Stochastic Analog Computation for Machine Learning

by

Yuan-Sheng Fang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Physics

in the

Graduate Division

of the

University of California, Berkeley

Spring 2020

Stochastic Analog Computation for Machine Learning

Abstract

Stochastic Analog Computation for Machine Learning

by

Yuan-Sheng Fang

Doctor of Philosophy in Physics

University of California, Berkeley

Associate Professor Michael R. DeWeese, Chair

Analog computers model logical and mathematical operations by exploiting the physical properties of continuously evolving systems. Variables are often directly represented by easily measurable quantities such as voltage, current, etc. By contrast, in typical digital computers, the representation is much more abstract and manipulated in discrete time. Despite certain advantages of analog computers, such as their power efficiency and speed, they were eventually made obsolete by the better scalability of their digital counterparts. However, with the recent advances in machine learning, specialized applications of analog computation, such as optical neural networks, are becoming more viable. Presented here is a collection of topics relevant to analog computing for machine learning. Rather than a set of algorithmic procedures, machine learning models can be treated as physical systems and studied accordingly. Specifically discussed are parameter estimation of the Ising spin glass, deep learning with photonic networks, and fully analog implementation of latent variable models.

For the most part, experimental physics is interested in the observation and measurement of a system under various conditions. On the other hand, much of machine learning is concerned with inferring the underlying properties of a system from known observations. This type of inference is often referred to as the inverse problem by physicists. In Chapter 1, one such concretely formalized task, parameter estimation, is used to study the Ising model and Hopfield network.

The next chapter explores certain practical problems associated with implementing neural networks with photonic components – optical neural networks (ONN). The analog noise and imprecisions are present in all analog computers and impact their performance. Accordingly, the proper characterization of ONNs require quantifying the effects of fabrication errors and other noise on their operation. The trade-off between expressivity and robustness is explored through comparison of two ONN architectures.

Rather than minimizing the effects of noise in analog systems, in the last chapter, it is demonstrated that they can be leveraged for more efficient computation. Manipulation and analysis of probabilistic models often require the generation of continuous random variables. Instead of using a deterministic, digital computer for this task, we demonstrate that this can be much more efficiently done with an analog computer with inherent variability.

A more, in depth, summary of these topics is presented in the introduction. While each chapter is self-contained, the common theme is a departure from discrete, deterministic approaches to computation and a step toward continuous and stochastic dynamics. Taken as a whole, the thesis acts as reference for further study in analog computation.

Contents

List of Figures

List of Tables

Acknowledgments

This thesis is only made possible by all the people who accompanied me during this nearly six-year journey. First, of course, Mom and Dad, who brought me into this world, led me half-way across and taught me to fall in love with its magics and wonders.

I want to thank my advisor, Mike DeWeese for his enthusiasm throughout. His unconditional support allowed me to work on a vast range of interdisciplinary topics.

I would also like to express my gratitude to the entirety of the Redwood Center. Here, I have benefited from the countless exchanges with so many amazing scientists and learned a tremendous amount about physics, machine learning and neuroscience. In particular, I am grateful to have worked with Bruno Olshausen. He is an immense source of inspiration to not only myself but to everyone at Redwood. With apologies in advance for anyone missed, I would like to thank all of my colleagues, James Arenmann, Connor Bybee, Yubei Chen, Brian Cheung, Vasha Dutell, Charles Frye, Louis Kang, Spencer Kent, Mayur Mudigonda, Pratik Sachdeva, Sophia Sanborn, Neha Wadia, Ryan Zarcone, I would not have made it through without their friendship and encouragement.

Outside of the lab, I am equally thankful to all my mentors for their guidance, Dmitri Budker, Hoi-Ying Holman, Cas Wierzynski, Sasikanth Manipatruni and many more. From them, I learned to be a better scientist and more importantly, become more than a scientist. In particular, I want to thank Amir Khosrowshahi for providing me the motivation and support towards graduation and beyond. Special thanks, also, to long-time friend and collaborator Arthur Montazeri for his companionship and energy.

Once again, I want to thank all of my friend friend, collaborators, and mentors, especially those I have forgotten to name. This PhD has been, without a doubt, more about the people involved than the science.

Abstract

Stochastic Analog Computation for Machine Learning

by

Yuan-Sheng Fang

Doctor of Philosophy in Physics

University of California, Berkeley

Associate Professor Michael R. DeWeese, Chair

Analog computers model logical and mathematical operations by exploiting the physical properties of continuously evolving systems. Variables are often directly represented by easily measurable quantities such as voltage, current, etc. By contrast, in typical digital computers, the representation is much more abstract and manipulated in discrete time. Despite certain advantages of analog computers, such as their power efficiency and speed, they were eventually made obsolete by the better scalability of their digital counterparts. However, with the recent advances in machine learning, specialized applications of analog computation, such as optical neural networks, are becoming more viable. Presented here is a collection of topics relevant to analog computing for machine learning. Rather than a set of algorithmic procedures, machine learning models can be treated as physical systems and studied accordingly. Specifically discussed are parameter estimation of the Ising spin glass, deep learning with photonic networks, and fully analog implementation of latent variable models.

For the most part, experimental physics is interested in the observation and measurement of a system under various conditions. On the other hand, much of machine learning is concerned with inferring the underlying properties of a system from known observations. This type of inference is often referred to as the inverse problem by physicists. In Chapter 1, one such concretely formalized task, parameter estimation, is used to study the Ising model and Hopfield network.

The next chapter explores certain practical problems associated with implementing neural networks with photonic components – optical neural networks (ONN). The analog noise and imprecisions are present in all analog computers and impact their performance. Accordingly, the proper characterization of ONNs require quantifying the effects of fabrication errors and other noise on their operation. The trade-off between expressivity and robustness is explored through comparison of two ONN architectures.

Rather than minimizing the effects of noise in analog systems, in the last chapter, it is demonstrated that they can be leveraged for more efficient computation. Manipulation and analysis of probabilistic models often require the generation of continuous random variables. Instead of using a deterministic, digital computer for this task, we demonstrate that this can be much more efficiently done with an analog computer with inherent variability.

A more, in depth, summary of these topics is presented in the introduction. While each chapter is self-contained, the common theme is a departure from discrete, deterministic approaches to computation and a step toward continuous and stochastic dynamics. Taken as a whole, the thesis acts as reference for further study in analog computation.

Chapter 0

Introduction

This thesis is motivated by an interest in studying machine learning with tools available to physicists. Various techniques in physics have been employed for better understanding of machine learning problems. As one example, mean field theory has been used to study batch normalization in neural networks [87]. Another direction is to discover new models and solutions. Both the Hopfield networks [32] and the restricted Boltzmann machine (RBM) [66] are closely related to the Ising spin glass and are known models for auto-associative memory and representing latent variables respectively. In Chapter 2, it is shown that the capacity of Hopfield networks to store patterns can be increased through a parameter estimation scheme called minimum probability flow (MPF) [74].

In addition to theory, expertise in experimental physics has successfully been applied in designing new hardware for acceleration of deep learning. Of particular interest are optical neural networks (ONNs) [69]. In Chapter 3, the fabrication imprecisions of [some optical devices] are modeled and their effects on ONNs are characterized. Strategies for mitigation of the effects of analog noise are also presented. Moreover, instead of minimizing such effects, the analog noise may actually be used to aid computation as well. A framework for how this may be implemented for sparse coding [52] – a latent variable model – is shown in Chapter 4.

Because each of the topics discussed are fairly specialized, they are presented in self-contained chapters that follow. In the current introductory chapter, brief descriptions of various concepts are presented to make the thesis more accessible to a general audience.

0.1 Parameter estimation

Parameter estimation is the goal of inferring parameter(s) θ of some model from observations $\mathbf{X}$. While very common in machine learning, this is considered the inverse problem in physics where often theory is developed to model observations from a system with given parameters. Consider the Ising spin glass model as an example. The system consists of N spins described by vector $\mathbf{s} \in \{-1, +1\}$ and has an energy of the form

$$E(\mathbf{s}; J) = \frac{1}{2}\mathbf{s}^T J\mathbf{s}. \tag{1}$$

Rather than understanding the statistics of $\mathbf{s}$, one might be interested in estimating the coupling matrix J from a set of observations $\{\mathbf{s}_i\}$ sampled from this system. One way of formalizing this goal is to select J^* that maximizes the likelihood of these observations. Generally, this can be very difficult. Various methods for parameter estimation have been developed and is an active area of research[78]. One such method is called minimum probability flow (MPF) and has found application in various machine learning tasks. In the subsequent chapter, its applicability in training Hopfield networks is demonstrated.

Maximum Likelihood

For a set of samples $\{\mathbf{s}_i | i = 1, \ldots, N\}$, the likelihood of it being drawn i.i.d. from distribution $p(\mathbf{s}; J)$ is

$$L(J) = p(\{\mathbf{s}_i\}; J) = \prod_{i=1}^{N} p(\mathbf{s}_i; J). \tag{2}$$

Conventionally, the negative log-likelihood is considered instead.

$$\mathcal{L}(J) = -\log L(J) = -\sum_{i=1}^{N} \log p(\mathbf{s}_i; J). \tag{3}$$

The max likelihood (ML) estimator for J is therefore

$$J^* = \arg\min \mathcal{L}(J) \tag{4}$$

Unfortunately, in the case of the Ising spin glass, along with many others, the normalized distribution is intractable. Formally,

$$p(\mathbf{s}; J) = \frac{e^{-E(\mathbf{s};J)}}{Z(J)} \tag{5}$$

where

$$Z(J) = \sum_{\mathbf{s}} e^{-E(\mathbf{s};J)} \tag{6}$$

is the partition function. and the sum is over all possible configurations of $\mathbf{s}$. The number of terms to be summed over is therefore 2^N and quickly becomes very large.

Nonetheless, because the energy is known, the ratio between likelihoods of two different configurations is directly calculable.

$$p(\mathbf{s}; J)/p(\mathbf{s}'; J) = \frac{e^{-E(\mathbf{s};J)}/Z(J)}{e^{-E(\mathbf{s}';J)}/Z(J)} = \exp(-(E(\mathbf{s}; J) - E(\mathbf{s}'; J)) \tag{7}$$

This fact is central to how MPF works.

Hopfield Network

A Hopfield network is a dynamical system consiting of N neurons $\mathbf{s} \in \{-1,+1\}^N$ with the same energy as the Ising spin glass (Eq. 1). At each time-step, a single neuron s_i is chosen, and either flips or stays to minimize the energy. More formally, for a given i, define $\mathbf{x}' = (x_1, \ldots, -x_i, \ldots, x_D)$ and then,

$$\mathbf{x} \leftarrow \begin{cases} \mathbf{x}' & \text{if } E(\mathbf{x}') < E(\mathbf{x}) \\ \mathbf{x} & \text{if } E(\mathbf{x}') \geq E(\mathbf{x}) \end{cases} \tag{8}$$

This process is iterated over all the dimensions $i = 1, \ldots, N$. Then repeated until convergence. Note that convergence is always possible as energy necessarily decreases whenever a neuron flips. The main topic of study is whether the local minimum of the energy landscape into which the neurons settle is meaningful.

As a model for memory, the aim is to store patterns $\{\xi^\alpha | \alpha = 1, \ldots, N\}$ as local minima. This way, given a corrupted version of one of the stored patterns, ξ^α, through the Hopfield update rule (Eq. 1.15), the corrupted pattern will converge to the stored memory (Fig. 1.1).

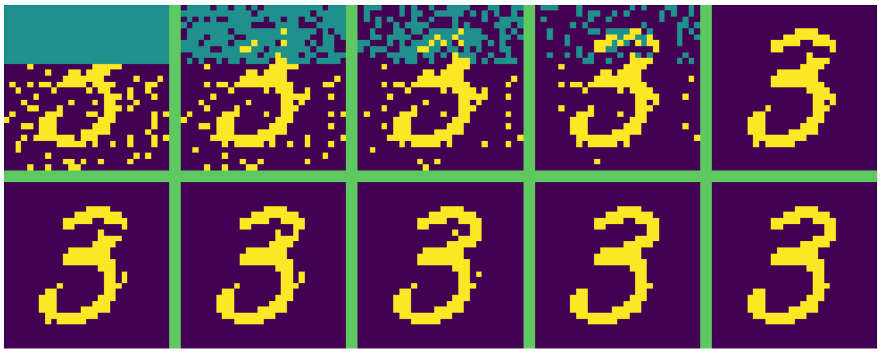


Fig. 0.1. Initialized to a pattern with both missing and noisy data, a Hopfield network eventually successfully converges to the desired, stored memory.

The challenge lies in devising a procedure for producing coupling matrix J for which $\{\xi^\alpha\}$ are local minima of the energy. There is a parallel between this and estimating J for which observations $\{\mathbf{s}_i\}$ are likely events. For this reason, one might attempt to use a parameter estimation scheme like MPF to train a Hopfield network. In Chapter 2, it is demonstrated that this is indeed possible and the procedure outperforms competing methods of training Hopfield networks.

0.2 Optical Neural Networks

An ONN is a physical implementation of artificial neural networks (ANNs) with optical components. While a variety of ONNs exists, here the focus is multilayer perceptrons (MLPs) based on photonic meshes of Mach-Zehnder Interferometers (MZIs).

Multilayer Perceptron

A multilayer perceptron (MLP) with K layers consists of weights $\{W_k \in \mathbb{R}^{N_k \times N_{k-1}} : k = 1 \dots K\}$ and biases $\{\theta_k \in \mathbb{R}^{N_k}\}$. Each layer is modeled as

$$f_k(\mathbf{x}) = \sigma(W_k \mathbf{x} + \theta_k). \tag{9}$$

σ is a nonlinear function applied element-wise (i.e. $\sigma(\mathbf{z})_i = \sigma(z_i)$). Then, the entire MLP can be expressed as

$$f(\mathbf{x}) = (f_K \circ f_{K-1} \circ \dots \circ f_1)(\mathbf{x}). \tag{10}$$

While the input and output dimension (N_0 and N_K) is typically fixed, the dimensions of intermediate layers (the width of the network) along with the total number of layers (the depth) can be arbitrary. This allows for a large number of tunable parameters. The nonlinearity σ is important because without it, the composition of linear functions is another linear function. For example, ignoring biases, we would have

$$f(\mathbf{x}) = W_K W_{K-1} \dots W_1 \mathbf{x} \equiv W\mathbf{x} \tag{11}$$

for $W \in \mathbb{R}^{N_K \times N_0}$. Regardless of the depth and width of a linear network, the number of non-redundant degrees of freedom is fixed at $N_K \times N_0$.

Roughly speaking, in training a MLP, the goal is to optimize parameters W_k, θ_k to minimize some loss function. This is often done through gradient descent. The architectural design of MLPs, the choice of loss functions and optimization strategies, while very important, is outside of the scope of this thesis.

Mach-Zehnder Interferometers

A single MZI is comprised of two 50-50 beamsplitters and two phaseshifters. Both photonic components act linearly on two inputs of light, with two outputs. Accordingly, they may be modeled by 2×2 transfer matrices,

$$U_{BS} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \tag{12}$$

and

$$U_{PS}(\theta) = \begin{pmatrix} e^{i\theta} & 0 \\ 0 & 1 \end{pmatrix}. \tag{13}$$

Monochrome light on two waveguides with amplitude $A_{1,2}$ and phase $\phi_{1,2}$ can be represented by the complex vector

$$\mathbf{z} = (A_1 e^{i\phi_1}, A_2 e^{i\phi_2})^T. \tag{14}$$

Going through a beamsplitter, the output is then

$$\mathbf{z}' = U_{BS}\mathbf{z}, \tag{15}$$

and for a phaseshifter,

$$\mathbf{z}'' = U_{PS}(\theta)\mathbf{z}. \tag{16}$$

Composing the transfer matrices of its constituent components, the MZI is represented as

$$\begin{aligned} U_{MZI}(\theta,\phi) &= U_{BS}U_{PS}(\theta)U_{BS}U_{PS}(\phi) && (17)\\ &= \frac{1}{2}\begin{pmatrix}1 & i\\ i & 1\end{pmatrix}\begin{pmatrix}e^{i\theta} & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}1 & i\\ i & 1\end{pmatrix}\begin{pmatrix}e^{i\phi} & 0\\ 0 & 1\end{pmatrix} && (18)\\ &= ie^{i\theta/2}\begin{pmatrix}e^{i\phi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2}\\ e^{i\phi}\cos\frac{\theta}{2} & -\sin\frac{\theta}{2}\end{pmatrix}. && (19)\end{aligned}$$

By tuning the phases (θ, ϕ) of the MZI, inputs can be made to interfere in a controlled manner. A mesh of MZIs, along with a few other photonic elements (Fig. 0.2) can implement any linear transformation of arbitrary dimensions.

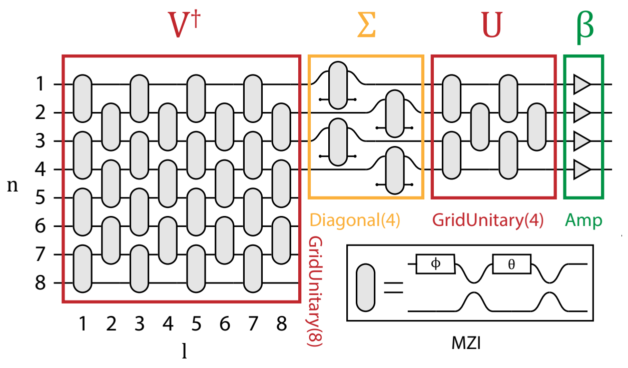


Fig. 0.2. A schematic of a universal 8×4 optical linear multiplier with two unitary multipliers (red) consisting of MZIs in a grid-like layout and a diagonal layer (yellow). The MZIs of GridUnitary multipliers are indexed according to their layer depth (l) and dimension (d). Symbols at the top represent the mathematical operations performed by the various modules. Inset: A MZI with two 50:50 beamsplitters and two tunable phaseshifters.

By interweaving optical nonlinearities such as saturable absorbers, a fully optical implementation of a multilayer perception is realizable. The content of Chapter 3 explores the effects of fabrication error and other sources of uncertainties on the operation of such optical neural networks.

0.3 Analog Sampling for Sparse Coding

The noise and error mentioned above is inherently present in all analog systems. These have been successfully leveraged into performing efficient sampling. In fact, an analog sampler built using photonic meshes similar to the one described in the previous section has been demonstrated [63]. In the last chapter, it is demonstrated that beyond just sampling, an analog system modeled by Langevin dynamics can be effectively used to train a sparse coding model.

Sparse Coding

Sparse coding is a simple yet efficient algorithm for dictionary learning. By adding an L_1 regularization to the reconstruction loss, sparsity of coefficients can be achieved. While the formulation is probabilistic, due to computational constraints, point estimates are typically used to represent posterior distributions. As a result, the theoretically desired distribution is not achieved. The main assumption is that data from some data set ($\mathbf{x} \in R^D$) is distributed as a linear combination of dictionary $A \in \mathbb{R}^{D\times K}$ with added Gaussian noise $\mathbf{n}$:

$$\mathbf{x} = A\mathbf{s} + \mathbf{n} \tag{20}$$

with $n_i \overset{iid}{\sim} N(0, \sigma^2)$. The coefficients $\mathbf{s}$ are further assumed to be L_1 sparse with a Laplacian prior:

$$p_s(s_i) \propto \exp(-\lambda|s_i|). \tag{21}$$

This probabilistic model can be described with the following energy function

$$E(A, \mathbf{s}, \mathbf{x}) = \frac{1}{2}\frac{||\mathbf{x} - A\mathbf{s}||_2^2}{\sigma^2} + \lambda||\mathbf{s}||_1 \tag{22}$$

such that $p(\mathbf{x}, \mathbf{s}|A) = Z^{-1}\exp(-E(A, \mathbf{s}, \mathbf{x}))$. The maximum likelihood estimator (MLE) of the dictionary is typically found through gradient descent:

$$A \leftarrow A - \eta_A \nabla_A \langle E(A, \mathbf{s}, \mathbf{x})\rangle_{\mathbf{s}} \tag{23}$$

However, the expectation over $\mathbf{s}$ is intractable and is typically approximated by the maximum *a posteriori* (MAP) estimator of $\mathbf{s}$

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} E(A, \mathbf{s}, \mathbf{x}). \tag{24}$$

The MAP of $\mathbf{s}$ is also foudn through gradient desecent:

$$\mathbf{s} \leftarrow \mathbf{s} - \eta_s \nabla_s E(A, \mathbf{s}, \mathbf{x}) \tag{25}$$

Before each step of A (Eq. 3.55), multiple step of $\mathbf{s}$ (Eq. 3.57) is run to obtain an estimate for $\mathbf{s}^*$. This set of discrete updates in such as nested loop makes implementation in analog devices highly infeasible.

Langevin Dynamics

Langevin dynamics is described by the following stochastic differential equation:

$$\dot{\mathbf{u}} = -\nabla E(\mathbf{u}) + \sqrt{2T}\xi(t), \tag{26}$$

where $\xi(t)$ is independent Gaussian white noise with $\langle \xi(t)\xi(t')^T \rangle = \mathbf{I}\delta(t-t')$. The distribution of $p(\mathbf{u}(t))$, over time, will asymptotically converge to

$$p^{(\infty)}(\mathbf{u}) \propto e^{-E(\mathbf{u})/T}. \tag{27}$$

If S obeyed the following dynamics

$$\tau_S \dot{S} = -\nabla_S E(A, S, X) + \sqrt{2T\tau_S}\xi(t), \tag{28}$$

for fixed A, X, over time, S will sample from $S|X$. Consider if at the same time, A also had some continuous dynamics described by

$$\tau_A \dot{A} = -\nabla_A E(A, S, X). \tag{29}$$

A useful property of (Eq. 28) is that the equilibrium distribution is independent of the time constant τ_s. By taking $\tau_A \gg \tau_S$, the assumption that A is fixed with respect to the dynamics of S can be upheld. Conversely, because S evolves much faster than A, the dynamics of A is well approximated by

$$\tau_A \dot{A} = -\langle \nabla_A E(A, S, X) \rangle_{S|A,X}. \tag{30}$$

This is the exact mean gradient desired in Eq. 3.55. This way, probabilistic sparse coding can be implemented by a system obeying Eq. 28, 29. Simulation of Langevin Sparse Coding (LSC) and related results are presented in Chapter 4.

Chapter 1

Training Higher Order Hopfield Networks with MPF

1.1 Introduction

Minimum probability flow (MPF) is a method for parameter estimation developed by Sohl-Dickenstein, et al. [73]. In this chapter, we expand upon a few theoretical properties of MPF and explore alternative, related objective functions. Furthermore, we demonstrate an application in learning higher order Ising models. Finally, we show that when used to train Hopfield networks,

1.2 Minimum Proability Flow

Define $\mathbf{p}$ to be a data probability distribution over a set of states S where p_i is the probability a random observation would be in state $i \in S$. We wish to find, from a family of parametrized data distribution $\{\mathbf{q}(\theta)|\theta \in \Theta\}$ the "closest" model distribution $\mathbf{q}(\hat{\theta})$ to the data distribution, in the sense of maximizing the likelihood that sample $\mathbf{p}$ was drawn from $\mathbf{q}(\theta)$. A well known method is to minimize the KL divergence[39] between $\mathbf{p}$ and $\mathbf{q}$:

$$\hat{\theta} =_{\theta\in\Theta} D_{KL}(\mathbf{p}||\mathbf{q}(\theta)) \tag{1.1}$$

where

$$D_{KL}(\mathbf{p}||\mathbf{q}) = -\sum_{i\in S} p_i \log\left(\frac{q_i}{p_i}\right). \tag{1.2}$$

The objective function D_{KL} however is often intractable.

MPF gets around this by considering the flow from $\mathbf{p}$ to $\mathbf{q}$. Let $\Gamma(\theta)$ be the probability flow matrix where Γ_{ij} can be interpreted as the fraction of probability at state j that will "flow" to state i per unit time. We will take the definition of $\Gamma_{ii} = -\sum_{j\neq i}\Gamma_{ji}$. Then we will choose Γ so that

$$\mathbf{q}(\theta) = \lim_{T\to\infty} \mathbf{p}_\theta(t) \equiv \lim_{t\to\infty} e^{\Gamma(\theta)t}\mathbf{p}. \tag{1.3}$$

With detailed balance in consideration, we define

$$\Gamma_{ij} = g_{ij}\sqrt{\frac{p_i}{p_j}} = g_{ij}\exp(-\frac{1}{2}(E_i - E_j)). \tag{1.4}$$

E_i is the energy of state i so that $p_i \propto \exp(-E_i)$ and $g_{ij} = g_{ji} \in \{0, 1\}$ is the connection matrix between i and j. This way,

$$\Gamma_{ij}p_j = \Gamma_{ji}p_i, \tag{1.5}$$

as is required by detailed balance. The idea of MPF is that for flow in infinitesimal time ϵ, the minimization of D_{KL} between $\mathbf{p}_\theta(\epsilon)$ and $\mathbf{p}$ is a sufficient objective function. We can then define an objective function K as follows:

$$\begin{aligned}
D_{KL}(\mathbf{p}||\mathbf{p}_\theta(\epsilon)) &= D_{KL}(\mathbf{p}||\mathbf{p} + \epsilon\Gamma\mathbf{p}) + \mathcal{O}(\epsilon^2)\\
&= -\sum_{i\in S} p_i \log(\frac{p_i + \epsilon(\Gamma\mathbf{p})_i}{p_i})\\
&= -\sum_{i\in D} p_i \log(1 + \epsilon\frac{(\Gamma\mathbf{p})_i}{p_i})\\
&= -\epsilon\sum_{i\in D} p_i\frac{(\Gamma\mathbf{p})_i}{p_i} + \mathcal{O}(\epsilon^2)\\
&= \epsilon\sum_{i\notin D}(\Gamma\mathbf{p})_i = \epsilon\sum_{i\notin D}\sum_{j\in D}\Gamma_{ij}p_j \equiv \epsilon K.
\end{aligned}$$

The set of data states D is the support of $\mathbf{p}$. That is the subset of S for which $\mathbf{p}$ is nonzero on. In the last line we made use of the fact that $\sum_{i\in S}\Gamma_{ij} = 0$.

An immediate problem with this objective function is that for dense data where $D = S$, K would be identically zero for all possible parameter. However, we could then just expand $D_{KL}(\mathbf{p}||\mathbf{p}_\theta(\epsilon))$ to the second or the lowest, non-vanishing order. The fundamental premise behind MPF is that

$$\hat{\theta}_{MPF} \equiv \arg\min_{\theta\in\Theta} D_{KL}(\mathbf{p}||e^{\Gamma(\theta)\epsilon}\mathbf{p})\Big|_{\epsilon\to 0} \approx \arg\min_{\theta\in\Theta} D_{KL}(\mathbf{p}||\mathbf{q}(\theta)) \equiv \hat{\theta}_{ML}. \tag{1.6}$$

Although it is never explicitly (stated without very strong conditions, namely that it is ever possible that $\mathbf{p} = \mathbf{q}(\theta)$ exactly) in the original paper that $\hat{\theta}_{MPF}$ will always be a consistent estimator, a lot of work has been done using the MPF. However, it can be shown, for certain trivial models that K can fail to be a useful objective function (See App. 1.A)

1.3 Ising Spin Glass

As an example, we apply MPF in estimating the couplings in an Ising spin glass with energy

$$E(\mathbf{x}; J, \theta) = -\frac{1}{2}\mathbf{x}^T J \mathbf{x} - \theta^T \mathbf{x}, \tag{1.7}$$

where $\mathbf{x} \in \mathbb{R}^D$ The probability flow matrix is then

$$\Gamma(\mathbf{x}, \mathbf{x}') = g(\mathbf{x}, \mathbf{x}') \exp\left(-\frac{1}{2}(E(\mathbf{x}; J, \theta) - E(\mathbf{x}'; J, \theta))\right). \tag{1.8}$$

One simple choice of the connectivity matrix is to connect all states exactly one bit flip apart. That is,

$$g(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \text{if } \mathbf{x}' \in N(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \tag{1.9}$$

where we define

$$N(\mathbf{x}) = \{(x_1, \ldots, -x_n, \ldots, x^D) | n = 1, \ldots, D\}. \tag{1.10}$$

The difference in energy due to a bit flip can also be calculated as

$$E(\mathbf{x}) - E(\mathbf{x}') = 2x_n \left(\sum_{j \neq n} J_{nj} x_j + \theta_n\right). \tag{1.11}$$

With N observations $\mathcal{D} = \{\mathbf{x}_n | n = 1, \ldots, N\}$, the MPF objective function is

$$K(J, \theta) = \sum_{\mathbf{x} \in \mathcal{D}} \sum_{\mathbf{x}' \in N(\mathbf{x})} \exp\left(-\frac{1}{2}\left(E(\mathbf{x}; J, \theta) - E(\mathbf{x}'; J, \theta)\right)\right) \tag{1.12}$$

$$= \sum_{\mathbf{x} \in \mathcal{D}} \sum_{n} \exp\left(-x_n \left[\sum_{j \neq n} J_{nj} x_j + \theta_n\right]\right) \tag{1.13}$$

1.4 Hopfield Networks

A Hopfield network[32] is a model for memory and consists of a collection of binary polar "neurons",

$$\mathbf{x} \in \{-1, 1\}^D \tag{1.14}$$

and the spin glass energy function (Eq. 1.7) Each neuron asynchronously updates if and only if doing so decreases the overall energy. More formally, for a given i, define $\mathbf{x}' = (x_1, \ldots, -x_i, \ldots, x_D)$ and then,

$$\mathbf{x} \leftarrow \begin{cases} \mathbf{x}' & \text{if } E(\mathbf{x}') < E(\mathbf{x}) \\ \mathbf{x} & \text{if } E(\mathbf{x}') \geq E(\mathbf{x}) \end{cases} \tag{1.15}$$

This process is iterated over all the dimensions $i = 1, \ldots, N$. Then repeated until convergence. Note that convergence is always possible as energy necessarily decreases whenever a neuron flips. However, whether the local minima in which the neurons settle in is meaningful is the main topic of study.

