access to the underlying calculations inside the reward function when considering the probabilistic modeling and inference procedure.

For instance, in Chapter 6, we adapted iterative amortized inference (Chapter 3) for policy optimization. In Chapter 3, in addition to gradient encoding models, which use $\nabla\mathcal{L}$, we considered Gaussian priors and conditional likelihoods, allowing us to define inference models that encode (prediction) errors, $\boldsymbol{\varepsilon}_{\mathbf{x}}$ and $\boldsymbol{\varepsilon}_{\mathbf{z}}$. However, in Chapter 6, because we were dealing with generic reward functions, we could only consider gradient encoding models. While similar squared error terms appear within the reward functions of the MuJoCo benchmark environments, these were effectively hidden from us. By more explicitly considering the origin and calculation of reward within the probabilistic modeling perspective, we may be able to devise more general principles for specifying and performing tasks.

This thesis has focused, in part, on revitalizing ideas developed under cybernetics using modern machine learning techniques. With its focus on both biological and non-biological systems, cybernetics framed objectives in terms of *homeostasis* (Cannon, 1929), or the maintenance of equilibria or "setpoints" on states (Ashby, 1952). Rather than a reward function, an agent is given a desired setpoint or goal state (e.g., a thermostat temperature), or more generally, a desired probability distribution over state variables. Indeed, this perspective is still prevalent in the control literature, where many reward and cost functions are defined in terms of squared errors, i.e., Gaussian log-likelihoods, from setpoints. We can even further generalize this notion to *allostasis* (Sterling, 1988), yielding desired distributions over state trajectories. Thus, while Sutton and Barto, 2018 describe reward maximizing agents as "*qualitatively different from equilibrium-seeking systems*," given the proper probabilistic framing, as discussed below, this distinction disappears. Defining tasks first in terms of state or trajectory distributions, rather than (simply) in terms of reward functions, could prove useful in building and understanding autonomous systems, potentially leading to improved methods for specifying and solving tasks.

Task Specification

Distribution-defined tasks have the benefit of naturally lending themselves toward probabilistic modeling and inference techniques. As discussed in Chapter 1, the various descendent fields of cybernetics each formulated control problems in terms of probabilistic inference. There is a rich line of work formulating planning (Attias, 2003; Botvinick and Toussaint, 2012; Piché et al., 2019) and control (Toussaint and Storkey, 2006; Todorov, 2008; Hoffman et al., 2009; Toussaint, 2009; Rawlik, Toussaint, and Vijayakumar, 2013) in terms of probabilistic inference. Within neuroscience, Friston's line of work on active inference (K. Friston, Daunizeau, and

Kiebel, 2009) similarly formulates control as the inference of actions to maximize the (homeostatic) likelihood of states (Morville et al., 2018). While each of these formulations offers unique perspectives on the control problem, the commonality is that many of these are formulated in terms of desired state distributions, which then give rise to reward (or cost functions) as the resulting log-likelihood.[2] In other words, for a desired state distribution, $p_d(S)$, the reward, r, that an agent receives in a particular state, s, is proportional to the log-likelihood of that state under the desired state distribution,

$$r(s) \propto \log p_d(S = s). \tag{9.5}$$

For instance, a quadratic reward function between the current state and some setpoint, s^*, i.e.,

$$r(s) = -\frac{1}{2}||s - s^*||_2^2,$$

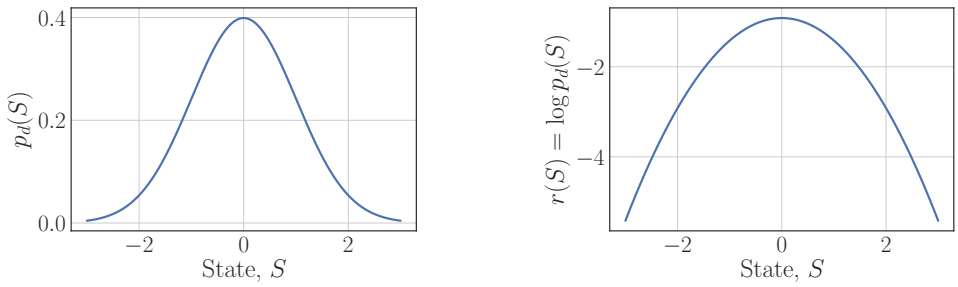
is equivalent to defining a Gaussian desired state distribution with mean s^*:

$$p_d(S) = \mathcal{N}(S; s^*, 1), \tag{9.6}$$

as shown in Figure 9.3. State distributions, rather than reward functions, may be a more natural way to specify tasks. This could range from simple distributions, as in the Gaussian case, to sharp distributions, such as a Dirac delta function for a particular state, e.g., a solved Rubik's cube. As discussed above, state distributions can also be time-varying, i.e., allostatic, defined over state trajectories. This is essential in situations where homeostatic variables need to change, e.g., blood pressure, to meet environmental requirements (Sterling, 2012).

As a final note here, it is worth mentioning that the notion of "reward" itself is an abstraction, a remnant of an earlier behaviorist paradigm. Although reward, as a concept, has served as a useful tool for making progress in computational approaches to control, we should not endow it with special status. At a more fundamental level, the common characteristic of organisms, machines, and "intelligent" systems broadly is their ability to perform work to alter the state of the environment (Levin et al., 2011). Defining such systems in terms of distributions over states, rather than an inferred scalar quantity (reward), perhaps offers a more congruent framing.

[2]Note that other formulations cast reward as the *change* in homeostatic log-likelihood (Keramati and Gutkin, 2014).



(a) (b)

Figure 9.3: **State Distributions & Rewards.** A quadratic reward function, $r(S)$, can be equivalently expressed as a Gaussian desired state distribution, $p_d(S)$.

Task Solving

The exponentiation interpretation above is highly similar to the control-as-inference perspective (Cooper, 1988), where we defined the likelihood of optimality as

$$p(O = 1) \propto \exp(r(s)).$$

However, unlike control-as-inference, where the variable of interest is the abstractly-defined "optimality," the state distribution approach considers states directly, yielding a more interpretable problem. The policy optimization objective then becomes the following cross-entropy:

$$\mathbb{E}_{a\sim\pi(A)}\mathbb{E}_{s\sim p_{\text{env}}(S|A=a)}\left[\log p_d(S = s)\right], \tag{9.7}$$

where, here, we have considered a single time step for simplicity. Unlike log-optimality, the objective in Eq. 9.7 has a simple interpretation: find a control policy that puts the environment in states with high log-probability under the desired distribution. In other words, *make the environment state look more like your desires*. Further, if we extend Eq. 9.7 to a KL divergence (Lee et al., 2019), this provides an additional state entropy term, encouraging maximum coverage over states, i.e., exploration and novelty-seeking.

Desired state distributions have the immediate benefit of exposing the calculation of reward to the probabilistic formulation and inference technique. Considering the Gaussian example in Eq. 9.6, the task error, $\varepsilon_s \equiv s - s^*$, is no longer hidden inside the black box of the reward function. Thus, through (iterative) amortization, we could potentially perform policy optimization by inverting this state error to yield an updated control policy. This is the idea behind negative feedback control techniques (Astrom and Murray, 2008), instantiated in so-called *comparator circuits*, a key concept from cybernetics (Wiener, 1948; Ashby, 1956).

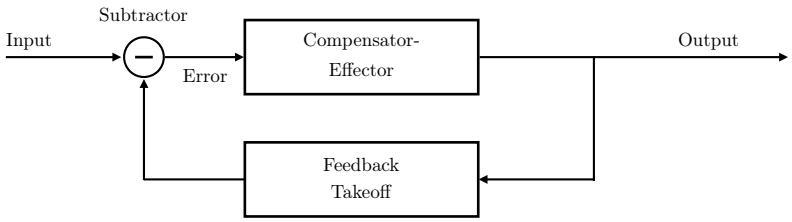


Figure 9.4: **Comparator Circuit**. Using the error between the current state and the input reference signal (or setpoint), a compensator-effector function updates the output control. The feedback takeoff function converts this control output into the current state. An ideal compensator-effector minimizes the discrepancy between the current state and the reference signal. Adapted from Wiener, 1948.

