Spring 5-13-2024

Analysis and Computation of Constrained Sparse Coding on Emerging non-von Neumann Devices

Kyle Henke

Kyle Henke

Candidate

Mathematics and Statistics

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Mohammad Motamed, Chairperson

Frank Gilfeather

Robyn Miller

Ben Migliori

Andrew Sornborger

Jacob Schroder

Analysis and Computation of Constrained Sparse Coding on Emerging non-von Neumann Devices

by

Kyle Garde Henke
B.A., Economics, University of New Mexico, 2015
M.S., Mathematics, University of New Mexico, 2021

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
MATHEMATICS

The University of New Mexico
Albuquerque, NM

May, 2024

Dedication

To the infinite love felt on the fateful night that changed my life forever.

"The first gulp from the glass of natural sciences will turn you into an atheist, but at the bottom of the glass God is waiting for you." - Werner Heisenberg

Acknowledgements

Crafting an acknowledgment for my dissertation somehow emerges as the most challenging task of my academic voyage, underscoring the impossibility of adequately thanking all the individuals who have stood by me or before me.

Nevertheless, I must first acknowledge the generations who have past. To my ancestors and grandfathers I wasn't old enough to know, you laid the foundation upon which I stand. Your sacrifices, dreams, and struggles have not gone unnoticed. The path you paved through your own lives has allowed me to embark on this journey toward understanding and discovery. It is upon your shoulders that I stand, with deep respect of the legacy you've passed down. Your resilience and hope are the bedrock of my achievements, and for that, I am eternally indebted.

To my Mother, Sue, your unwavering support has been my constant. Aunt Sarah, the meticulous edits you and mom provided along with your passion for teaching mathematics were indispensable. To my sisters, Kayci and Karla, you are the epitome of unconditional love. To my father, Steve, your diverse motivational tactics have always spurred me on. My credit extends to Zach Stuart, Kellin Rumsey, and Gabriella Dalton for your fellowship, motivation, and support throughout my graduate studies and beyond. Dalton Robinson, a friend like no other, your presence in my toughest times and belief in surpassing limits continue to be invaluable. To my grandmothers, extended family, and in-laws, your countless acts of kindness were crucial to reaching this milestone.

An immense thanks to my advisor, Professor Mohammad Motamed. Your efforts to cultivate my passion for neuroscience and mathematics into something rigorous and tangible were truly a blessing. I am honored to have been your first PhD student. Professor Frank Gilfeather, your encouragement during my undergraduate struggles and ongoing support have been a beacon of hope and inspiration. My mentors at Los Alamos National Laboratory—Garrett Kenyon, Ben Migliori, and Andrew Sornborger—your patience and enthusiasm were instrumental in my growth to become a scientist. Robyn Miller, our conversations and your willingness to venture into new intellectual territories were vital.

Lastly, to my wife, Makenna, your love and support transcend all else, proving to be the most precious gift throughout this endeavor.

Thank you all for never giving up on me!

Analysis and Computation of Constrained Sparse Coding on Emerging non-von Neumann Devices

Kyle Garde Henke

B.A., Economics, University of New Mexico, 2015

M.S., Mathematics, University of New Mexico, 2021

Ph.D., Mathematics, University of New Mexico, 2024

Abstract

This dissertation seeks to understand how different formulations of the neurally inspired Locally Competitive Algorithm (LCA) represent and solve optimization problems. By studying these networks mathematically through the lens of dynamical and gradient systems, the goal is to discern how neural computations converge and link this knowledge to theoretical neuroscience and artificial intelligence (AI). Both classical computers and advanced emerging hardware are employed in this study. The contributions of this work include:

1. Theoretical Work: A comprehensive convergence analysis for networks using both generic Rectified Linear Unit (ReLU) and Rectified Sigmoid activation functions. Exploration of techniques to address the binary sparse optimization problem, especially when the problem landscape is non-convex. Non-autonomous systems with time-varying sigmoid activation that approaches the step function have been proposed due to the challenge of proving step function convergence.

2. Computational Work: Numerical tests on classical computers confirm the theoretical analysis. In mapping the problem to the spiking domain, it is shown spike rates can represent continuous valued neuron activations. The binary sparse optimization problem is reformulated into a Quadratic Unconstrained Binary Optimization (QUBO) problem. Solutions are then sought using quantum annealing and spiking neuromorphic devices.

Contents

Chapter 1

Introduction

The objective of this dissertation is to delve into the intricacies of dynamical and gradient systems, interpreted as special types of recurrent neural networks, for solving constrained sparse optimization problems. Specifically, we analyze Hopfield networks where neurons interact with each other and themselves to self organize into a representation of the problem's minimizer. By rigorously analyzing these networks from their mathematical principles, we aim to connect these concepts together and shed light on convergence behavior of neural computations in the broader contexts of theoretical neuroscience and artificial intelligence. We proceed with implementation on classical computers to verify the analysis and then solve on spiking neural network and quantum hardware.

At their core, dynamical systems are described by state variables and an associated evolution rule. These rules provide the trajectory the system will follow over time. Depending on the parameters and initial conditions, dynamical systems can display various phenomena, including stability, periodicity, or even chaotic behavior. Gradient systems are a subset of dynamical systems where the evolution is guided by the gradient (or steepest descent) of a potential function. The trajectory inherently seeks local minima of this function, making them pivotal for optimization problems.

A Hopfield network is a fully connected, recurrent neural network and can be viewed as a gradient system. Each configuration of the network corresponds to a specific energy value. The dynamics of the network push the configurations to those of lower energy, creating a metaphorical landscape of valleys (attractors) and hills. Patterns are stored in a Hopfield network by adjusting the weights between neurons. Ideally, each stored pattern corresponds to a local minimum or an attractor in the energy landscape. When presented with a noisy or partial version of a stored pattern, the network dynamics lead it to evolve towards the closest stored pattern, showcasing its capability as an associative memory.

A specific type of Hopfield network, the locally competitive algorithm, solves the sparse coding problem of reconstructing input signals from linear combinations of a few features from a large overcomplete dictionary. Interestingly, the mathematical representation of these systems correspond to the dynamics of neurons measured in the V1 layer of the visual cortex in mammals [45]. Each neuron represents one of these features. The network is fully connected (making it a Hopfield network) and neurons compete by inhibiting and exciting other neurons for the lowest energy representation of the input. The dynamics of the competitive process stabilize into a configuration where only a subset of neurons are active, representing the input in a sparse manner. Within these networks, the full dynamics are governed by the non-linear activation functions assigned to each of the neurons. We aim to analyze the convergence behavior of networks with varying activations and use the insight as motivation to find solutions to the binary sparse coding problem where the energy landscapes are non-convex and cannot be directly solved using the gradient system approach.

To the best of the author's knowledge, research to date primarily focus more on addressing the computational aspects of LCA for non-negative sparse optimization using only unit ReLU activation functions. There have been attempts to address the convergence properties for this class of activation functions, but the analysis is incomplete for reasons which will be discussed in later sections [27, 49, 45]. The first contribution of this dissertation is a complete analysis of convergence for generic Rectified Linear Unit (ReLU) (including the unit slope from previous literature) and generic Rectified Sigmoid activation functions. Next, techniques for solving the binary sparse optimization problem are analyzed. The lack of provable convergence of step activation functions motivates the creation of non-autonomous systems comprised of time-varying sigmoidal activation that converge to the step function. Numerical experiments on classical computers are then performed to verify the analysis. Finally, numerical experiments are performed on emerging quantum annealing and spiking neuromorphic non-von Neumann architectures.

The remainder of this dissertation is organized as follows. In Chapter 2 we present a detailed formal description of the problems of interest. We prove the convergence of the corresponding dynamical systems with generic Rectified Linear Unit (ReLU) and generic Rectified Sigmoid activation function. We also present non-autonomous systems for solving the binary optimization problem. Chapter 3 contains several numerical experiments performed on CPU to verify our theoretical results. In Chapter 4, we map a network of unit slope ReLU activation functions from the continuous domain into a domain where activations of neurons are represented as spike rates. The equivalence of the spiking approach to its continuous counterpart is demonstrated on a spiking neuromorphic processor. In Chapter 5 we will reformulate the binary sparse optimization problem into a Quadratic Unconstrained Binary Optimization (QUBO) problem. Solutions are found using quantum annealing and a spiking neuromorphic devices to compare with classical techniques and our newly developed gradient system approaches. Chapter 6 consists of a collection of recent related papers published by the author. Finally, we list a few potential future works, motivated by the findings of this dissertation, and summarize our conclusions in Chapter 7 and Chapter 8, respectively.

Chapter 2

Constrained Sparse Optimization

2.1 Problem Setup

Let $\boldsymbol{x} \in \mathbb{R}^m$ represent a (vector-valued) signal, and let $D \in \mathbb{R}^{m \times p}$ be a dictionary with $p > m$ column vectors, typically of unit norm. The general problem that we are interested in concerns finding a sparse vector $\mathbf{a} = (a_1, \ldots, a_p) \in \mathbb{R}^p$, referred to as a *sparse code*, such that $\boldsymbol{x}$ is as close as possible to $D\,\boldsymbol{a}$ while constraining the number of nonzero elements in $\mathbf{a}$. Using the L^2 norm for measuring the distance between $\boldsymbol{x}$ and $D\,\boldsymbol{a}$, often referred to as reconstruction error, and the L^1 norm as a penalty term to enforce sparsity, we can formulate the problem as an unconstrained sparse optimization problem:

$$\min_{\boldsymbol{a} \in \mathbb{R}^p} E(\boldsymbol{a}), \qquad E(\boldsymbol{a}) := \frac{1}{2} ||\boldsymbol{x} - D\,\boldsymbol{a}||_2^2 + \lambda\, ||\boldsymbol{a}||_1,$$

where $\lambda > 0$ is a scaling or regularization parameter that determines the relative importance of sparsity compared to the reconstruction error. The above problem is a convex optimization problem, for which a solution exists. In the statistics and machine learning literature, this problem is known as LASSO (least absolute shrinkage and selection operator) or sparse coding [52]. In this dissertation, we will consider two types of constraints, amounting to two types of constrained sparse optimization problems:

- Non-negative sparse optimization;
- Binary sparse optimization.

2.2 Non-negative Sparse Optimization

Non-negative sparse optimization (NSO) constrains the sparse solution values $a_i \geq 0$ for all $i = 1, ..., p$. Formally we will write

$$\min_{\boldsymbol{a} \in \mathbb{R}^p_+} E(\boldsymbol{a}), \qquad E(\boldsymbol{a}) := \frac{1}{2} ||\boldsymbol{x} - D\,\boldsymbol{a}||_2^2 + \lambda\,||\boldsymbol{a}||_1. \tag{2.1}$$

It is to be noted that the functional E is convex with respect to its argument $\boldsymbol{a}$, and it is quadratic in $\boldsymbol{a}$ when $\boldsymbol{a} \in \mathbb{R}^p_+$. NSO is of particular interest when mapping the sparse coding problem onto neuromorphic hardware. In this case, the outputs of the spiking processors are constrained to be positive spike rates [28, 21, 24, 23]. More details will be presented in Section 2.8.

2.3 Binary Sparse Optimization

Binary sparse optimization (BSO) constrains the sparse solution values $a_i \in \{0, 1\}$ for all $i = 1, ..., p$. Our formal definition of BSO reads:

$$\min_{\boldsymbol{a} \in \{0,1\}^p} E(\boldsymbol{a}), \qquad E(\boldsymbol{a}) := \frac{1}{2} ||\boldsymbol{x} - D\,\boldsymbol{a}||_2^2 + \lambda\,||\boldsymbol{a}||_1. \tag{2.2}$$

It is worth noting the binary constraint makes the L^1 norm sparsity penalty equivalent to an L^0 norm, and the functional E becomes non-convex. BSO is of interest because such a constraint allows us to map the optimization problem onto quantum annealing devices [23, 20]. Further details will be presented in Chapters 3 and 5.

2.4 Karush-Kuhn-Tucker Conditions and Convex Optimization

Karush-Kuhn-Tucker (KKT) conditions give necessary and sufficient conditions for the existence of optimal solutions to convex optimization problems, given in the following theorem.