As a model for memory, the aim is to store patterns $\{\xi^\alpha | \alpha = 1, \ldots, N\}$ as local minima. This way, given a corrupted version of one of the stored patterns, ξ^α, through the Hopfield update rule (Eq. 1.15), the corrupted pattern will converge to the stored memory (Fig. 1.1)

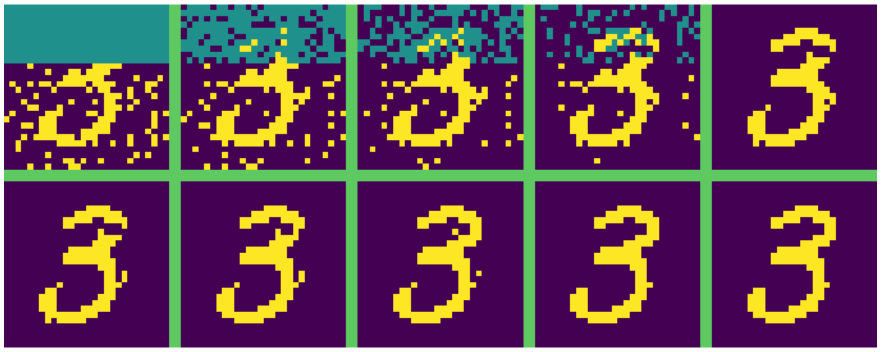


Fig. 1.1. Initialized to a pattern with both missing and noisy data, a Hopfield network eventually successfully converges to the desired, stored memory.

We say that for a memory ξ^α, $\mathbf{x}$ is in its basin of attraction when initialized at $\mathbf{x}$, network converges to ξ^α. In the scenario described above, The corrupted pattern was in the basin of the stored memory. However, it is possible for the pattern to be too corrupted to end outside said basin (Fig. 1.2). In such case, the network will instead converge to another, spurious minimum. Worse still, the network can converge to a spurious minimum even when initialized exactly at a stored pattern. In such case, the memory is said to not have been successfully stored.

When the number of stored patterns – N becomes too large, it becomes increasingly difficult to store patterns in what is known as catastrophic forgetting[37].

1.5 Learning rules

In the section above, it was assumed that the network has already stored memories $\{\xi^\alpha\}$. Here, we discuss how the weights J_{ij}, θ_i are set to store such memories.

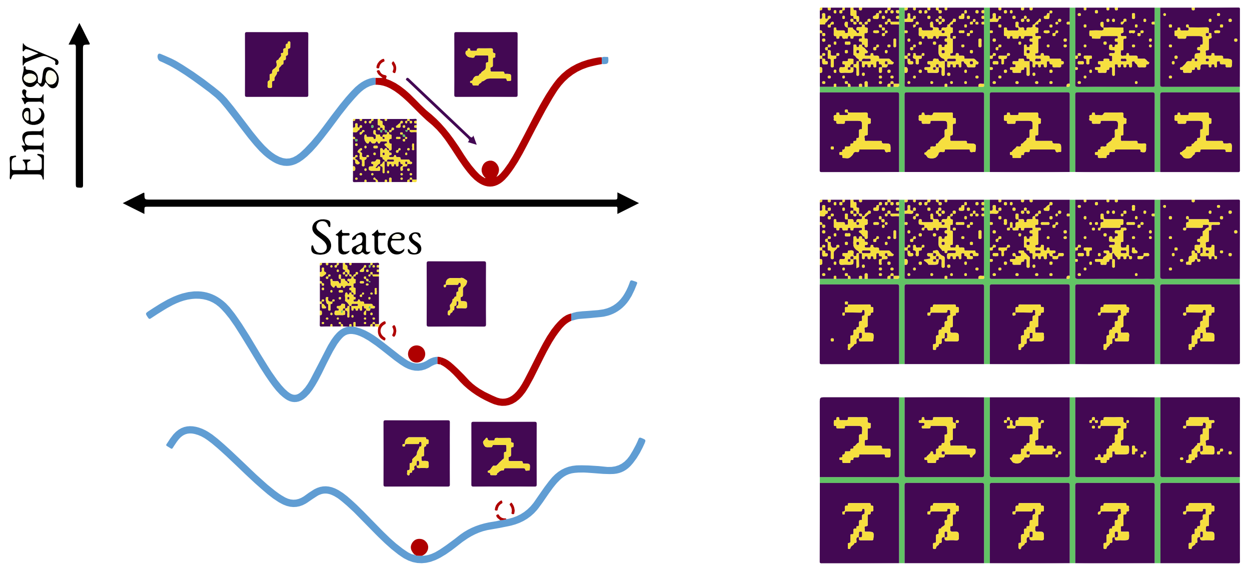


Fig. 1.2. Three scenarios illustrating the basin of attraction of a stored pattern "2". In the first case, the network was initialized within the basin of attraction and successfully converges. In the second case, the starting pattern was outside the basin of attraction and ends up stuck at an undesired state. In the last case, the network is over capacity and even when initialized exactly at the stored pattern, the network converges to something else.

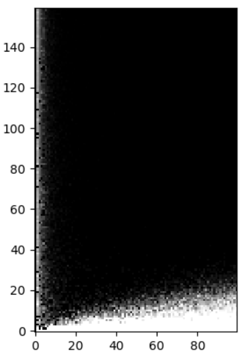
The original method for learning the couplings J as given by Hopfield is the outer product rule (OPR) where we set

$$J_{ij} = \sum_{\alpha=1}^{N} \xi_i^\alpha \xi_j^\alpha \tag{1.16}$$

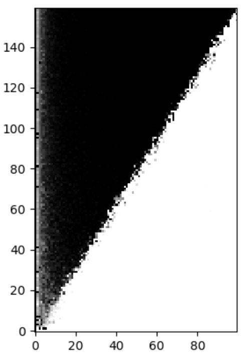
$$= \Xi\Xi^T. \tag{1.17}$$

OPR is very fast to calculate but yields a fairly low capacity. The capacity ends up being a small fraction the total number of neurons in the network – $C \approx 0.14D$[45].

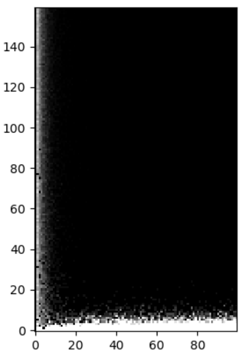
An alternative learning rule can be provided by MPF. We can treat the memories, $\{\xi^\alpha\}$ as data drawn from an spin glass model and use MPF, as described in Sec. 1.3, to infer the coupling responsible for generating these "samples". A Hopfield network trained this way has a capacity of approximately ten times higher[31] ($C \approx 1.4D$). See Fig. 1.3 for comparison.



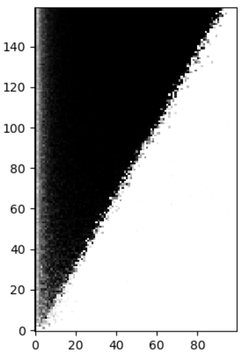
((**a**)) OPR on random data with $p = 0.5$



((**b**)) MPF on random data with $p = 0.5$



((**c**)) OPR on random data with $p = 0.3$



((**d**)) MPF on random data with $p = 0.3$

Fig. 1.3. Comparison of network trained using OPR and MPF on random data with quadratic energy. The x-axis is the number of neurons. The y-axis is the number of patterns. The color denote fraction of patterns that were successfully stored. We see that although both MPF and OPR training allows for linear capacity, MPF gives a much better rate.

1.6 JK Model

It has been reported that using energies with higher order interactions increases the capacity of Hopfield networks. Specifically, the capacity is proportional to

$$C \propto D^{d-1}, \tag{1.18}$$

where d is the order of the polynomial (e.g. $C \propto D$ for second order energy that we have studied). To leverage this effect, we introduce a higher, fourth-order interaction. The

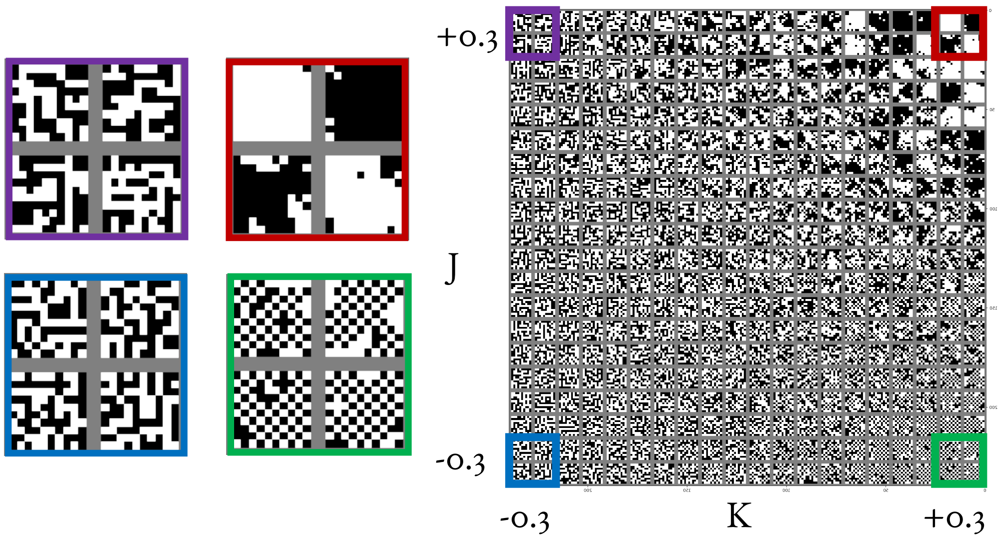


Fig. 1.4. Samples drawn from the JK model with various values of J and K. The samples sampled from extreme values of J, K are enlarged to the right.

simplest way of doing so is through extension of the nearest neighbor Ising model.

$$E(\mathbf{x}; J, K) = -\frac{1}{2} J \sum_{\langle i,j \rangle} x_i, x_j - K \sum_{ijkl \in \square} x_i x_j x_k x_l. \tag{1.19}$$

The notation $\langle i, j \rangle$ denotes nearest neighbors and $ijkl \in \square$ denotes sites in a 2×2 square configuration. A collection of samples were drawn at various values of J and K and are shown in Fig. 1.4. Note that multiple samples were drawn while only a single representative sample is shown in the figure.

Based on the data drawn at specific J, K, MPF was used to estimate the parameters $\hat{J}, \hat{K}$. The parameters, estimators and errors are show in Fig. 1.5. Note that the error is relatively evenly distributed.

We can then generalize the JK model to have the following energy:

$$E(\mathbf{x}; \mathbf{J}, \mathbf{K}) = -\frac{1}{2} \sum_{n=1}^{4} \left(\sum_{\langle ij \rangle_n} J_n x_i x_j \right) - \sum_{ijkl \in \square} K_{ijkl} \cdot x_i x_j x_k x_l. \tag{1.20}$$

We have used $\langle \cdot \rangle_n$ to denote the $n-$th nearest neighbors. Each grouping of fourth-order interaction also has a separate coupling K_{ijkl}. Fast computation of $x_i x_j x_k x_l$ can be found in App. 1.B. Four networks of $D = 100$ neurons with second order and fourth order energies (Eq. 1.20) were trained using MPF then OPR. Figure 1.6 demonstrates that MPF performs

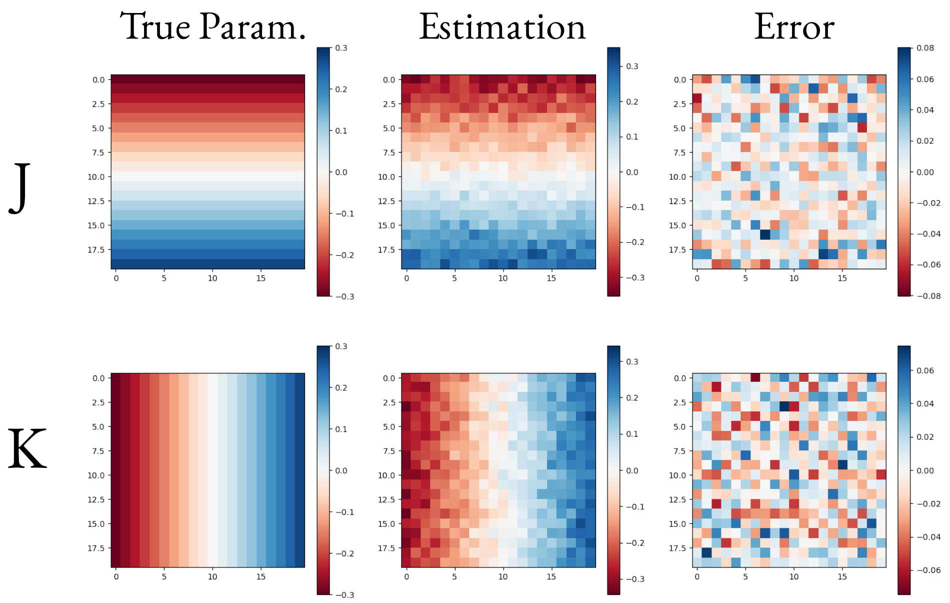


Fig. 1.5. Estimation of J and K by MPF along with the true parameter and error. Note that the error has a color scale much less than those of the parameters.

better than OPR using second order energies. Furthermore, MPF maintains its capacity when used to store sparser patterns.

1.7 Conclusion

In this chapter, we have exhibited variants of the MPF objective function and benchmarked them with simple models. Subsequently, MPF was used to train Hopfield networks. A higher capacity was verified for quadratic energy. Furthermore, using the JK-Ising model, a higher capacity is also achieved when compared against a fourth-order Hopfield network when trained with OPR for sparse patterns.

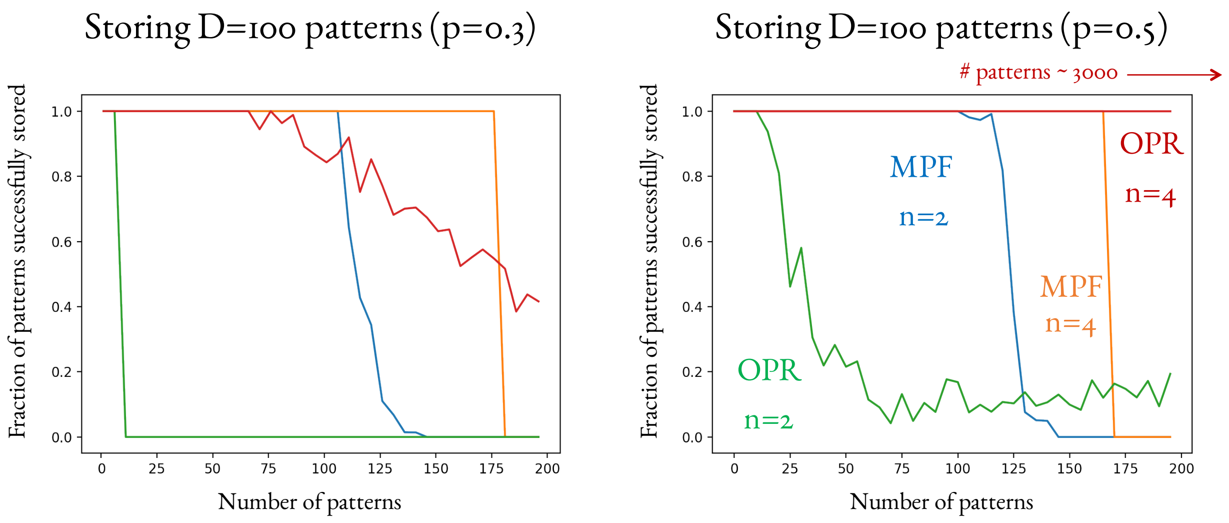


Fig. 1.6. Comparision of capacity using MPF vs OPR for second order and fourth order energies.

Appendix

1.A Modified MPF Objective Functions

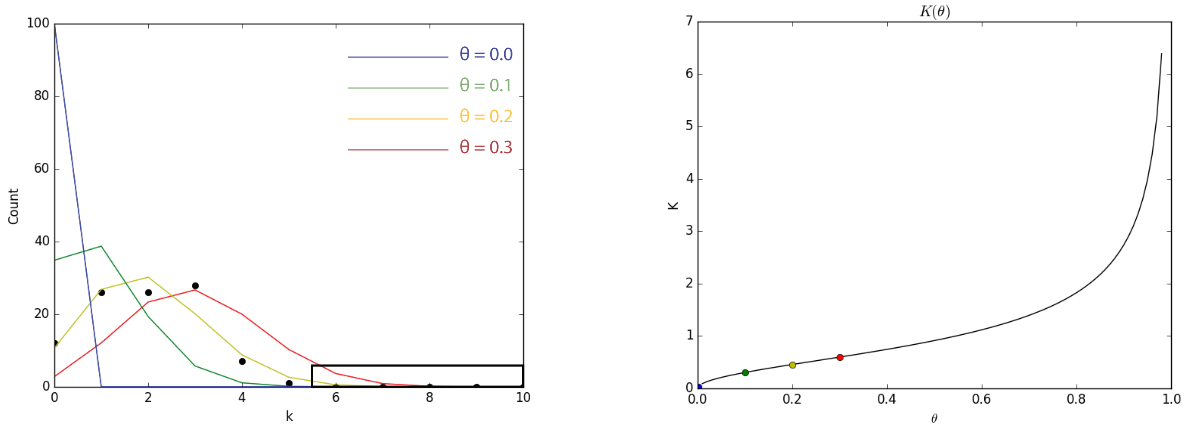
Binomial distribution

Consider a the very simple family of n-trial binomial distributions $\{\mathbf{q}(\theta)|\theta \in [0,1]\}$. Where

$$q_k(\theta) = \binom{n}{k}\theta^k(1-\theta)^{n-k}. \tag{1.21}$$

This is, for example, the probability of obtaining k heads when flipping a biased coin n times where the coin has a θ probability of landing on heads. Consider a sample of $N_s = 100$ i.i.d. binomial trials with $n = 10$ and $\theta_0 = 0.20$. A typical sample is shown in Fig. 1.A.1(a). Superimposed are four model distributions with different parameters $\theta = 0.0, 0.1, 0.2$ and 0.3 respectively. Note that the data states are $D = \{0 \leq k < 6\}$ and $\{6 \leq k \leq 10\}$ are the non-data states. This is fairly typical of binomial samples of the specific prescribed parameters.

We will now apply MPF on this sample to obtain an estimator for θ. The hope is that we will have $\hat{\theta}_{MPF} \approx \theta_0 = 0.2$.



((a)) Sample data ($\theta_0 = 0.2$) denoted by black dots. Model distribution of various parameters shown in colored lines. The black box shows the non-data states.

((b)) The MPF objective function.

Fig. 1.A.1. MPF estimation of binomial trials

For simplicity, use nearest neighbors connectivity. That is $g_{ij} = 1$ if and only if $|i-j| = 1$. Fig. 1.A.4(b) shows the objective function K calculated over all values of $\theta \in [0,1]$. $K(\theta)$ at $\theta = 0.0, 0.1, 0.2, 0.3$ are illustrated as well. We notice right away that the objective function is minimized not when $\theta = 0.2$ but rather when $\theta = 0$.

In fact, in this case, $K(\theta)$ has a very simple form. As there is only a single connection between the data and non-data sets (j = 5 to i = 6) thus,

$$K = \Gamma_{56} p_6 = \sqrt{\frac{p_\theta(6)}{p_\theta(5)}} p_6 \propto \sqrt{\frac{\theta^6(1-\theta)^4}{\theta^5(1-\theta)^5}} = \sqrt{\frac{\theta}{1-\theta}}$$

which is minimized when $\theta = 0$. Note that despite the somewhat pathological partition of data and non-data sets, in no way is the data dense. Indeed, unlike the case when $D = S$, K does not vanish and the inclusion of the second order expansion does not remedy the situation since K, as the lowest, non-zero, term will dominate over all higher order terms. In this case,

$$\hat{\theta}_{MPF} \equiv {}_{\theta \in \Theta} D_{KL}(\mathbf{p} || e^{\Gamma(\theta)\epsilon}\mathbf{p})\big|_{\epsilon \to 0}$$

will not be a consistent estimator regardless of the order that it is expanded to.

Changing the structure of the connection between states (i.e. g_{ij}) will not change this either. We can see this by regarding K as

$$K(\theta) = \sum_{i \neq D} (\Gamma p)_i = \sum_{i \neq D} p_i t.$$

This is the net change in probability of non-data states, towards the model distribution. The problem then reduces to the minimization model probability in non-data states only. Looking at the region in the black box in Fig. 1.A.1(a), it's clear that moving θ closer to 0 minimizes the probability in non-data states. In particular, when $\theta = 0$, it is exactly zero. Of course, the problem is that absolutely no consideration is given to data states.

Visualization of MPF with n = 3 toy model

For a more graphical and explicit look at what went wrong, let's look at one of the simplest non-trivial set of distributions, the $n = 3$ binomial distributions. The family of model distributions are of the form

$$\mathbf{q}(\theta) = ((1-\theta)^2,\ 2(1-\theta)\theta,\ \theta^2).$$

The data distribution, in the most general case will take the form of

$$\mathbf{p} = (a,\ 1-a-b,\ b).$$

Because $\mathbf{p}$ must have unit norm, it is completely parameterized by a and b. We also require that $0 \le a + b \le 1$. Therefore, in this toy model, all possible data distributions must reside in a 2D simplex shown as the colored region in Fig. 1.A.2 below:

Consider a model distribution at $a = 0.2, b = 0$ (i.e. $\mathbf{p} = (0.2, 0.8, 0)$), which is depicted as an orange point above. The flow from $\mathbf{p}$ to a set of model distributions $\{\mathbf{q}(\theta)|\theta = 0.1, 0.2, \cdots, 0.9$ are represented by colored lines from the orange dot to a place along the black curve (the model distribution space). The flow to $\mathbf{q}(\theta = 0.4)$ and $\mathbf{q}(\theta = 0.1)$ are of particular interest and are colored green and red respectively. Once again, the connection matrix g_{ij} was chosen to connect nearest neighbors. That is $g_{01} = g_{10} = g_{12} = g_{21} = 1$ and all other off diagonal elements vanish. The KL divergence from $\mathbf{p}$ (i.e. $D_{KL}(\mathbf{p}, \cdot)$) is visualized as a set of contours. This provides for us a sense of distance from the data distribution. By inspection, we see that of the subset of model distributions shown, $\mathbf{q}(0.4)$ is the "closest" to the data distribution. In fact, $\hat{\theta}_{ML} = 0.4$ is the parameter that minimizes the KL divergence out of *all* possible $\theta \in [0, 1]$. However, the point of MPF is to only consider an infinitesimal flow from $\mathbf{p}$.

Flowing $\mathbf{p}$ towards each of $\mathbf{q}(\theta)$ for $t = 0.01$, we get $\mathbf{p}_\theta(0.01)$. They are shown above in Fig. 1.A.3 as black points. In this case, we can clearly see that the "closest" point to $\mathbf{p}$ is actually $\mathbf{p}_{0.1}(0.01)$ despite $\mathbf{p}_{0.4}(\infty)$ being closer than $\mathbf{p}_{0.1}(\infty)$.

This provides an alternative, visual explanation for the exact same problem in the pervious section. The reason MPF fails for these cases is because the KL divergence from $\mathbf{p}$ gets heavily "distorted" when near the edge (i.e. when b = 0). Points along the edge are seen much "closer" than those further away.

While the "distortion" becomes less severe as we move away from the edge, it is still enough to pose problems. For example take $b = 0.005$ instead of zero. The same plots above are repeated for $\mathbf{p} = (0.2, 0.795, 0.005)$. In Fig. 1.A.4b, we see indeed, that the KL

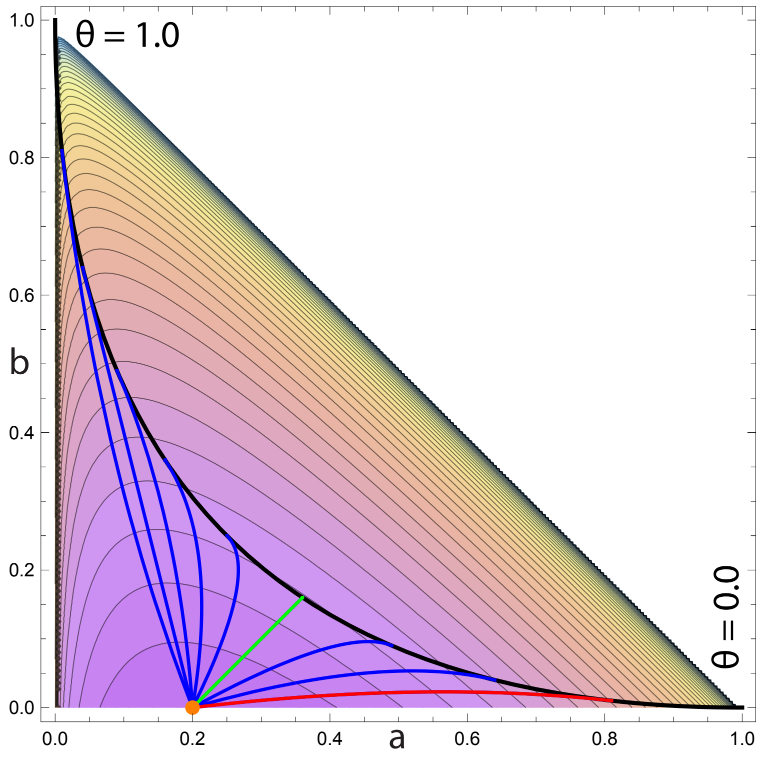


Fig. 1.A.2. The colored region denotes the simplex where all probability distributions must reside. The black curve is the family of parametrized model distributions $\{\mathbf{q}(\theta)\}$. The various blue, red and green lines represent the flow from the data distribution p (orange dot) to $\{\mathbf{q}(\theta)|\theta = 0.1, 0.2, \cdots, 0.9\}$. The colored contour shows the distance in the sense of KL divergence from the data distribution.

divergence contours are significantly less squashed than when $b = 0$ for just a small change in b. Nonetheless, the effects are still enough to provide a very incorrect estimate of $\hat{\theta}_{MPF}$ which is closer to 0.2 (red line) than 0.4 (green line).

Alternative objective functions

In the section, a few alternative but closely related objective functions to K are presented and at least in the case of binomial distributions, they fare much better than the original MPF objective function K.

K_Δ: Total change in probability

The first one, which we'll call K_Δ, is due to Chris Hillar. Let A be a $n \times n$ matrix with eigenvalues $\lambda_1 = 0 < \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n$ and corresponding eigenvectors $\{\mathbf{u}_i\}$. For any

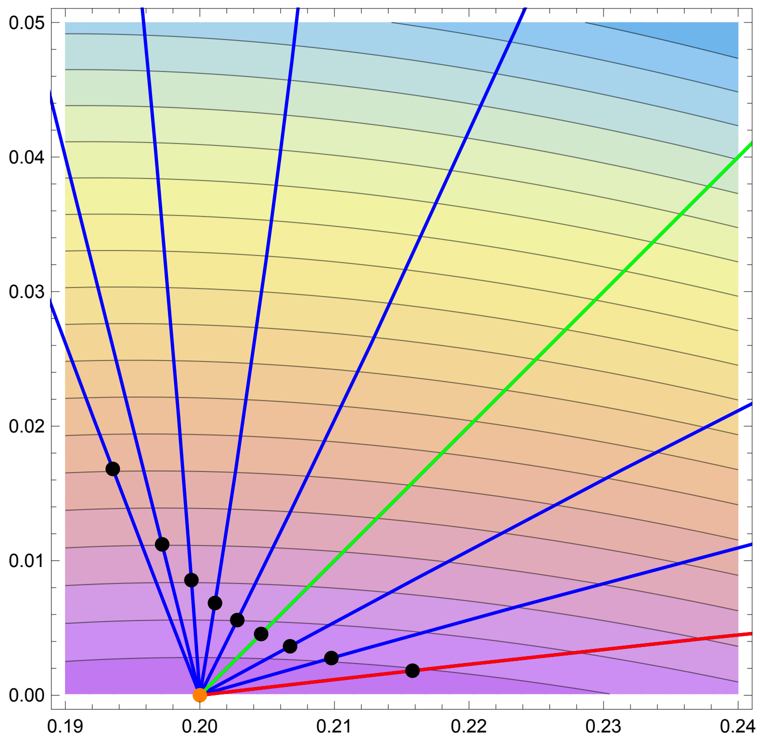


Fig. 1.A.3. Zoomed in on $\mathbf{p}$ by a factor of 20. The flow from $\mathbf{p}$ towards $\mathbf{q}(\theta)$ for a small time ($t = 0.01$) are shown as the black dots. The resolution of the contours of KL divergence is updated for this new scale.

given vector $\mathbf{x} \in \mathbb{R}^n$, let's define

$$\mathbf{x}_\perp = \mathbf{x} - (\mathbf{x} \cdot \mathbf{u}_1)/||\mathbf{u}_1||^2.$$

This is the components of $\mathbf{x}$ perpendicular to the first eigenvector. Working in the eigenbasis of A, we have

$$A = \mathrm{diag}(0, \lambda_2, \cdots, \lambda_n)$$

and

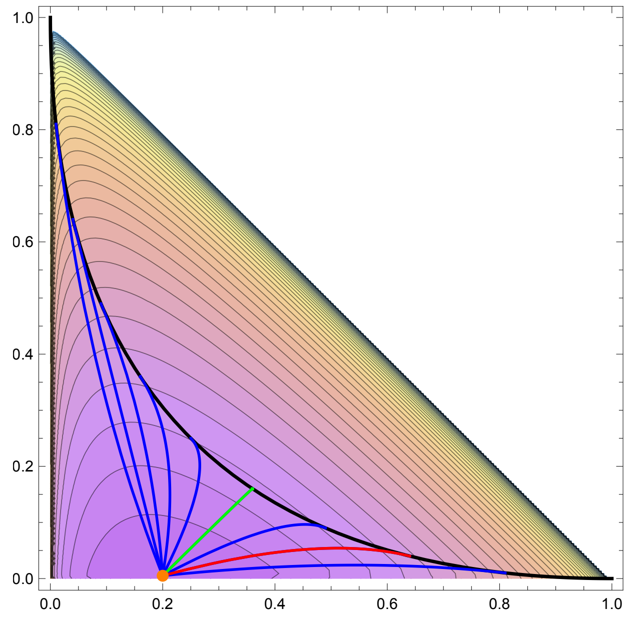
$$\mathbf{u}_i = \mathbf{e}_i.$$

Defining $x_i \equiv \mathbf{x} \cdot u_i$,

$$\mathbf{x}^T A \mathbf{x} = \lambda_2 x_2^2 + \cdots + \lambda_n x_n^2 \geq \lambda_2 (x_2^2 + \cdots + x_n^2) = \lambda_2 ||\mathbf{x}_\perp||^2.$$

Therefore,

$$||\mathbf{x}_\perp||^2 \leq \frac{\mathbf{x}^T A \mathbf{x}}{\lambda_2}.$$


((a)) Flow to $\mathbf{q}(\theta)$. $\theta = 0.4$ is still the closest to $\mathbf{p}$ out of the subset shown.