Comparator circuits (Figure 9.4) consist of an input reference signal (or setpoint), a compensator-effector function, a feedback takeoff function, and a control output. The current state, given by the feedback takeoff function, is compared with the reference signal. The error is then fed to the compensator-effector function, which adjusts the control output to minimize this error (negative feedback). Broadly, we can identify the input reference signal as the mean of a Gaussian desired state distribution (s^*), the compensator-effector function as a type of iterative inference model, and the feedback takeoff function as the environment (or model). While the basic comparator circuit neglects many of the underlying mathematical details and assumptions, it nevertheless provides a useful example of feedback control that appears to fit nicely with iterative amortization, and importantly, desired state distributions, where an explicit error can be calculated.

Biological organisms contain negative feedback processes operating at multiple scales, altering the internal state of the organism and the external environment. From an evolutionary point of view, such homeostatic feedback control processes are only useful insofar as they enable genetic propagation (Dawkins, 1976). Thus, the homeostatic desired state distribution is itself learned through evolution, translating genetic survival into distributions over internal states (or trajectories). From this perspective, nervous systems are merely an extension of this trend, enabling more flexible forms of feedback over a range of time horizons. As argued by Damasio, 2019, this requires some method for estimating the (log) likelihood of current and future states under the homeostatic distribution, which he identifies as *feelings*.

Hierarchical Control

The setup discussed thus far has only considered a single desired state distribution, recapitulating the standard RL setup from an alternative mathematical perspective. However, by framing tasks in terms of desired distributions, we can readily extend this formulation to hierarchical setups, in which higher levels assign desired *perceptual* distributions to lower levels (i.e., perceptual goals). These lower levels then assign desired perceptual distributions to even lower levels (or actions), attempting to maximize the likelihood of perceptions under the distribution from above. In this way, each level operates semi-autonomously as an "agent," performing tasks in perceptual state-spaces that are increasingly coarse-grained over space and time. These perceptual spaces could, in principle, be defined in terms of generative mappings, as in Part II. Under this scheme, negative feedback and feedforward perception and control processes would operate in tandem at each level, updating perceptual estimates and goals, respectively.

The brain structures associated with homeostatic regulation are evolutionarily ancient, located in the brain stem. More recent structures, such as the limbic system and cortex, provide additional layers of regulation, estimating and controlling more abstract external states. The pinnacle of this abstraction is located in prefrontal cortex, which is hypothesized to infer a hierarchy of increasingly abstract high-value (goal) states (Pezzulo, Rigoli, and K. J. Friston, 2018). Through a descending hierarchy of proprioceptive "predictions," these goal states are translated into muscle tensions, enacted in the spinal cord through (negative feedback) reflex arcs (Adams, Shipp, and K. J. Friston, 2013). This *cognitive control* setup broadly agrees with perceptual control theory (Powers, 1973), which posits that actions are engaged in controlling a hierarchy of perceptual estimates via negative feedback, with homeostatic variables dictating the top of the hierarchy. This overall architecture has also been described by Meystel and Albus, 1997 from a more control-theoretic perspective.