Theorem 1. *Let $E : \mathbb{R}^p \to \mathbb{R}$ be a convex function. Consider the optimization problem $\min_{\boldsymbol{a}} \; E(\boldsymbol{a})$, subject to constraints $h_i(\mathbf{a}) \leq 0$, $i = 1, \ldots, p$, where all h_i's are affine maps, consisting of a linear transformation followed by a translation. Then $\boldsymbol{a}^* \in \mathbb{R}^p$ is an optimal solution to the constrained optimization problem if and only if there exists a $\boldsymbol{\mu}^* \in \mathbb{R}^p$ such that the following conditions hold:*

1. *Stationarity:* $\mathbf{0} \in \partial E(\boldsymbol{a}^*) + \sum_{i=1}^p \mu_i^* \nabla_a \, h_i(\boldsymbol{a}^*)$.
2. *Complimentarity:* $\mu_i^* h_i(\boldsymbol{a}^*) = 0, \quad i = 1, 2, ..., p$.
3. *Feasibility:* $h_i(\boldsymbol{a}^*) \leq 0, \; \mu_i^* \geq 0 \quad i = 1, 2, ..., p$.

Here, ∂E denotes the generalized gradient of E [9].

Proof. For the proof, we refer to [6]. □

We also refer to [4] for more details on KKT conditions. We note that while NSO is a convex optimization problem to which the KKT theorem can be applied, BSO is not a convex optimization problem. Later, we will connect BSO to other convex optimization problems where we can utilize KKT conditions for convergence studies of BSO.

2.5 Gradient System Approach

After establishing a problem that satisfies the KKT conditions, we now focus on transforming the constrained sparse optimization problem into a dynamical system with a solution flow

$$\boldsymbol{u} = (u_1, \ldots, u_p) : \; t \in \mathbb{R}_+ \mapsto \boldsymbol{u}(t) \in \mathbb{R}^p.$$

An important part of the dynamics is the introduction of a transfer function

$$\sigma(\boldsymbol{u}(t)) = (\sigma(u_1(t)), ..., \sigma(u_p(t))) =: \boldsymbol{a}(t),$$

that acts component-wise on the components of the solution flow $\boldsymbol{u}(t)$, outputting a time-dependent sparse code $\boldsymbol{a}$ whose limit

$$\boldsymbol{a} = \lim_{t \to \infty} \boldsymbol{a}(t),$$

will be the sparse code $\boldsymbol{a}$ that solves the desired constrained sparse optimization problem. The dynamical systems that we will obtain are of gradient form, i.e., their force terms are given as the negative gradient of a real-valued function; see e.g. (2.9). We refer to this method of computing the sparse code $\boldsymbol{a}$ as the gradient system approach.

To this end, we define

$$\widetilde{E}(\boldsymbol{u}(t)) := E(\sigma(\boldsymbol{u}(t))), \tag{2.3}$$

and instead of solving (2.1) or (2.2) with related constraints in the form NSO or BSO, we look for a solution flow and an activation function that gives us the sparse code $\boldsymbol{a}$. This requires finding a dynamical system with solution flow $\boldsymbol{u}(t)$ that converges to the set of stationary points of $\widetilde{E}(\boldsymbol{u}(t))$; see the following discussion for precise definitions. Let us first expand (2.3) before proceeding for clarity.

$$
\begin{aligned}
\widetilde{E}(\boldsymbol{u}(t)) &= \frac{1}{2}||\boldsymbol{x} - D\,\sigma(\boldsymbol{u}(t))||_2^2 + \lambda\,||\sigma(\boldsymbol{u}(t))||_1 \\
&= \frac{1}{2}(\boldsymbol{x} - D\sigma(\boldsymbol{u}(t)))^\top(\boldsymbol{x} - D\sigma(\boldsymbol{u}(t))) + \lambda||\sigma(\boldsymbol{u}(t))||_1 \\
&= \frac{1}{2}(\boldsymbol{x}^\top\boldsymbol{x} - \boldsymbol{x}^\top D\sigma(\boldsymbol{u}(t)) - \sigma(\boldsymbol{u}(t))^\top D^\top\boldsymbol{x} + \sigma(\boldsymbol{u}(t))^\top D^\top D\sigma(\boldsymbol{u}(t))) + \lambda||\sigma(\boldsymbol{u}(t))||_1 \\
&= \frac{1}{2}(\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top D\sigma(\boldsymbol{u}(t)) + \sigma(\boldsymbol{u}(t))^T D^\top D\sigma(\boldsymbol{u}(t))) + \lambda\sum_{i=1}^{p}\sigma(u_i(t)) \qquad (2.4)
\end{aligned}
$$

The last equality comes because our constraints make all activations strictly non-negative in both regimes, turning the norm into a sum.

2.6 Dynamical Systems

We will establish the underlying theory necessary to prove convergence of a dynamical system to a global minimum and subsequently apply the machinery to various problems of interest. First we consider a generic system of differential equations for $\boldsymbol{u} : \mathbb{R}_+ \to \mathbb{R}^p$:

$$\dot{\boldsymbol{u}}(t) = \boldsymbol{f}(\boldsymbol{u}(t)) = (f_1(\boldsymbol{u}(t)), \cdots, f_p(\boldsymbol{u}(t))), \qquad (2.5)$$

where $\boldsymbol{f} : \mathbb{R}^p \to \mathbb{R}^p$ is a p-dimensional vector field, to be determined in this section. We augment the system (2.5) with an initial condition,

$$\boldsymbol{u}(0) = \boldsymbol{u}^{(0)}, \qquad (2.6)$$

arriving to an initial value problem (IVP).

When $\boldsymbol{f}(\boldsymbol{u}(t)) : \mathbb{R}^p \to \mathbb{R}^p$ is sufficiently smooth, for instance when it is Lipschitz-continuous, the (local) existence and uniqueness of the solution to the IVP (2.5)-(2.6) follows from the classical theory, and the solution $\boldsymbol{u}$ is typically continuously differentiable; see e.g., [39, 40]. The type of vector fields that we deal with in this section are however not smooth; they are smooth almost everywhere, with isolated jump discontinuities. Hence, solutions need to be defined in a weaker sense: the equality (2.5) only holds for almost every $t \geq 0$. To this end, we will closely follow [17], which provides a comprehensive theoretical analysis of systems featuring specific types of discontinuous vector fields.

We begin by considering specific types of discontinuous vector fields, which adhere to the following assumption. We note that the vector fields that we derive here satisfy this assumption.

Assumption. The vector field $\boldsymbol{f}(\boldsymbol{u}(t)) : \mathbb{R}^p \to \mathbb{R}^p$ is locally Lipschitz-continuous in $\boldsymbol{u}(t)$ except at $\boldsymbol{u}(t) = \lambda \, \mathbb{1}_p$, with $\mathbb{1}_p$ being the p-dimensional vector of 1's, where it has a bounded jump discontinuity.

We next define the solution to the IVP (2.5)-(2.6) in weak sense, following Section 4 of [17]. For any piecewise continuous vector-valued function $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{u})$ with a set M (of measure zero) of points of discontinuity, such as a vector field that satisfies the above assumption, we specify a set $\mathbf{F}(\boldsymbol{u})$ in a p-dimensional space as follows.

- At each point $\boldsymbol{u}$ where $\boldsymbol{f}$ is continuous, the set $\mathbf{F}(\boldsymbol{u})$ consists of one point coinciding with the value of the function $\boldsymbol{f}(\boldsymbol{u})$ at that point.
- At each point $\boldsymbol{u}$ where $\boldsymbol{f}$ is discontinuous, the set $\mathbf{F}(\boldsymbol{u})$ is given by the smallest convex closed set containing all the limit values of $\boldsymbol{f}(\boldsymbol{v})$ where $\boldsymbol{v} \notin M$ and $\boldsymbol{v} \to \boldsymbol{u}$.

A solution of equation (2.5) is then defined as an absolutely continuous vector-valued function $\boldsymbol{u} = \boldsymbol{u}(t)$ such that $\dot{\boldsymbol{u}}(t) \in \mathbf{F}(\boldsymbol{u}(t))$ for almost every $t \geq 0$. Such a solution is referred to as a differential inclusion solution. The IVP (2.5)-(2.6) has a unique differential inclusion solution, provided the corresponding set-valued function $\mathbf{F}(\boldsymbol{u}(t))$ is nonempty, compact, and convex. These conditions are referred to as the basic conditions [17] (see page 76). We refer to Section 7 (Theorem 1) and Section 10 of [17] for details on the existence and uniqueness proofs. We also refer to [47] where the convexity condition is relaxed.

We denote by $\boldsymbol{U}(t, \boldsymbol{u}^{(0)})$ the solution to the system (2.5) at time t with the initial solution $\mathbf{u}^{(0)} = \boldsymbol{U}(0, \boldsymbol{u}^{(0)})$. A set of definitions and theorems follow.

Definition 1. *A point $\boldsymbol{u} \in \mathbb{R}^p$ is a fixed point of the ODE system* (2.5) *if and only if $\boldsymbol{f}(\boldsymbol{u}) = \mathbf{0}$.*

Definition 2. *A set $\boldsymbol{K} \subseteq \mathbb{R}^p$ is positive invariant if for any $\boldsymbol{u}^{(0)} \in \boldsymbol{K}$, the continuous sequence of points in time (also known as a flow) $\boldsymbol{u}(t) = \boldsymbol{U}(t, \boldsymbol{u}^{(0)}) \in \boldsymbol{K}$ for all $t \geq 0$.*

Definition 3. *For any set $\boldsymbol{K} \subseteq \mathbb{R}^p$ a flow $\boldsymbol{U}(t, \boldsymbol{u}^{(0)})$ converges to $\boldsymbol{K}$ if*

$$\lim_{t\to\infty} dist(\boldsymbol{U}(t, \boldsymbol{u}^{(0)}), \boldsymbol{K}) = \lim_{t\to\infty} inf_{\boldsymbol{y}\in\boldsymbol{K}} \parallel \boldsymbol{U}(t, \boldsymbol{u}^{(0)}) - \boldsymbol{y} \parallel_2 = 0.$$

Definition 4. *$\boldsymbol{B} \subseteq \mathbb{R}^p$ is a domain of bounded flows if for each $\boldsymbol{u}^{(0)} \in \boldsymbol{B}$, there exists a $C > 0$ such that $\parallel \boldsymbol{U}(t, \boldsymbol{u}^{(0)}) \parallel_2 \; < C$ for all $t \geq 0$.*

When the ODE system (2.5) is defined strongly, such as when the vector field $\boldsymbol{f}$ is smooth, we have the following well-known result.

Theorem 2. *(La Salle) Given a dynamical system* (2.5)*, let $\boldsymbol{K} \subseteq \mathbb{R}^p$ be a compact and positive invariant set, and let $\widetilde{E} : \mathbb{R}^p \to \mathbb{R}$ be a scalar functional of $\boldsymbol{u}$ with continuous first partial derivatives. Suppose that*

$$\dot{\widetilde{E}}(\boldsymbol{u}(t)) = \nabla_{\boldsymbol{u}(t)} \; \widetilde{E}(\boldsymbol{u}(t)) \cdot \boldsymbol{f}(\boldsymbol{u}(t)) = \sum_{i=1}^{p} \partial_{u_i} \widetilde{E}(\boldsymbol{u}(t)) f_i(\boldsymbol{u}(t)) \leq 0, \quad \forall \boldsymbol{u}(t) \in \mathbb{R}^p.$$

Then for all $\boldsymbol{u}^{(0)} \in \mathbb{R}^p$, the solution $\boldsymbol{U}(t, \boldsymbol{u}^{(0)})$ to the system (2.5) *converges to $\boldsymbol{M}$, which is the largest positive invariant set in $\boldsymbol{S} = \left\{ \boldsymbol{u}(t) | \dot{\widetilde{E}}(\boldsymbol{u}(t)) = 0 \right\}$.*

Proof. For the proof, we refer to [46]. □

For our systems of interest, however, the system (2.5) is not defined strongly. Indeed, our vector fields are not smooth, and some partial derivatives of $\widetilde{E}$ may not exist. For example, a partial derivative $\partial_{u_i}\widetilde{E}$ may not exist when $\sigma'(u_i(t))$ does not exist; see (2.4). Hence, we need to use an extension of the La Salle Theorem as follows:

Theorem 3. *(Modified La Salle) Given a domain $\boldsymbol{K} \subseteq \mathbb{R}^p$ of bounded flows that are closed and positive invariant, suppose there exist scalar functionals $\widetilde{E}, W : \mathbb{R}^p \to \mathbb{R}$ with $\widetilde{E}$ being continuous and W being upper semicontinous and non-positive for all $\boldsymbol{u}(t) \in \mathbb{R}^p$. Suppose further that*

$$\dot{\widetilde{E}}(\boldsymbol{u}(t)) = W(\boldsymbol{u}(t)) \leq 0, \qquad \forall \boldsymbol{u} \in \mathbb{R}^p, \text{ a.e. in } [0, \infty).$$

Then for all $\boldsymbol{u}^{(0)} \in \boldsymbol{K}$, the solution $\boldsymbol{U}(t, \boldsymbol{u}^{(0)})$ to the system (2.5) *converges to $\boldsymbol{M}$, which is the largest positive invariant set in $\boldsymbol{S} = \{\boldsymbol{u}(t) | W(\boldsymbol{u}(t)) = 0\}$.*

Proof. For the proof, we refer to [49]. □

Consider the index set

$$I(t) = \left\{ i \in \{1, 2, ..., p\} : \partial_{u_i}\widetilde{E}(\boldsymbol{u}(t)) \ \text{ exists} \right\}. \tag{2.7}$$

We note that the set $I(t)$ depends on t, that is, a different point in time may give a different index set $I(t)$. Motivated by Theorem 2 and Theorem 3, we set:

$$W(\boldsymbol{u}(t)) = \begin{cases} \sum\limits_{i \in I(t)} \partial_{u_i}\widetilde{E}(\boldsymbol{u}(t)) \cdot f_i(\boldsymbol{u}(t)) \text{ , when } I(t) \neq \emptyset \\ \\ 0 \quad \text{, when} \quad I(t) = \emptyset. \end{cases} \tag{2.8}$$

Assuming the set of time points at which the gradient does not exist is of measure zero, we will have $I(t) = \{1, 2, \ldots, p\}$ for almost every $t \geq 0$. We note that this assumption holds for the type of transfer functions that we consider in this work. By (2.8) we therefore arrive at $\dot{\widetilde{E}}(\boldsymbol{u}(t)) = W(\boldsymbol{u}(t)),$ a.e. in $[0,\infty)$. All that is left to do is to choose $f_i(\boldsymbol{u})$ such that $W \leq 0$ so that Theorem 3 can be utilized. Let D_i represent the i-th column of the dictionary, D. One particular way is to set $f_i(\boldsymbol{u}) = -\partial_{u_i}\widetilde{E}(\boldsymbol{u})$, whenever the partial derivative exists. This brings us to the following formula (see the appendix for the derivation),

$$f_i(\boldsymbol{u}(t)) = \left(D_i^\top \boldsymbol{x} - D_i^\top D\sigma(\boldsymbol{u}(t)) - \lambda\right)\sigma'(u_i(t)), \tag{2.9}$$

when $\sigma'(u_i(t))$ exists. When $\sigma'(u_i(t))$ does not exist, we are free to choose $f_i(\boldsymbol{u}(t))$ because the contribution of $f_i(\boldsymbol{u}(t))$ will be excluded from the sum in the definition of $W(\boldsymbol{u}(t))$, preserving the nonpositivity constraint; see (2.8). Nevertheless, for convenience, we use the same formula (2.9) also in this case, but we replace strong derivatives by weak derivatives; see details in the forthcoming sections.

The gradient system that we consider here reads

$$\dot{\boldsymbol{u}}(t) = \boldsymbol{f}(\boldsymbol{u}(t)) = (D^\top \boldsymbol{x} - D^\top D\sigma(\boldsymbol{u}(t)) - \lambda \mathbb{1}_p) \odot \sigma'(\boldsymbol{u}(t)), \tag{2.10}$$

where $\sigma'(\boldsymbol{u}(t)) = (\sigma'(u_1(t)), \cdots, \sigma'(u_p(t)))$ is understood in the weak sense. Specific examples of σ' can be found in (2.12), (2.14) and (2.18) for activation functions found in (2.11), (2.13) and (2.17), respectively.

We are now ready to impose our constraints for NSO and a system that asymptotically converges to BSO by modifying the transfer function to fit our needs.

2.7 NSO Transfer Functions

The standard transfer function used in the literature is a shifted unit slope Rectified Linear Unit (ReLU) function defined as,

$$a_i = \sigma(u_i(t)) = \begin{cases} u_i(t) - \lambda \quad \text{when} \quad u_i(t) \geq \lambda \\ 0 \quad \text{otherwise.} \end{cases} \tag{2.11}$$

Although the derivative at time points where $u_i(t) = \lambda$ does not exist, motivated by the flexibility of the choice for σ' discussed in the previous sections and the notion of weak derivatives, we set

$$\sigma'(u_i(t)) = \begin{cases} 1 \quad \text{, when} \quad a_i \neq 0 \\ 0 \quad \text{, when} \quad a_i = 0. \end{cases} \tag{2.12}$$

We may then consider the gradient system (2.10) with the activation and its derivative given in (2.11)-(2.12). It is important to note that the gradient system that we are considering here is different from the dynamical system used in the literature; see e.g., [21, 23, 49],

$$\dot{\boldsymbol{u}}(t) = D^\top \boldsymbol{x} - D^\top D\sigma(\boldsymbol{u}(t)) - \lambda \mathbb{1}_p = D^\top \boldsymbol{x} - \boldsymbol{u}(t) - (D^\top D - I)\,\sigma(\boldsymbol{u}(t)).$$

This system is obtained under the assumption that σ' is identically one, independent of the value of a_i, and through a different formulation of W than our formulation in (2.8), by replacing the index set $I(t)$ in our formulation by $A(t) = \{i \in \{1, \ldots, p\} : a_i(t) \neq 0\}$. There is, however, a crucial problem with this formulation: it invalidates the condition $\dot{\tilde{E}}(\boldsymbol{u}(t)) = W(\boldsymbol{u}(t))$, a.e. in $[0, \infty)$, which is necessary for the result of Theorem 3 to hold. As a result, the convergence analysis of the optimization problem by the gradient system method cannot be studied making use of the (modified) La Salle theorem.

It is to be noted that the above approach applies to general transfer functions and is not restricted to the specific form (2.11). Importantly, if we change the form of the transfer function, our dynamical system changes and so do its fixed points. For example, we may consider ReLU functions with general positive slopes $c > 1$:

$$a_i = \sigma_c(u_i(t)) = \begin{cases} c(u_i(t) - \lambda), & \text{when} \quad u_i(t) \geq \lambda \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

In this case, following the discussion above we set

$$\sigma_c'(u_i(t)) = \begin{cases} c & \text{, when} \quad a_i \neq 0 \\ 0 & \text{, when} \quad a_i = 0. \end{cases} \tag{2.14}$$

We may then consider the gradient system (2.10) with the activation and its derivative given in (2.13)-(2.14).

ReLU functions with different slopes can be seen in figure 2.1.

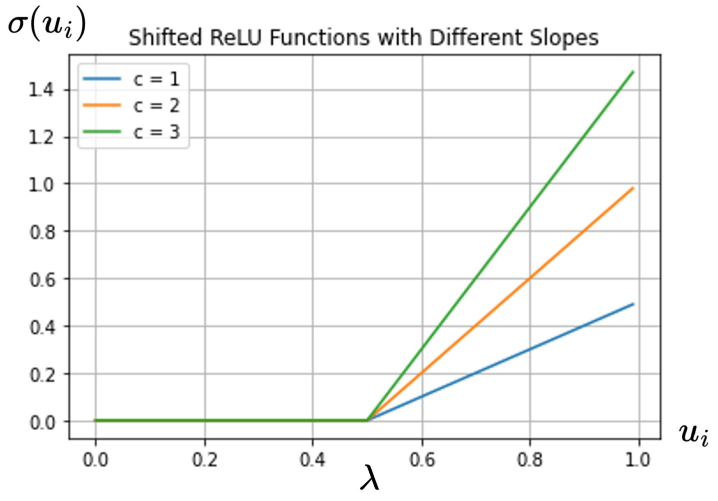


Figure 2.1: **ReLU functions with sparsity penalty parameter $\lambda = 0.5$ and different slopes.**

2.8 BSO Transfer Function and Approximating NSO Sequences

The BSO transfer function of interest forces the coefficients to be binary and takes the form

$$a_i = \sigma(u_i(t)) = \begin{cases} 1 & \text{when} \quad u_i(t) \geq \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

The derivative of the activation function (2.15) exists almost everywhere and is zero, except at time points where $u_i(t) = \lambda$. This makes all components of the force field $\boldsymbol{f}$ in (2.10) equal to zero, delivering a trivial dynamical system $\dot{\boldsymbol{u}}(t) = \mathbf{0}$, and hence, preventing a direct study of the BSO problem through the gradient system approach.

In order to study the BSO problem by the gradient system approach, we consider a sequence of problems that "approach" the BSO problem, by considering a sequence of activation functions that "approach" the binary activation function (2.15). Specifically, we introduce a more restricted NSO problem,

$$\min_{\boldsymbol{a} \in [0,1]^p} E(\boldsymbol{a}), \qquad E(\boldsymbol{a}) := \frac{1}{2} ||\boldsymbol{x} - D\,\boldsymbol{a}||_2^2 + \lambda\,||\boldsymbol{a}||_1. \tag{2.16}$$

We then select a sequence of activation functions with the range $[0, 1]$ that converge to the original BSO activation functions. To this end, we consider two different strategies: one by defining a time-independent sequence, and one by defining a time-dependent sequence, as follows.

In the first approach, we consider a sequence of time-independent activation functions, generated by a family of sigmoid functions, which are strictly increasing,

$$a_i = \sigma_k(u_i(t)) = \begin{cases} \frac{2}{1+e^{-k(u_i(t)-\lambda)}} - 1, & \text{when } u_i(t) \geq \lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{2.17}$$

where $k \in \{1, 2, \dots\}$. We refer to such functions as rectified sigmoid functions. The weak derivative of the rectified sigmoid function (2.17) can be computed using the product rule as,

$$\sigma_k'(u_i(t)) = \begin{cases} \frac{2ke^{-k(u_i(t)-\lambda)}}{(e^{-k(u_i(t)-\lambda)}+1)^2}, & a_i \neq 0, \\ 0, & a_i = 0. \end{cases} \tag{2.18}$$

We may then consider the gradient system (2.10) with activation and its derivative given in (2.17)-(2.18).

We note that as k increases, $\sigma_k(u_i(t))$ gets closer to the step function; see figure 2.2. Hence, one strategy to solve for the BSO problem would be to solve the approximating autonomous system (2.10) with the derivatives given in (2.18) with a large k (for example $k = 100$) and then perform post-processing to force the resulting solution $\boldsymbol{a}$ to be binary.

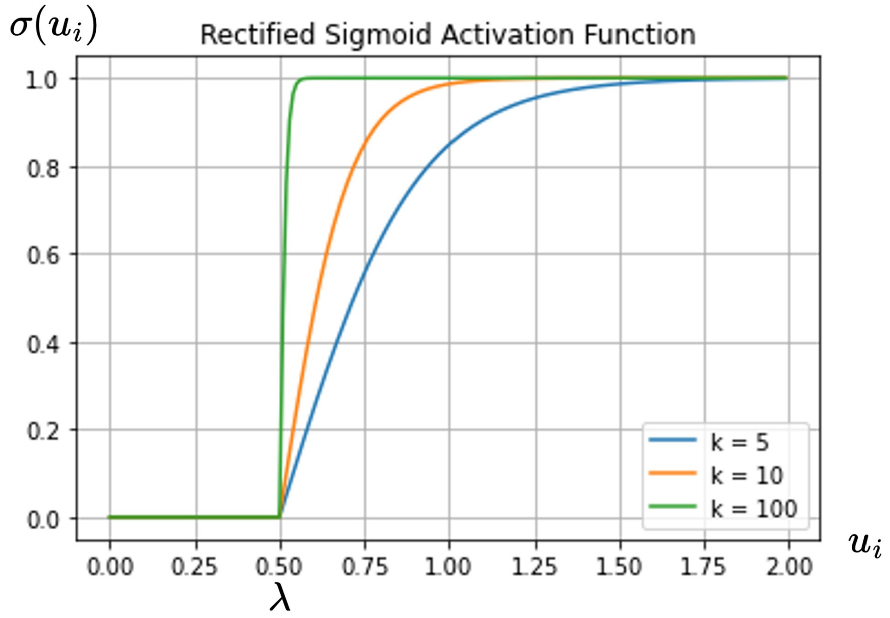


Figure 2.2: **Rectified sigmoids with different decay parameters k and sparsity penalty parameter $\boldsymbol{\lambda} = 0.5$.** As the constant k increases, the transfer function approaches the step function used in BSO.