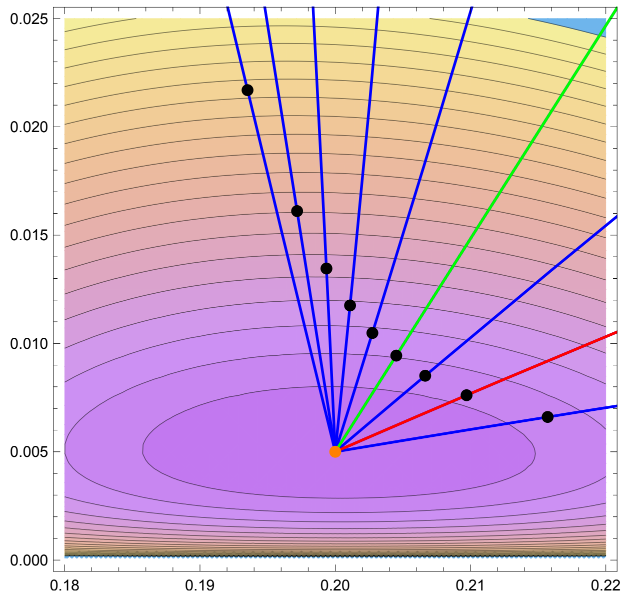

((b)) The MPF objective function.

Fig. 1.A.4. Flow visualization for $(a, b) = (0.2, 0.005)$

If we take $A = \Gamma^T\Gamma$, we can see that it does fulfill the above conditions. Specifically, $A\mathbf{q} = \Gamma^T\Gamma\mathbf{q} = \Gamma^T(0) = 0$ so $\mathbf{u}_1 = \mathbf{q}$ and $\mathbf{x}^T A\mathbf{x} = \mathbf{x}^T\Gamma^T\Gamma\mathbf{x} = ||\Gamma||^2 \geq 0$ so it has no negative eigenvalues. Γ also has a unique zero eigenvalue due to the Perron-Frobenius theorem (proof omitted) so all other eigenvalues are non-zero. Then, for $\mathbf{p}_\perp = \mathbf{p} - (\mathbf{p}\cdot\mathbf{q})/||\mathbf{q}||^2$, we have the following:

$$||\mathbf{p}_\perp||^2 \leq \frac{\mathbf{p}^T\Gamma^T\Gamma\mathbf{p}}{\lambda_2} = \left(\frac{||\Gamma\mathbf{p}||}{\sqrt{\lambda_2}}\right)^2$$

or

$$||\mathbf{p}_\perp|| \leq \frac{||\Gamma\mathbf{p}||}{\sqrt{\lambda_2}} \leq \frac{|\Gamma\mathbf{p}|}{\sqrt{\lambda_2}}.$$

Note that λ_2, being second singular value of Γ, very much depends on the choice of Γ. However, if it can be bounded from below by some value, as $|\Gamma\mathbf{p}| \to 0$ so will $||\mathbf{p}_\perp||$ and thus $\mathbf{q} \to \mathbf{p}$. This gives us the objective function

$$K_\Delta \equiv |\Gamma\mathbf{p}| = \sum_{i\in S} |(\Gamma\mathbf{p})_i|.$$

This is the sum of the magnitude of initial change in probabilities at $t = 0$ and does take into account both data and non-data states. Reassuringly, when $\mathbf{q} = \mathbf{p}$, $\Gamma\mathbf{q} = 0$ by definition and $K_\Delta = 0$. This does not of course imply that K_Δ is always consistent as $K = 0$ in this case as well.

K_f: Total flow of probability

Confusingly, in this section, "flow" refers to the flow between probability states (i.e. $\Gamma_{ij}p_j$ is the flow in probability from state j to state i) and not the flow in probability space as discussed previously. As K_Δ was a bound on $\mathbf{p}_\perp$, we can derive a bound on K_Δ:

$$\begin{aligned}
K_\Delta = \sum_{i\in S}\left|\sum_{j\in S}\Gamma_{ij}p_j\right| &\leq \sum_{i\in S}\sum_{j\in S}|\Gamma_{ij}p_j| \\
&= \sum_{i\in S}\left(\sum_{j\neq i}|\Gamma_{ij}p_j| + |\Gamma_{ii}p_i|\right) \\
&= \sum_{i\in S}\left(\sum_{j\neq i}\Gamma_{ij}p_j + \left|-\sum_{j\neq i}\Gamma_{ji}p_i\right|\right) \\
&= 2\sum_{i\neq j}\Gamma_{ij}p_j \equiv 2K_f.
\end{aligned}$$

In the first line, we used the triangle inequality. In the third line, the fact that $\Gamma_{ij} > 0$ for $i \neq j$ and $p_j > 0$ for all j was used. Lastly, we exchanged indices i, j to get the final result. K_f is the total flow between all states and unlike K_Δ will never be zero even when $p = q$. However, it can be shown that K_f reaches a minimum when $q = p$:

$$K_f = \sum_{i\neq j} g_{ij}\sqrt{\frac{q_i}{q_j}}p_j = \sum_{i<j} g_{ij}\left(\sqrt{\frac{q_i}{q_j}}p_j + \sqrt{\frac{q_j}{q_i}}p_i\right) = \sum_{i<j} g_{ij}\sqrt{p_ip_j}\left(\sqrt{\frac{q_ip_j}{q_jp_i}} + \sqrt{\frac{q_jp_i}{q_iq_j}}\right).$$

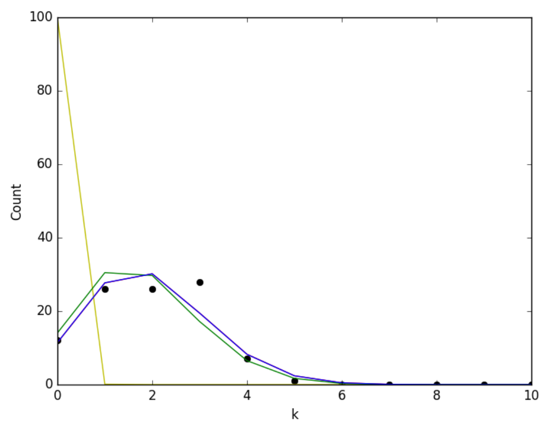
Since $x + 1/x$ is minimized when $x = 1$, each term in the sum will be minimized when $q_i/q_j = p_i/p_j$ for all pairs of i, j. As both probabilities must be normalized, this happens when $\mathbf{p} = \mathbf{q}$. As with K and K_Δ, this also does not imply consistency. K_f has the advantage over K_Δ of being linear so its gradient is easier to calculate. Also, in the limit where the data set D is sparse, K_f approaches K as $S \setminus D \to S$ in that limit.

Comparison of objective functions

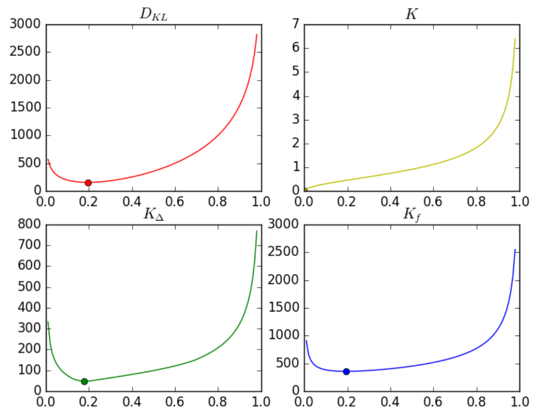
Four objective functions, D_{KL}, K, K_Δ, K_f were compared empirically for the same binomial model with $n = 10, \theta_0 = 0.2$. The results are shown below:

We see that K_Δ and K_f both do a much better job than K. In this example, the estimates given by K_f and K were almost identical. 1000 random i.i.d. samples data distributions were taken and a parameter estimate by each objective function was generated. The mean and variance of the parameter estimates are in the table below:

In this case, K_Δ and K_f seem to be consistent (D_{KL} is known to be consistent and saturates the Cramer-Rao bound for the variance). K_f however has a variance that's smaller than K_Δ by an order of magnitude and is very close to that of D_{KL}.



((a)) The model distribution chosen to fit the data by each objective function.



((b)) The various objective functions with its minimum labeled by a dot

Fig. 1.A.5. Comparison of objective functions. Red: D_{KL}, Yellow: K, Green: K_Δ, Blue: K_f

Objective functions	D_{KL}	K	K_Δ	K_f
Mean: $\langle\hat{\theta}\rangle$	0.19973	0.02701	0.20248	0.19982
Variance: $\langle(\hat{\theta}-\langle\hat{\theta}\rangle)^2\rangle$	1.58×10^{-4}	8.17×10^{-3}	1.32×10^{-3}	1.62×10^{-4}

Table 1.A.1: Comparison of parameter estimation by each of the objective functions

f-divergence

The f-divergence from a distribution q to a distribution p is defined[16] as

$$D_f(\mathbf{p}||\mathbf{q}) = \sum_{i\in S} \tilde{f}\left(\frac{p_i}{q_i}\right) q_i$$

for some convex function $\tilde{f}$ with $\tilde{f}(1) = 0$. For example, the KL-divergence is a type of a f-divergence with

$$\tilde{f}(t) = t\log t.$$

Through scaling and addition of the term $t-1$, we can ensure that for some value of c_1, c_2.

$$\tilde{f}(t) \to f(t) = c_1(f(t) + c_2(x-1))$$

will also satisfy $f'(1) = f''(1) = 1$ while staying convex. We will assume that f is normalized in such a way. Note that in the case of the KL-divergence, the f function is already normalized.

Second order expansion

We are interested in the case where

$$\mathbf{q}(t) = e^{\Gamma t}\mathbf{p}.$$

Specifically, the MPF objective function can be generalized to

$$D_f(\mathbf{p}||e^{\Gamma\epsilon}\mathbf{p}).$$

Expanding to second order and remembering that $f(1) = 0, f'(1) = f''(1) = 1$:

$$D_f(\mathbf{p}||e^{\Gamma\epsilon}\mathbf{p}) = -\epsilon\sum_i(\Gamma\mathbf{p})_i + \frac{1}{2}\epsilon^2\sum_i\left(\frac{(\Gamma\mathbf{p})_i^2}{p_i} - (\Gamma^2\mathbf{p})_i\right).$$

This is almost identical to the first and second order MPF objective functions with the KL-divergence. However, to make the expansion more rigorous, we should write the f-divergence as

$$D_f(\mathbf{p}||\mathbf{q}) = \sum_{i\in F} f\left(\frac{p_i}{q_i}\right)q_i$$

where

$$F = \left\{i \in S \middle| f\left(\frac{p_i}{q_i}\right)q_i \neq 0\right\}.$$

Then,

$$D_f(\mathbf{p}||e^{\Gamma\epsilon}\mathbf{p}) = \epsilon\sum_{i\notin F}(\Gamma\mathbf{p})_i + \frac{1}{2}\epsilon^2\sum_{i\in F}\left(\frac{(\Gamma\mathbf{p})_i^2}{p_i} - (\Gamma^2\mathbf{p})_i\right).$$

Two types of f-MPF objective functions

To see what F can be note that since f is convex, it must grow faster than any linear function. Thus, as $q_i \to 0$,

$$f\left(\frac{p_i}{q_i}\right)q_i \to \infty.$$

It follows that

$$f\left(\frac{p_i}{q_i}\right)q_i = 0,$$

we must have

$$f\left(\frac{p_i}{q_i}\right) = 0.$$

We already have $f(1) = 0$. As f is convex, it has, at most, one other root. If the non-trivial root exists, let's call it $\alpha \neq 1$ (i.e. $f(\alpha) = 0$). In our case, however, we are dealing with $q_i = p_i + \epsilon(\Gamma\mathbf{p})_i$ so $p_i/q_i = 1 + \delta$ is infinitesimally close to 1. This means $p_i/q_i \neq 1, \alpha$. The only way that $f(p_i/q_i)$ can be exactly zero is if $p_i = 0$ and $\alpha = 0$.

For our consideration, there are two types of f-divergences. One for which $\alpha = 0$ and $F = D = \{i \in S | p_i \neq 0\}$ and the other where $\alpha \neq 0$ or that f only has the trivial root at $t = 1$. Since $f_{KL}(t) = t \log t$ has $\alpha = 0$, it of the first type.

The two corresponding MPF objective function are

$$D_f^{(1)}(\mathbf{p}||e^{\Gamma\epsilon}\mathbf{p}) = \epsilon \sum_{i \notin D} (\Gamma \mathbf{p})_i + \frac{1}{2}\epsilon^2 \sum_{i \in D} \left(\frac{(\Gamma \mathbf{p})_i^2}{p_i} - (\Gamma^2 \mathbf{p})_i \right)$$

and

$$D_f^{(2)}(\mathbf{p}||e^{\Gamma\epsilon}\mathbf{p}) = \frac{1}{2}\epsilon^2 \sum_{i \in S} \left(\frac{(\Gamma \mathbf{p})_i^2}{p_i} - (\Gamma^2 \mathbf{p})_i \right).$$

It turns out that using f-divergences does not generalize MPF by much. This is because f-divergences all locally look like the Fisher information matrix.

Comments and future work

In practice when the space of possible distributions is very large, we will always be dealing with sparse data. In that limit, as mentioned previously, there is very difference between K_f and K. However, the point of this write-up is to illustrate that the premise upon which K is derived is incorrect in a few trivial cases and potentially others as well.

As shown in Sec. 1.A, the local landscape of D_{KL} very close to $\mathbf{p}$ does not accurate describe the situation further away. It would be interesting to see if there's a way of fixing this through information geometry. Also, empirically, MPF does work and quite well. There needs to be a better explanation for why it does. If it's simply because K and K_f are identical for sparse data, then why does K_f work? Or is it because in more realistic settings, and higher dimensions, D_{KL} is much more likely to be "well-behaved"? If that's the case, is there a way of showing this?

1.B Convolution Trick

Here, we describe a fast method for calculating local higher order interactions of the form

$$E = -\sum_{\alpha} K_\alpha \prod_{(i,j) \in N_\alpha} X_{i,j}.$$

To be concrete, consider a set of spins

$$\mathbf{x} = \begin{pmatrix} +1 & -1 & +1 \\ +1 & +1 & -1 \\ -1 & +1 & -1 \end{pmatrix}$$

and a local interaction involving three spins in the shape of Γ. We can iterate over all possible groups to get the higher order correlation:

$$\begin{pmatrix} \boxed{+1} & \boxed{\text{-1}} & +1 \\ \boxed{+1} & +1 & -1 \\ -1 & -1 & -1 \end{pmatrix} \to \begin{pmatrix} -1 & * \\ ** & * \end{pmatrix}$$

$$\vdots$$

$$\begin{pmatrix} +1 & -1 & +1 \\ +1 & \boxed{+1} & \boxed{\text{-1}} \\ -1 & \boxed{\text{-1}} & -1 \end{pmatrix} \to \begin{pmatrix} -1 & -1 \\ -1 & +1 \end{pmatrix} \equiv C$$

The energy can then be written as $E = \sum K_{ij}C_{ij}$. However, rather than looping over all possible interactions explicity in calculating $\mathbf{C}$, we can define a matrix $\mathbf{M}$ that describes the interaction:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

We can use a cross-correlation to calculate the sum of terms in interacting sites:

$$\mathbf{S} = \mathbf{X} \star \mathbf{M}.$$

We are, of course, interested in the product. However, it's easy to see that with only spins of value ± 1, the product is $(-1)^{N_-}$ where N_- is the number of spin -1. The total number of spins is the number of 1's in $\mathbf{M}$.

$$\sum M_{ij} = N = N_- + N_+.$$

The sum of the values is

$$S_{ij} = N_+ - N_-.$$

With those two relations, we get

$$N_- = \frac{1}{2}(N - S_{ij}).$$

This way, $C_{ij} = (-1)^{\frac{1}{2}(N - S_{ij})}$. The energy is easily calculated as

$$E = -\sum K_{ij}C_{ij} \equiv -\sum K_\alpha C_\alpha$$

where $\alpha = (i, j)$. We are, however, interested in the difference in energy due to a bit flip. A bit flip will result in flipping the values of all relavent C_α. For example, consider flipping the central bit of $\mathbf{X}$:

$$\begin{pmatrix} +1 & -1 & +1 \\ +1 & +1 & -1 \\ -1 & +1 & -1 \end{pmatrix} \to \begin{pmatrix} +1 & -1 & +1 \\ +1 & -1 & -1 \\ -1 & +1 & -1 \end{pmatrix}.$$

The patterns of interactions that this affects are the following three:

$$\begin{pmatrix} +1 & -1 & +1 \\ +1 & \boxed{\text{-1}} & \boxed{\text{-1}} \\ -1 & \boxed{\text{-1}} & -1 \end{pmatrix} \rightarrow \begin{pmatrix} -1 & +1 \\ +1 & \boxed{\text{-1}} \end{pmatrix}$$

$$\begin{pmatrix} +1 & -1 & +1 \\ \boxed{+1} & \boxed{\text{-1}} & -1 \\ \boxed{\text{-1}} & -1 & -1 \end{pmatrix} \rightarrow \begin{pmatrix} -1 & +1 \\ \boxed{+1} & -1 \end{pmatrix}$$

$$\begin{pmatrix} +1 & \boxed{\text{-1}} & \boxed{+1} \\ +1 & \boxed{\text{-1}} & -1 \\ -1 & -1 & -1 \end{pmatrix} \rightarrow \begin{pmatrix} -1 & \boxed{+1} \\ +1 & -1 \end{pmatrix}.$$

The corresponding three elements of $\mathbf{C}$,

$$\begin{pmatrix} -1 & \boxed{+1} \\ \boxed{+1} & \boxed{\text{-1}} \end{pmatrix},$$

is in the shape of the interaction matrix $\mathbf{M}$ but inverted. Since these elements are taken to be negative of their original value, the total change in energy is

$$\Delta\mathbf{E} = -2 \cdot (\mathbf{C} * \mathbf{M})$$

Chapter 2

Design of optical neural networks with component imprecisions

For the benefit of designing scalable, fault resistant optical neural networks (ONNs), we investigate the effects architectural designs have on the ONNs' robustness to imprecise components. We train two ONNs – one with a more tunable design (GridNet) and one with better fault tolerance (FFTNet) – to classify handwritten digits. When simulated without any imperfections, GridNet yields a better accuracy ($\sim 98\%$) than FFTNet ($\sim 95\%$). However, under a small amount of error in their photonic components, the more fault tolerant FFTNet overtakes GridNet. We further provide thorough quantitative and qualitative analyses of ONNs' sensitivity to varying levels and types of imprecisions. Our results offer guidelines for the principled design of fault-tolerant ONNs as well as a foundation for further research.

2.1 Introduction

Motivated by the increasing capability of artificial neural networks in solving a large class of problems, optical neural networks (ONNs) have been suggested as a low power, low latency alternative to digitally implemented neural networks. A diverse set of designs have been proposed, including Hopfield networks with LED arrays [20], optoelectronic implementation of reservoir computing[58, 1], spiking recurrent networks with microring resonators[77, 76], convolutional networks through diffractive optics[10], and fully connected, feedforward networks using Mach-Zehnder interferometers (MZIs) [69].

We will focus on the last class of neural networks, which consist of alternating layers of modules performing linear operations and element-wise nonlinearities[26]. The N-dimensional complex-valued inputs to this network are represented as coherent optical signals on N single-mode waveguides. Recent research into configurable linear optical networks[**carolan2015unive** 61, 12, 5, 27] enables the efficient implementation of linear operations with photonic devices. These linear multipliers, layered with optical nonlinearities form the basis of the physical design of ONNs. In Sec. 2.2, we provide a detailed description of two specific architectures – GridNet and FFTNet – both built from MZIs.

While linear operations are made much more efficient with ONNs in both power and speed, a major challenge to the utility of ONNs lies in their susceptibility to fabrication errors and other types of imprecisions in their photonic components. Therefore, realistic considerations of ONNs require that these imprecisions be taken into account. Previous analyses of the effects of fabrication errors on photonic networks were in the context of post-fabrication optimization of unitary networks [56, 65, 8]. Our study differs in three main areas.

First, In the previous work, unitary optical networks were optimized to simulate randomly sampled unitary matrices. We, instead, train optical neural networks to classify structured data. ONNs, in addition to unitary optical multipliers, include nonlinearities, which add to its complexity.

Second, rather than optimization towards a specific matrix, the linear operations learned for the classification task is not, *a priori*, known. As such, our primary figure of merit is the classification accuracy instead of the fidelity between the target unitary matrix and the one learned.

Lastly, the aforementioned studies mainly focused on the optimization of the networks after fabrication. The imprecisions introduced generally reduced the expressivity of the network – how well the network can represent arbitrary transformations. Evaluation of this reduction in tunability and mitigating strategies were provided. However, such post-fabrication optimization requires the characterization of every MZI, the number of which scales with the dimension (N) of the network as N^2. Protocols for self configuration of imprecise photonic networks have been demonstrated [46, 85]. While measurement of MZIs were not necessary in such protocols, each MZI needed to be configured progressively and sequentially. Thus, the same N^2 scaling problem remained. Furthermore, if multiple ONN devices are fabricated, each device, with unique imperfections, has to be optimized separately. The total computational power required, therefore, scales with the number of devices produced.

In contrast, we consider the effects of imprecisions introduced after software training of ONNs (Code 1, Ref. [19]), details of which we present in Sec. 2.3. This pre-fabrication training is more scalable, both in network size and fabrication volume. An ideal ONN (i.e., one with no imprecisions) is trained in software only once and the parameters are transferred to multiple fabricated instances of the network with imprecise components. No subsequent characterization or tuning of devices are necessary. In addition to the benefit of better scalability, fabrication of static MZIs can be made more precise and cost effective compared to re-configurable ones.

We evaluate the degradation of ONNs from their ideal performances with increasing imprecision. To understand how such effects can be minimized, we investigate the role that the architectural designs have on ONNs' sensitivity to imprecisions. The results are presented in Sec. 2.4. Specifically, we study the performance of two ONNs in handwritten digit classification. GridNet and FFTNet are compared in their robustness to imprecisions.

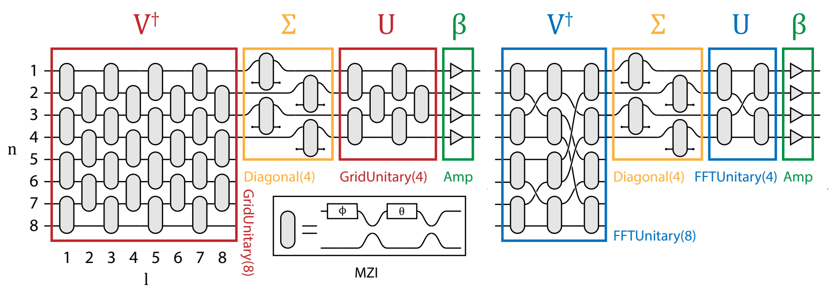
We found that GridNet achieved a higher accuracy ($\sim$ 98%) when simulated with ideal components compared to FFTNet ($\sim$ 95%). However, FFTNet is much more robust to imprecisions. After the introduction of realistic levels of error, the performance of GridNet quickly degrades to below that of FFTNet. We also show, in detail, the effect that specific levels of noise has on both networks.

In Sec. 2.4, we demonstrate that this is due to more than the shallow depth of FFTNet and that FFT-like architectures is more robust to error when compared to Grid-like architectures of the same depth.

In Sec. 2.4, we investigate the effects localized imprecisions have on the network by constraining the imprecisions to specific groups of MZIs. We demonstrate that the network's sensitivity to imprecisions is dependent on algorithmic choices as well as its physical architecture.

With a growing interest in optical neural networks, a thorough analysis of the relationship between ONNs' architecture and its robustness to imprecisions and errors is necessary. From the results that follow, in this article, we hope to provide a reference and foundation for the informed design of scalable, error resistant ONNs.

2.2 Physical design of optical neural networks



((a)) GridNet linear layer **((b))** FFTNet linear layer

Fig. 2.2.1. a) A schematic of a universal 8×4 optical linear multiplier with two unitary multipliers (red) consisting of MZIs in a grid-like layout and a diagonal layer (yellow). The MZIs of GridUnitary multipliers are indexed according to their layer depth (l) and dimension (d). Symbols at the top represent the mathematical operations performed by the various modules. Inset: A MZI with two 50:50 beamsplitters and two tunable phaseshifters b) An FFT-like, non-universal multiplier with FFTUnitary multipliers (blue).

The ONN consists of multiple layers of programmable optical linear multipliers with intervening optical nonlinearities (Fig. 2.3.1). The linear multipliers are implemented with two unitary multipliers and a diagonal layer in the manner of a singular-value decomposition

(SVD). These are, in turn, comprised of arrays of configurable MZIs, which each consist of two phaseshifters and two beamsplitters (Fig. 2.2.1(a)).

Complex-valued $N-$dimensional input vectors are encoded as coherent signals on N waveguides. Unitary mixing between the channels is effected by MZIs and forms the basis of computation for ONNs. A single MZI consists of two beamsplitters and two phaseshifters (PS) (Fig. 2.2.1(a) inset). While the fixed 50:50 beamsplitters are not configurable, the two phaseshifters, parameterized by θ and ϕ, are to be learned during training. Each MZI is characterized by the following transfer matrix (see App. 2.A for details):

$$U_{MZ}(\theta,\phi) = U_{BS}U_{PS}(\theta)U_{BS}U_{PS}(\phi) = ie^{i\theta/2}\begin{pmatrix} e^{i\phi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \\ e^{i\phi}\cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \end{pmatrix}. \tag{2.1}$$

Early work has shown that universal optical unitary multipliers can be built with a triangular mesh of MZIs[61]. These multipliers enabled the implementation of arbitrary unitary operations and were incorporated into the ONN design by Shen et al. [69]. Its asymmetry prompted the development of a symmetric grid-like network with more balanced loss[12]. By relaxing the requirement on universality, a more compact design, inspired by the Cooley-Tukey FFT algorithm [14], has been proposed[5]. It can be shown that FFT transforms, and therefore convolutions, can be achieved with specific phase configurations (see appendix 2.H). We allow the phase configurations to be learned for implementation of a greater class of transformations.

In this section, we focus on the last two designs, referring to them as GridUnitary (Fig. 2.2.1(a)) and FFTUnitary (Fig. 2.2.1(b)), respectively. GridUnitary can implement unitary matrices directly by setting the phaseshifters using an algorithm by Clements et al. [12]. Despite being non-universal and lacking a decomposition algorithm, FFTUnitary can be used to reduce the depth of the unitary multipliers from N to $\log_2(N)$. Reducing the number of MZIs leads to lower overall noise and loss in the network. However, due to the FFT-like design, waveguide crossings are necessary. To overcome this challenge, low-loss crossings[43] or 3D layered waveguides[25, 57] could be utilized.

MZIs can also be used to attenuate each channel separately without mixing. This way, a diagonal multiplier can be built. Because signals can only be attenuated by MZIs, subsequent global optical amplification[13] is needed to emulate arbitrary diagonal matrices. Through SVD, a universal linear multiplier can be created from two unitary multipliers and a diagonal multiplier (Fig. 2.2.1(a)). Formally, a linear transformation represented by matrix M can be decomposed as

$$M = \beta \cdot U\Sigma V^{\dagger}. \tag{2.2}$$

Here both U and $V^{\dagger}$ are unitary transfer matrices of GridUnitary multipliers while Σ represents a diagonal layer with eigenvalues no greater than one. β is a compensating scaling factor.

Along with linear multipliers, nonlinear layers are required for artificial neural networks. In fact, the presence of nonlinearties sets the study of ONNs apart from earlier research in linear photonic networks [47]. One possible implementation is by saturable absorbers such as monolayer graphene [4]. This is has the advantage of being easily approximated with a Softplus function (see Sec. 2.3 for details on implementation). However, it has been demonstrated that Softplus underperforms, in many regards, when compared to rectified linear units (ReLU)[48]. Indeed, a complex extension of ReLU, ModReLU, has been proposed [2]. While it is physically unrealistic to implement ModReLU, the nonoptimality of Softplus functions still motivates the exploration of other optical nonlinearities, such as optical bistability in microring resonators[86], and two-photon absorption [35, 3] as alternatives.

2.3 Neural network architecture and software implementation

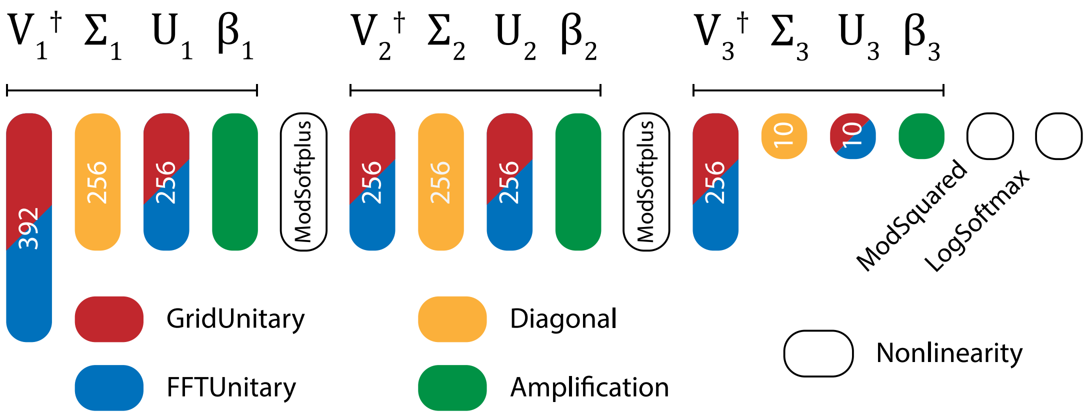


Fig. 2.3.1. Network design used for the MNIST classification task. GridNet used universal unitary multipliers while FFTNet used FFT-Unitary multipliers. See Fig. 2.2.1 for details of physical implementation of the three linear layers.