Although ideas on the relationship between feedback and the brain have existed since the dawn of cybernetics (Rosenblueth, Wiener, and Bigelow, 1943; Ashby, 1952; Powers, 1973), it is only through recent discoveries in neuroscience that a more detailed picture of higher levels, outlined above, has started to emerge (Cools, 1985; Yin, 2014; Pezzulo and Cisek, 2016; Badre, 2020). Through developments in machine learning, in part advanced through this thesis, implementation of these ideas may be on the horizon. However, while there has been progress in goal-based reinforcement learning (Schaul et al., 2015; Andrychowicz et al.,

2017; Eysenbach, Salakhutdinov, and Levine, 2019) and goal hierarchies (Nachum et al., 2018a; Nachum et al., 2018b), substantial insights are still needed. For instance, temporal abstraction (Sutton, Precup, and Singh, 1999), despite its agreed upon importance, remains computationally elusive. Inspiration from neuroscience, particularly regarding perceptual control, as opposed to the conventional dichotomy of model-free and model-based control, may provide useful directions for progress. We have purposefully left the formulation in this section somewhat vague, as many details are left to be worked out. Yet, we feel strongly that this perspective, starting from homeostasis and scaling up toward hierarchical perceptual control, is the most promising path toward building capable systems and understanding the brain.

9.4 Bridging Neuroscience and Machine Learning

The approach taken in this thesis, applying ideas from neuroscience to machine learning (and vice versa), is, unfortunately, fairly uncommon. If nothing else, we hope that the reader has come to appreciate the connections between these fields, first emphasized by cybernetics. The contributions within this thesis were inspired by ideas from theoretical neuroscience, which, in turn, originated from information theory, control theory, and cybernetics more broadly. Looking at the history of these ideas, it becomes clear that there is a proven track record of making progress through interaction between neuroscience and machine learning. Bridging these fields is difficult, but it is a well-documented path toward further progress.

Indeed, there are *multiple* bridges between neuroscience and machine learning, each operating at different scales or in different brain regions. In this thesis, we have explored a circuit-level description for inspiration and correspondences, primarily drawn from existing work in predictive coding (Srinivasan, Laughlin, and Dubs, 1982; Rao and Ballard, 1999). These circuits are far more complex than generic neural network architectures, involving multiple objective terms, multiple functions, and specific probabilistic computations. Yet, these models are entirely abstracted away from ion currents, actions potentials, and neuromodulators. In our view, this is a useful level of abstraction for describing the computational principles underlying neural systems. Some previous works have attempted to compare similar learned models with neural data (Lotter, Kreiman, and Cox, 2018), however, additional work is needed. By scaling up these models, our hope is that future researchers can explore and test aspects of predictive coding in detail, alongside neuroscience experiments.

To conclude, we will briefly review the neuroscience implications of the ideas

presented in this thesis. Building on the cybernetic tradition, we have investigated improved approaches to feedback and feedforward perception and control. While the structure (or circuit architecture) of these approaches is predefined, the processes themselves are learned from data. That is, we *learn* feedforward predictions via autoregressive flows, and we *learn* negative feedback updates via iterative amortization. This demonstrates that a limited set of computational operations, combined appropriately, is capable of adapting to environment-specific perception and control tasks. In a similar way, predefined neural circuit architectures, particularly in cortex, are capable of adapting to the specific perceptual and control tasks encountered by an organism during its lifetime.

Notably, in addition to the basic arithmetic operations for probabilistic computations, e.g., subtraction for error and multiplication for precision-weighting, these approaches rely on learned non-linear mappings, parameterized by deep networks. By identifying the probabilistic computations as occurring *across* neurons, as in predictive coding, the non-linear mappings are implicated in occurring *within* neurons. Importantly, this interpretation hinges on implementing negative feedback (inference) optimization using amortization. Thus, a significant portion of neurons may be devoted to learning to optimize the perceptual and control estimates of other neural circuits. While this idea has a long history (Rosenblueth, Wiener, and Bigelow, 1943), amortization provides a new lens for understanding how such corrective processes can be *learned*.

Much work remains to be done in formulating and understanding neural systems, and biological systems generally, in computational terms. Humans, in particular, through a complex array of perceptual and control processes, are capable of bringing about wildly improbable, high-value environmental states. While it is tempting to think that we are close to "solving intelligence," we are still only scratching the surface of basic engineering principles. Substantial work remains to be done in integrating and scaling these systems. May we continue to look toward the complexity and wonder of biological systems as a guiding source of inspiration.