An alternative approach is to define a sequence of dynamical systems using a sequence of time-dependent activation functions converging to the step activation function (2.15) as time increases. This would make the force field $\boldsymbol{f}$ an explicit function of both $\boldsymbol{u}(t)$ and t, and thus amount to a non-autonomous system. Although the convergence theory of general non-autonomous systems is not well developed (see e.g. [34]), we can still numerically study their limiting solution as time increases to study the BSO problem. First, to find the full time-derivative of $\widetilde{E}$, we differentiate it with respect to time,

$$\begin{aligned}\partial_t\widetilde{E}(\boldsymbol{u}(t),t) &= \lambda\sum_{i=1}^{p}\partial_t\sigma_i(u_i(t)(t)) - 2\boldsymbol{x}^\top D\partial_t\sigma(\boldsymbol{u}(t)) + 2\sigma(\boldsymbol{u}(t))^\top D^\top D\partial_t\sigma(\boldsymbol{u}(t)) \\ &= (\lambda\mathbb{1}_p^\top - 2\boldsymbol{x}^\top D + 2\sigma(\boldsymbol{u}(t))^\top D^\top D)\partial_t\sigma(\boldsymbol{u}(t))\cdot \end{aligned} \tag{2.19}$$

Assuming $\partial_{u_i(t)}\widetilde{E}$ and $\partial_t\widetilde{E}$ exist a.e. in $[0,\infty)$,

$$\dot{\widetilde{E}}(\boldsymbol{u}(t),t) = \sum_{i=1}^{p}\partial_{u_i(t)}\widetilde{E}(\boldsymbol{u}(t),t)\,\dot{u}_i(t) + \partial_t\widetilde{E}(\boldsymbol{u}(t),t) = W(\boldsymbol{u}(t),t) \text{ a.e. in } [0,\infty). \tag{2.20}$$

If we replace k in (2.17) with a time-dependent function $k(t)$ such that $\lim_{t\to\infty}k(t)=\infty$, then we will have created a non-autonomous system whose force terms asymptotically converge to 0 with the step activation function seen in figure 2.2. The general form for this case becomes

$$a_i = \sigma(u_i(t),t) = \begin{cases} \frac{2}{1+e^{-k(t)(u_i(t)-\lambda)}} - 1 & \text{when} \quad u_i \geq \lambda \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

with derivative with respect to $u_i(t)$ when $a_i \neq 0$ set as

$$\partial_{u_i(t)}\sigma(u_i(t),t) = \frac{2k(t)e^{-k(t)(u_i(t)-\lambda)}}{(e^{-k(t)(u_i(t)-\lambda)}+1)^2}, \tag{2.22}$$

and corresponding time derivative when $a_i \neq 0$

$$\partial_t\sigma(u_i(t),t) = \frac{2(u_i(t)-\lambda)e^{-k(t)(u_i(t)-\lambda)}k'(t)}{(e^{-k(t)(u_i(t)-\lambda)}+1)^2}. \tag{2.23}$$

Following the strategy from before, we set the force term equal to zero when $a_i = 0$, and when $a_i \neq 0$ to

$$f_i(\boldsymbol{u}(t), t) = -\partial_{u_i(t)} \widetilde{E}(\boldsymbol{u}(t), t) = \\ \left(\boldsymbol{x}^\top D_i - D_i^\top (\sum_{k \neq i}^{p} a_k D_k) - D_i^\top D_i (\frac{2}{1 + e^{-k(t)(u_i(t) - \lambda)}} - 1) - \lambda \right) \frac{2k e^{-k(t)(u_i(t)-\lambda)}}{(e^{-k(t)(u_i(t)-\lambda)} + 1)^2}. \tag{2.24}$$

The partial time-derivative $\partial_t \sigma(u_i(t), t)$ in (2.24) vanishes as $t \to \infty$ because the exponential terms approach zero. Therefore, $\partial_t \widetilde{E}(\boldsymbol{u}, t)$ in (2.19) also vanishes as $t \to \infty$. Hence, by (2.20), in the limit $W(\boldsymbol{u}(t), t) \leq 0$. Motivated by the theory of autonomous systems (Theorem 3), after integrating the non-autonomous system for a sufficiently long time, the convergence of the solution flow may be attained.

2.9 Analysis of Gradient System Approach

In this section, we will present the convergence analysis of the gradient system (2.10) with the activation and its derivative given in either (2.13)-(2.14) or (2.17)-(2.18), for solving the NSO and BSO problems, respectively.

First a lemma that enables us to use Theorem 3 and show the convergence of our gradient systems.

Lemma 1. *Let $\widetilde{E}$ and W be given respectively by* (2.4) *and* (2.8) *with two generic activation functions in* (2.13) *and* (2.17)*, where $\boldsymbol{u}(t) = \boldsymbol{U}(t, \boldsymbol{u}^{(0)})$, with $\boldsymbol{u}^{(0)} \in \mathbb{R}^p$, is the flow of the gradient system* (2.10) *with the activation and its derivative given in either* (2.13)-(2.14) *or* (2.17)-(2.18)*. Then the following statements hold,*

1. *$\widetilde{E}(\boldsymbol{u}(t))$ is continuous.*

2. *$W(\boldsymbol{u}(t)) \leq 0$ is upper semicontinuous.*

3. $\dot{\widetilde{E}}(\boldsymbol{u}(t)) = W(\boldsymbol{u}(t))$, *a.e. in* $[0,\infty)$.

4. $\dot{\widetilde{E}}(\boldsymbol{u}(t)) \le 0$.

5. *The set* $\boldsymbol{K} = \{\boldsymbol{u}(t) | \widetilde{E}(\boldsymbol{u}(t)) \le \widetilde{E}(\boldsymbol{u}^{(0)})\}$ *is closed, positive invariant, and a domain of bounded flows.*

Proof.

1. All of the terms in (2.4) are continuous because activation functions of the form in (2.13) and (2.17) are continuous. Since $\widetilde{E}$ is a linear combination of continuous functions, it is also continuous.

2. Define an indicator function $\mathbb{I}_i(\boldsymbol{u}(t)) = 1$ if $\sigma(u_i(t)) > 0$ and 0 otherwise for $i = 1, \ldots, p$. Since we have $f_i(\boldsymbol{u}(t)) = -\partial_{u_i(t)}\widetilde{E}(\boldsymbol{u}(t))$ with σ' given in either (2.14) or (2.18), from (2.8) we get,

$$W(\boldsymbol{u}(t)) = \sum_{i \in I} \partial_{u_i}\widetilde{E}(\boldsymbol{u}(t)) \cdot f_i(\boldsymbol{u}(t)) = \sum_{i=1}^{p} -(f_i(\boldsymbol{u}(t)))^2 \mathbb{I}_i(\boldsymbol{u}(t)) \le 0.$$

   Since $-\mathbb{I}_i(\boldsymbol{u}(t))$ is upper semicontinuous, so is $-(f_i(\boldsymbol{u}(t)))^2\mathbb{I}_i(\boldsymbol{u}(t))$, and thus $W(\boldsymbol{u})$ is upper semicontinuous because it is the sum of upper semicontinuous functions.

3. This follows directly from the definition of $W(\boldsymbol{u}(t))$ in (2.8) and noting the activation functions (2.13) and (2.17) are differentiable almost everywhere.

4. From items 2 and 3 we know $\dot{\widetilde{E}}(\boldsymbol{u}(t)) \le 0$ a.e. in $[0,\infty)$. For the points where $\dot{\widetilde{E}}(\boldsymbol{u}(t)) \ne W(\boldsymbol{u}(t))$, i.e., the points where the $\partial u_i \widetilde{E}(\boldsymbol{u}(t))$ doesn't exist, we set $\sigma' = 0$, causing the force term $f_i = 0$. Hence, these points do not contribute to $\dot{\widetilde{E}}(\boldsymbol{u}(t))$, and it remains less than or equal to zero.

5. Let $\boldsymbol{u}^{(0)} \in \mathbb{R}^p$ be an arbitrary initial point and define $\widetilde{E}(\boldsymbol{u}^{(0)}) := e^{(0)}$. Since $\widetilde{E}$ is continuous, $\boldsymbol{K}$ is closed because $\widetilde{E}^{-1}([0, e^{(0)}])$ (the inverse map of $\widetilde{E}$) is closed.

Since $\dot{\widetilde{E}}(\boldsymbol{u}(t)) = W(\boldsymbol{u}(t)) \leq 0$ a.e., $\widetilde{E}(\boldsymbol{u}(t))$ is non-increasing in t . Hence for any flow corresponding to an arbitrary initial condition, we can say,

$$\widetilde{E}(\boldsymbol{U}(t, \boldsymbol{u}^{(0)})) \leq \widetilde{E}(\boldsymbol{U}(0, \boldsymbol{u}^{(0)})) = e^{(0)}$$

which implies $\boldsymbol{K}$ is positive invariant and bounded. Hence $\boldsymbol{U}(t, \boldsymbol{u}^{(0)}) \to \boldsymbol{K}$ for any $\boldsymbol{u}^{(0)} \in \mathbb{R}^p$.

□

Our first main result follows.

Theorem 4. *Consider the gradient system* (2.10) *with the activation and its derivative given in either* (2.13)-(2.14) *or* (2.17)-(2.18). *For all* $\boldsymbol{u}^{(0)} \in \boldsymbol{K}$*, the solution flow* $\boldsymbol{U}(t, \boldsymbol{u}^{(0)})$ *to the gradient system converges to the fixed points of the system.*

Proof. From Lemma 1, we know the hypotheses of Theorem 3 have been satisfied for both systems. Hence, for any $\boldsymbol{u}^{(0)} \in \boldsymbol{K}$ for either system, $\boldsymbol{U}(t, \boldsymbol{u}^{(0)})$ will converge to the largest positive invariant set $\boldsymbol{M}$ inside the set $\boldsymbol{S} = \{\boldsymbol{u}(t) | W(\boldsymbol{u}(t)) = 0\}$. Let F be the set of fixed points of the system. We will show that $\boldsymbol{M} \subset F$. To this end, we let $\boldsymbol{u}^* \in \boldsymbol{M}$, which implies that $W(\boldsymbol{u}^*) = 0$. Hence, by the definition of W in (2.8) and the definition of f_i in (2.9), we obtain $f_i(\boldsymbol{u}^*) = 0$ for all $i \in \{1, \ldots, p\}$ for which $\partial_{u_i} \tilde{E}(\boldsymbol{u}^*)$ exists. It remains to note that we always set $f_i(\boldsymbol{u}^*) = 0$ for all $i \in \{1, \ldots, p\}$ for which $\partial_{u_i} \tilde{E}(\boldsymbol{u}^*)$ does not exist, because in such cases we set $\sigma' = 0$. Hence, we get $\boldsymbol{u}^* \in F$, and the proof is complete. □

Our second main result follows.

Theorem 5. *Let C be the set of optimal solutions to* (2.1) *where* $a_i \in [0, \infty)$ *or to* (2.16) *when* $a_i \in [0, 1]$ *for all* $i \in \{1, \ldots, p\}$. *Let F be the set of fixed points for the gradient system* (2.10) *with the activation and its derivative given in either* (2.13)-(2.14) *or* (2.17)-(2.18). *Then the following statements hold:*

1. *If* $\boldsymbol{u}^* \in F$ *and* $\boldsymbol{a}^* = \sigma(\boldsymbol{u}^*)$*, then* $\boldsymbol{a}^* \in C$*, assuming* $-D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* \geq -\lambda$ *for* i *when* $a_i^* = 0$*;*

2. *If* $\boldsymbol{a}^* \in C$*, then* $\exists \boldsymbol{u}^* \in F$ *s.t.* $\sigma(\boldsymbol{u}^*) = \boldsymbol{a}^*$*.*

Proof. To show 1, let $\boldsymbol{u}^* \in F$. Then $\boldsymbol{f}(\boldsymbol{u}^*) = 0$. In order to show $\boldsymbol{a}^* = \sigma(\boldsymbol{u}^*) \in C$, we need to show that the criteria in Theorem 1 are satisfied, with $h_i(\boldsymbol{a}) = -a_i \leq 0$. It suffices to show that for every $i \in \{1, \ldots, p\}$, there exists $\mu_i^* \geq 0$ such that $0 \in \partial_{a_i} E(\boldsymbol{a}^*) - \mu_i^*$ and $\mu_i^* \, a_i^* = 0$, because then all three conditions of Theorem 1 will be satisfied, noting that $h_i(\boldsymbol{a}) = -a_i \leq 0$. We consider two cases as follows:

Case 1: $a_i^* > 0$. In order to satisfy $\mu_i^* \, a_i^* = 0$, we need to take $\mu_i^* = 0$. Moreover, from $f_i(\boldsymbol{u}^*) = 0$, noting that $\sigma'(u_i^*) \neq 0$, we obtain $D_i^\top \boldsymbol{x} - D_i^\top D\boldsymbol{a}^* - \lambda = 0$. The desired condition is then satisfied noting that $\partial_{a_i} E(\boldsymbol{a}^*) - \mu_i^* = -D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* + \lambda = 0$.