We considered a standard deep learning task of MNIST handwritten digit classification [41]. Fully connected feedforward networks with two hidden layers of 256 complex-valued neurons each were implemented with GridNet and FFTNet architectures (Fig. 2.3.1) and simulated in PyTorch [59]. The $28^2 = 784$ dimensional real-valued input was converted into $392 = 784/2$ dimensional complex-valued vectors by taking the top and bottom half of the image as the real and imaginary part. This was done to ensure the data is distributed evenly throughout the complex plane rather than just along the real number line.

Each network consists of linear multipliers followed by nonlinearities. The linear layers of GridNet and FFTNet were described in the previous section and illustrated in Fig. 2.2.1. The response curve of the saturable absorption is approximated by the Softplus function[18] (App. 2.C), a commonly used nonlinearity available in most deep learning libraries such as

PyTorch. The nonlinearity is applied to the modulus of the complex numbers. A modulus squared nonlinearity modeling an intensity measurement is then applied. The final SoftMax layer allows the (now real) output to be interpreted as a probability distribution. A cross-entropy[15] loss function is used to evaluate the output distribution against the ground truth.

An efficient implementation of GridNet requires representing matrix-vector multiplications as element-wise vector multiplications [36]. Nevertheless, training the phaseshifters directly was still time consuming. Instead, a complex-valued neural network [80] was first trained. An SVD (Eq. (2.2)) was then performed on each complex matrix. Finally, phase-shifters were set to produce the unitary $(U, V^\dagger)$ and diagonal (Σ) multipliers through a decomposition scheme by Clements et al. [12].

However, note that SVD is ambiguous up to permutations (Π) of the singular values and the columns of U and V.

$$U\Sigma V^\dagger = (U\Pi^{-1})(\Pi\Sigma\Pi^{-1})(\Pi V^\dagger). \tag{2.3}$$

Conventionally, the ambiguity is resolved through ordering the singular values from largest to smallest. In Sec. 2.4 we show that randomizing the singular values increases the error tolerance of GridNet. FFTNet is trained directly and its singular values are naturally unordered. For a fair comparison, we randomly permute the singular values of GridNet.

After 10 training epochs with standard stochastic gradient descent[62], classification accuracies of 97.8% (GridNet) and 94.8% (FFTNet) were achieved. Better accuracies can be achieved through convolutional layers [71], Dropout regularization[75], better training methods, etc. However, we omitted these in order to focus purely on the effects of architecture.

The networks were trained assuming ideal components represented with double-precision floating point values. Under realistic conditions, due to imprecision in fabrication, calibration, etc., the realizable accuracy could be much lower. During inference, we modeled these imprecisions by adding independent zero-mean Gaussian noise of standard deviation σ_{PS} and σ_{BS} to the phases (θ, ϕ) of the phaseshifters and the transmittance T of the beamsplitters, respectively. Reasonable values for such imprecisions can be taken to be approximately $\sigma_{PS} \approx 0.01\text{rad}$ and $\sigma_{BS} \approx 1\% = 0.01$ [22, 23]. Note that the dynamical variation due to laser phase noise can be modeled by σ_{PS} as well. However, we show in App. 2.B that typical values would be well below 0.01 rad.

2.4 Results

Degradation of network accuracy

To investigate the degradation of the networks due to imprecisions, we started by simulating 100 instances of imprecise networks with $\sigma_{BS} = 1\%$ and $\sigma_{PS} = 0.01\text{rad}$. Identical inputs of a digit "4" (Fig. 2.4.1(a) inset) are fed through each network. The mean and spread of the output of the ensemble is plotted and compared against the output from the ideal network (Fig. 2.4.1).

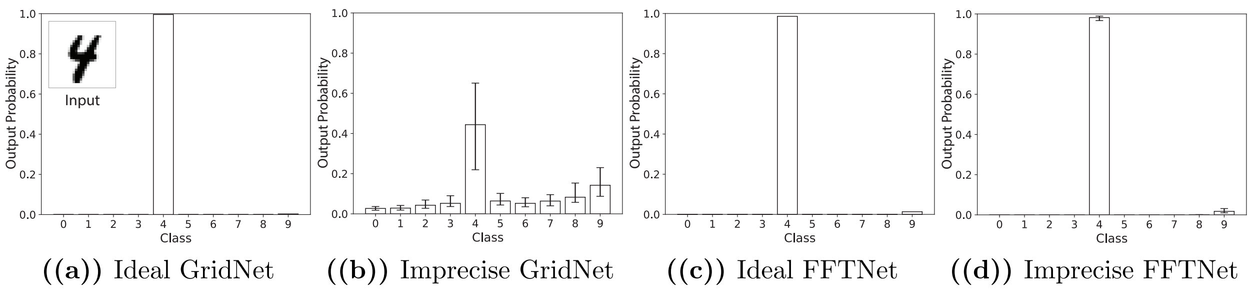


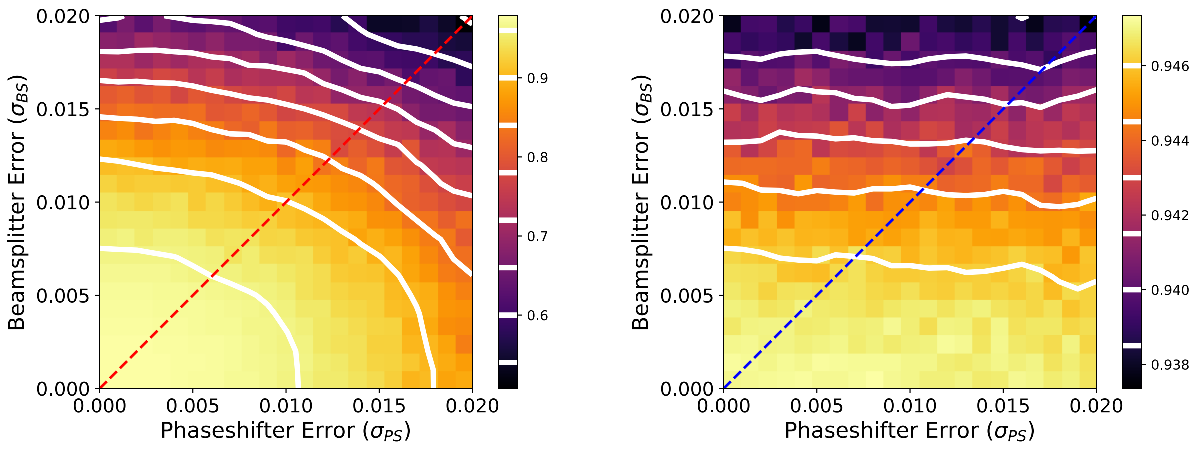
Fig. 2.4.1. Visualizing the degradation of ONN outputs, FFTNet is seen to be much more robust than GridNet. Identical input is fed through GridNet (a, b) and FFTNet (c, d), simulated with ideal components (a, c) and imprecise components (b, d) with $\sigma_{BS} = 0.01$ and $\sigma_{PS} = 0.01\text{rad}$. Imprecise networks are simulated 100 times and their mean output is represented by bar plots. Error bars represent the 20th to 80th percentile range.

The degradation of classification output is significant for GridNet. Without imprecisions in the photonic components, the digit is correctly classified with near 100% confidence (Fig. 2.4.1(a)). When imprecisions are simulated, we see a large decrease in classification confidence (Fig. 2.4.1(b)). In particular, the image is often misclassified when the prediction probability for class "9" is greater than that for class "4". Repeating these experiments on FFTNet demonstrated that they were much more resistant to imprecisions (Fig. 2.4.1(c), 2.4.1(d)). In Appendix 2.D, we show confusion matrices of both networks with increasing error to further support this conclusion.

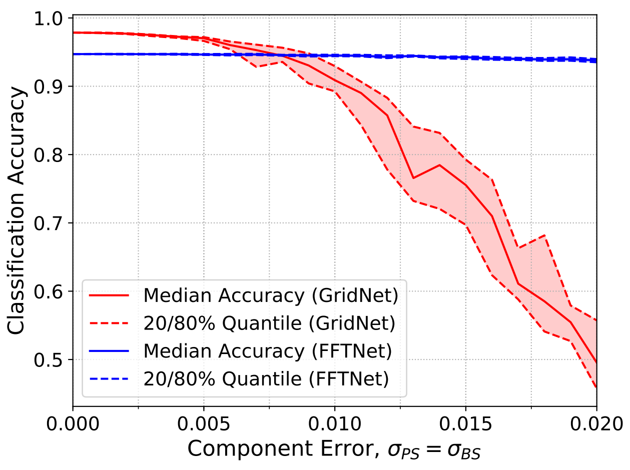
Evaluating the two networks on overall classification accuracy confirms the superior robustness to imprecisions of FFTNet. GridNet and FFTNet were tested at levels of imprecisions with of imprecisions with σ_{PS}/rad and σ_{BS} ranging from 0 to 0.02 with a step size of 0.001. At each level of imprecision, 20, instances of each network were created and tested. The mean accuracies are plotted in Fig. 2.4.2(a), 2.4.2(b). A direct comparison between the two networks along the diagonal (i.e., $\sigma_{PS} = \sigma_{BS}$ cut line, taking $1\% = 0.01$ rad) is shown in Fig. 2.4.2(c).

Starting at roughly 98% with ideal components, the accuracy of GridNet rapidly drops with increasing σ_{PS} and σ_{BS}. By comparison, very little change in accuracy is seen for FFTNet despite starting with a lower ideal accuracy. Also of note are the qualitatively different levels of sensitivity of the different components to imprecision. In particular, FFTNet is much more resistant to phaseshifter error compared to beamsplitter error.

The experiments described in this section confirm the significant effect component imprecisions have on the overall performance of ONNs, as well as the importance of architecture in determining the network's robustness of the network to these imprecisions. Despite having a better classification accuracy in the absence of imprecisions, GridNet is surpassed by FFTNet when a small amount of error ($\sigma_{PS} = 0.01, \sigma_{BS} = 1\%\text{rad}$) is present. In Appendix



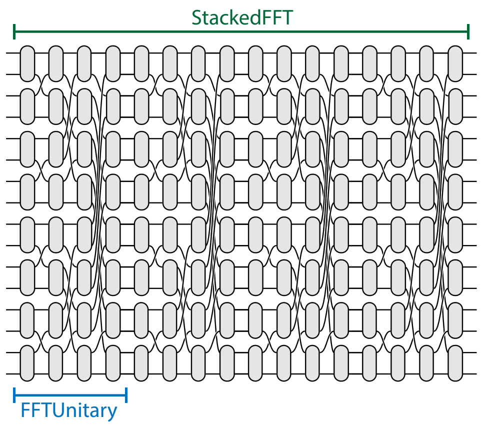
((**a**)) GridNet ((**b**)) FFTNet



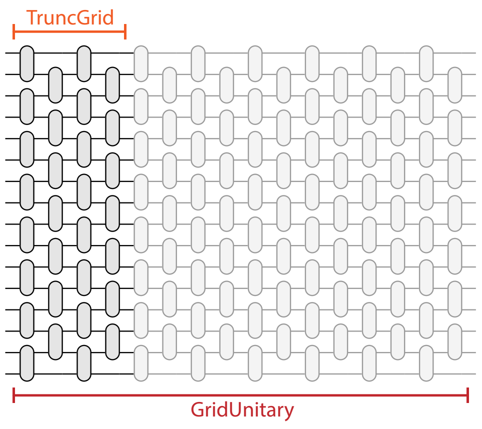
((**c**)) Direct Comparison of GridNet and FFT Accuracy

Fig. 2.4.2. The decrease in classification accuracy is visualized for GridNet and FFTNet. (a,b) The two networks were tested with simulated noise of various levels for 20 runs. The mean accuracy is plotted as a function of σ_{PS} and σ_{BS}. Note the difference in color map ranges between the two plots. (c) The accuracies of GridNet and FFTNet are compared along the $\sigma_{PS} = \sigma_{BS}$ cutline.

2.E, we demonstrate that FFTNet is also more robust to quantization error that GridNet.



((a)) StackedFFT and FFTUnitary



((b)) TruncGrid and GridUnitary

Fig. 2.4.3. The architecture of a) StackedFFT and b) TruncGrid shown with FFTUnitary and GridUnitary from which they were derived. For clarity, the dimension, here, is $N = 2^4 = 16$ so FFTUnitary was stacked four times and GridUnitary was truncated at the fourth layer. In the experiments described in this section, the dimension was taken to be $N = 2^8 = 256$.

Stacked FFTUnitary and truncated GridUnitary

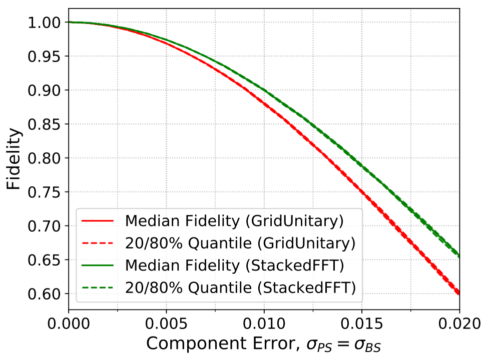
One obvious reason why FFTNet would be more robust than GridNet is its much lower number of MZI layers. Their respective, constituent unitary multipliers, FFTUnitary and GridUnitary contains $\log_2(N)$ and N layers respectively. For $N = 2^8 = 256$, GridUnitary is 32 times deeper than FFTUnitary which contains only 8 layers.

To demonstrate that FFTUnitary is more robust due architectural reasons beyond its shallow depth, in this section, we introduce two unitary multipliers – StackedFFT (Fig. 2.4.3(a)) and TruncGrid (Fig. 2.4.3(b)). StackedFFT consists of FFTUnitary multipliers stacked end-to-end 32 times and TruncGrid is the GridUnitary truncated after 8 layers of MZIs. This way, FFTUnitary and TruncGrid have the same depth as do GridUnitary and StackedFFT.

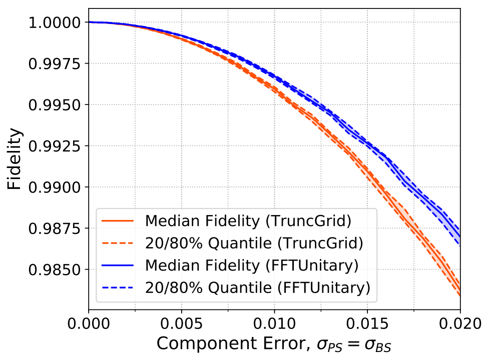
Unitary multipliers by themselves are not ONNs and cannot be trained for classification tasks. Instead, after introducing imprecisions to the each multiplier, we evaluated the fidelity $F(U_0, U)$ between the original, error-free transfer matrix U_0 and the imprecise transfer matrix U. The fidelity, a measure of "closeness" between two unitary matrices, is defined as[81]

$$F(U_0, U) = \left| \frac{\mathrm{Tr}(U^\dagger U_0)}{N} \right|^2 . \tag{2.4}$$

Ranging from 0 to 1, $F(U_0, U) = 1$ only when $U = U_0$. Using this metric of fidelity, we show that StackedFFT is more robust to error than GridUnitary (Fig. 2.4.4(a)) and TruncGrid more than FFTUnitary (Fig. 2.4.4(b)). Both comparisons are between multipliers with the



((**a**)) StackedFFT and GridUnitary



((**b**)) TruncGrid and FFTUnitary

Fig. 2.4.4. With the same layer depth, multipliers with FFT-like architectures are shown to be more robust. The fidelity between the error-free and imprecise transfer matrices is plotted as a function of increasing error. Two sets of comparisons between unitary multipliers of the same depth are made. a) Both StackedFFT and GridUnitary have $N = 256$ layers of MZIs. b) TruncGrid and FFTNet have $\log N = 8$ layers.

same number of MZI layers. Yet, the FFT-like architectures are still more robust to their grid-like counterparts.

One possible explanation could be the better mixing facilitated by FFTUnitary. GridUnitary and thus TruncGrid, at each MZI layer, only mixes neighboring waveguides. After P layers, each waveguide is connected to, at most, to its $2P$ nearest neighbors. In comparison, after P layers, FFTUnitary connects $N = 2^P$.

Here, we have compared the robustness of different unitary multipliers in isolation. We stress that the overall robustness of neural networks is a much more complex and involved problem. A rough understanding can be formulated as follows. A trained neural network defines a decision boundary throughout the input space. Introduction of errors perturbs the decision boundary which can lead to misclassification. To reduce this effect, we can make the decision boundary of ONNs more robust to errors. However, it is also important to consider the robustness of misclassification due to perturbations of decision boundaries. Indeed, it has been shown that robustness of neural networks are dependent on the geometry of the boundary [21].

A complete analysis of the robustness of neural networks to various forms of perturbations is outside the scope of this paper. Nonetheless, it is important to understand the dependence of ONNs on both architectural and algorithmic design.

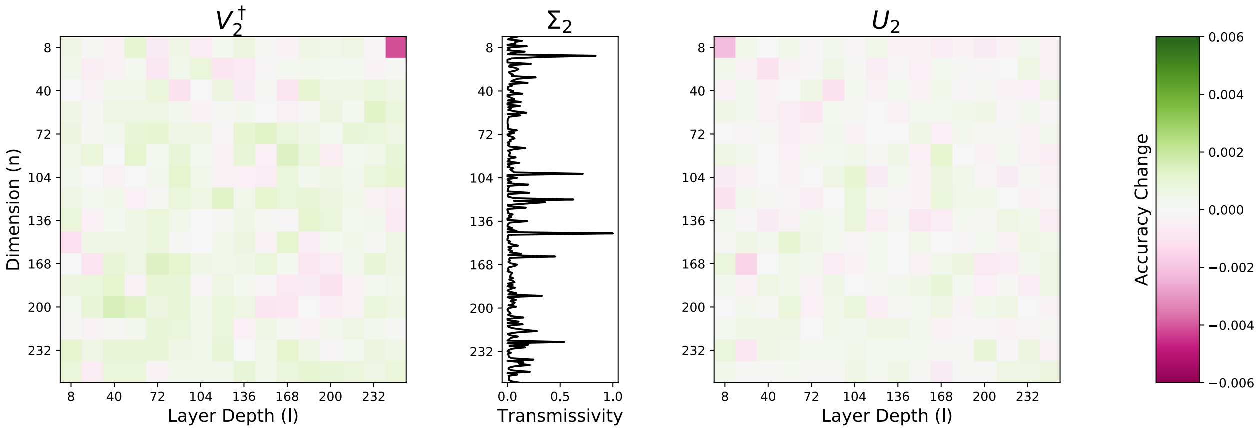


Fig. 2.4.5. Change in accuracy due to localized imprecision in layer 2 of GridNet with randomized singular values. A large amount of imprecision ($\sigma_{PS} = 0.1$rad) is introduced to 8×8 blocks of MZIs in an otherwise error-free GridNet. The resulting change in accuracy of the network is plotted as a function of the position of the MZI block in GridUnitary multipliers $V_2^\dagger$ and U_2 (coordinates defined as in Fig. 2.2.1(a)). The transmissivity of each waveguide through the diagonal layer Σ_2 is also plotted (center panel).

Localized imprecisions

To better understand the degradation of network accuracy, we mapped out the sensitivity of GridNet to specific groups of MZIs. A relatively large amount of imprecision ($\sigma_{PS} = 0.1$rad) was introduced to 8×8 blocks of MZIs in layer 2 (Fig. 2.3.1) of an otherwise error-free GridNet. The resulting change in classification accuracy is plotted as a function of the position of the MZI block (Fig. 2.4.5). We see no strong correlation between the change in accuracy and the spatial location of the introduced error. In fact, error in many locations led to small increases in accuracy, suggesting that much of the effect is due to chance.

This result seems to contradict previous studies on the spatial tolerance of MZIs in a GridUnitary multiplier [56, 65, 8]. It was discovered that the central MZIs of the multiplier had a much lower tolerance than those near the edges. When learning randomly sampled unitary matrices, the central MZIs needed to have phase shift values very close to 0 (π, following the convention used in this paper). This would only be achievable with MZIs with extremely high extinction ratios and thus low fabrication error.

Empirically, this distribution of phases was observed in GridUnitary multipliers of trained ONNs (See app. 2.F). However, the idea of tolerance of a MZI to beamsplitter fabrication imprecision, while related, is not the same as the network sensitivity to localized imprecisions. To elaborate, tolerance is implicitly defined, in references[56, 65, 8], as roughly the allowable beamsplitter imperfection (deviation from 50:50) that still permits post-fabrication optimization of phaseshifter towards arbitrary unitary matrices. In our pre-fabrication optimization approach, we take sensitivity to be the deviation from ideal classification accuracy when imprecision is introduced to the MZI with no further reconfiguration. See App. 2.G for this difference further illustrated by experiments with another architecture.

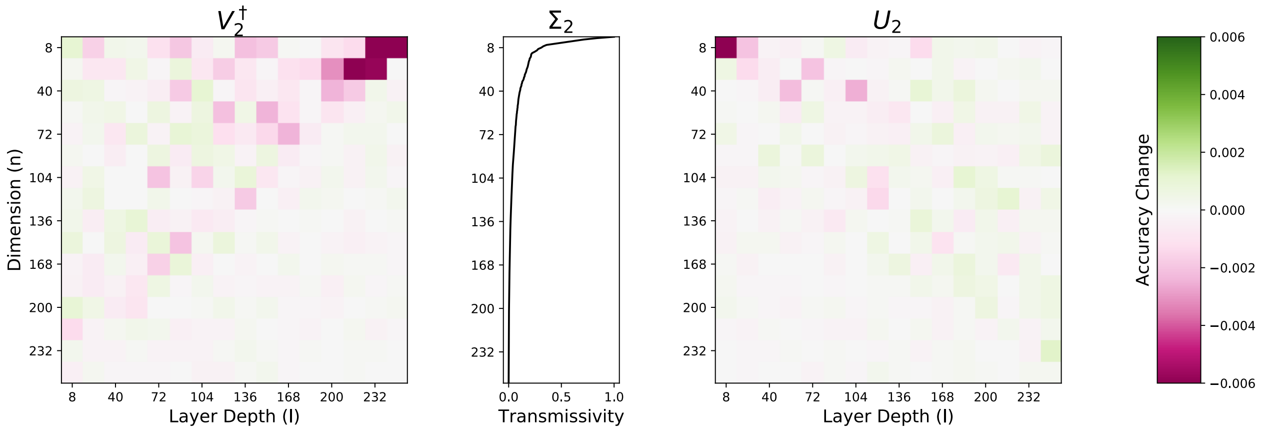


Fig. 2.4.6. Effects of localized imprecision in layer 2 of GridNet with ordered singular values. Similar to Fig. 2.4.5, except GridNet has its singular values ordered. Therefore, the transmissivity is also ordered (center panel).

Recall that the singular values Σ of the GridNet's linear layers could be permuted together with columns and rows of U and $V^\dagger$ respectively without changing the final transfer matrix (Eq. (2.3)). The singular values were randomized to provide a fair comparison with FFTNet. We then performed the same experiment on GridNet where the singular values of each layer were not randomized but ordered from largest to smallest. Therefore, the transmissivity $T = |\sin(\theta/2)|^2$ of the diagonal multiplier Σ is also ordered (Fig. 2.4.6). In this case, there is a significant, visible pattern because most of the signal travels through the top few waveguides of Σ_2 due to the ordering of transmissivities. Only MZIs connected to those waveguides have a strong effect on the network. In fact, the network is especially sensitive to imprecisions in MZIs closest to this bottleneck (Fig. 2.4.6, top-right of $V_2^\dagger$ and top-left of U_2). It is important to note that this bottleneck only exist due to the locality of connections in GridNet where only neighboring waveguides are connected by MZIs. In FFTNet, due to crossing waveguides, no such locality exist.

In addition to, and likely due to the spatial non-uniformity in error sensitivity, GridNet with ordered singular values is more susceptible to uniform imprecisions (Fig. 2.4.7). The same GridNet architecture, could be made more resistant by shuffling its singular values. This difference between two identical architectures implementing identical linear and non-linear transformations demonstrates that the resistance to error in ONNs is effected by more than architecture.

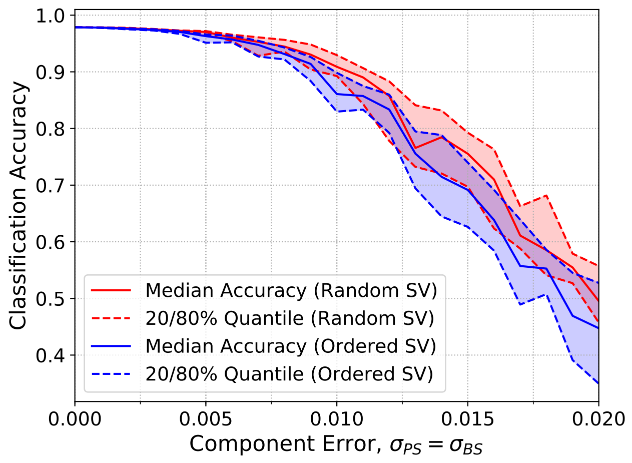


Fig. 2.4.7. The degradation of accuracies with increasing $\sigma_{PS} = \sigma_{BS}$ compared between two GridNets one with ordered and another with randomized (but fixed) singular values.

2.5 Conclusion

Having argued that pre-fabrication, software optimization of ONNs is much more scalable than post-fabrication, on-chip optimization, we compared two types of networks–GridNet and FFTNet in their robustness to error. These two networks were selected to showcase the trade-off between expressivity and robustness. We demonstrated in Sec. 2.4 that the output of GridNet is much more sensitive to errors than FFTNet. We have illustrated the robustness of FFTNet by a providing a thorough evaluation of both networks operating with imprecisions ranging between $0 \leq \sigma_{BS}, \sigma_{PS} \leq 0.02$. With ideal accuracies of 97.8% and 94.8% for GridNet and FFTNet respectively, GridNet accuracy dropped rapidly to below 50% while FFTNet maintained near constant performance. Under conservative assumptions of errors associated with the beamsplitter ($\sigma_{BS} > 1\%$) and phaseshifter ($\sigma_{PS} > 0.01$ rad), a more robust network (FFTNet) can be favorable over one with greater expressivity (GridNet).

We then demonstrate, in Sec. 2.4, through modified unitary multipliers, TruncGrid and StackedFFT, that controlling for MZI layer depth, FFT-like designs are inherently more robust than grid-like ones.

To gain a better understanding of GridNet's sensitivity to imprecision, in Sec. 2.4, we probed the response of the network to localized imprecisions by introducing error to small groups of MZIs at various locations. The sensitivity to imprecisions was found to be less affected by the MZIs' physical position within the grid and more so by the flow of the optical signal. We then demonstrated that beyond architectural designs, small procedural changes to the configuration of an ONN, such as shuffling the singular values, can change affect the its robustness.

Our results, presented in this paper, provide clear guidelines for the architectural design of efficient, fault-resistant ONNs. In looking forward, it would be important to investigate algorithmic and training strategies as well. A central problem in deep learning is to design neural networks complex enough to model the data while being regularized to prevent overfitting of noise in the training set[26]. To this end, a wide variety of regularization techniques such as Dropout[75], Dropconnect[82], data augmentation, etc. have been developed. This problem parallels the trade-off between an ONN's expressivity and its robustness to imprecisions presented here. Indeed, an important conclusion in Sec. 2.4 is that in addition to architecture, even minor changes in the configuration of ONNs also have a great effect on the network's robustness to faulty components.

The robustness of neural networks to perturbations [21] is a well studied and open problem that is outside of the scope of this article on architectural design. Nevertheless, a complete analysis of ONNs with imprecise components requires an understanding of robustness due to architectural design as well as due to software training, possibly under a unifying framework. A natural direction for further exploration is to consider analogies to regularization in the context of imprecise photonic components and to focus on the development of algorithms and training strategies for error-resistant optical neural networks.

2.6 Funding

MRD was partially supported by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract W911NF-13-1-0390.

2.7 Code repository and results

The code repository, results and scripts used to generate figures in this paper are freely available at `https://github.com/mike-fang/imprecise_optical_neural_network`

Appendix

2.A MZI transfer matrix

Because MZIs are comprised of beampslitters and phaseshifters, we state their respective transfer matrix first.

$$U_{BS}(r) = \begin{pmatrix} r & it \\ it & r \end{pmatrix} \tag{2.5}$$

where $t \equiv \sqrt{1-r^2}$ and

$$U_{PS}(\theta) = \begin{pmatrix} e^{i\theta} & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.6}$$

With the construction of PS-BS-PS-BS (Fig. 2.2.1(a), inset), the MZI transfer matrix is the following matrix product:

$$U_{MZI}(\theta,\phi;r,r') = U_{BS}(r)U_{PS}(\theta)U_{BS}(r')U_{PS}(\phi) \tag{2.7}$$

$$= \begin{pmatrix} r & it \\ it & r \end{pmatrix} \begin{pmatrix} e^{i\theta} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} r' & it' \\ it' & r' \end{pmatrix} \begin{pmatrix} e^{i\phi} & 0 \\ 0 & 1 \end{pmatrix} \tag{2.8}$$

$$= \begin{pmatrix} e^{i\phi}\left(e^{i\theta}rr' - tt'\right) & i\left(t\rho + e^{i\theta}rt'\right) \\ ie^{i\phi}\left(e^{i\theta}tr' + rt'\right) & r\rho - e^{i\theta}tt' \end{pmatrix} \tag{2.9}$$

Assuming that the beamsplitter ratios are 50:50, we can take $r = t = 1/\sqrt{2}$ so that

$$U_{BS} \equiv U_{BS}(\pi/2) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \tag{2.10}$$

and therefore,

$$U_{MZI}(\theta,\phi) = ie^{i\theta/2}\begin{pmatrix} e^{i\phi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \\ e^{i\phi}\cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \end{pmatrix} \tag{2.11}$$

In our convention, the transmission and reflection coefficient is

$$T = |\cos\theta/2|^2 \text{ and } R = |\sin\theta/2|^2 \tag{2.12}$$

respectively. In particular, the MZI is in the bar state ($T = 0$) when $\theta = \pi$ and in the cross state ($T = 1$) when $\theta = 0$.