References

Adams, Rick A, Stewart Shipp, and Karl J Friston (2013). "Predictions not commands: active inference in the motor system". In: *Brain Structure and Function* 218.3, pp. 611–643.

Albus, James, Anthony J Barbera, and Roger N Nagel (1980). *Theory and practice of hierarchical control*. National Bureau of Standards.

Andrychowicz, Marcin et al. (2017). “Hindsight experience replay”. In: *Advances in neural information processing systems*, pp. 5048–5058.

Ashby, W Ross (1952). *Design for a brain: The origin of adaptive behaviour*.

– (1956). *An Introduction to Cybernetics*. Chapman and Hall.

Astrom, Karl Johan and Richard M Murray (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

Attias, Hagai (2003). “Planning by probabilistic inference.” In: *AISTATS*. Citeseer.

Badre, David (2008). “Cognitive control, hierarchy, and the rostro–caudal organization of the frontal lobes”. In: *Trends in cognitive sciences* 12.5, pp. 193–200.

– (2020). “On Task”. In: *On Task*. Princeton University Press.

Botvinick, Matthew and Marc Toussaint (2012). “Planning as inference”. In: *Trends in cognitive sciences* 16.10, pp. 485–488.

Cannon, Walter B (1929). “Organization for physiological homeostasis”. In: *Physiological reviews* 9.3, pp. 399–431.

Cools, AR (1985). “Brain and behavior: hierarchy of feedback systems and control of input”. In: *Perspectives in ethology*. Springer, pp. 109–168.

Cooper, Gregory F (1988). “A method for using belief networks as influence diagrams”. In: *Fourth Workshop on Uncertainty in Artificial Intelligence*.

Damasio, Antonio (2019). *The strange order of things: Life, feeling, and the making of cultures*. Vintage.

Dawkins, Richard (1976). *The selfish gene*. Oxford university press.

Dayan, Peter and Geoffrey Hinton (1993). “Feudal reinforcement learning”. In: *Advances in Neural Information Processing Systems*.

Eysenbach, Ben, Russ R Salakhutdinov, and Sergey Levine (2019). “Search on the replay buffer: Bridging planning and reinforcement learning”. In: *Advances in Neural Information Processing Systems*, pp. 15246–15257.

Friston, Karl, Jean Daunizeau, and Stefan J Kiebel (2009). “Reinforcement learning or active inference?” In: *PloS one* 4.7, e6421.

Greff, Klaus et al. (2019). “Multi-Object Representation Learning with Iterative Variational Inference”. In: *International Conference on Machine Learning*, pp. 2424–2433.

Ha, David and Jürgen Schmidhuber (2018). “Recurrent world models facilitate policy evolution”. In: *Advances in Neural Information Processing Systems*, pp. 2450–2462.

Hafner, Danijar et al. (2019). “Learning Latent Dynamics for Planning from Pixels”. In: *International Conference on Machine Learning*, pp. 2555–2565.

Hoffman, Matthew et al. (2009). "An expectation maximization algorithm for continuous Markov decision processes with arbitrary reward". In: *Artificial Intelligence and Statistics*, pp. 232–239.

Keramati, Mehdi and Boris Gutkin (2014). "Homeostatic reinforcement learning for integrating reward collection and physiological stability". In: *Elife* 3, e04811.

Lee, Lisa et al. (2019). "Efficient exploration via state marginal matching". In: *arXiv preprint arXiv:1906.05274*.

Levin, Robert et al. (2011). *Work meets life: exploring the integrative study of work in living systems*. MIT Press.

Levine, Sergey (2018). "Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review". In: *arXiv preprint arXiv:1805.00909*.

Lotter, William, Gabriel Kreiman, and David Cox (2018). "A neural network trained to predict future video frames mimics critical properties of biological neuronal responses and perception". In: *arXiv preprint arXiv:1805.10734*.