Case 2: $a_i^* = 0$. It can be seen $\mu_i^* \, a_i^* = 0$ is satisfied for any $\mu_i^* \geq 0$. The generalized derivative of E is

$$\partial_{a_i} E(\boldsymbol{a}^*) = -D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* + \lambda \, \partial_{a_i} |a_i^*| = -D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* + [-\lambda, \lambda].$$

We need show that there exists $\lambda_i^* \geq 0$ such that $\partial_{a_i} E(\boldsymbol{a}^*) - \lambda_i^*$ contains zero. This is true when $-D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* \geq -\lambda$.

To show 2, let $\boldsymbol{a}^* \in C$. Then by Theorem 1, for all $i = \{1, \ldots, p\}$, there exists $\mu_i^* \geq 0$ such that

$$0 \in \begin{cases} -D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* + \lambda, & \text{when } \; a_i^* \neq 0 \\ -D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* + [-\lambda, \lambda] - \mu_i, & \text{when } \; a_i^* = 0. \end{cases} \tag{2.25}$$

We now define $\boldsymbol{u}^*$ as follows, considering two cases.

Case 1: $a_i^* \neq 0$. We choose $u_i^* = \sigma^{-1}(a_i^*)$. Hence, $a_i^* = \sigma(u_i^*)$ and by (2.25) we will have $f_i(\boldsymbol{u}^*) = -D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* + \lambda = 0$.

Case 2: $a_i^* = 0$. By (2.25) there exists $\beta_i \in [-\lambda, \lambda]$ such that $0 = -D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* + \beta_i - \mu_i$. Hence, if we choose $u_i^* = \beta_i - \mu_i$, then we will have $u_i^* \leq \lambda$ and hence $a_i^* = \sigma(u_i^*) = 0$, noting that $\mu_i \geq 0$ and that $\beta_i \leq \lambda$. In this case, we also have $f_i(\boldsymbol{u}^*) = 0$.

The proof is complete. $\square$

It is to be noted that in all the stable numerical examples performed in this work, the assumption $-D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* \geq -\lambda$ for i when $a_i^* = 0$ is satisfied. This can also be achieved if, for example, we let σ' be a very small number (instead of zero) such that reaching a fixed point with $f_i(\boldsymbol{u}^*) = 0$ would imply $-D_i^\top \boldsymbol{x} + D_i^\top D\boldsymbol{a}^* + \lambda = 0$.

Chapter 3

Numerical Methods and Results

Theorem 4 and Theorem 5 enable the computation of both the NSO problem with the generic ReLU activation function and restricted NSO problem with the sequence of sigmoid activation functions (as an approximate BSO problem) by computing the corresponding gradient systems. Specifically, by Theorem 4, we know the solution flows of the gradient system (2.10), with the activation and its derivative given in either (2.13)-(2.14) or (2.17)-(2.18), converge to F, the set of fixed points of the gradient system. Hence if we integrate the gradient system in time and compute the flow at a large terminal time, we would expect to have an approximation of a fixed point of the system, say $\boldsymbol{u}^*$. Next, by Theorem 5, we know that $\boldsymbol{a}^* = \sigma(\boldsymbol{u}^*)$ will be an approximate minimizer of the sparse optimization problems found in (2.1) and (2.16).

It is important to note the IVP of (2.5)-(2.6) for this work is defined in weak sense, and hence special numerical treatment may be required. However, we ignore jump discontinuities for the initial numerical examples and simply use standard Runge-Kutta (RK) methods for proof of concept. The implementation of more specified methods, discussed in the review paper addressing discontinuous right hand sides of IVPs for ODEs [36], will be the subject of future work.

3.1 NSO Process

We can summarize the approach to solving NSO, (2.1), as follows:

1. Define any positive constraints on the solution vector $\boldsymbol{a}$.

2. Find a strictly increasing activation function σ defined to be 0 when less than λ and also mapping any input into the constrained range of $\boldsymbol{a}$.

3. Using σ, define a dynamical system of differential equations $\boldsymbol{f}(\boldsymbol{u}(t))$ with solution flow $\boldsymbol{u}(t)$ that converges to the stationary points of (2.3).

4. Numerically solve for $\boldsymbol{u}(t)$.

5. Apply σ to the resulting converged solution flow $\boldsymbol{u}(t)$ to get a final sparse code $\boldsymbol{a}$.

3.2 Approximating BSO

When solving for BSO, (2.2), we cannot use the gradient system approach directly. There are two separate procedures for approximation.

3.2.1 NSO Solve and Post Process

One approach is:

1. Define the constraints $\boldsymbol{a} \in [0, 1]^p$.

2. Find a strictly increasing activation function σ defined to be 0 when less than λ and also maps any input into [0,1].

3. Using σ, define a dynamical system of differential equations $\boldsymbol{f}(\boldsymbol{u}(t))$ with solution flow $\boldsymbol{u}(t)$ that converges to the stationary points of (2.3).

4. Numerically solve for $\boldsymbol{u}(t)$.

5. Apply σ to the resulting converged solution flow $\boldsymbol{u}(t)$ to get final sparse code $\boldsymbol{a}$.

6. Force any component of solution vector $a_i > 0$ to be exactly 1, or apply some threshold value in (0,1) where anything below threshold goes to 0 and anything equal to the threshold or higher is defined to be 1.

We know that this first approach is guaranteed to converge to the solutions of (2.16), and any post processing should give reasonable results to the BSO problem.

3.2.2 NSO Non-autonomous System Converging to BSO

Alternatively, we can define a non-autonomous, which has a time-dependent activation function $\sigma(u_i(t), t)$ that converge to the non-differentiable step function. Here we proceed by,

1. Define the constraints $\boldsymbol{a} \in [0, 1]^p$.

2. Find a series of strictly increasing activation functions $\sigma(u_i(t), t)$ defined to be 0 when less than λ and also mapping any input to [0,1]. $\sigma(u_i(t), t)$ must converge to the step activation function.

3. Using $\sigma(u_i(t), t)$, define a dynamical system of differential equations $\boldsymbol{f}(\boldsymbol{u}(t), t)$ with solution flow $\boldsymbol{u}(t)$.

4. Numerical solve for $\boldsymbol{u}(t)$ for an extended period of time when the activation functions are likely to be very close to the step function.

5. Apply $\sigma(u_i(t), t)$ to the resulting converged solution flow $\boldsymbol{u}(t)$ to get final sparse code $\boldsymbol{a}$.

3.3 Numerical Examples

Here we show a few numerical examples of the different techniques to demonstrate their efficacy. For this work, we solve for a single 7×7 pixel patch of a Fashion MNIST [54] image using a trained dictionary D of size 49×64 and set our $\lambda = 1.4$. Details can be found in Chapter 5.

3.3.1 NSO Numerical Examples

First we demonstrate the convergence of our generic ReLU activation function (2.13) for solving (2.1) where the solution coefficients can be any positive value which can be seen in Figure 3.1. The original image and final reconstruction of the 16 7×7 patches can be seen in Figure 3.2.

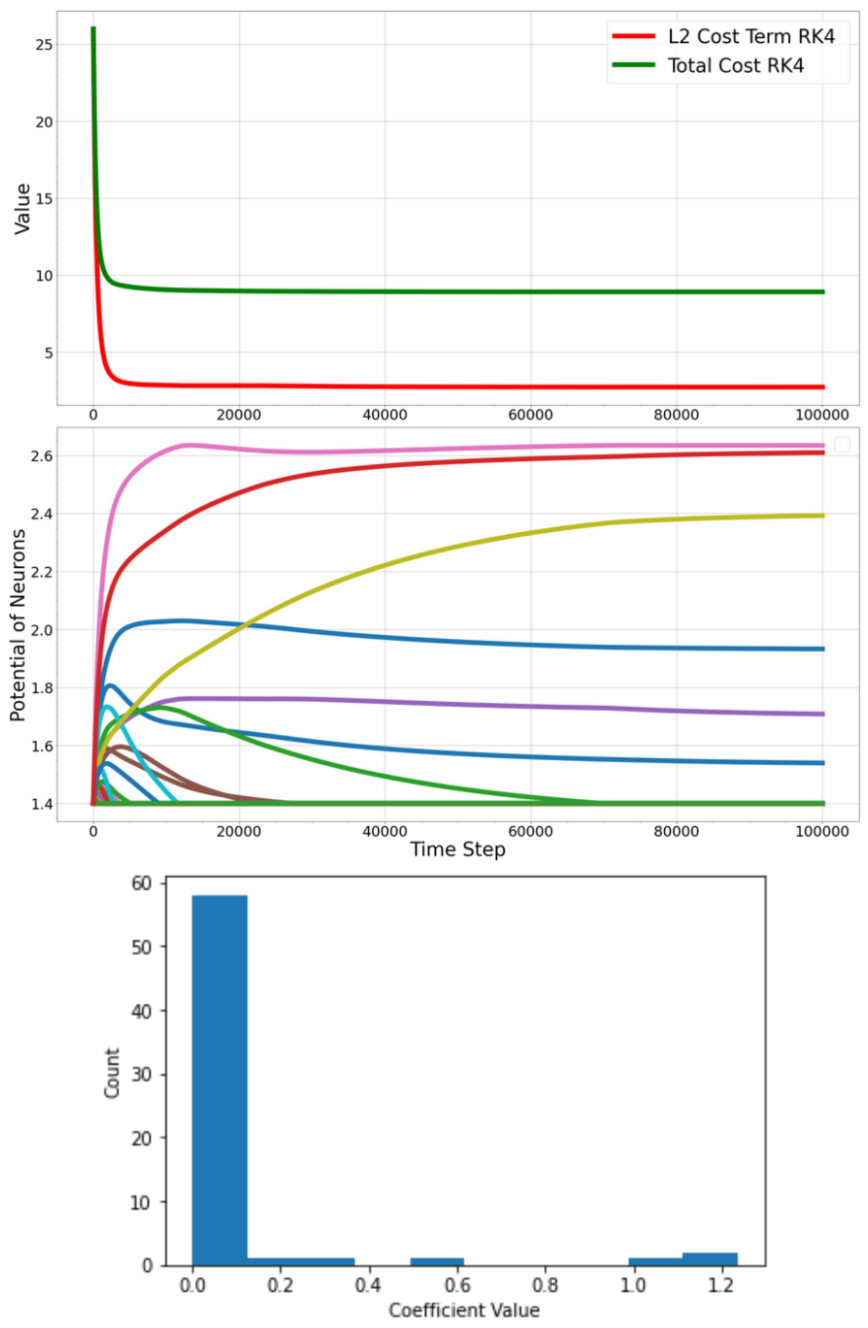


Figure 3.1: **Top Panel: Convergence of reconstruction error and total cost for** (2.1)**. Middle Panel: Trajectories of the 64 components of the solution flow in** (2.10)**. Bottom Panel: Histogram of coefficients for active elements.** Notice they are not constrained to be between 0 and 1. There are 6 active features making the solution 91 percent sparse.