However, in other conventions, the beamsplitter is often taken to be the Hardamard gate.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.13}$$

We note however, that

$$U_{BS} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} H \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = U_{PS}(-\pi/2) H U_{PS}(-\pi/2) \tag{2.14}$$

up to a global phase. We then can express the MZI transfer matrix as

$$U_{MZ}(\theta, \phi) = U_{PS}(-\pi/2) H U_{PS}(\theta - \pi) H U_{PS}(\phi - \pi/2). \tag{2.15}$$

Note in this convention the internal phase shift is now $\theta + \pi$ and thus the bar and cross states are now at $\theta = 0$ and $\theta = \pi$ respectively.

2.B Laser phase noise

The variance in phase for typical lasers can be modeled as[38]

$$\sigma_\phi(\tau)^2 = 2\pi \cdot \delta f \cdot \tau. \tag{2.16}$$

Here, τ is the time of integration and δf the linewidth of the laser. For an order or magnitude calculation, we ignore the refractive index and take $\tau = L/c$ where L is the distance between two subsequent phaseshifters on an MZI. Again, as an order of magnitude estimate, we take $L = 100\mu\text{m} = 10^{-4}\text{m}$ and thus $\tau \approx 3 \times 10^{-13}$. We wish to solve for the linewidth required for $\sigma_\phi = 0.01\text{rad}$:

$$\begin{aligned} \sigma_\phi^2 = 10^{-4} &= 2\pi \cdot \delta f \cdot \tau & (2.17)\\ &\approx 6 \cdot 3 \times 10^{-13} s \delta f & (2.18)\\ \delta f &\approx 5 \times 10^{7}\text{Hz} & (2.19)\\ &= 50\text{MHz}. & (2.20) \end{aligned}$$

A linewidth of 50 MHz is easily achieved by modern lasers. For example, Bragg reflector lasers have been shown to achieve a linewidth of 300 kHz [40]. Thus, the contribution to phase noise from the laser is roughly two orders of magnitude smaller than that from MZIs.

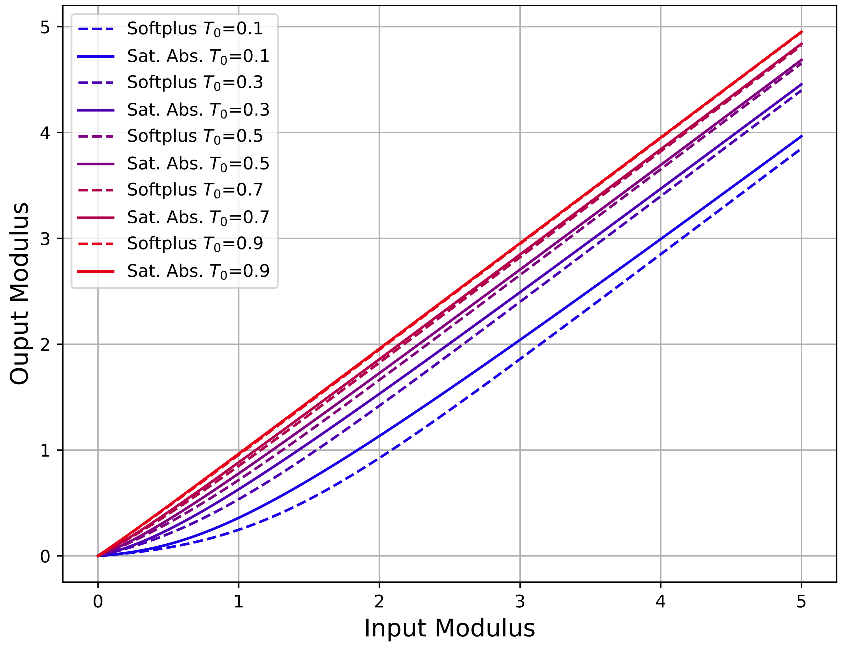


Fig. 2.C.1. The saturable absorption response curve compared to the corresponding Softplus approximation with various values of T.

2.C Approximating saturable absorption

Saturable absorption can be modeled by the relation[67]

$$u_0 = \frac{1}{2}\frac{\log(T/T_0)}{1-T} \tag{2.21}$$

where $T = u/u_0$ and $u = \sigma\tau_s I$ and $u_0 = \sigma\tau_s I_0$. I_0, I are the incidental and transmitted intensities, respectively. The above equation can be solved to be

$$u = \frac{1}{2}W(2T_0 u_0 e^{2u_0}) \equiv f(u_0) \tag{2.22}$$

Where W is the product log function or Lambert W function. However, since W is not readily available in most deep learning libraries and difficult to implement, we wish to approximate the above by the shifted and biased Softplus non-linearity of the form

$$\sigma(u) = \beta^{-1}\log\left(\frac{1+e^{\beta(u-u_0)}}{1+e^{-\beta u_0}}\right). \tag{2.23}$$

The bias of $-\beta^{-1}\log(1+e^{-\beta u_0})$ was chosen to ensure that $\sigma(0)=f(0)=0$. We now choose β and u_0 to ensure that

1. $\sigma'(0)=f'(0)=T_0$,
2. $\lim_{u\to\infty}\sigma(u)-u=\lim_{u\to\infty}f(u)-u=\frac{1}{2}\log T_0$.

The derivative of $\sigma(u)$ is easily found to be

$$\begin{aligned}\sigma'(0)&=\frac{e^{-\beta u_0}}{1+e^{-\beta u_0}} && (2.24)\\ &=(1+e^{\beta u_0})^{-1}. && (2.25)\end{aligned}$$

Requiring that it equals to $f'(0)=T_0$ allows us to solve for

$$u_0=\beta^{-1}\log\left(T_0^{-1}-1\right). \tag{2.26}$$

Next, in the large u limit, the biased Softplus converges to

$$\sigma(u)\to(u-u_0)-\beta^{-1}\log(1+e^{-\beta u_0}). \tag{2.27}$$

Solving for equality with $f(u)\to u+\frac{1}{2}\log T_0$ gives

$$\begin{aligned}u_0+\beta^{-1}\log(1+e^{-\beta u_0})&=-\frac{1}{2}\log T_0 && (2.28)\\ \beta u_0+\log\left(1+\frac{1}{T_0^{-1}-1}\right)&=-\frac{1}{2}\beta\log T_0 && (2.29)\\ \log(T_0)&=-\frac{1}{2}\beta\log T_0 && (2.30)\\ \beta&=2. && (2.31)\end{aligned}$$

Going back to Eq. (2.26), we obtain

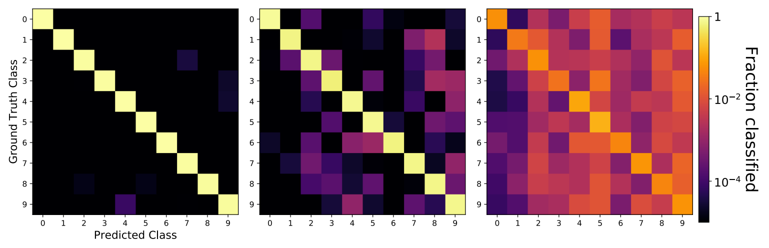
$$u_0=\frac{1}{2}\log(T_0^{-1}-1). \tag{2.32}$$

Fig. 2.C.1 plots the saturable absorption response curve compared to the Softplus approximation derived above.

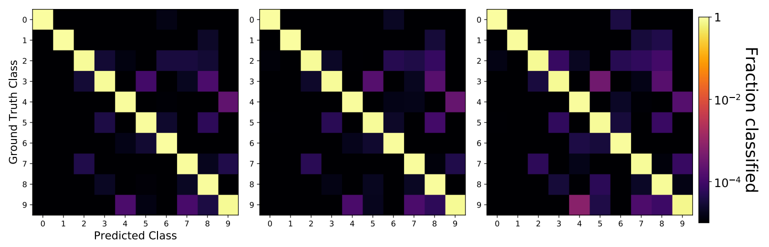
2.D Confusion matrices

To investigate the degradation of the networks due to imprecisions, we produce confusion matrices for both networks in the ideal case, with no imprecisions, and with different levels of error. $\sigma_{BS}=1\%$, $\sigma_{PS}=0.01$rad and $\sigma_{BS}=2\%$, $\sigma_{PS}=0.02$rad (Fig. 2.D.1).

The imprecisions were simulated 10 times and the mean of the output was used in generating the confusion matrices.



((a)) Ideal GridNet **((b))** GridNet : $\sigma = 0.01$ **((c))** GridNet : $\sigma = 0.02$



((d)) Ideal FFTNet **((e))** FFTNet : $\sigma = 0.01$ **((f))** FFTNet : $\sigma = 0.02$

Fig. 2.D.1. The degradation of ONN outputs visualized through confusion matrices. Each confusion matrix shows how often each target class (row) is predicted as each of the ten possible classs (column). Both networks, GridNet (a, b, c) and FFTNet (d, e, f) are evaluated. First in the ideal case (a, d) then, with increasing errors (b, e and c ,f). Note the logarithmic scaling.

2.E Quantization error

In this section, we explore the quantization error introduced by thermo-optic phaseshifters. Assuming a linear relationship between refractive index and temperature and quadratic relationship between temperature and voltage, we have

$$\begin{aligned} \theta &\propto V^2 \\ \theta &= 2\pi \left(\frac{V}{V_{2\pi}}\right)^2 \\ &\equiv 2\pi u^2 \\ \sqrt{\frac{\theta}{2\pi}} &= u. \end{aligned}$$

We have taken $V_{2\pi}$ to be the voltage required for a 2π phaseshift and defined the dimensionless voltage $u = V/V_{2\pi}$. Assuming that the voltage can be set with B-bit precisions, u must take

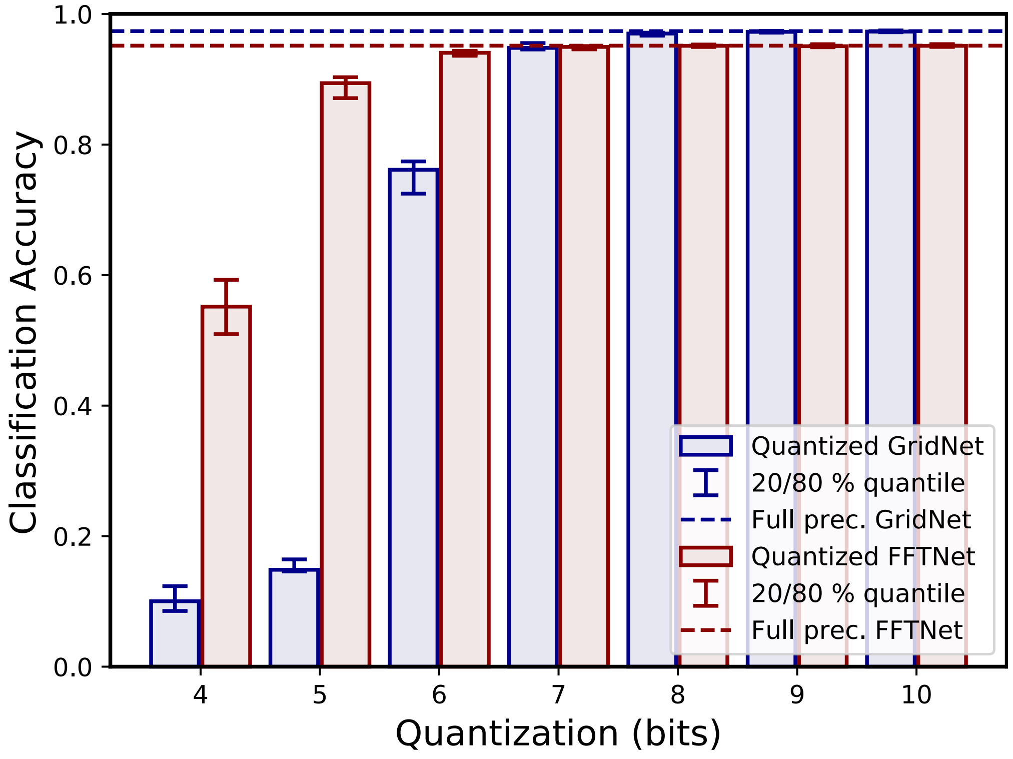


Fig. 2.E.1. The effects of quantization is shown for both GridNet and FFTNet. 10 instances of GridNet (blue) and FFTNet (red) were trained then quantized to varying levels. The mean classification accuracy at each level is shown by bar plots. The 20-80% quantiles are shown with error bars. The dotted horizontal line denotes the full precision accuracy.

on values of

$$u \in \{2^{-B} i : i = 0, \ldots, 2^B - 1\}.$$

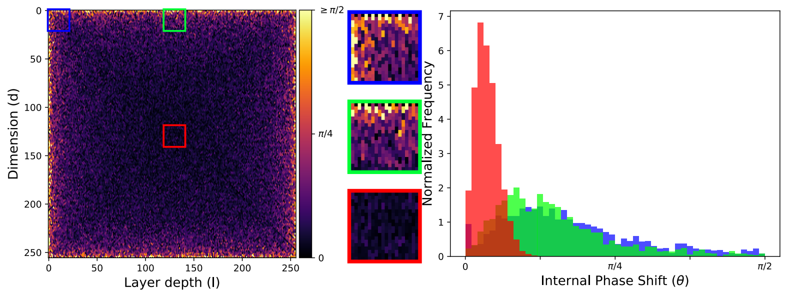
The quantization procedure then takes

$$\theta \to \tilde{\theta} \in \left\{ \frac{2\pi}{2^{2B}} i^2 : i = 0, \ldots, 2^B - 1 \right\}.$$

To evaluate the sensitivity to quantization, we quantized GridNet and FFTNet with varying levels of precision. Since quantization is deterministic, we trained 10 instances of both networks with randomized initialization and thus different configuration but similar ideal accuracies ($\sim 95\%$ and $\sim 98\%$). The networks were then quantized at varying levels – from 4 to 10 bits. Their classification accuracy at each level is shown in Fig. 2.E.1.

Similar to results with simulated Gaussian noise, FFTNet is more robust than GridNet. Note that in this case, the quantization was applied after training has finished. Neural networks in which quantization happens as part of the training procedure has been demonstrated to have accuracies very near their full precision counterpart, down to even binary weights [34, 60].

2.F Empirical distribution of phases



((a)) ((b))

Fig. 2.F.1. The central MZIs of GridNet has lower variance in internal phase shifts (θ). a) The spatial distribution of internal phase shift ($\theta_{d,l}$) of MZIs in U_2 of GridNet. Reference Fig. 2.2.1(a) for coordinates and Fig. 2.3.1 for location of U_2 in context of network architecture. b) Histogram of phase shifts near the center (red), edge (green), and corner (blue) of the GridUnitary multiplier. These phases are obtained from multiple instances of trained GridNets with random initialization.

Analyses has been done on the distribution of the internal phase shift (θ) of MZIs of GridUnitary multipliers when used to implement randomly sampled unitary matrices [65, 8, 56]. It was shown that the phases are not uniformly distributed spatially. To be more concrete, We denote d the waveguide number and l the layer number (see Fig. 2.2.1(a)). The distribution of the MZI reflectivity ($r = \sin(\theta/2)$) is[65]

$$r_{d,l} \sim \text{Beta}(1, \beta_{d,l}). \tag{2.33}$$

For large dimensions N,

$$\begin{aligned} \beta &\approx N - 2\max(|d - N/2|, |l - N/2|) & (2.34) \\ &= N - 2||(d,l) - (N/2, N/2)||_\infty. & (2.35) \end{aligned}$$

β decreases from N at the center of the grid layout to 0 at the edge of the grid. For large β (i.e. near the center), the mean and variance of $r_{d,l}$ are approximately

$$\mu_r \approx \beta^{-1}; \sigma_r^2 \approx \beta^{-2}.$$

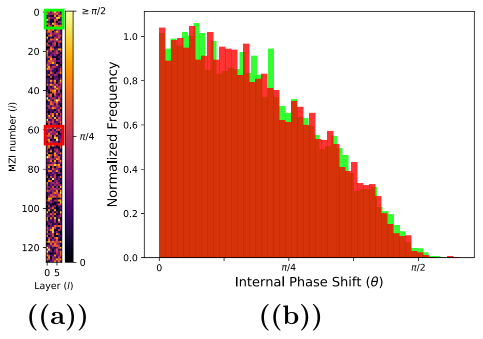


Fig. 2.F.2. The variance of internal phase shifts of FFTNet is uniform spatially (a) Spatial distribution of phase shifts for a FFTUnitary multiplier. The MZIs are ordered as shown in Fig. 2.2.1(b). (b) Histogram of phase shifts of FFTUnitary near the center (red) and top (green). These phases are obtained from mulitple trained FFTNets with random initialization.

Consequently, the reflectivity, and therefore the internal phases, of MZIs near the center of Gird Unitary multipliers are distributed very close to 0, with low variance. This effect is magnified with larger dimensions N.

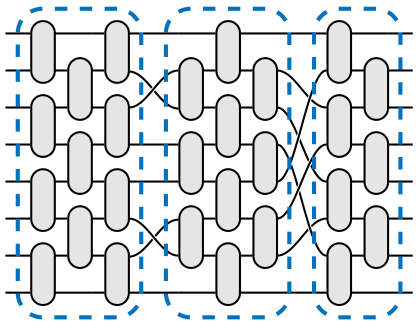
This result was derived with the assumption of Haar-random unitary matrices. Such a distribution is not guaranteed and not expected for layers of trained neural networks. (Fig. 2.F.1(a)) shows the spatial distribution of phases in the GridUnitary multiplier U_2 (see Fig. 2.3.1). While the empirical histogram (Fig. 2.F.1(b)) does not match the theoretical distribution (Eq. (2.33)), the general trend of lower variance near the center of GridUnitary multipliers is evident. This is claimed to translates to a lower tolerance for error[56].

A similar analysis was conducted for FFTNet. Immediately we notice that the distribution of phase shifts is mostly uniform across the MZIs (Fig. 2.F.2(a)). This can be attributed to the non-local connectivity of FFTUnitary multipliers. Histograms constructed from an ensemble of 100 trained FFTNets with random initial weights (Fig. 2.F.2(b)) confirms this observation. The histogram for the region near the center (red) is nearly identical to the top (green).

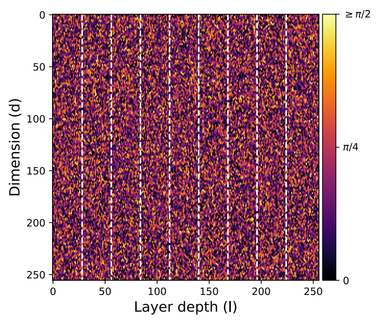
We reiterate the distinction, made in Section 2.4, between pre-fabrication error tolerance and the sensitivity of error introduced post-fabrication. Pertinent to the first concept is how well the network can be optimized after a known set of imperfections are introduced to the network. The latter concept, which is relevant for our discussion, describes the sensitivity of the network with no further reconfiguration to unknown errors. In contrast to pre-fabrication error tolerance, our analysis in 2.4 does not show significant spatial dependence for post-fabrication error sensitivity.

2.G BlockFFTNet

We introduce a network with similar depth as GridUnitary but with non-local, crossing waveguides in between as those seen in FFTUnitary (Fig. 2.G.1(a)). This is similar to the coarse-grained rectangular design mesh in [56] which was motivated to produce a spatially uniform distribution of phase and thus better tolerance for post-fabrication optimization. We also empirically observe that when incorporated as part of a ONN (BlockFFTNet), the distribution of phases are also uniformly distributed (Fig. 2.G.1(b)). We directly demon-



((a))



((b))

Fig. 2.G.1. a) A schematic of BlockFFTUnitary. Blocks of MZIs in dashed, blue boxes are similar to GridNet. The crossing waveguide, similar to those in FFTNet are between the blocks. b) The distribution of phases after being trained. The dashed white lines denote the locations of the crossing waveguides.

strate that better tolerance for post-fabrication optimization does not directly to better error-resistance for a network optimized pre-fabrication. The accuracy loss due to increasing imprecision is shown in Fig. 2.G.2.

2.H FFT algorithm and convolution

We show that the actual Cooley-Tukey FFT algorithm can be implemented with appropriate configurations of the phases of FFTUnitary multiplier.

If we denote the input as $x_n \in \mathbb{C}^N$, its Fourier transform is

$$X_k = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk}. \tag{2.36}$$

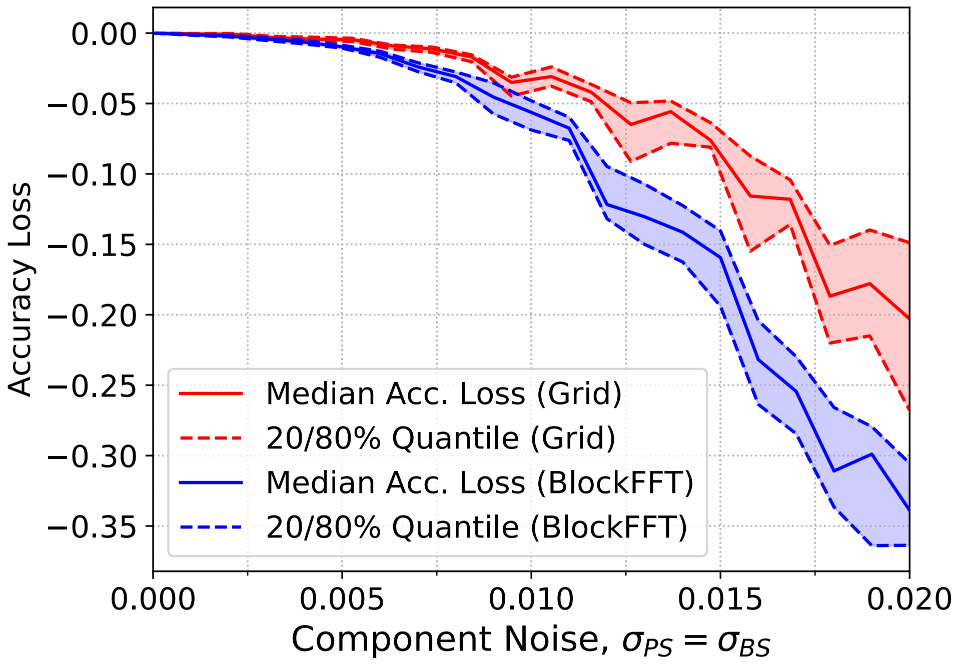


Fig. 2.G.2. No improvement in robustness to imprecision is seen with BlockFFTNet over GridNet. In fact, there is a significant decrease.

The FFT algorithm, in short, is to rewrite the above as

$$X_k = \frac{1}{\sqrt{2}}\left(E_k + e^{-\frac{2\pi i}{N}k}O_k\right) \tag{2.37}$$

$$X_{k+N/2} = \frac{1}{\sqrt{2}}\left(E_k - e^{-\frac{2\pi i}{N}k}O_k\right). \tag{2.38}$$

Here, we have defined O_k and E_k to be the Fourier transform on the odd and even elements of x_n respectively. The calculation of E_k and O_k are done recursively. For $N = 2^K$, a total of K iterations are needed. It is well known that if x_n is in bit-reversed order, the calculations can be done in place.

Furthermore, in matrix form,

$$\begin{pmatrix} X_k \\ X_{k+N/2} \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & e^{-\frac{2\pi i}{N}k} \\ 1 & -e^{-\frac{2\pi i}{N}k} \end{pmatrix}\begin{pmatrix} E_k \\ O_k \end{pmatrix} \equiv U_k \begin{pmatrix} E_k \\ O_k \end{pmatrix}.$$

From Eq. (2.1), we note that $U_k = U_{MZ}(\theta = \pi/2, \phi = 2\pi k/N)$, up to some global phase. Therefore, if x_n is in bit-reversed order, and passed through a FFTUnitary multiplier where the kth layer is configured with $\theta = \pi/2, \phi = 2\pi k/N$, FFT can be performed.

Going further, a convolution can be easily performed through multiplication of the Fourier transformed signal by the Fourier transformed convolutional kernel, followed by a inverse Fourier transform.

Chapter 3

Sparse Dictionary Learning through Analog Sampling

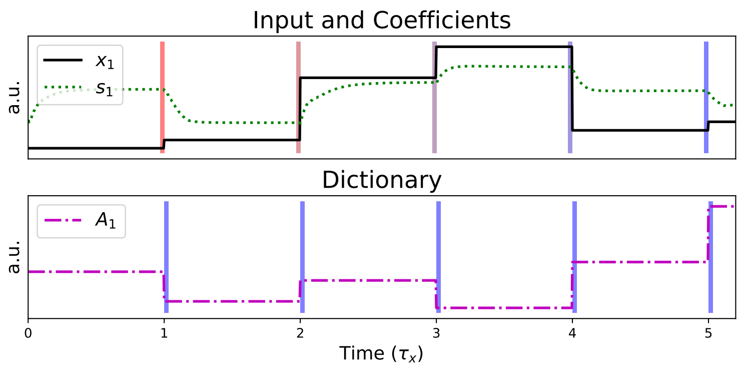
3.1 Introduction

Latent variable models such as sparse coding [52] and restricted Boltzmann machines [72] have been shown to be powerful and flexible tools in machine learning. However, training such models properly requires sampling from probability distributions over the variables. Typically, instead of sampling, a point estimate or other heuristic are used since most sampling algorithms are laboriously slow and have convergence guarantees only under limited conditions. The time cost in large part comes from simulating stochastic dynamics of state transitions on deterministic, discrete-logic based hardware, requiring random number generation and fine sampling intervals to avoid discretization errors.

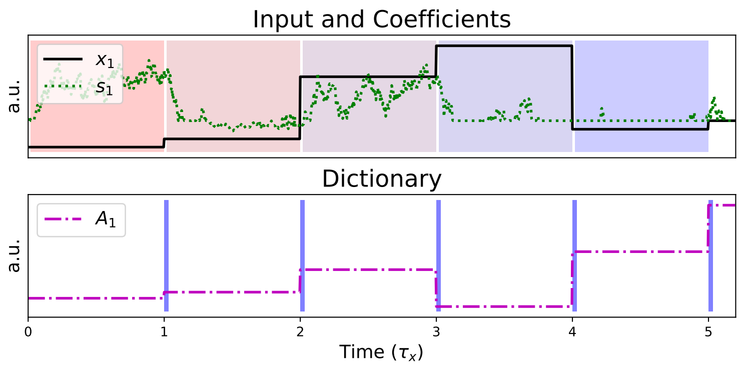
Hardware Samplers of various types[83, 44] have been demonstrated, in an effort to develop more efficient sampling methods (See Sec. 3.6 for more examples). The noise inherent in these systems is used as a source of stochasticity to sample over the latent variables. One direct way of exploiting these recent advances in analog sampling is to update parameters using the samples collected from the sampler (Fig. 3.1.1(b)). However, this requires a digital accumulator. Interfacing between analog and digital hardware is often a bottleneck for sampling. For example, in recent work by [63], the limiting component for a photonic sampler was identified as the photodetector.

Here we propose a novel, fully analog framework in which the update of parameters occurs alongside the sampling of latent variables through continuous time dynamics (Fig. 3.1.1(c)). Rather than waiting for the collection of samples for each discrete parameter update, the effective accumulation of samples is achieved through a longer time constant.

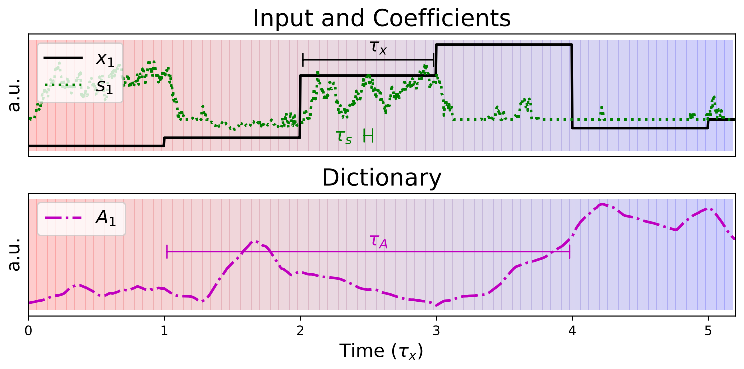
In neuroscience it has been theorized that seemingly random fluctuations in neural activity can be interpreted as a process for sampling from posterior distributions [33, 7, 55]. The results presented here are consistent with this idea, and moreover we propose specific analog circuits that could enable this, along with a more general framework that encompasses



((a)) Discrete Dictionary Update with MAP



((b)) Discrete Update with Sampling



((c)) Continuous Update with Sampling

Fig. 3.1.1. Illustration of three approaches to learning latent variable models. a) Data $\mathbf{x}$ is presented at regular intervals. A MAP estimate of latent variables $\mathbf{s}$ is calculated (green trace). This is used for a discrete update to the dictionary A. The colored vertical bars illustrate the computational inefficiency where only a single point estimate of the coefficients are used to make a discrete dictionary update. b) For each data interval, samples are collected for a discrete dictionary update. The colored regions in the top panel show that many samples are collected to approximate the posterior distribution. However, the discrete dictionary updates (at corresponding vertical bands) make a fully analog implementation difficult. c) Rather than waiting for the accumulation of samples, The dictionary A is updated continuously alongside both the latent variables $\mathbf{s}$. The slow timescale of the dictionary compared to the latent variables $\tau_A \gg \tau_s$ allows for effective averaging. Note that the hyper-parameters such as learning rates were not optimized for best convergence or regularization but instead

dynamics for both inference and learning.