Marino, Joseph and Yisong Yue (2019). "An Inference Perspective on Model-Based Reinforcement Learning". In: *ICML Workshop on Generative Modeling and Model-Based Reasoning for Robotics and AI*. URL: https://drive.google.com/open?id=1uc1FdjHVkkEAaU6jz7aVrP3khGyqkdKi.

Meystel, Al M and James Albus (1997). "Intelligent Systems". In: *Architecture, Designand Control*.

Morville, Tobias et al. (2018). "The homeostatic logic of reward". In: *bioRxiv*, p. 242974.

Nachum, Ofir et al. (2018a). "Data-efficient hierarchical reinforcement learning". In: *Advances in Neural Information Processing Systems*, pp. 3307–3317.

– (2018b). "Near-Optimal Representation Learning for Hierarchical Reinforcement Learning". In: *International Conference on Learning Representations*.

Pezzulo, Giovanni and Paul Cisek (2016). "Navigating the affordance landscape: feedback control as a process model of behavior and cognition". In: *Trends in cognitive sciences* 20.6, pp. 414–424.

Pezzulo, Giovanni, Francesco Rigoli, and Karl J Friston (2018). "Hierarchical active inference: A theory of motivated control". In: *Trends in cognitive sciences* 22.4, pp. 294–306.

Piché, Alexandre et al. (2019). "Probabilistic Planning with Sequential Monte Carlo methods". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=ByetGn0cYX.

Powers, William T (1973). *Behavior: The control of perception*. Aldine Chicago.

Rao, Rajesh PN and Dana H Ballard (1999). "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects." In: *Nature neuroscience* 2.1.

Rawlik, Konrad, Marc Toussaint, and Sethu Vijayakumar (2013). "On stochastic optimal control and reinforcement learning by approximate inference". In: *Twenty-Third International Joint Conference on Artificial Intelligence*.

Rosenblueth, Arturo, Norbert Wiener, and Julian Bigelow (1943). "Behavior, purpose and teleology". In: *Philosophy of science* 10.1, pp. 18–24.

Schaul, Tom et al. (2015). "Universal value function approximators". In: *International conference on machine learning*, pp. 1312–1320.

Srinivasan, Mandyam Veerambudi, Simon Laughlin, and Andreas Dubs (1982). "Predictive coding: a fresh view of inhibition in the retina". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 216.1205, pp. 427–459.

Sterling, Peter (1988). "Allostasis: a new paradigm to explain arousal pathology". In: *Handbook of life stress, cognition and health*.

– (2012). "Allostasis: a model of predictive regulation". In: *Physiology & behavior* 106.1, pp. 5–15.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, Richard S, Doina Precup, and Satinder Singh (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* 112.1-2, pp. 181–211.

Todorov, Emanuel (2008). "General duality between optimal control and estimation". In: *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, pp. 4286–4292.

Toussaint, Marc (2009). "Robot trajectory optimization using approximate inference". In: *International conference on machine learning*, pp. 1049–1056.

Toussaint, Marc and Amos Storkey (2006). "Probabilistic inference for solving discrete and continuous state Markov Decision Processes". In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 945–952.

Vezhnevets, Alexander Sasha et al. (2017). "Feudal networks for hierarchical reinforcement learning". In: *International Conference on Machine Learning*. PMLR, pp. 3540–3549.

Wiener, Norbert (1948). *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press.

Wolpert, Daniel M, Zoubin Ghahramani, and J Randall Flanagan (2001). "Perspectives and problems in motor learning". In: *Trends in cognitive sciences* 5.11, pp. 487–494.

Yin, Henry H (2014). "How basal ganglia outputs generate behavior". In: *Advances in neuroscience*.

Yin, Henry H (2016). “The basal ganglia and hierarchical control in voluntary behavior”. In: *The Basal Ganglia*. Springer, pp. 513–566.