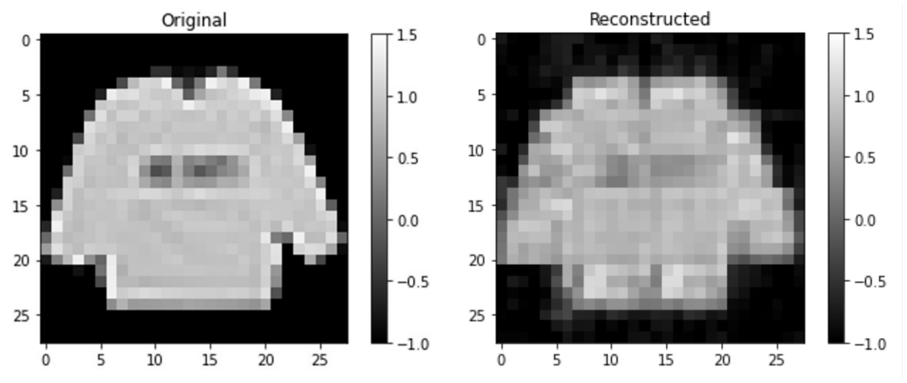


Figure 3.2: **Original image and final reconstruction of the 16 7×7 patches demonstrating the efficacy of the technique with 90% sparsity on average.**

Generic ReLU activation functions converge to the same minimum, but at different rates and with varying numerical stabilites. The relationship can be seen in Figure 3.3.

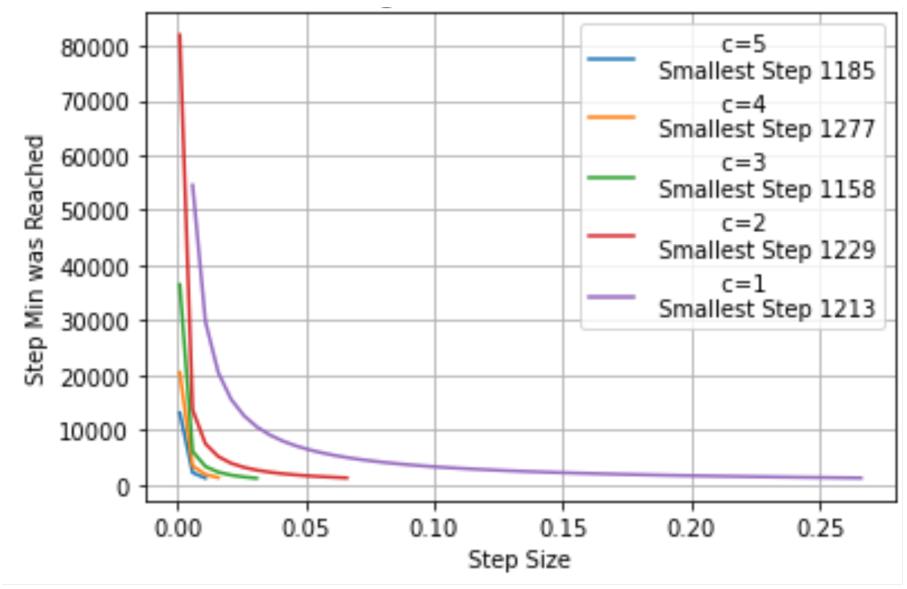


Figure 3.3: **Relationship between the convergence rate and stability of different slopes c on the generic ReLU activation functions.** All systems converge as in Figure 3.1. The steeper the slope, the faster the convergence, but less numerically stable. Points not shown either didn't converge or became numerically unstable.

3.3.2 BSO Numerical Examples

Next, we will study varying constants k for the autonomous system in (2.17). The convergence behaviour of the total cost function and potentials for all k values are similar to what is seen in Figure 3.1. In Figure 3.4, we observe that reconstructions remain valid for various levels of k. With an increase in k, the coefficient distributions tend to be more binary. Following post processing, both the average values of the cost function and the average feature activities exhibit a decreasing trend as k ascends. Sequentially from left to right, the total average cost values stand at 33.17, 32.8, and 9.7, and the average feature activities from the 64-sized dictionary are at 8.56, 8.44, and 3.75, respectively.

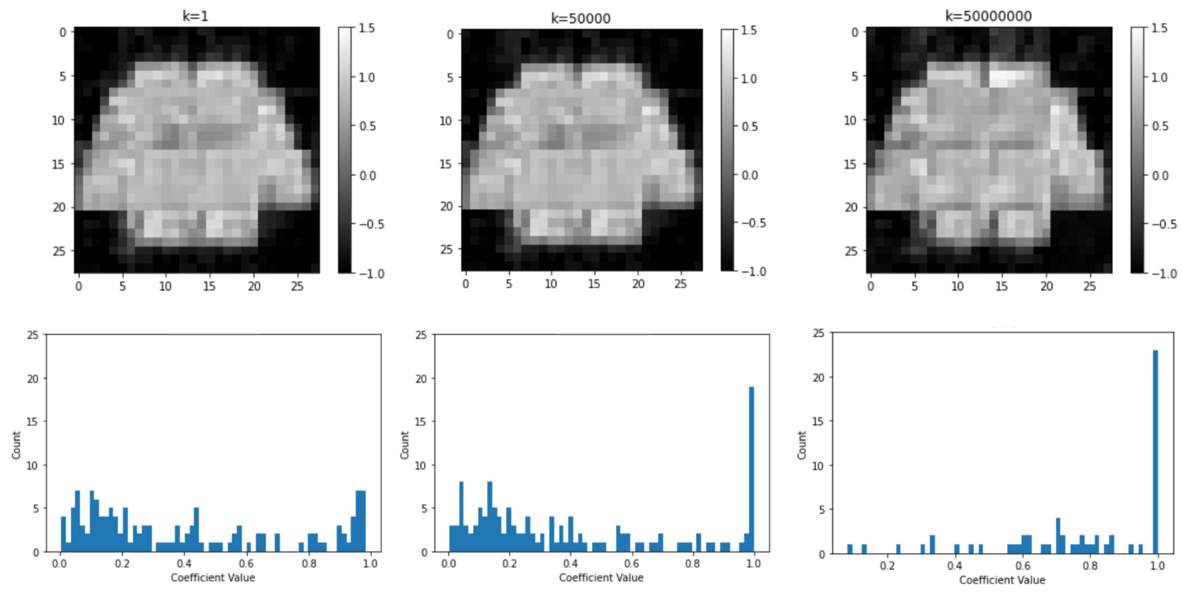


Figure 3.4: **Top panel: Reconstruction for different values of k. Bottom panel: Histograms of active coefficients over 16 patches (zero valued coefficients are omitted).** As k increases, the distribution of coefficients becomes more binary. After post processing, the average cost function values and average activity of features also decrease as k increases. In order from left to right, total average cost values are 33.17, 32.8, and 9.7 while the average feature activities out of the size 64 dictionary are 8.56, 8.44, and 3.75.

3.4 Convergence of NSO to BSO

As stated before, we are unable to show convergence for non-autonomous systems of the form in (2.24). However, the numerical example in Figure 3.5 shows convergence of a linear function $k(t)$.

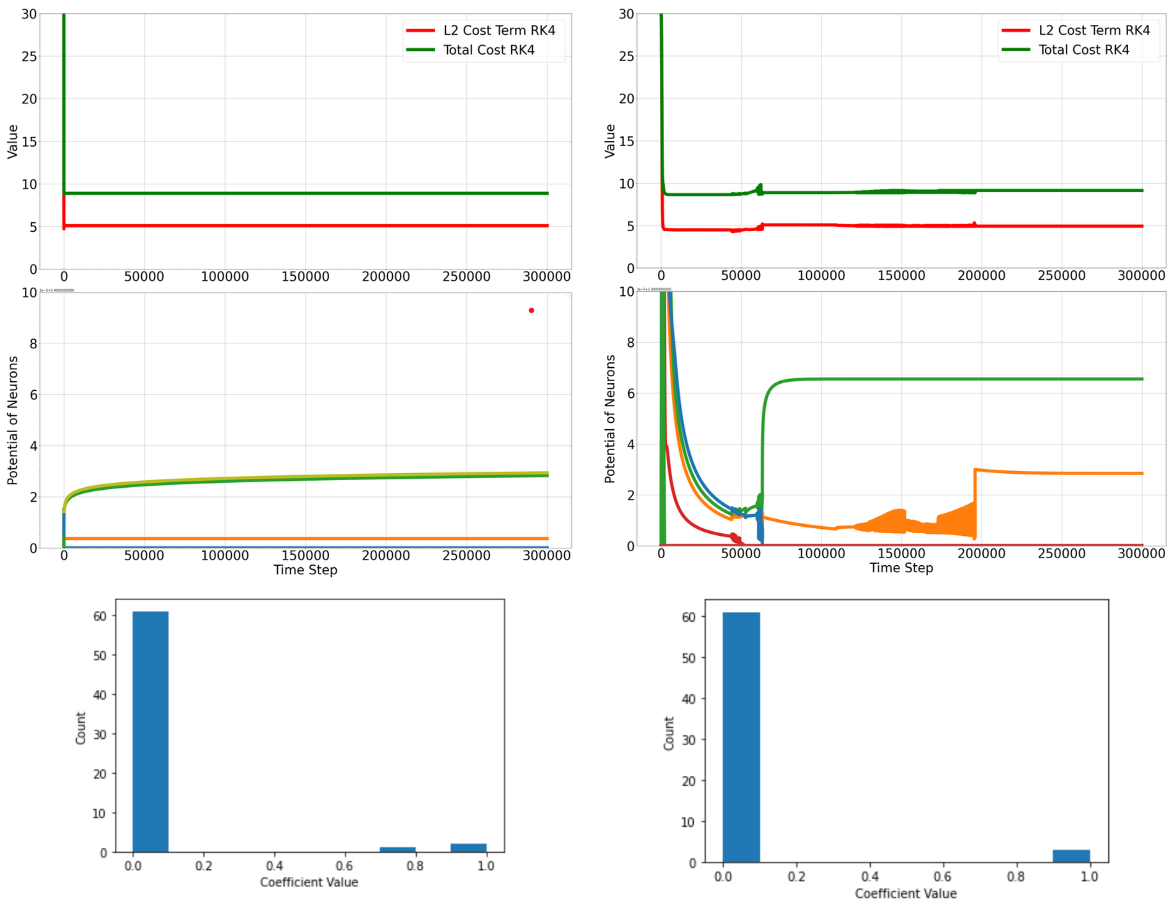


Figure 3.5: **Left Panel: Convergence of fixed Convergence rate of $k = 50000$ and final solution histogram. Right Panel: Convergence of fixed Convergence rate of $k(t) = 2.5t$ and final solution histogram.** Fixed k does not converge to a binary solution, but finds the same binary solution as the non-autonomous system (which always converges to binary) after post processing.

Further numerical experiments show tuned fixed k values do not always get to the same quality of binary solutions after post processing as the non-autonomous approach across multiple images. Further details on the comparisons will be shown in Chapter 5 in Figure 5.5.

Chapter 4

Approximating NSO on a Spiking Neuromorphic Processor

4.1 Background

Computational neuroscience frequently delves into understanding how intricate phenomena arise from neural networks. This understanding typically comes from crafting models that closely depict the physics that control communication within extensive neuronal systems. For practical applications, a solid mathematical base is essential to offer theoretical insight into the convergence and efficiency of synthetic systems. In this context, we spotlight sparse coding models, known to mirror the response traits and receptive field patterns of neurons in the primary visual cortex (V1) [41, 45]. The sparse coding problem described in previous sections has also been described by Rozell et al. [45] as a recurrent network termed the Locally Competitive Algorithm (LCA) to represent the local lateral competition that is observed in V1 [8].

Sparse coding models with a biological basis, such as LCA, captivate the neuromorphic community due to their capability to emulate significant aspects of biological sensory processing, as noted by [56, 57]. Additionally, they maintain relevance in machine learning contexts [38, 51]. As spiking implementations on neuromorphic systems often diverge substantially from non-spiking versions on conventional computing platforms, it is pivotal to discern these differences, especially when considering critical deployments.

Previously, [48] highlighted that the sparse coding objective function in a simulated spiking neural network (SNN) with integrate-and-fire neurons converged to the same solution as NSO when the activation functions are unit slope ReLUs. Following this, Fair and colleagues showed that a spiking LCA on the TrueNorth neuromorphic platform [7] aligned closely with its non-spiking counterpart on traditional hardware, albeit under stringent model conditions [16].

More recently, an LCA demonstration on the Intel's Loihi neuromorphic processor was conducted by [15], using leaky-integrate-and-fire (LIF) neurons. Interestingly, their model solely contained inhibitory connections, in contrast to the non-spiking LCA, which incorporates both excitatory and inhibitory links. Their findings showed a consistent decline in the sparse coding objective function when assessing average firing rates as a neural activity metric. Yet, a direct coefficient comparison between NSO and the spiking LCA remains unexplored, leading to an unresolved question about the fidelity of spiking LCA on neuromorphic hardware when applied to intricate, authentic challenges.