Langevin Sparse Coding; To study this analog sampling framework, we apply it specifically to sparse coding. Sparse coding [79, 28] is a simple yet expressive probabilistic model with an explicit prior over the latent variables. It has also been shown to account for the neural representation of natural images in visual cortex [53]. In Section 3.3, we present Langevin Sparse Coding (LSC), a fully continuous model making use of Langevin dynamics to sample latent variables ($\mathbf{s}$) and slower dynamics to update the dictionary (A).

Sampling with Langevin dynamics is well studied both in theory [9] and in application to Bayesian learning [84]. However, to our knoweldge, this is the first fully analog approach to inference and learning for sparse coding. Prior sampling-based approaches utilized a mixture-of-Gaussians model and employed Gibbs sampling over the mixture weights [54] or a method for preselecting parts of the space to sample via MCMC [68].

A key assumption of the sparse coding model is that the latent variables $\mathbf{s}$ assigned to any given data sample $\mathbf{x}$ should be mostly zero. Traditionally this is enforced by imposing an L_1 cost function on $\mathbf{s}$, which is used as a proxy for L_0 since it allows for convex optimization. However, in the probabilistic setting an L_1 cost corresponds to a Laplacian prior, which only weakly captures the notion of sparsity. We show in Section 3.4 that LSC allows for a straightforward implementation of an L_0 sparse prior. In Section 3.5, with a synthetic dataset, we show that our approach is truly probabilistic by recovering the expected prior and sparsity. Furthermore in Section 3.5, we demonstrate that our approach also allows for learning the size of the dictionary, which was attempted in previous work using variational approximation of the posterior [6].

Finally, in Section 3.5, we apply LSC in fitting our L_0-sparse prior to the Van Hateren dataset of natural images. In addition to learning the dictionary elements, we provide an estimate for the sparsity of natural images, a challenging problem. To summarize, the main contributions presented are:

- A theoretical formulation of mixed time-scale analog dynamics for simultaneously sampling from and training latent variable models.
- LSC – A continuous-time, probabilistic model for training sparse coding.
- Using LSC to efficiently implement sampling of the posterior with a L_0 prior.
- Using LSC to learn a dictionary for representing natural images as well as the associated sparsity level.
- In addition to learning said dictionary, showing that LSC allows for learning the parameters of an L_0 prior over the latent variables, as well as the size of the dictionary.

3.2 Sparse Coding

Sparse coding is a simple yet efficient algorithm for dictionary learning. By adding an L_1 regularization to the reconstruction loss, sparsity of coefficients can be achieved. While the formulation is probabilistic, due to computational constraints, point estimates are typically used to represent posterior distributions. As a result, the theoretically desired distribution is not achieved.

In this section, starting from the canonical discrete sparse coding (DSC), sequential modifications are introduced to produce a progressively more analog algorithm. First, with continuous time sparse coding (CTSC), dictionary updates are made continuous and concurrent with the dynamics of the coefficients. With this modification, the nested loop structure is made unnecessary and the input can be fed in asynchronously. By not only accounting for, but also taking advantage of the inherent noise in analog systems to perform sampling, in Sec. 3.3, we arrive at LSC.

Discrete Sparse Coding

Sparse coding assumes that data from some data set ($\mathbf{x} \in R^D$) is distributed as a linear combination of dictionary $A \in \mathbb{R}^{D\times K}$ with added Gaussian noise $\mathbf{n}$.

$$\mathbf{x} = A\mathbf{s} + \mathbf{n} \tag{3.1}$$

with $n_i \overset{iid}{\sim} N(0, \sigma^2)$. The coefficients $\mathbf{s}$ are further assumed to be L_1 sparse with a Laplacian prior:

$$p_s(s_i) \propto \exp(-\lambda|s_i|). \tag{3.2}$$

This probabilistic model can be described with the following energy function

$$E(A, \mathbf{s}, \mathbf{x}) = \frac{1}{2}\frac{||\mathbf{x} - A\mathbf{s}||_2^2}{\sigma^2} + \lambda||\mathbf{s}||_1 \tag{3.3}$$

such that $p(\mathbf{x}, \mathbf{s}|A) = Z^{-1}\exp(-E(A, \mathbf{s}, \mathbf{x}))$. Note that conveniently, the partition function $Z = (\sqrt{8\pi}\sigma/\lambda)^D$ and is a constant, specifically independent of A. This yields the nice property that

$$\nabla_A \log p(\mathbf{x}, \mathbf{s}|A) = -\nabla_A E(A, \mathbf{s}, \mathbf{x}). \tag{3.4}$$

Our goal is to solve for the maximum likelihood estimator (MLE) of the dictionary

$$A^* = \arg\max_A \langle \log p(\mathbf{x}|A)\rangle_{\mathbf{x}\sim\mathcal{D}}. \tag{3.5}$$

The expectation $\langle\cdot\rangle_{\mathbf{x}\sim\mathcal{D}}$ denotes expectation over dataset $\mathcal{D}$ (e.g. natural images). The MLE can be found through gradient descent, where the gradient is

$$-\nabla_A\langle\log p(\mathbf{x}|A)\rangle = \Big\langle\langle\nabla_A E(A,\mathbf{s},\mathbf{x})\rangle_{\mathbf{s}|\mathbf{x}}\Big\rangle_{\mathbf{x}\sim\mathcal{D}} \tag{3.6}$$

$$= \Big\langle\big\langle(A\mathbf{s}-\mathbf{x})\mathbf{s}^T\big\rangle_{\mathbf{s}|\mathbf{x}}\Big\rangle_{\mathbf{x}\sim\mathcal{D}} \tag{3.7}$$

In practice, the expectation over the data is approximated via stochastic gradient descent (SGD). On a drawn data $\{\mathbf{x}_n\}_{n=1\ldots N}$ of batch size N, the update rule is

$$\Delta A = \frac{1}{N}\sum_{n=1}^{N}\big\langle\ p(\mathbf{s}_n|\mathbf{x}_n)\cdot(A\mathbf{s}_n-\mathbf{x}_n)\mathbf{s}_n^T\big\rangle_{\mathbf{s}_n|\mathbf{x}_n} \tag{3.8}$$

However, the expectation over $\mathbf{s}_n$ is also intractable and is typically approximated by the maximum *a posteriori* (MAP) estimator of $\mathbf{s}_n$

$$\mathbf{s}_n^* = \arg\min_{\mathbf{s}_n} E(A,\mathbf{s}_n,\mathbf{x}_n). \tag{3.9}$$

This can also be found via gradient descent with updates

$$\Delta\mathbf{s}_n = -\nabla_{\mathbf{s}_n}E(A,\mathbf{s}_n,\mathbf{x}_n) \tag{3.10}$$

$$= -\frac{1}{\sigma^2}A^T(A\mathbf{s}_n-\mathbf{x}_n)-\lambda\cdot\mathrm{sgn}(\mathbf{s}_n). \tag{3.11}$$

Both updates ΔA and $\Delta\mathbf{s}_n$ can be expressed more efficiently through gradient descent on a batch energy function:

$$E(A,S,X)\equiv\sum_n E(A,\mathbf{s}_n,\mathbf{x}_n) \tag{3.12}$$

$$= \frac{1}{2}\frac{||AS-X||_F^2}{\sigma^2}+\lambda||S||_{1,1}. \tag{3.13}$$

We have defined matrices $S\in\mathbb{R}^{K\times N}$ and $X\in\mathbb{R}^{D\times N}$. Recall that $A\in\mathbb{R}^{D\times K}$ with D being the dimension of the the data, K the number of dictionary bases and N the batch size of X. Above, $||\cdot||_F$ and $||\cdot||_{1,1}$ are the Frobenius and $(1,1)$ matrix norms respectively. With the batch energy defined, the update rules are

$$S\leftarrow S-\eta_s\frac{\partial E(A,S,X)}{\partial S} \tag{3.14}$$

$$A\leftarrow A-\eta_A\frac{1}{N}\frac{\partial E(A,S,X)}{\partial A}. \tag{3.15}$$

Note that we can redefine η_A to be η_A/N for simplicity. To coordinate the two updates, a nested loop must be used (Alg. 3). The inner loop approximates the MAP estimator S^* while the outer loop finds the MLE of A.

Algorithm 1 Algorithm for discrete sparse coding. Note line 6 was included purely to emphasize S^* as an estimate of the MAP.

```
1: for $k \leftarrow 1$ to $N_A$ do
2:     for $n \leftarrow 1$ to $N_s$ do
3:         $X \leftarrow$ SampleBatch()
4:         $S \leftarrow S - \eta_S \cdot \frac{\partial E}{\partial S}(A, S, X)$
5:     end for
6:     $S^* \leftarrow S$
7:     $A \leftarrow A - \eta_A \frac{\partial E}{\partial A}(A, S^*, X)$
8: end for
```

Continuous Time Sparse Coding

Rather than updating the dictionary A at the end of the loop when S has converged to the MAP estimator S^*, CTSC updates A continuously.

$$S \leftarrow S - \eta_S \nabla_S E(A, S, X) \tag{3.16}$$

$$A \leftarrow A - \frac{\eta_A}{N_S} \nabla_A E(A, S, X). \tag{3.17}$$

To account for the fact that A will be updated multiple times instead of only once at the end of the inner loop, the learning rate was divided by N_S, the number of loop iterations. In search of dynamics amenable to analog computation, we take the step sizes to be infinitesimally small, and arrive at the following set of differential equations.

$$\tau_S \dot{S} = -\nabla_S E(A, S, X(t)) \tag{3.18}$$

$$\tau_A \dot{A} = -\nabla_A E(A, S, X(t)). \tag{3.19}$$

Here, we take $X(t)$ to be updated synchronously at regular intervals of τ_X. At each update, a new batch of samples is drawn. To see the similarity between CTSC and DSC, consider the following simulation for CTSC using Euler Method.

Algorithm 2 Euler Method simulation of CTSC with stepsize of Δt and regular interval input of X

```
1: for $t \leftarrow 1$ to $t_{\max}/\Delta t$ do
2:     $dS \leftarrow \frac{\partial E}{\partial S}(A, S, X(t))$
3:     $dA \leftarrow \frac{\partial E}{\partial A}(A, S, X(t))$
4:     $S \leftarrow S - \frac{\Delta t}{\tau_S} \cdot dS$
5:     $A \leftarrow A - \frac{\Delta t}{\tau_A} \cdot dA$
6: end for
```

Comparing Alg. 3 and Alg. 2, the timescales, τ can be related to the learning rates η, and the number of iterations N.

With continuous time dynamics, there is no longer need for synchronous, regular input of X. While maintaining the advantages conferred by parallel, batched inputs the asynchronous behavior allows for easier implementation in analog devices.

3.3 Langevin Sparse Coding

Langevin dynamics is described by the following stochastic differential equation:

$$\dot{\mathbf{u}} = -\nabla E(\mathbf{u}) + \sqrt{2T}\xi(t), \tag{3.20}$$

where $\xi(t)$ is independent Gaussian white noise with $\langle \xi(t)\xi(t')^T \rangle = \mathbf{I}\delta(t-t')$. The distribution of $p(\mathbf{u}(t))$, over time, will asymptotically converge to

$$p^{(\infty)}(\mathbf{u}) \propto e^{-E(\mathbf{u})/T} \tag{3.21}$$

We can change the dynamics of CTSC (Eq. 3.19) by injecting noise to $\dot{S}$ so that

$$\tau_S \dot{S} = -\nabla_S E(A, S, X) + \sqrt{2T\tau_S}\xi(t) \tag{3.22}$$

For a fixed A, over time, S will sample from the desired distribution.

$$p_S(S(t)) \propto e^{-E(A,S,X)/T}. \tag{3.23}$$

With $T = 0$, we recover the CTSC dynamics where the dynamics converge to the MAP estimation of S. A useful property of (Eq. 3.22) is that the equilibrium distribution is independent of the time constant τ_s. By taking $\tau_A \gg \tau_S$, the assumption that A is fixed with respect to the dynamics of S can be upheld. Conversely, because S evolves much faster than A, the dynamics of A is well approximated by

$$\tau_A \dot{A} = -\langle \nabla_A E(A, S, X) \rangle_{S|A,X}. \tag{3.24}$$

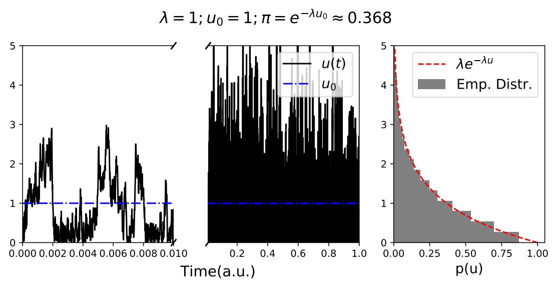
This is the exact mean gradient we wish to calculate.

While all of the results are derived from computer simulations of Langevin dynamics (see App. 3.D), our aim is not to produce another MCMC algorithm but to demonstrate the feasibility of training latent variable models with analog devices. In the sections that follow, we illustrate utility of systems capable of implementing LSC.

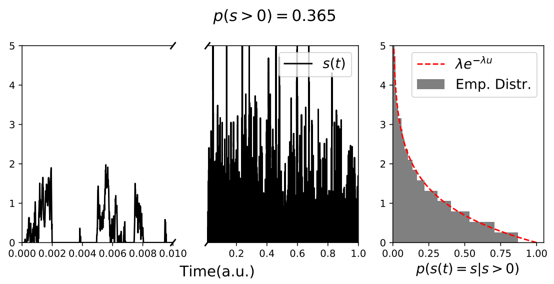
3.4 L_0 Sparse Prior

Due to the difficulty of MAP estimation for L_0 priors, a L_1 prior is typically used. Consider, instead, the following prior:

$$p_0(s)ds = \pi\lambda e^{-\lambda s} + (1 - \pi)\delta(s). \tag{3.25}$$



((a)) Sampling $p(u)$ using Langevin dynamics



((b)) Sampling $p(s)$

Fig. 3.4.1. An L_0 can be sampled from using Langevin dynamics and a hidden variable. a) The hidden variable u is made to follow an exponential distribution. b) By applying a threshold function, $s = f(u)$ achieves L_0 sparsity.

With π as the activity, $1 - \pi$ quantifies the L_0 sparsity, how likely s is to be zero. When nonzero, s is exponentially distributed. Were we to find the MAP estimator with this prior, using gradient descent, we would always eventually end up with $s = 0$, regardless of the likelihood. However, with Langevin dynamics, LSC can efficiently implement this L_0 sparse prior (Fig. 3.4.1).

First, we define a hidden variable $\mathbf{u}$ such that u_i independently follows the exponential distributed:

$$p_U(u_i) = \lambda e^{-\lambda u_i}. \tag{3.26}$$

We then take the coefficients to be $u_i = f(s_i)$ where f is a biased ReLU function:

$$s_i = f(u_i) = \begin{cases} u_i - u_0 & \text{if } u_i > u_0 \\ 0 & \text{if } u_i < u_0 \end{cases} \tag{3.27}$$

for some positive u_0. We then evaluate the distribution of random variable $S = f(U)$. First,

$$P(S = 0) = \int_0^{u_0} du\ p_U(u) \tag{3.28}$$

$$= 1 - e^{-\lambda u_0} \equiv 1 - \pi. \tag{3.29}$$

We have defined $\pi = e^{-u_0}$. For positive $s > 0$, $du/ds = 1$, so

$$p_S(s)ds = p_U(u)du \tag{3.30}$$

$$= \lambda e^{-\lambda(s+u_0)} \tag{3.31}$$

$$= \pi\lambda e^{-\lambda s}. \tag{3.32}$$

This means that S is distributed according to the desired L_0-sparse prior provided that

$$u_0 = -\lambda^{-1}\log(\pi). \tag{3.33}$$

To continue with L_0-LSC, we modify the energy function to be

$$E(A, \mathbf{u}, \mathbf{x}) = \frac{1}{2}\frac{||\mathbf{x} - Af(\mathbf{u})||_2^2}{\sigma^2} + \lambda||\mathbf{u}||_1. \tag{3.34}$$

Here, $\mathbf{s}$ is still the coefficients relevant for the reconstruction

$$\hat{\mathbf{x}} = A\mathbf{s} = Af(\mathbf{u}). \tag{3.35}$$

However, the prior is defined over $\mathbf{u}$ instead. Furthermore, the variable undergoing Langevin dynamics is also $\mathbf{u}$. This way, conditioned on $\mathbf{x}$, the distribution of $\mathbf{u}$ will converge to

$$\begin{aligned} p(\mathbf{u}|\mathbf{x}) &\propto \exp\left(-||Af(\mathbf{u}) - \mathbf{x}||_2^2/\sigma^2 - \lambda||\mathbf{u}||_1\right) && (3.36)\\ &= p(\mathbf{x}|f(\mathbf{u}))p_U(\mathbf{u}) && (3.37)\\ &= p(\mathbf{x}|\mathbf{s})p_0(\mathbf{s}). && (3.38)\end{aligned}$$

The distribution

$$p(\mathbf{u}, \mathbf{x}|A) \propto e^{-E(A,\mathbf{u},\mathbf{x})} \tag{3.39}$$

has a partition function that is constant with respect to A and π as well. By following the derivation in Sec. 3.A, we can also optimize for u_0, and therefore, π via

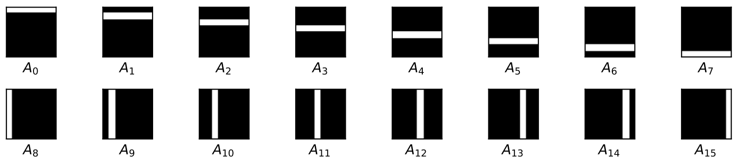
$$\begin{aligned} \dot{u}_0 &\propto -\left\langle\left\langle\frac{\partial E}{\partial u_0}\right\rangle_{\mathbf{s}|\mathbf{x}}\right\rangle_{X\sim\mathcal{D}} && (3.40)\\ &= \left\langle\left\langle A^T(A\mathbf{s} - \mathbf{x})\cdot\mathbf{1}(\mathbf{s} > 0)\right\rangle_{\mathbf{s}|\mathbf{x}}\right\rangle_{\mathbf{x}\sim\mathcal{D}} && (3.41)\end{aligned}$$

3.5 Results

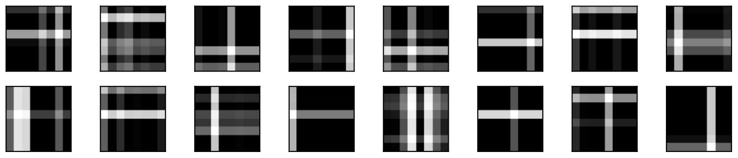
To study the efficacy of L_0 sparsity with LSC, we consider dictionary learning on the bars dataset and natural scenes. The bars dataset presents itself as a nice control since we know all the *causes* that generate the data.

Inference on Bars Dataset

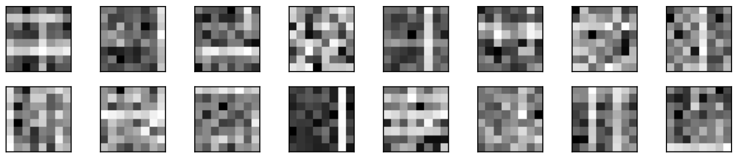
For the bars dataset, samples are generated from a dictionary A consisting of vertical and horizontal lines (Fig. 3.5.1(a)). We compare results obtained on this dataset against DSC as



((**a**)) Bars Dictionary



((**b**)) Bars Sample: $\lambda = 1, \pi = 0.3, \sigma = 0$



((**c**)) Bars Sample: $\lambda = 1, \pi = 0.3, \sigma = 0.5$

Fig. 3.5.1. The synthetic Bars dataset used as a toy problem. a) The dictionary is the collection of vertical and horizontal lines. b) An example of a sample drawn from the dataset. c) Another sample with noise introduced.

well as another method for training sparse coding, the locally competitive algorithm (LCA) [64].

We synthetically generate data as a linear combination of the dictionary with additive Gaussian noise:

$$\mathbf{x} = A\mathbf{s} + \mathbf{n} \tag{3.42}$$

where, $n_i \sim N(0, \sigma^2)$ and the coefficients are distributed according to L_0 zero-inflated exponential prior (Eq. 3.25). A sample drawn from this model without noise and with noise is shown in Fig. 3.5.1(b) and 3.5.1(c).

When trained on this dataset, all three algorithms were successful at learning the correct dictionary. However, LSC can better capture the posterior distribution than either DSC or LCA. As a consequence, LSC can enforce L_0 sparsity much more easily and directly. The sparsity is controlled through adjusting a parameter λ in both DSC and LCA. However, the relationship between λ and L_0 (Fig. 3.5.2(a)) is rather indirect and no analytic expression is known. On the other hand, a specific level of L_0-sparsity $(1 - \pi)$ can be directly enforced by setting $u_0 = -\lambda^{-1} \log(\pi)$ (Eq. 3.33).

Moreover, the activity π can be learned by LSC without any guesswork or parameter search (Eq. 3.40). Figure 3.5.2(b) shows the approximate convergence of model parameter

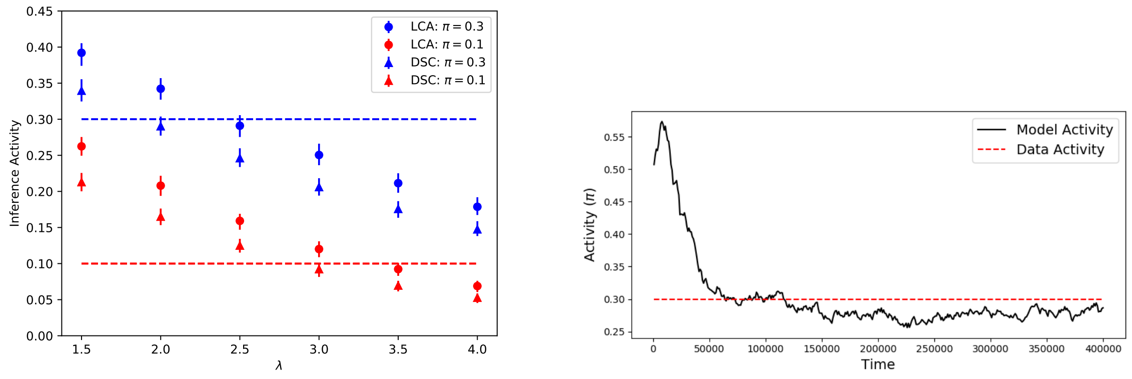


Fig. 3.5.2. a) A correspondence between the parameters λ for both DSC and LCA is mapped out above. There is however, no obvious, analytic relationship between λ and the observed activity π. The parameter λ is swept for both LCA and DSC when trained on data drawn with activity $\pi = 0.3$ and $\pi = 0.1$. b) With LSC, the activity π can be learned directly without a parameter search.

π to the actual data activity. To further characterize the coefficients, the distributions of the non-negative coefficients of the three algorithms were also plotted in Fig. 3.5.3. Using a fixed dictionary, the algorithm was run to infer either the MAP (DSC and LCA) or to sample from the posterior of input data. This was done with a correctly learned dictionary (Fig. 3.5.1(a)) as well as a random dictionary (i.e. uncorrelated gaussian noise). In addition to having the correct L_0-sparsity, LSC also samples correctly the desired prior (Fig. 3.5.3(c)), in contrast to the other non-stochastic algorithms. In fact, this is only true with the correctly learned dictionary. A more quantified analysis is provided in Section 9 in the Supplimentary Materials.

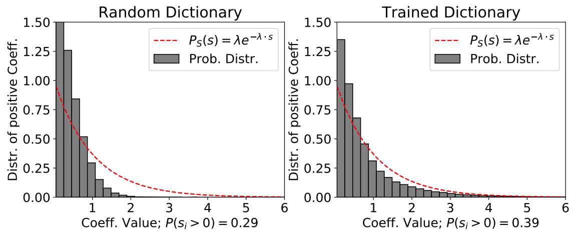
Learning the Dictionary Norm

For both DSC and LCA, at each dictionary update, it is necessary to normalize the dictionary elements. For $A = (\mathbf{A}_1, \ldots, \mathbf{A}_K)$, after updating, we reassign

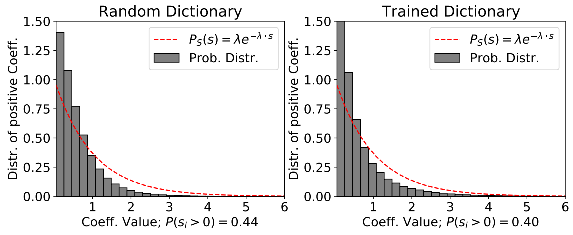
$$\mathbf{A}_i \leftarrow \frac{\mathbf{A}_i}{||\mathbf{A}_i||_2} \equiv \hat{\mathbf{A}}_i. \tag{3.43}$$

This is because the MAP estimator $\mathbf{s}^*$ is biased and will consistently underestimate s_i. To compensate, $\mathbf{A}_i$ will grow unless normalized (See Fig. 12 in the Suppl. Materials). In fact, growing without bound, DSC and LCA cannot learn the correct dictionaries this way.

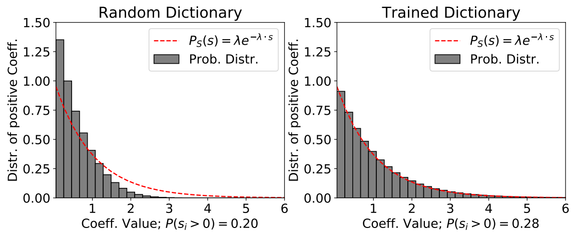
LSC, however, performs sampling and does not suffer from the same problem. As a result, when using LSC, there is no need for normalization. Instead, the dictionary element norms $||\mathbf{A}_i||$ will grow or shrink to optimize the likelihood.



((a)) DSC



((b)) LCA

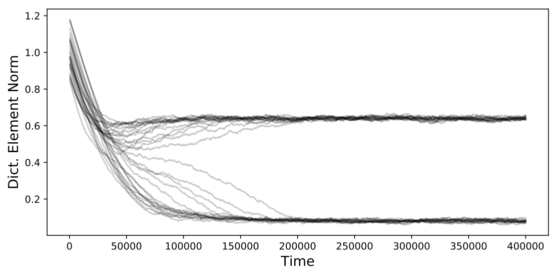


((c)) LSC

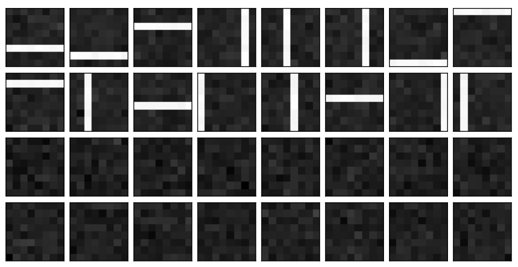
Fig. 3.5.3. The distribution of non-zero coefficients of each of the four algorithms. The dotted red line shows the prior of coefficients used in generating the dataset. The left panel of each subfigure shows the distribution when each algorithm is run with random dictionaries. The right shows the the distribution with trained dictionaries. Only LSC, with the correctly trained dictionary has the a distribution matching the prior.

The adaptive norm property can be used to automatically select for the number of dictionary elements needed. For data of dimension D, we consider a dictionary of size $K = \Omega \times D$, to have an (over)completeness of Ω. A $2\times$ overcomplete model was trained using the LSC algorithm using a fixed activation probability pi, without normalizing the dictionary A. The resulting, learned dictionary is shown in Fig. 3.5.4(b). In previous work by [6], Annealed Importance Sampling (AIS) [49] was used to approximate the marginal likelihood in order to find the optimal dictionary elements. However, LSC, without additional procedures, can be used to effectively do the same through attenuation of unnecessary dictionary elements.

The learned dictionary contains exactly the bars dictionary and the remaining elements were made to vanish. A clear illustration of this is shown in Fig. 3.5.4(a). When both $||\mathbf{A}_i||$ and π are being learned, a more stable solution is to have duplicated dictionary elements



((a)) Evolution of dictionary norms



((b)) Unnecessary dictionary elements vanish.

Fig. 3.5.4. Fixing, the activity π, LSC is use to learn the dictionary of a twice overcomplete model. a) Dictionary norms bifurcate. b) Half of the elements vanish leaving exactly one copy of the generating dictionary elements.

with a reduced activity. is shown in Fig 3.5.5(a) with a duplicated dictionary but halved activity (Fig. 3.5.5(b)).

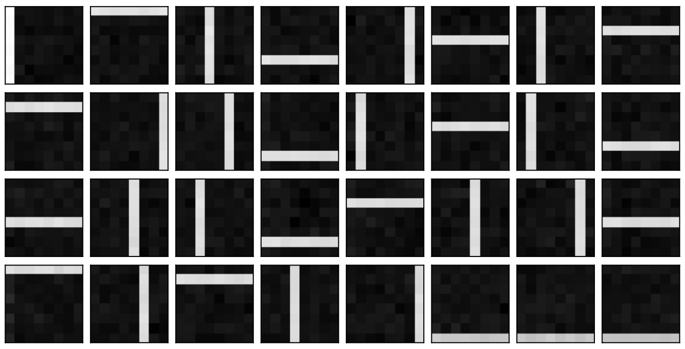
Natural Image Patches

We ran the LSC algorithm on 8×8 patches of natural scenes from the Van Hateren dataset[29] (see Fig. 3.5.6). First, the model activity was fixed at $\pi = 0.5$ and we used LSC to learn a $4\times$ overcomplete dictionary ($K = 4 \times 64 = 256$). We can see in Fig. 3.5.7 that only a fraction of the dictionary was utilized. The "inactive" dictionary elements had a comparatively insignificant norm. In contrast to prior work by [6], this is emergent from dictionary learning with sampling; no algorithms were used to determine the optimal number of dictionary elements.