In this chapter, we build upon the earlier spiking version of LCA, denoted as S-LCA [15]. We present an enhanced S-LCA that incorporates both excitatory and inhibitory lateral connections (illustrated in Figure 4.1). We demonstrate that this modification aligns closely with the non-spiking, analog LCA (referred to as A-LCA). Undertaking a detailed neuron-to-neuron analysis, we compared the S-LCA executed on cutting-edge neuromorphic hardware to the solutions obtained on a CPU, specifically Intel's Loihi, against A-LCA on traditional computing platforms (e.g., NSO with unit ReLU on a CPU). Our findings highlight that our S-LCA rendition closely mirrors A-LCA at both individual neuron and system scales. This superior resemblance, in comparison to the earlier S-LCA model, is largely attributed to the addition of excitatory lateral connections.

4.2 S-LCA and Convergence to A-LCA

First, we present an overview of the convergence proof provided by Tang et al. [50] for a LASSO problem with strictly positive connectivity weights $w_{i,j}$ and extend the result into a regime where both positive and negative weights are present.

Define the only independent variable in our spiking network as the soma currents $\mu_i(t)$ for the p neurons, which receive a constant input bias $b_i = D_i^T \boldsymbol{x}$ and maintain an internal electric potential $v_i(t)$. When an electric potential reaches a firing threshold ν_f at a time $t = k$, the corresponding neuron simultaneously fires a spike to either inhibit or excite the other $p-1$ neurons and resets its potential to ν_r. Let $\alpha = e^{-t}$ and define the soma currents of the other neurons to change in the following manner:

$$\mu_j(t) = \mu_j(t) - w_{ji}\alpha(t - t_{i,k}) \tag{4.1}$$

Next, define $\varphi_i(t) = \sum_k \delta(t - t_{i,k})$ as the sum of Dirac delta functions δ whenever the neuron spikes over the simulation time. This leads to the final defining equations of soma currents:

$$\mu_i(t) = b_i - \sum_{j \neq i} w_{ij}(\alpha * \varphi_j)(t) \tag{4.2}$$

$$\dot{\mu}_i(t) = b_i - \mu_i(t) - \sum_{j \neq i} w_{ij}\varphi_j(t) \tag{4.3}$$

The instantaneous spike rate $a_i(t)$ and average soma current $u_i(t)$ are defined as:

$$a_i(t) = \frac{1}{t - t_0} \int_{t_0}^{t} \varphi_i(s)ds \tag{4.4}$$

$$u_i(t) = \frac{1}{t - t_0} \int_{t_0}^{t} b_i - \sum_{i \neq j} w_{i,j} (\alpha_u * \varphi_j)(s) ds \tag{4.5}$$

Leading to the spiking analog of our unit slope ReLU NSO system found in equation (2.13) with $c = 1$ as:

$$\dot{u}_i = b_i - u_i - \sum_{j \neq i} w_{ij} a_j(t) - \frac{(u_i(t) - u_i(t_0))}{t - t_0} \tag{4.6}$$

4.3 S-LCA With Excitatory Connections

Here we make a distinction and extend the previous work and summarize a previously published paper [24]. Originally, only inhibitory connections were allowed in order to ensure the soma current magnitudes and corresponding average potentials are bounded. For a strictly inhibitory network, the maximum bound on current is defined as $B_+ = max_i b_i$ since the largest value obtainable in Equation 12 requires zero inhibition from other neurons. Moreover, [50] also showed there is a lower bound and the existence of some $R > 0$ such that $t_{i,k+1} - t_{i,k} \geq 1/R$ for all $i = 1, 2, ..., n$ and $k \geq 0$ whenever two spike times exist. We can leverage this information to show that the soma currents of our updated model are also bounded above and below. First let $A > max_{i,j} |w_{i,j}|$ and $B = max_j |b_j|$ since we know the inner product of features and biases are bounded. Using the fact $(\alpha * \varphi_j)(t) \leq \sum_{l=0}^{\infty} e^{-\frac{l}{R}} < \infty$, we can show:

$$\begin{aligned}
\|\mu_i(t)\| &= \left\| b_i - \sum_{j \neq i} w_{ij}(\alpha * \varphi_j)(t) \right\| \\
&\leq \left\| |b_i| + \sum_{j \neq i} |w_{ij}| (\alpha * \varphi_j)(t) \right\| \\
&\leq \left\| max_j |b_j| + \sum_{j \neq i} |w_{ij}| (\alpha * \varphi_j)(t) \right\| \\
&\leq \|B + nA(\alpha * \varphi_j)(t)\| \\
&\leq \left\| B + nA \sum_{l=0}^{\infty} e^{-\frac{l}{R}} \right\| < \infty, \qquad (4.7)
\end{aligned}$$

implying the soma currents are bounded from above and below. Equipped with this knowledge, we can follow the proof by [50] and state $\boldsymbol{u}(t) = [u_1(t), u_2(t), ..., u_p(t)]^T$ has at least one limit point $\boldsymbol{u}^* \in \mathbb{R}^p$ such that $\boldsymbol{u}(t_k) \rightarrow \boldsymbol{u}^*$ as the sequence $t_k \rightarrow \infty$ when $k \rightarrow \infty$ from the Bolzano-Weierstrass theorem [5].

This implies:

$$\lim_{t \to \infty} \dot{u}_i(t) = \lim_{t \to \infty} \frac{1}{t - t_0}(\mu_i - u_i) = 0. \qquad (4.8)$$

Hence $\sigma(\boldsymbol{u}(t_k)) \rightarrow \sigma(\boldsymbol{u}^*) = \boldsymbol{a}^*$, we can conclude the system converges to the same limit found in A-LCA:

$$0 = \boldsymbol{b} - \boldsymbol{u}^* - (D^T D - I)\boldsymbol{a}^*. \qquad (4.9)$$

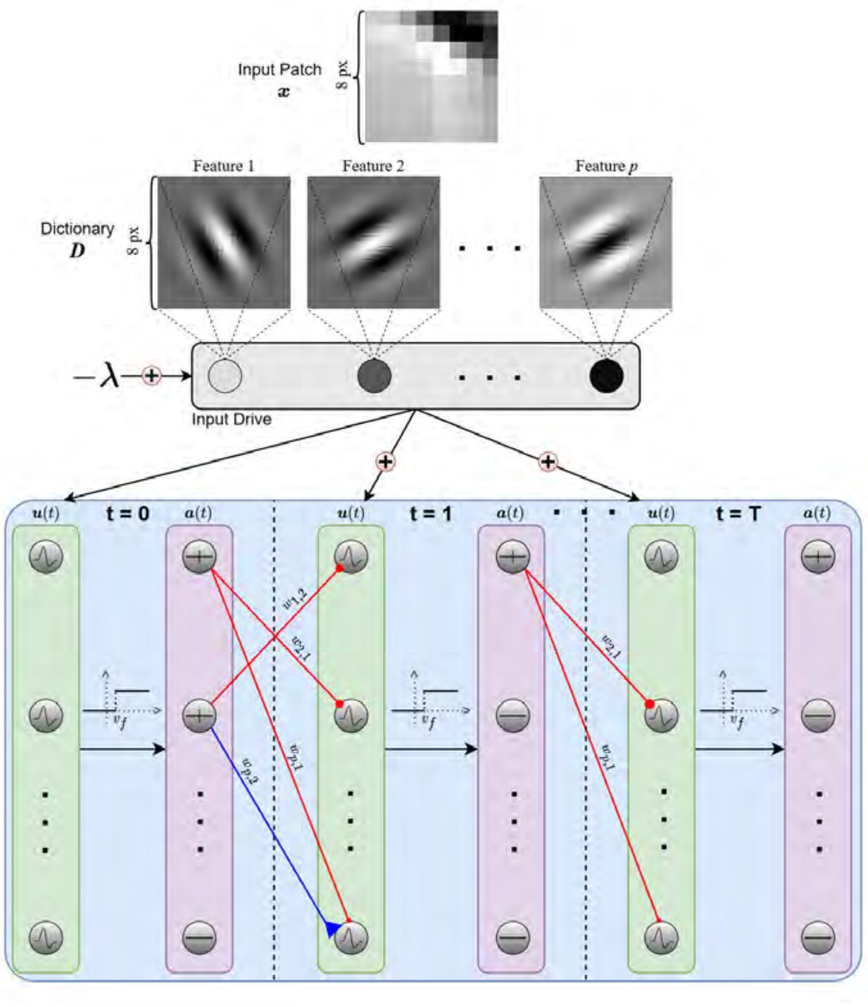


Figure 4.1: Our S-LCA implementation on Loihi. A general depiction of the S-LCA algorithm as implemented on Loihi using a single 8×8 patch as input. The input drive, which is the dot product between each neuron's feature vector and the input patch, is computed and used to initialize the membrane potentials after subtracting the A-LCA trade-off parameter λ. At each timestep, each neuron's membrane potential is charged up (or down) by the input drive and compared to a spiking threshold ν_f. Any neuron whose membrane potential is greater than ν_f will "spike", and thus inhibit (red) or excite (blue) neurons whose features overlap with its own, depending on whether the features are aligned or anti-aligned, respectively. The previous S-LCA implementation only contained inhibitory (red) lateral connections. The membrane potential is reset to zero after every spike. After T iterations, typically only a few neurons remain active. The average firing rate of each active neuron in the S-LCA model is computed over the last 1,000 timesteps for comparison with the A-LCA model. Our comparisons are performed on a 56×56 pixel image, but we use 8×8 features and a stride of 8, which is the same process depicted here but with $7 \times 7 = 49$ patches.

4.4 Unsupervised Dictionary Learning

The subsequent phase of the optimization procedure entails determining the optimal dictionary D for the specified dataset. Initially, random features populate the dictionary. Then, a stochastic gradient descent algorithm combined with a local Hebbian Learning Rule adjusts the feature vectors of active neurons, enhancing the sparse reconstruction [19]. First, let us look at the objective function for a sparse coding problem. $\boldsymbol{x} \in \mathbb{R}^m$ is the input $D \in \mathbb{R}^{mxp}$ is our dictionary and $\boldsymbol{a} \in \mathbb{R}^p$ is the sparse code:

$$E(\boldsymbol{a}(t)) = \frac{1}{2}||\boldsymbol{x} - D\boldsymbol{a}(t)||_2^2 + \lambda||\boldsymbol{a}(t)||_1.$$

We expand on the reconstruction error term for purposes of gradient descent because the sparsity penalty drops after differentiation with respect to D.

$$\begin{aligned} E(\boldsymbol{a}(t))^* &= \frac{1}{2}||\boldsymbol{x} - D\boldsymbol{a}(t)||_2^2 \\ &= \frac{1}{2}(\boldsymbol{x} - D\boldsymbol{a}(t))^T(x - D\boldsymbol{a}(t)) \\ &= \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{x} - \boldsymbol{x}^T D\boldsymbol{a}(t) - (D\boldsymbol{a}(t))^T\boldsymbol{x} + (D\boldsymbol{a}(t))^T D\boldsymbol{a}(t)) \\ &= \frac{1}{2}\sum_{i=1}^{m} x_i^2 - 2\sum_{i=1}^{m}(x_i \sum_{j=1}^{p} D_{ij}a_j(t)) \\ &\quad + \sum_{i=1}^{m}(\sum_{j=1}^{p} D_{ij}a_j(t))^2). \end{aligned} \tag{4.10}$$

Now we can differentiate E^* with respect to D_{yz} to see how each individual dictionary element changes:

$$\begin{aligned}\frac{\partial E^*}{\partial D_{yz}} &= \frac{1}{2}(-2x_y a_z(t) + 2(\sum_{j=1}^{p} D_{yj}a_j)a_z(t))\\ &= (\sum_{j=1}^{p} D_{yj}a_j(t) - x_y)a_z(t)\\ &= -r_y a_z(t).\end{aligned}$$

Where r_y represents the y'th component of the residual, we can then expand into matrix form:

$$\frac{\partial E}{\partial D} = -\begin{pmatrix} r_1a_1(t) & r_1a_2(t) & ... & r_1a_p(t)\\ r_2a_1(t) & . & & .\\ . & & . & .\\ r_ma_1 & & & r_ma_p \end{pmatrix} \tag{4.11}$$

$$= -\boldsymbol{r}\boldsymbol{a}(t)^T \tag{4.12}$$

$$= -(\boldsymbol{x} - D\boldsymbol{a}(t))\boldsymbol{a}(t)^T. \tag{4.13}$$

The Hebbian learning algorithm [19] given a single input, $\boldsymbol{x}$, is summarized in Algorithm 1. In practice, a mini-batch (i.e. an average over a fixed number of input samples) of input samples are used for each update instead of a single input sample. Since our gradient system is proportional to the derivative of the energy w.r.t D of the NSO problem, we know the learning process will descend the gradient of our neurophysiological representation. The final learned dictionary can be seen in figure 4.3.