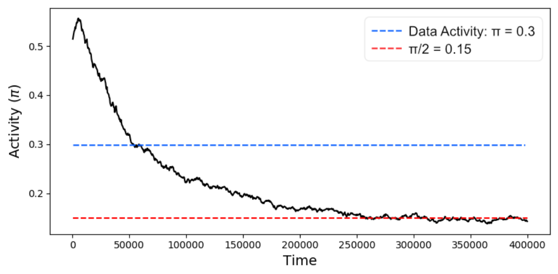
Then, unfixing π, we allow the activity to be learned. Repeating the experiment at different levels of overcompleteness Ω, a correspondence between the activity and overcompleteness is plotted in Figure 3.5.8(a). This relationship happens to be very well modeled by $\pi \propto \Omega^{-1}$. As a consequence, the expected number of active dictionary elements, $\pi \times K = \pi \times \Omega \times D$ stayed near constant irrespective of the the completeness Ω.

3.6 Hardware implementation

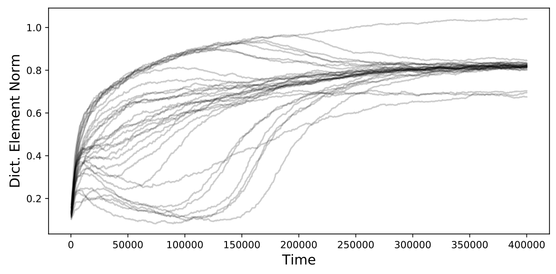
A major shortcoming of most sampling approaches to latent variable models has been their intractability. Indeed, when simulating LSC on a digital computer using the Euler-Maruyama algorithm, the run time needed was significantly longer when compared to the point estimate counterparts (DSC and LCA). However, the LSC algorithm was designed for eventual hardware implementation. Presented here is a variety of candidates for such implementation.



((a)) Duplicated dictionary learned



((b)) Activity learned by twice overcomplete model

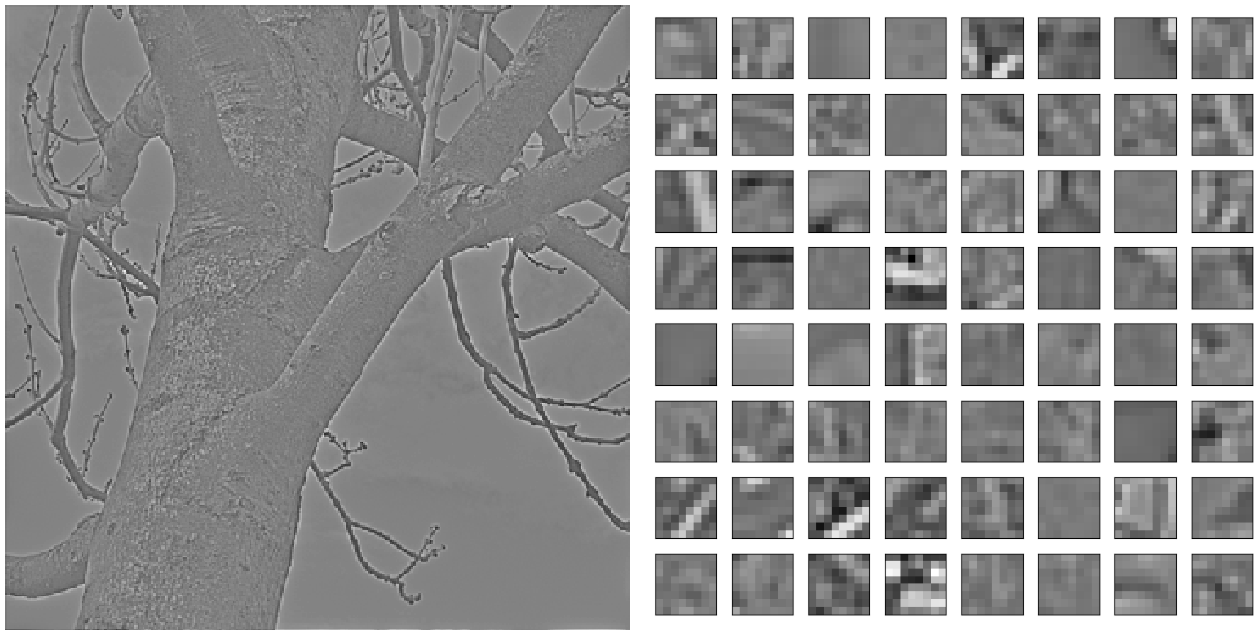


((c)) Evolution of dictionary norms

Fig. 3.5.5. LSC is used to learn both the dictionary and activity of the same overcomplete model a) The dictionary the learned is duplicated. b) But the activity π is half of the actual activity.

A major requirement is an analog method of implementing matrix-vector multiplication. This has been demonstrated with photonic meshes of Mach-Zehnder interferometers (MZIs)[61]. In fact, MZI meshes have been shown capable of sampling from Ising models[63]. Another good candidate is resistor crossgrid arrays. Memristors[11], in particular, allow for fast, configurable coupling, which is necessary for fast weight updates. Finally, coupled oscillators have also been shown viable in sampling Ising models [83] as well as Hopfield networks[50].

Specific to sparse coding, previous work by [70] have implemented the LCA algorithm for sparse coding on a memristor network with a field-programmable gate array (FPGA) for weight updates. To take advantage of an algorithm like LCA would require a fully analog implementation and is a direction for further research.



((a)) Whitened Image ((b)) Image patches

Fig. 3.5.6. a) An example of whitened Van Hateren image. b) A sample of 8×8 patches taken from the Van Hateren images.

3.7 Discussion

We have presented a novel mixed time-scale analog sampling framework. Using this, in Sec. 3.3, we have introduced Langevin Sparse Coding (LSC), a novel method of training sparse coding that is especially amenable to implementation on analog devices. Designed to make use of inherent stochasticity found in analog systems it is able to sample from a given posterior distribution. Furthermore, through application of a threshold function to the stochastic dynamics, we demonstrate that a truly L_0-sparse prior can be implemented (Sec. 3.4).

Applied to a synthetic dataset, LSC was compared against two alternative sparse coding algorithms (DSC and LCA). From the results presented in Section 3.5, LSC was shown to be better at sampling from the posterior distribution as well as capable of learning the mean activity π of the latent variables $\mathbf{s}$. We then used LSC for dictionary learning of natural images (Sec. 3.5). One notable result found was that the mean number of of dictionary elements was mostly invariant to the total dictionary size. This ran contrary to previous results by [51] where, on average, the number of elements required for reconstructing a given image went down with larger dictionaries.

In part, this discrepancy can be reconciled by the fact that the previous work was not done with a truly probabilistic model of sparse coding. In such case, larger dictionaries do yield

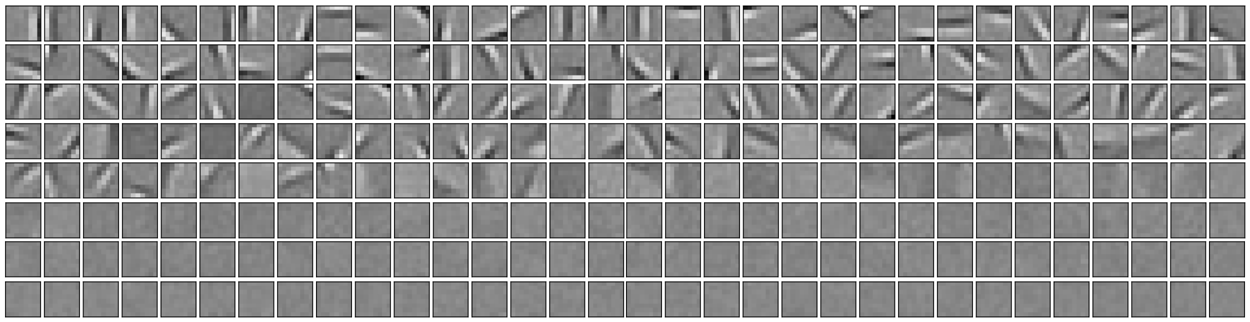
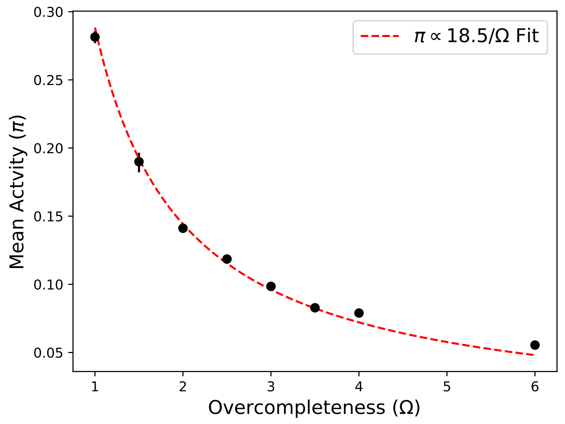
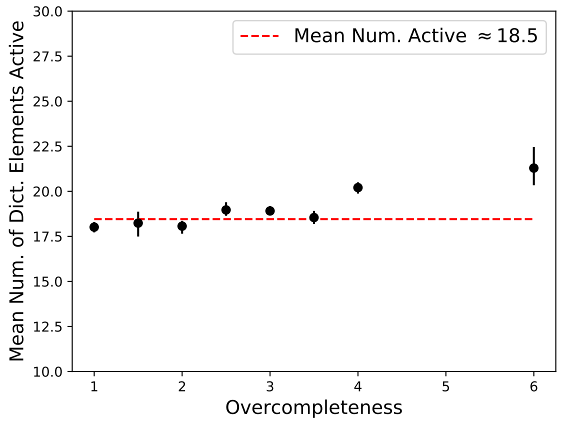


Fig. 3.5.7. With activity fixed ($\pi = 0.5$), only a fraction of the total dictionary elements are "active". The remaining have been made to vanish. The dictionary elements are sorted by their respective norms.



((a)) Mean activity at different completeness



((b)) Mean active dictionary elements

Fig. 3.5.8. a) Using LSC to learn dictionaries for natural scenes at multiple times at different levels of completeness α, a $\pi \propto 1/\alpha$ relationship is obtained. b) This implies that the mean number of dictionary elements active is constant irrespective of the total number of dictionary elements learned. Error bars on both plots denote the 10% - 90% range

fewer necessary elements for a MAP reconstruction. One can imagine a massive dictionary where every possible natural image is represented. A single element is then required in representing any image. However, in the case of LSC, each element has a prior probability of π to be active regardless of the image.

Nonetheless, it is still curious that the mean number of dictionary elements active was near constant suggesting that the overcompleteness is an under-utilized degree of freedom. One hypothesis is that the prior of i.i.d. exponential distributions is overly simplified and ill-suited to represent the statistics of natural images. While we have only used the L_0 sparse exponential prior, LSC with more sophisticated priors needs to be explored. For example, previous work by [24] used Laplacian Scale Mixture (LSM) to model dependencies across dictionary elements.

The mixed time-scale analog sampling framework on which LSC is based goes beyond just sparse coding. We hope to develop analogous procedures for learning other latent variable models such as Restricted Boltzmann Machines, hierarchical Bayesian models[42], etc.

A review of potential hardware systems for implementation is presented in Section 3.6. This is not exhaustive and more work needs to be done in designing a scalable analog system for implementing LSC. Nonetheless, the work presented here highlights the benefits that could be brought by implementation of probabilistic models on analog systems.

Appendix

3.A Discrete Sparse Coding

Sparse coding assumes that data from some data set ($\mathbf{x} \in R^D$) is distributed as a linear combination of dictionary $A \in \mathbb{R}^{D \times K}$ with added Gaussian noise $\mathbf{n}$.

$$\mathbf{x} = A\mathbf{s} + \mathbf{n} \tag{3.44}$$

with $n_i \overset{iid}{\sim} N(0, \sigma^2)$. The coefficients $\mathbf{s}$ are further assumed to be L_1 sparse with a Laplacian prior:

$$p_s(s_i) \propto \exp(-\lambda |s_i|). \tag{3.45}$$

This probabilistic model can be described with the following energy function

$$E(A, \mathbf{s}, \mathbf{x}) = \frac{1}{2} \frac{||\mathbf{x} - A\mathbf{s}||_2^2}{\sigma^2} + \lambda ||\mathbf{s}||_1 \tag{3.46}$$

such that

$$p(\mathbf{x}, \mathbf{s}|A) = Z^{-1} \exp(-E(A, \mathbf{s}, \mathbf{x})) \tag{3.47}$$

Note that conveniently, the partition function $Z = (\sqrt{8\pi}\sigma/\lambda)^D$ and is a constant, specifically independent of A. This yields the nice property that

$$\nabla_A p(\mathbf{x}, \mathbf{s}|A) = -\nabla_A E(A, \mathbf{s}, \mathbf{x}) \cdot p(\mathbf{x}, \mathbf{s}|A). \tag{3.48}$$

Our goal is to solve for the maximum likelihood estimator (MLE) of the dictionary

$$A^* = \arg\max_A \langle \log p(\mathbf{x}|A) \rangle_{\mathbf{x} \sim \mathcal{D}}. \tag{3.49}$$

The expectation $\langle\cdot\rangle_{\mathbf{x}\sim\mathcal{D}}$ denotes expectation over dataset $\mathcal{D}$ (e.g. natural images). The MLE can be found through gradient descent. Where the gradient is

$$-\nabla_A \langle \log p(\mathbf{x}|A)\rangle = -\left\langle \frac{\nabla_A p(|A)}{p(\mathbf{x}|A)} \right\rangle \tag{3.50}$$

$$= -\left\langle \frac{\nabla_A \int d\mathbf{s}\ p(\mathbf{x},\mathbf{s}|A)}{p(\mathbf{x}|A)} \right\rangle \tag{3.51}$$

$$= -\left\langle \int d\mathbf{s}\ \nabla_A E(A,\mathbf{s},\mathbf{x}) \cdot \frac{p(\mathbf{x},\mathbf{s}|A)}{p(\mathbf{x}|A)} \right\rangle \tag{3.52}$$

$$= -\left\langle \int d\mathbf{s}\ \nabla_A E(A,\mathbf{s},\mathbf{x}) \cdot p(\mathbf{s}|\mathbf{x},A) \right\rangle \tag{3.53}$$

$$= -\left\langle \int d\mathbf{s}\ (A\mathbf{s}-\mathbf{x})\mathbf{s}^T \cdot p(\mathbf{s}|\mathbf{x},A) \right\rangle. \tag{3.54}$$

In practice, the expectation of the data is approximated with the empirical mean of a sample $\{\mathbf{x}_n\}_{n=1\ldots N}$ from dataset D. So we have

$$\Delta A = \frac{1}{\sigma^2}\frac{1}{N}\sum_{n=1}^{N}\int d\mathbf{s}_n\ p(\mathbf{s}_n|\mathbf{x}_n)\cdot(A\mathbf{s}_n-\mathbf{x}_n)\mathbf{s}_n^T. \tag{3.55}$$

The integration over $\mathbf{s}_n$ is also intractable and is approximated by the maximum *a posteriori* (MAP) estimator of $\mathbf{s}_n$

$$\mathbf{s}_n^* = \arg\min_{\mathbf{s}_n} E(A,\mathbf{s}_n,\mathbf{x}_n). \tag{3.56}$$

This can also be found via gradient descent with updates

$$\Delta\mathbf{s}_n = \nabla_{s_n} E(A,\mathbf{s}_n,\mathbf{x}_n). \tag{3.57}$$

Both updates ΔA and $\Delta\mathbf{s}_n$ can be expressed more efficiently through gradient descent on a batch energy function:

$$E(A,S,X) \equiv \sum_n E(A,\mathbf{s}_n,\mathbf{x}_n) \tag{3.58}$$

$$= \sum_n \frac{1}{2}\frac{||A\mathbf{s}_n-\mathbf{x}_n||_2^2}{\sigma^2} + \lambda||\mathbf{s}_n||_1 \tag{3.59}$$

$$= \frac{1}{2}\frac{||AS-X||_F^2}{\sigma^2} + \lambda||S||_{1,1}. \tag{3.60}$$

We have defined matrices $S\in\mathbb{R}^{K\times N}$ and $X\in\mathbb{R}^{D\times N}$. Recall that $A\in\mathbb{R}^{D\times K}$ with D being the dimension of the the data, K the number of dictionary bases and N the batch size of X. Above, $||\cdot||_F$ and $||\cdot||_{1,1}$ are the Frobenius and $(1,1)$ matrix norms respectively.

With the batch energy defined,

$$S \leftarrow S - \eta_s \frac{\partial E(A,S,X)}{\partial S} \tag{3.61}$$

$$A \leftarrow A - \eta_A \frac{1}{N} \frac{\partial E(A,S,X)}{\partial A}. \tag{3.62}$$

Note that we can redefine η_A to be η_A/N for simplicity. To coordinate the two updates, a nested loop must be used (Alg. 3). The inner loop approximates the MAP estimator S^* while the outer loop finds the MLE of A.

Algorithm 3 Algorithm for discrete sparse coding. Note line 8 was included purely to emphasize s^* as an estimate of the MAP.

```
1: for $k \leftarrow 1$ to $N_A$ do
2:     for $n \leftarrow 1$ to $N_s$ do
3:         $X \leftarrow$ SampleBatch()
4:         $S \leftarrow S - \eta_S \cdot \frac{\partial E}{\partial S}(A,S,X)$
5:     end for
6:     $S^* \leftarrow S$        ▷ Included to emphasize that we have obtained a MAP estimator of $S$
7:     $A \leftarrow A - \eta_A \frac{\partial E}{\partial A}(A,S^*,X)$
8: end for
```

3.B Locally Competitive Algorithm

The locally competitive algorithm (LCA) is another method for training sparse coding which solves a large set of similar problems that LSC addresses[64]. Explicitly, it is suitable for parallel analog implementation and that it achieves L_0 sparsity. In the context of LCA, The thresholding function in Eq. 3.27 is known as a soft threshold. In this section, we provide a brief introduction to LCA with a soft threshold and compare LCA to LSC.

The dynamics of LCA is given by

$$\dot{\mathbf{u}} \propto -A^T(A\mathbf{s} - \mathbf{x}) - (\mathbf{u} - \mathbf{s}). \tag{3.63}$$

This has many similarities to the LSC dynamics:

$$\dot{\mathbf{u}} \propto -A^T(A\mathbf{s} - \mathbf{x}) \cdot \mathbf{1}(|\mathbf{u}| > \mathbf{u}_0) - \lambda \cdot \mathrm{sgn}(\mathbf{u}) + \sqrt{2T}\xi(t). \tag{3.64}$$

Specifically, for $T = 0$ and $u_i > u_0$ we have

$$\dot{u}_i^{LCA} \propto -\left[A^T(A\mathbf{s} - \mathbf{x})\right]_i - u_0. \tag{3.65}$$

$$\dot{u}_i^{LSC-0T} \propto -\left[A^T(A\mathbf{s} - \mathbf{x})\right]_i - \lambda. \tag{3.66}$$

$$\tag{3.67}$$

In taking $u_0 = \lambda$, the two algorithms will have identical dynamic in this region. However for $|u_i| < u_0$, the two differs significantly.

$$\dot{u}_i^{LCA} \propto -\left[A^T(A\mathbf{s} - \mathbf{x})\right]_i - u_i. \tag{3.68}$$

$$\dot{u}_i^{LSC-0T} \propto -0 - \lambda. \tag{3.69}$$

$$\tag{3.70}$$

In this case, the LSC algorithm will quickly converge to $u_i \to 0$ since the reconstruction loss is no longer relevant. This is rectified through the noise term $\xi(t)$ with non-zero temperature T.

In addition to the stochasticity, the LSC dynamics is also derived from steepest descent of its associated energy function. This, by design, is not true for LCA. In other words, the LCA flow field is not conservative. The fact that LSC has an associated energy functions allows us to sample from the exact prior distribution through Langevin Dynamics.

If we assumed $\dot{\mathbf{u}} = -\nabla E_{LCA}$ for some energy E_{LCA}, we have

$$\frac{\partial^2 E_{LCA}}{\partial u_1} \equiv \partial_1 E_{LCA} = (A^TA)_{11}f(u_1) + (A^TA)_{12}f(u_2) + \cdots - A^T_{11}x_1 - A^T_{12}x_2 - \cdots + u_1 - f(u_1) \tag{3.71}$$

$$\partial_2\partial_1 E_{LCA} = (A^TA)_{12}f'(u_2). \tag{3.72}$$

However,

$$\partial_1\partial_2 E_{LCA} = (A^TA)_{21}f'(u_1) \neq \partial_2\partial_1 E_{LCA} \tag{3.73}$$

unless $f'(u_1) = f'(u_2)$. Note that this is not an issue with LSC where

$$\partial_1\partial_2 E_{LSC} = (A^TA)_{21}f'(u_1)f'(u_2) = (A^TA)_{12}f'(u_2)f'(u_1) = \partial_2\partial_1 E_{LSC}. \tag{3.74}$$

Of course, this should be the case by construction.

3.C DSC, LSC, Asynch, LSC

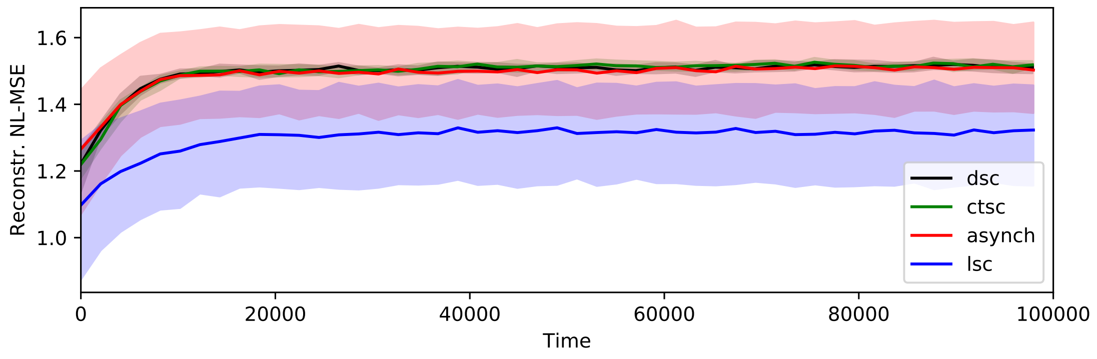
in Fig. 3.C.1, we see the evolution of the four algorithms when trained with the Bars dataset. The mean squared reconstruction error (MSE) was calculated. Specifically

$$\epsilon_{\text{recon}} = \langle ||A\mathbf{s} - \mathbf{x}||_2^2 \rangle. \tag{3.75}$$

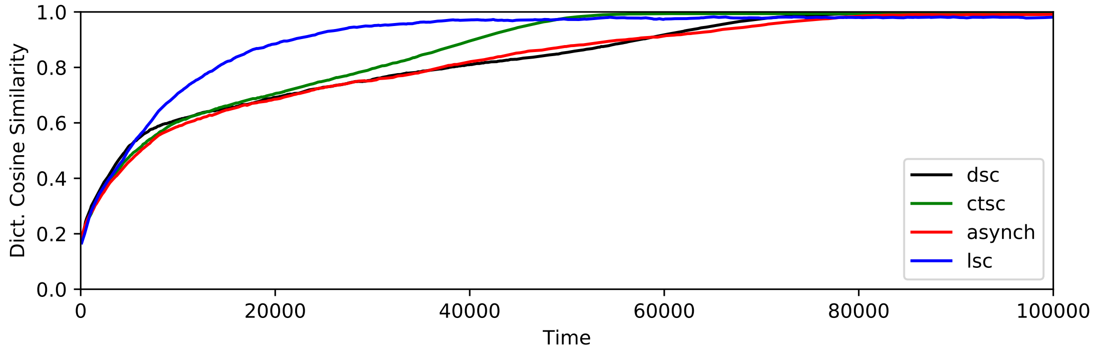
The mean is taken over the batch of inputs and reconstructions as well as a small interval of time. We then took the negative log MSE (NL-MSE) as a metric of reconstruction fidelity. Up to a constant, NL-MSE is equal to the peak signal to noise ratio PSNR.

Over the dynamics of each algorithm, their respective NL-MSE is plotted in Fig. 3.C.1. We notice similar rates of convergence for DSC and CTSC and Async. The higher variance associated with Asynch can be explained by the variable time for which each input is presented. LSC by contrast has a lower NL-MSE with significantly higher variance. This result is, however, expected as in contrast to the other algorithms, LSC does not return a single MAP estimate. Instead, the dynamics samples the posterior distribution.

Indeed, all four algorithms does successfully the correct dictionaries (Fig. 3.5.1(a)). The mean cosine similarity of the learned dictionary with the correct dictionary is plotted in (Fig. 3.C.1). We also see similar rates of convergence.



((a)) Negative log MSE



((b)) Mean cosine similarity between learned dictionary and true dictionary.

Fig. 3.C.1. Two metrics demonstrating the convergence rates for each algorithm in training with the Bars dataset. a) Negative log MSE – a metric of reconstruction fidelity. The median along with the 20%-80% range are plotted. b) The cosine similarity of learned dictionary to the “true” dictionary.

3.D Numerical Simulation of Langevin Dynamics

In simulating Eq. **??**, we use the Euler-Maruyama method. First, for $\mu = 0$, we have

$$dx = -\nabla V du + \sqrt{2T du} \cdot \eta(u) \tag{3.76}$$

$$x \leftarrow x + dx \tag{3.77}$$

where du is a suitably small time step and $\eta(u) \overset{iid}{\sim} N(0,1)$. In the case of $\mu > 0$, we can solve the second order equation as two first order equations with a dimensionless momentum defined as

$$\pi \equiv \mu\tau\dot{x}. \tag{3.78}$$

Then, we have

$$dx = \frac{\pi}{\mu} du \tag{3.79}$$

$$d\pi = -\frac{\pi}{\mu} du - \nabla V du + \sqrt{2T du} \cdot \eta(u) \tag{3.80}$$

$$x \leftarrow x + dx \tag{3.81}$$

$$\pi \leftarrow \pi + d\pi. \tag{3.82}$$

For better numerical stability, we may choose to do

$$x \leftarrow dx/2 \tag{3.83}$$

$$\pi \leftarrow d\pi \tag{3.84}$$

$$x \leftarrow dx/2. \tag{3.85}$$

3.E Circuit Implementation

Resistor cross-grid

We seek to implement an optimizer for both objective functions.

Matrix Inversion

As a starting point, it has been demonstrated that an analog circuit can be used to invert a matrix G. Given current $\mathbf{I} = GV$, it can be used to solve for V, (see Fig. 3.E.1)

Note that the op amps act as virtual grounds while preventing current from passing through.

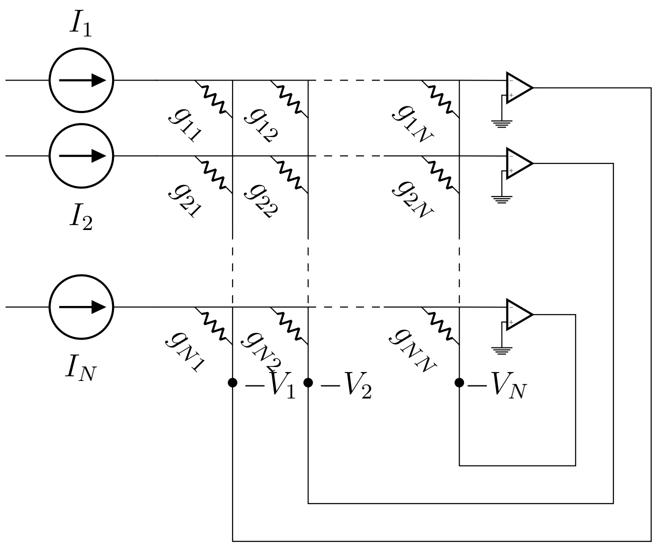


Fig. 3.E.1. Circuit for matrix inversion

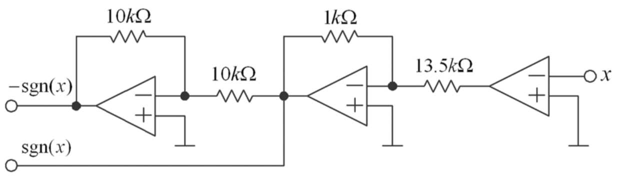


Fig. 3.E.2. Circuit realization of op amp (Tang, 2008).

L_1-SC Analog Circuit

Taking the derivative of the energy, we derive as solutions the following equalities

$$\nabla E_1(\mathbf{s}) = A^T(A\mathbf{s} - \mathbf{x}) + \lambda \cdot \operatorname{sgn}(s) \tag{3.86}$$

$$\equiv G\mathbf{s} - \mathbf{y} + \lambda \cdot (s) \tag{3.87}$$

$$G\mathbf{s} = \mathbf{y} - \lambda \cdot (s) \tag{3.88}$$

where $G = A^T A$ and $\mathbf{y} = A^T\mathbf{x}$. Given y, G, we wish to solve for sparse coefficients s wich satisfy the condition above. This is very similar to the matrix inversion problem above with the modification of an additional sign function. A circuit realization of a sign function is achievable with three op amps (Fig. 3.E.2). To see how the matrix inversion matrix can be modified to include the $(\mathbf{s})$ term, the reduced circuit in Fig. 3.E.3 illustrates the idea with

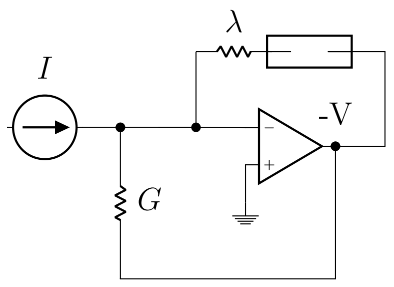


Fig. 3.E.3. Sign function modification

a single resistor G (i.e. $N = 1$). Here the total current I is split between I_G and I_λ.

$$I = GV + \lambda \cdot (V) \tag{3.89}$$

$$GV = I - \lambda \cdot (V). \tag{3.90}$$

Modifying all of the op amps in the full $N \times N$ circuit, gives the desired result.

L_0-SC Circuit

To implement the L_0 sparse prior, we modify the condition to

$$Gs = y - \lambda \cdot (u) \tag{3.91}$$

where $s = f(u)$ and u is taken to be the internal state. Note that f can be modeled as two ReLU functions

$$f(u) = \text{ReLU}(u - u_0) - \text{ReLU}(-u - u_0) \tag{3.92}$$

and CMOS implementations of ReLU functions exist and have been demonstrated (see Fig. 3.E.4). The threshold function can be easily implemented by combing two biased ReLU in parallel (and in different directions). We can then modify op amp again (this time the lower lead), see Fig. 3.E.5. Here, the relevant voltage to the resistors G and determining the current is $f(V)$ and we have again the desired result. Note that true to its interpretation, the voltage V, representing u now, is indeed "hidden" from the resistor array.

3.F Coupling Function DSC to CTSC

To connect CTSC to DSC, we can define a τ_x-periodic coupling function

$$\gamma(\tilde{t}) = \frac{\beta\tau_x}{1 - e^{-\beta\tau_x}} e^{\beta\tilde{t}} \tag{3.93}$$

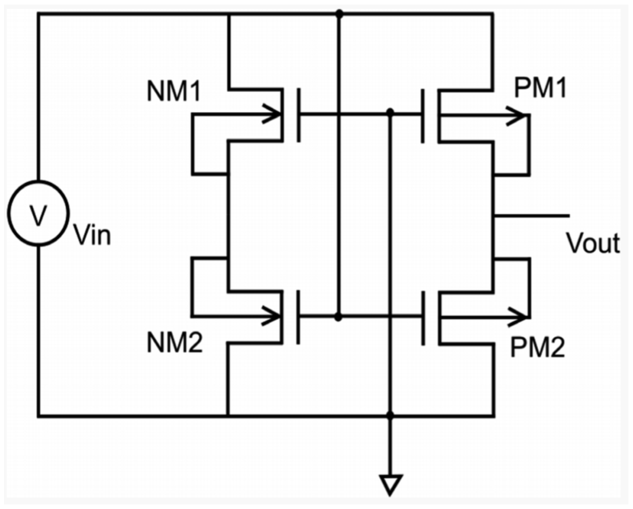


Fig. 3.E.4. CMOS implementation of ReLU

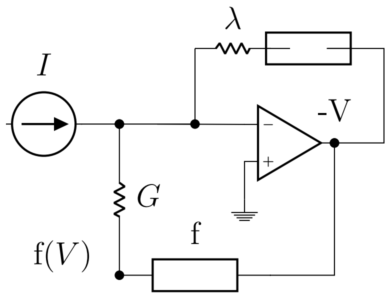


Fig. 3.E.5. Sign function and threshold modification

where $\tilde{t} \equiv (t - \tau_x) \mod \tau_x$. Note that from this definition,

$$\int_0^{\tau_x} d\tilde{t}\gamma(\tilde{t}) = \tau_x = \int_0^{\tau_x} d\tilde{t}. \tag{3.94}$$

Now, we modify the dynamics of A to be

$$\tau_A \cdot \dot{A} = -\gamma(t)\nabla_A E(A, S, X(t)). \tag{3.95}$$

The coupling γ weighs the dynamics of A to be more significant at later times. In taking the limit $\beta \to \infty$, we end up with

$$\tau_A \cdot \dot{A} = \tau_x \delta(\tilde{t})\nabla_A E(A, S, X(t) \tag{3.96}$$

and A is only updated when $t \equiv 0 \mod \tau_x$ with

$$\Delta A = \eta_A \nabla_A E(A, S, X(t)). \tag{3.97}$$

In the opposite limit $\beta \to 0$, we recover

$$\tau_A \dot{A} = -\nabla_A E(A, S, X(t)). \tag{3.98}$$

3.G Convergence Properties

(Dalalyan, 2016) [17] derived convergence gaurantees for Lagevin sampling if the energy is strongly convex with Lipschitz continuous gradient. That is

$$E(\theta + \delta\theta) - E(\theta) - \nabla E(\theta) \cdot \delta\theta \geq \frac{m}{2} ||\delta\theta||^2; \tag{3.99}$$

$$||\nabla E(\theta + \delta\theta) - \nabla E(\theta)|| \leq M ||\delta\theta|| \tag{3.100}$$

for some positive constants m and M. The informal result is that the Langevin dynamics has a characteristic time scale of

$$\tau = m^{-1}. \tag{3.101}$$

Furthermore, in simulating Langevin dynamics with the discrete algorithm, we require that the time step be

$$\Delta t < M^{-1}. \tag{3.102}$$

To demonstrate this, consider a quadratic energy of the form

$$E(s) = 0.5||As||^2. \tag{3.103}$$

This, of course, corresponds to a normal distribution with covariance matrix

$$\Sigma = (A^T A)^{-1}. \tag{3.104}$$

It can be shown that in this case, the constants are

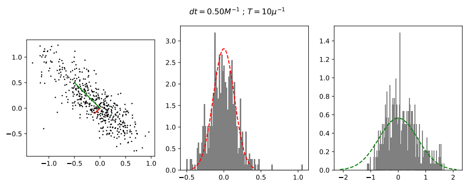
$$M = \sigma_{\min}^{-2}; \tag{3.105}$$

$$m = \sigma_{\max}^{-2} \tag{3.106}$$

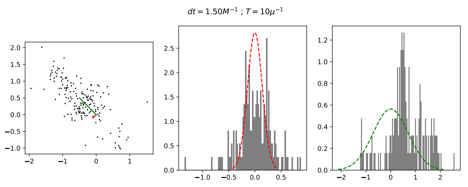
where $\sigma_{\max}^2$ is the largest uncorrelated variance and $\sigma_{\min}^2$, the smallest. This gives us

$$\Delta t < \sigma_{\min}^2; \tag{3.107}$$

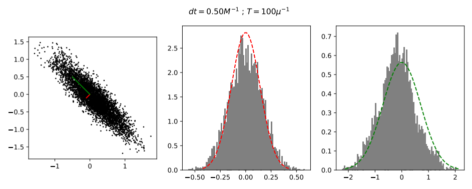
$$\tau = \sigma_{\max}^2. \tag{3.108}$$



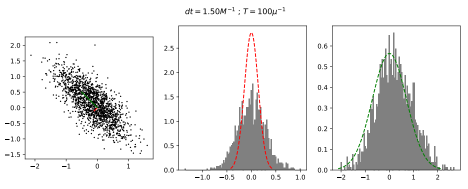
((a))



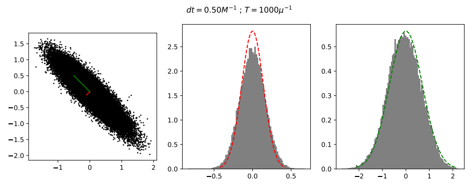
((b))



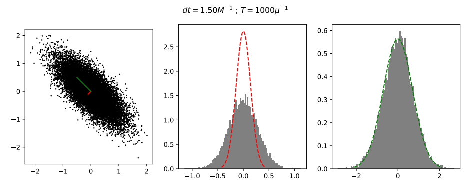
((c))



((d))



((e))



((f))

Fig. 3.G.1. With increasing time (a, c, f), the position converges to the desired distribution if the step size is small enough. When the simulation step size is too large, (b, d, f), the dynamics will not converge.

3.H Proof for L0 prior

Take $\mathbf{s} = f(\mathbf{u})$ with k elements zero. That is, without loss of generality,

$$\mathbf{s} = (0, \ldots, 0, s_{k+1}, \ldots, s_N). \tag{3.109}$$

We must then have

$$u_i \in \begin{cases} (-u_0, u_0) & \text{if} \quad 1 \leq i \leq k \\ \{s_i - u_0 \cdot \operatorname{sgn}(s_i)\} & \text{if} \quad k < i \leq N. \end{cases} \tag{3.110}$$

Then, the conditional probability of $f(\mathbf{u}) = \mathbf{s}|\mathbf{x}$ is

$$\begin{aligned} p(\mathbf{x}, f(\mathbf{u}) = \mathbf{s}) &= \int_{-u_0}^{u_0} du_1 \cdots du_k \; p_R(\mathbf{x}|\mathbf{s}) \cdot \prod_i \frac{1}{2\lambda} e^{-\lambda |u_i|} & (3.111) \\ &= p_R(\mathbf{x}|\mathbf{s}) \cdot \prod_{i=k+1}^{N} \frac{1}{2\lambda} e^{-\lambda |s_i| - \lambda u_0} \cdot \left(\int_{-u_0}^{u_0} du_i \; \frac{e^{-\lambda u_i}}{2\lambda} \right)^k & (3.112) \\ &= p_R(\mathbf{x}|\mathbf{s}) \prod_{i=1}^{N} e^{-\lambda s_i} \left(e^{-\lambda u_0} \right)^{N-k} \left(1 - e^{-\lambda u_0} \right)^k & (3.113) \\ &= p_R(\mathbf{x}|\mathbf{s}) p_S(\mathbf{s}) \cdot \pi^k (1-\pi)^{N-k}. & (3.114) \end{aligned}$$

We can easily interpret the final term as a $L0$ sparsity prior on $\mathbf{s}$.

Note that using MAP estimation, the estimator u^* will always be zero due to the delta functions with infinite height albeit with finite mass.

3.I Quantifying convergence to prior

To better quantify the convergence to the desired prior, we estimate the KL-divergence from $p(s_i|\lambda)$, the target prior to $p(s_i|A)$ the learned prior based on dictionary A. Because the learned prior cannot be easily calculated, we rely on samples taken at regular time intervals. The samples are then binned in the same way that generated the histograms in (Fig. 3.5.3) .

$$\begin{aligned} D_{KL}(p(s|\lambda)||p(s|A)) &= \left\langle \log \left(\frac{p(s|\lambda)}{p(s|A)} \right) \right\rangle_{s|\lambda} & (3.115) \\ &\approx \sum_n p_n(\lambda) \log \left(\frac{p_n(\lambda)}{q_n(A)} \right) & (3.116) \end{aligned}$$

where

$$p_n(\lambda) = P(n\delta s < s < (n+1)\delta s) \qquad (3.117)$$

with δs being the bin width. Figure 3.I.1 shows the evolution of the estimated D_{KL} over time. As expected, only with LSC does the KL-divergence approach 0.

3.J Learned Dictionary from Natural Scenes

Dictionary elements learned from the Van Hateren dataset is presented in Fig. ??. Only a random selection of 64 dictionary elements are shown

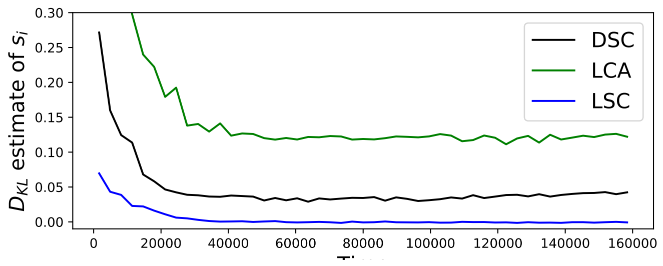
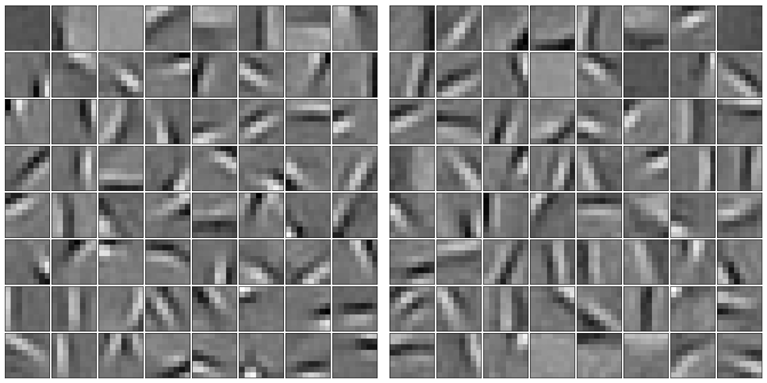


Fig. 3.I.1. The KL-Divergence for coefficients s_i is compared for each of the three sparse coding methods. Only with LSC, does the D_{KL} approach zero.



((a)) Complete dictionary; $\pi \approx 0.16$

((b)) $2\times$ overcomplete dictionary; $\pi \approx 0.08$

3.K Convergence Tests

In using LSC to infer the dictionary element norms, we show in the sections various experiments done to explore the convergence towards a known dictionary under various hyperparameters.

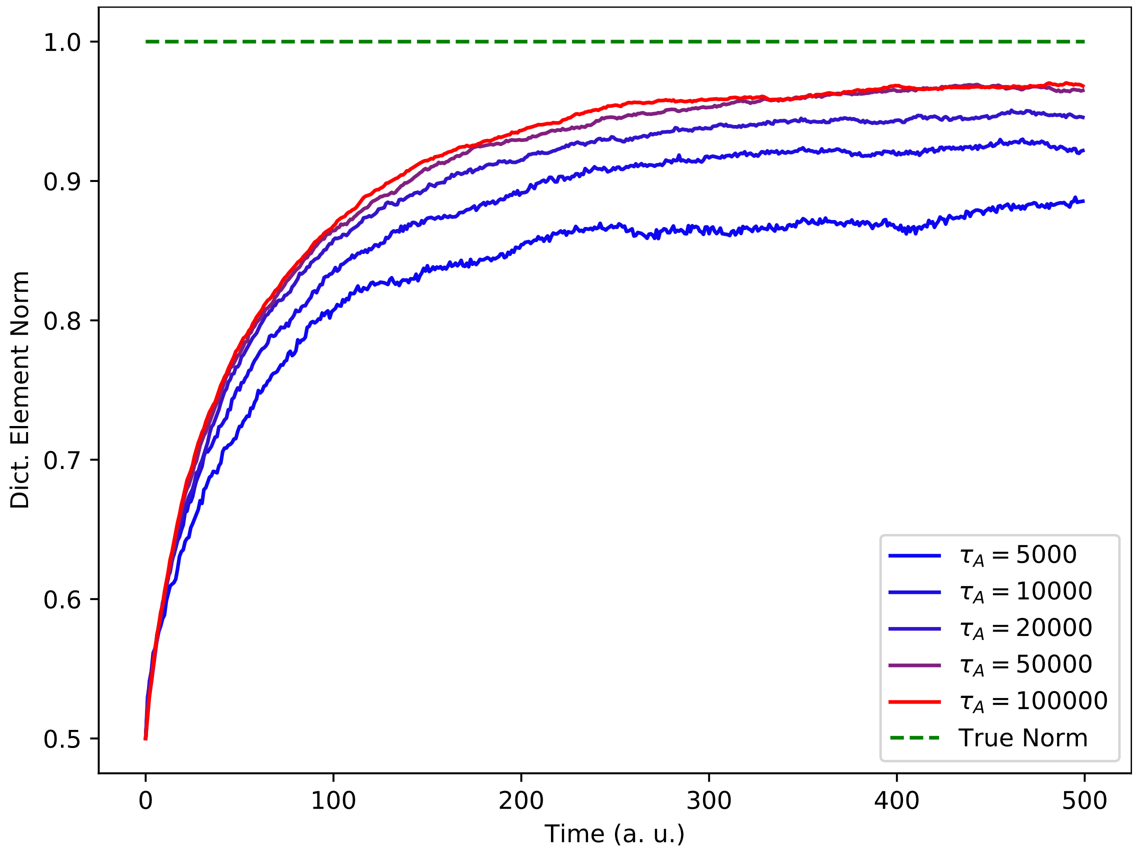


Fig. 3.K.1. The convergence towards the true norm of 1 is shown for various values of τ_A. As the ratio of τ_A to τ_s becomes larger, LSC does a better job at inference.

Chapter 4

Conclusion

A variety of ideas from physics have been shown to have applicability in machine learning. Specifically relevant to the work presented in this thesis are training Hopfield networks with MPF[31], optical neural networks[69], and analog sampling[83, 63]. These results were extended in this thesis.

In summary, alternative objective functions for MPF were first presented along with an efficient method for training higher order Hopfield networks. In the next chapter, the effects of fabrication error on the performance of ONNs were presented with architectural designs mitigating these adverse effects. Lastly, A fully analog framework for training latent variable models was developed with specific implementation applied to sparse coding for natural images.

Due to the interdisciplinary nature of the work, there is much room for further extension. Beyond using MPF to train higher order Hopfield networks on random data, it would be important to see application in solving specific problems. One example would be discovering temporal structure in time-series data, previously demonstrated with second order Hopfield energy[30].

The study of robustness of optical neural networks was focused primarily on optimization of architectural designs of ONNs. A more interesting follow up would involve an algorithmic approach in minimizing the fabrication error present. Robustness of neural networks to noise in input data is already a widely studied topic[21], generalizing the work to account for noise in the network weights would be one approach.

By comparison, a more complete study of the Langevin Sparse Coding (LSC) algorithm was presented. However, a concrete physical implementation needs to be developed. In the absence of such physical system, LSC is idealized and more accurate modeling of the noise present in specifc analog devices would be necessary.

While the topics studied are diverse, they were all inspired by a more probabilistic treatment of existing problems. MPF is closely related to contrastive divergence. Instead of sampling, however, the objective function is built from analytic manipulation of the conditional distribution. Furthermore, making connections between the deterministic Hopfield network to the probabilistic Ising model allowed MPF to be used for training. Then, in the case of optical neural networks, while previous acknowledged, a detailed treatment of the effect of fabrication error on the network performance was yet to be done. This understanding led to an alternative architectural design that is significantly more robust to such errors. In the case of sparse coding, while a inherently a probabilistic model, point estimates typically replace sampling in most implementations due to the intractability of sampling on digital computers. By utilizing the noise in analog systems, both sampling and training can be achieved.

There is great value in continuing in this direction of studying machine learning problems as stochastic, physical systems. Beyond further contributions to both physics and machine learning, a better understanding of analog computation for machine learning is especially useful for modeling computation in biology.

Bibliography

[1] Lennert Appeltant et al. “Information processing using a single dynamical node as complex system”. In: *Nature communications* 2 (2011), p. 468.

[2] Martin Arjovsky, Amar Shah, and Yoshua Bengio. “Unitary evolution recurrent neural networks”. In: *International Conference on Machine Learning*. 2016, pp. 1120–1128.

[3] Masoud Babaeian et al. “Nonlinear optical components for all-optical probabilistic graphical model”. In: *Nature communications* 9.1 (2018), p. 2128.

[4] Qiaoliang Bao et al. “Monolayer graphene as a saturable absorber in a mode-locked laser”. In: *Nano Research* 4.3 (2011), pp. 297–307.

[5] Ronen Barak and Yacob Ben-Aryeh. “Quantum fast Fourier transform and quantum computation by linear optics”. In: *JOSA B* 24.2 (2007), pp. 231–240.

[6] Pietro Berkes, Richard Turner, and Maneesh Sahani. “On sparsity and overcompleteness in image models”. In: *Advances in neural information processing systems*. 2008, pp. 89–96.

[7] Pietro Berkes et al. “Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment”. In: *Science* 331.6013 (2011), pp. 83–87.

[8] Roel Burgwal et al. “Using an imperfect photonic network to implement random unitaries”. In: *Optics Express* 25.23 (2017), pp. 28236–28245.

[9] Giovanni Bussi and Michele Parrinello. “Accurate sampling using Langevin dynamics”. In: *Physical Review E* 75.5 (2007), p. 056707.

[10] Julie Chang et al. “Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification”. In: *Scientific reports* 8.1 (2018), p. 12324.

[11] Leon Chua. “Memristor-the missing circuit element”. In: *IEEE Transactions on circuit theory* 18.5 (1971), pp. 507–519.

[12] William R Clements et al. “Optimal design for universal multiport interferometers”. In: *Optica* 3.12 (2016), pp. 1460–1465.

[13] Michael J Connelly. *Semiconductor optical amplifiers*. Springer Science & Business Media, 2007.

[14] James W Cooley and John W Tukey. “An algorithm for the machine calculation of complex Fourier series”. In: *Mathematics of computation* 19.90 (1965), pp. 297–301.

[15] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. New York, NY, USA: Wiley-Interscience, 2006. ISBN: 0471241954.

[16] Imre Csiszár. “Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten”. In: *Magyer Tud. Akad. Mat. Kutato Int. Koezl.* 8 (1964), pp. 85–108.

[17] Arnak S Dalalyan. “Theoretical guarantees for approximate sampling from smooth and log-concave densities”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 79.3 (2017), pp. 651–676.

[18] Charles Dugas et al. “Incorporating second-order functional knowledge for better option pricing”. In: *Advances in neural information processing systems*. 2001, pp. 472–478.

[19] Micael Y.-S. Fang. *Imprecise Optical Neural Networks*. `https://github.com/mike-fang/imprecise_optical_neural_network`. Mar. 2019.

[20] Nabil H Farhat et al. “Optical implementation of the Hopfield model”. In: *Applied optics* 24.10 (1985), pp. 1469–1475.

[21] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. “Robustness of classifiers: from adversarial to random noise”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 1632–1640.

[22] Fulvio Flamini et al. “Benchmarking integrated linear-optical architectures for quantum information processing”. In: *Scientific Reports* 7.1 (2017), p. 15133.

[23] Fulvio Flamini et al. “Thermally reconfigurable quantum photonic circuits at telecom wavelength by femtosecond laser micromachining”. In: *Light: Science & Applications* 4.11 (2015), e354.

[24] Pierre Garrigues and Bruno A Olshausen. “Group sparse coding with a laplacian scale mixture prior”. In: *Advances in neural information processing systems*. 2010, pp. 676–684.

[25] Rafael R Gattass and Eric Mazur. “Femtosecond laser micromachining in transparent materials”. In: *Nature photonics* 2.4 (2008), p. 219.

[26] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT Cambridge, 2016.

[27] Nicholas C Harris et al. “Quantum transport simulations in a programmable nanophotonic processor”. In: *Nature Photonics* 11.7 (2017), p. 447.

[28] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[29] J. H. van Hateren and A. van der Schaaf. “Independent Component Filters of Natural Images Compared with Simple Cells in Primary Visual Cortex”. In: *Proceedings: Biological Sciences* 265.1394 (Mar. 1998), pp. 359–366.

[30] Christopher Hillar and Felix Effenberger. “Robust discovery of temporal structure in multi-neuron recordings using Hopfield networks”. In: *Procedia Computer Science* 53 (2015), pp. 365–374.

[31] Christopher Hillar, Jascha Sohl-Dickstein, and Kilian Koepsell. “Efficient and optimal binary Hopfield associative memory storage using minimum probability flow”. In: *arXiv preprint arXiv:1204.2916* (2012).

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

[33] Patrik O Hoyer and Aapo Hyvärinen. “Interpreting neural response variability as Monte Carlo sampling of the posterior”. In: *Advances in neural information processing systems*. 2003, pp. 293–300.

[34] Itay Hubara et al. “Quantized neural networks: Training neural networks with low precision weights and activations”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6869–6898.

[35] Yunshan Jiang, Peter TS DeVore, and Bahram Jalali. “Analog optical computing primitives in silicon photonics”. In: *Optics letters* 41.6 (2016), pp. 1273–1276.

[36] Li Jing et al. “Tunable efficient unitary neural networks (EUNN) and their application to RNNs”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1733–1741.

[37] Iakov Karandashev, Boris Kryzhanovsky, and Leonid Litinskii. “Properties of the hopfield model with weighted patterns”. In: *International Conference on Artificial Neural Networks*. Springer. 2012, pp. 9–16.

[38] Kazuro Kikuchi. “Characterization of semiconductor-laser phase noise and estimation of bit-error rate performance with low-speed offline digital coherent receivers”. In: *Optics Express* 20.5 (2012), pp. 5291–5302.

[39] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[40] MC Larson et al. “Narrow linewidth high power thermally tuned sampled-grating distributed Bragg reflector laser”. In: *2013 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*. IEEE. 2013, pp. 1–3.

[41] Yann LeCun. “The MNIST database of handwritten digits”. In: *http://yann.lecun.com/exdb/mnist/* ().

[42] Tai Sing Lee and David Mumford. "Hierarchical Bayesian inference in the visual cortex". In: *JOSA A* 20.7 (2003), pp. 1434–1448.

[43] Yangjin Ma et al. "Ultralow loss single layer submicron silicon waveguide crossing for SOI optical interconnect". In: *Optics express* 21.24 (2013), pp. 29374–29382.

[44] Vikash K Mansinghka, Eric M Jonas, and Joshua B Tenenbaum. "Stochastic digital circuits for probabilistic inference". In: *Massachussets Institute of Technology, Technical Report MITCSAIL-TR* 2069 (2008).

[45] ROBERTJ McEliece et al. "The capacity of the Hopfield associative memory". In: *IEEE transactions on Information Theory* 33.4 (1987), pp. 461–482.

[46] David AB Miller. "Perfect optics with imperfect components". In: *Optica* 2.8 (2015), pp. 747–750.

[47] David AB Miller. "Silicon photonics: Meshing optics with applications". In: *Nature Photonics* 11.7 (2017), p. 403.

[48] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[49] Radford M Neal. "Annealed importance sampling". In: *Statistics and computing* 11.2 (2001), pp. 125–139.

[50] Dmitri E Nikonov et al. "Coupled-oscillator associative memory array operation for pattern recognition". In: *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 1 (2015), pp. 85–93.

[51] Bruno A Olshausen. "Highly overcomplete sparse coding". In: *Human Vision and Electronic Imaging XVIII*. Vol. 8651. International Society for Optics and Photonics. 2013, 86510S.

[52] Bruno A Olshausen and David J Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (1996), pp. 607–609.

[53] Bruno A Olshausen and David J Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision research* 37.23 (1997), pp. 3311–3325.

[54] Bruno A Olshausen and K Jarrod Millman. "Learning sparse codes with a mixture-of-Gaussians prior". In: *Advances in neural information processing systems*. 2000, pp. 841–847.

[55] Gergő Orbán et al. "Neural variability and sampling-based probabilistic representations in the visual cortex". In: *Neuron* 92.2 (2016), pp. 530–543.

[56] Sunil Pai et al. "Matrix optimization on universal unitary photonic devices". In: *arXiv preprint arXiv:1808.00458* (2018).

[57] Giulia Panusa et al. “Photoinitiator-free multi-photon fabrication of compact optical waveguides in polydimethylsiloxane”. In: *Optical Materials Express* 9.1 (2019), pp. 128–138.

[58] Yvan Paquot et al. “Optoelectronic reservoir computing”. In: *Scientific reports* 2 (2012), p. 287.

[59] Adam Paszke et al. “Automatic differentiation in PyTorch”. In: *NIPS-W*. 2017.

[60] Mohammad Rastegari et al. “Xnor-net: Imagenet classification using binary convolutional neural networks”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 525–542.

[61] Michael Reck et al. “Experimental realization of any discrete unitary operator”. In: *Physical review letters* 73.1 (1994), p. 58.

[62] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *Herbert Robbins Selected Papers*. Springer, 1985, pp. 102–109.

[63] Charles Roques-Carmes et al. “Photonic recurrent Ising sampler”. In: *CLEO: QELS_Fundamental Science*. Optical Society of America. 2019, FTu4C–2.

[64] Christopher J Rozell et al. “Sparse coding via thresholding and local competition in neural circuits”. In: *Neural computation* 20.10 (2008), pp. 2526–2563.

[65] Nicholas J Russell et al. “Direct dialling of Haar random unitary matrices”. In: *New Journal of Physics* 19.3 (2017), p. 033007.

[66] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. “Restricted Boltzmann machines for collaborative filtering”. In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 791–798.

[67] AC Selden. “Pulse transmission through a saturable absorber”. In: *British Journal of Applied Physics* 18.6 (1967), p. 743.

[68] Jacquelyn A Shelton et al. “Select and sample-a model of efficient neural inference and learning”. In: *Advances in neural information processing systems*. 2011, pp. 2618–2626.

[69] Yichen Shen et al. “Deep learning with coherent nanophotonic circuits”. In: *Nature Photonics* 11.7 (2017), p. 441.

[70] Patrick M Sheridan et al. “Sparse coding with memristor networks”. In: *Nature nanotechnology* 12.8 (2017), p. 784.

[71] Patrice Y Simard, Dave Steinkraus, and John C Platt. “Best practices for convolutional neural networks applied to visual document analysis”. In: *null*. IEEE. 2003, p. 958.

[72] Paul Smolensky. *Information processing in dynamical systems: Foundations of harmony theory*. Tech. rep. Colorado Univ at Boulder Dept of Computer Science, 1986.

[73] Jascha Sohl-Dickstein, Peter B Battaglino, and Michael R DeWeese. “New method for parameter estimation in probabilistic models: minimum probability flow”. In: *Physical review letters* 107.22 (2011), p. 220601.

[74] Jascha Sohl-Dickstein, Peter Battaglino, and Michael R DeWeese. “Minimum probability flow learning”. In: *arXiv preprint arXiv:0906.4779* (2009).

[75] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[76] Alexander N Tait et al. “Broadcast and weight: an integrated network for scalable photonic spike processing”. In: *Journal of Lightwave Technology* 32.21 (2014), pp. 3427–3439.

[77] Alexander N Tait et al. “Neuromorphic photonic networks using silicon photonic weight banks”. In: *Scientific Reports* 7.1 (2017), p. 7430.

[78] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation.* Vol. 89. siam, 2005.

[79] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[80] Chiheb Trabelsi et al. “Deep complex networks”. In: *arXiv preprint arXiv:1705.09792* (2017).

[81] Daniel F Walls and Gerard J Milburn. *Quantum optics.* Springer Science & Business Media, 2007.

[82] Li Wan et al. “Regularization of neural networks using dropconnect”. In: *International Conference on Machine Learning.* 2013, pp. 1058–1066.

[83] Tianshi Wang and Jaijeet Roychowdhury. “Oscillator-based ising machine”. In: *arXiv preprint arXiv:1709.08102* (2017).

[84] Max Welling and Yee W Teh. “Bayesian learning via stochastic gradient Langevin dynamics”. In: *Proceedings of the 28th international conference on machine learning (ICML-11).* 2011, pp. 681–688.

[85] Callum M Wilkes et al. “60 dB high-extinction auto-configured Mach–Zehnder interferometer”. In: *Optics letters* 41.22 (2016), pp. 5318–5321.

[86] Qianfan Xu and Michal Lipson. “Optical Bistability based on the Carrier dispersion effect in SOI Ring Resonators”. In: *Integrated Photonics Research and Applications.* Optical Society of America. 2006, p. IMD2.

[87] Greg Yang et al. “A mean field theory of batch normalization”. In: *arXiv preprint arXiv:1902.08129* (2019).