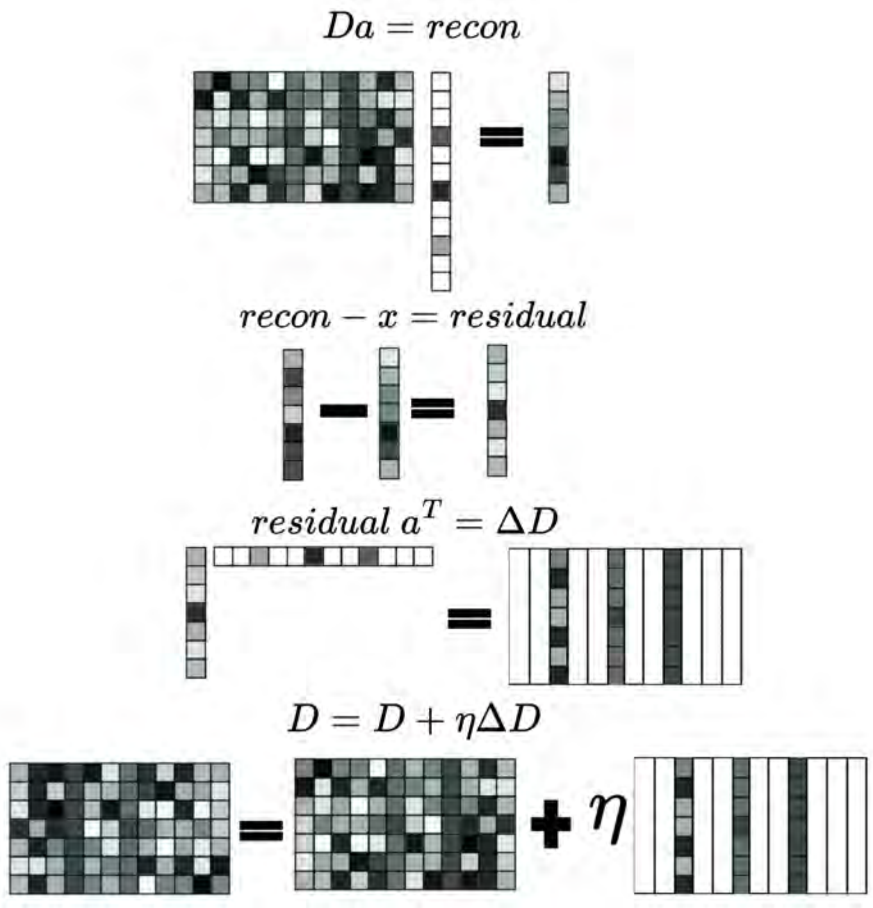


Figure 4.2: **Sparse dictionary learning.** Sparse coding is often combined with dictionary learning in an alternating fashion. After computing $\boldsymbol{a}(t)$ by minimizing 2.10 with fixed D, D is then updated as in (4.13). At each update, the dictionary only changes in the directions of the active neurons.

Algorithm 1 Dictionary Update

```
Require $\boldsymbol{D} \in \mathbb{R}^{m \times p}$, $\boldsymbol{a} \in \mathbb{R}^p$, $\boldsymbol{x} \in \mathbb{R}^m$, $\eta \in \mathbb{R}^+$
Ensure $\boldsymbol{D} \in \mathbb{R}^{m \times p}$
UPDATE_DICTIONARY($D, a, x, \eta$)
    $recon \leftarrow D \cdot a$
    $residual \leftarrow x - recon$
    $\Delta D \leftarrow residual \cdot a^T$
    $D \leftarrow D + \eta \cdot \Delta D$
    for $i = 1$ TO $p$ do
        $D_i \leftarrow \frac{D_i}{\|D_i\|_2}$
    end
    return $D$
```

4.5 A-LCA Implementation

To draw a comparison between S-LCA on Loihi and the non-spiking LCA [45], we utilized PyTorch to implement a single LCA layer through the LCA-PyTorch [42] package. More precisely, we set up a convolutional LCA layer equipped with valid padding, comprising 450 features of dimensions 8×8, and 8-stride, and a rectified soft threshold. Employing this configuration, we refined a dictionary over 5,000 updates (as per Algorithm 1) based on 50,000 grayscale images, each of 56×56 size, sourced from the COCO dataset [35] on CPU, setting λ at 0.5. This trained dictionary (as illustrated in figure 5.4) was then incorporated into both the non-spiking A-LCA and the spiking S-LCA models for benchmarking using reserved images from our COCO selection. To align the sparsity of S-LCA on Loihi to match A-LCA, we tuned the regularization parameter λ to the value of 0.73.

4.6 S-LCA Loihi Implementation and Modifications

The previous S-LCA implementation on Loihi that used only inhibitory lateral connections [**article**, 15] was structured the following way:

Neurons in the spiking network are driven by a respective bias current $\boldsymbol{b}$ (not a spiking input) that is calculated once, at the beginning of a run, as the dot product of the dictionary element and the respective patch and is scaled then scaled to the available bit space.

The weights in [**article**, 15] are chosen to be positive definite and made to work via the construction of an expanded dictionary twice the size of the original, consisting of strictly positive dictionary elements in the top half of the dictionary and inverted negative elements in the lower half. This S-LCA implementation converged towards the minimum of a different objective function in which the feature vectors lacked negative sub-units. The lack of anti-aligned sub-units prohibited more biologically realistic environments where neurons can also excite one other.

We demonstrate the addition of these excitatory sub-units, in combination with the inhibitory sub-units, give rise to a dynamical spiking system that behaves more closely to a conventional non-spiking A-LCA model (Fig. 4.4). Specifically, when features contain both excitatory and inhibitory sub-units, both positive and negative lateral connections arise naturally via taking the transpose of the dictionary dotted with itself. A given spiking neuron will now inhibit neurons with similar explanations of the same patch (positive inner product) but will excite neurons with dissimilar explanations (negative inner product). In addition, we re-implemented the ranges of biases, weights and activations such that there were no longer sign flips (integer overflow) due to the limited bit ranges on Loihi.

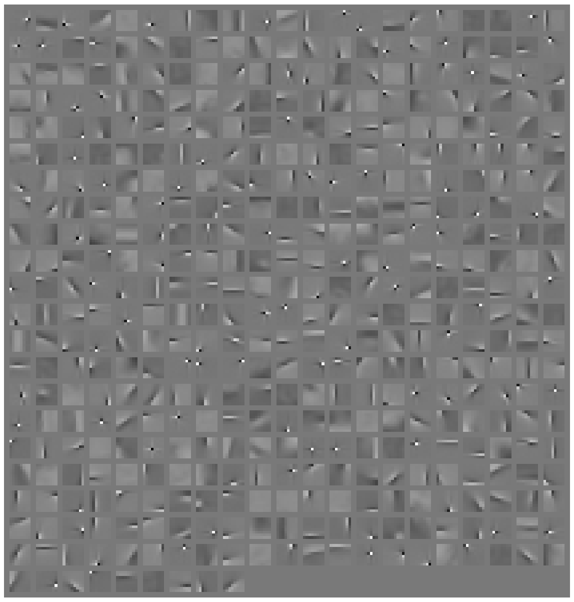
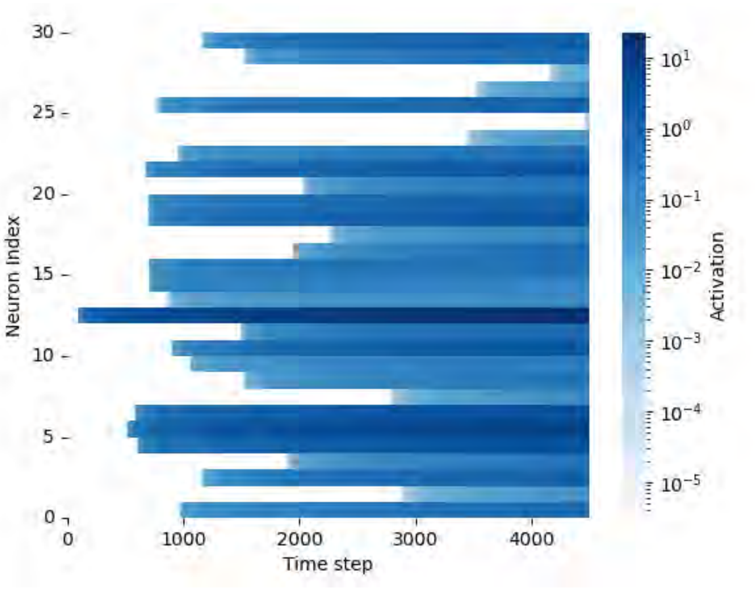


Figure 4.3: **The dictionary used by both the S-LCA and A-LCA models in our experiments.** The dictionary (D) is composed of 450 features of size 8×8.



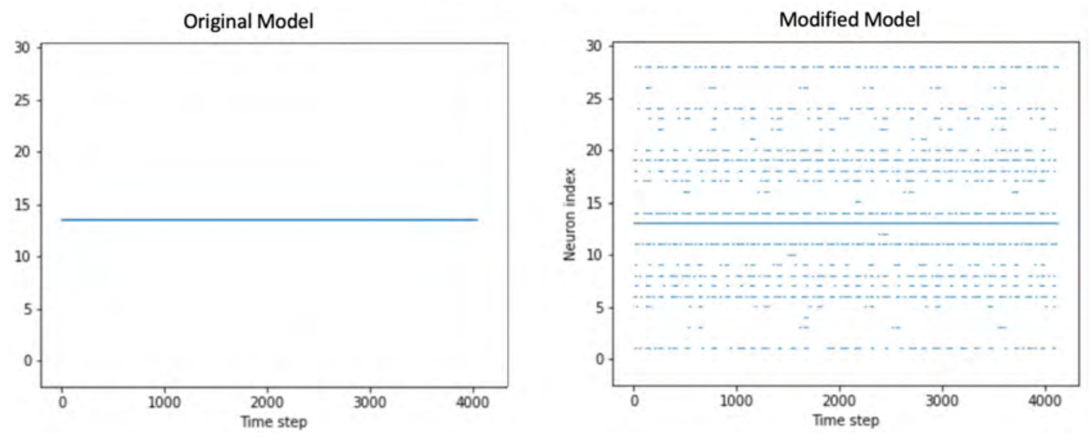


Figure 4.4: **Our modified S-LCA model contains excitatory connections as in A-LCA.** Activation when only the neuron best aligned with the input patch receives bias drive; all other biases were set to zero. A-LCA model (top) and our implementation of S-LCA on Loihi (bottom right) exhibit activity for neurons with zero input drive, while the previous S-LCA implementation on Loihi (bottom left) [15] does not excite activity of other neurons (bottom left), confirming an absence of excitatory connections.

After initializing both the S-LCA and A-LCA models with the dictionary learned in Section 4.5 (Figure 4.3), both models were run on their respective hardware using the same test image with the parameters outlined in Sections 4.5 and 4.6.

Our S-LCA exhibits closer dynamics to A-LCA than previous implementations of S-LCA by allowing only one neuron in each model to receive an input drive while all other neurons received no input drive. Since earlier S-LCA architectures contained no excitatory lateral connections, we hypothesized that only the neuron receiving input drive would be active in those models. In contrast, our S-LCA and A-LCA contains excitatory lateral connections, which should raise the membrane potential of some of the other neurons above threshold even without input drive. In Figure 4.4, we confirm this, as both A-LCA (top) and our S-LCA (bottom right) have multiple neurons active, whereas the previous S-LCA (bottom left) only has one active neuron (the only one with a non-zero input drive). In both A-LCA and our S-LCA, the same neurons appear to be active at qualitatively similar activity levels as the system converges.

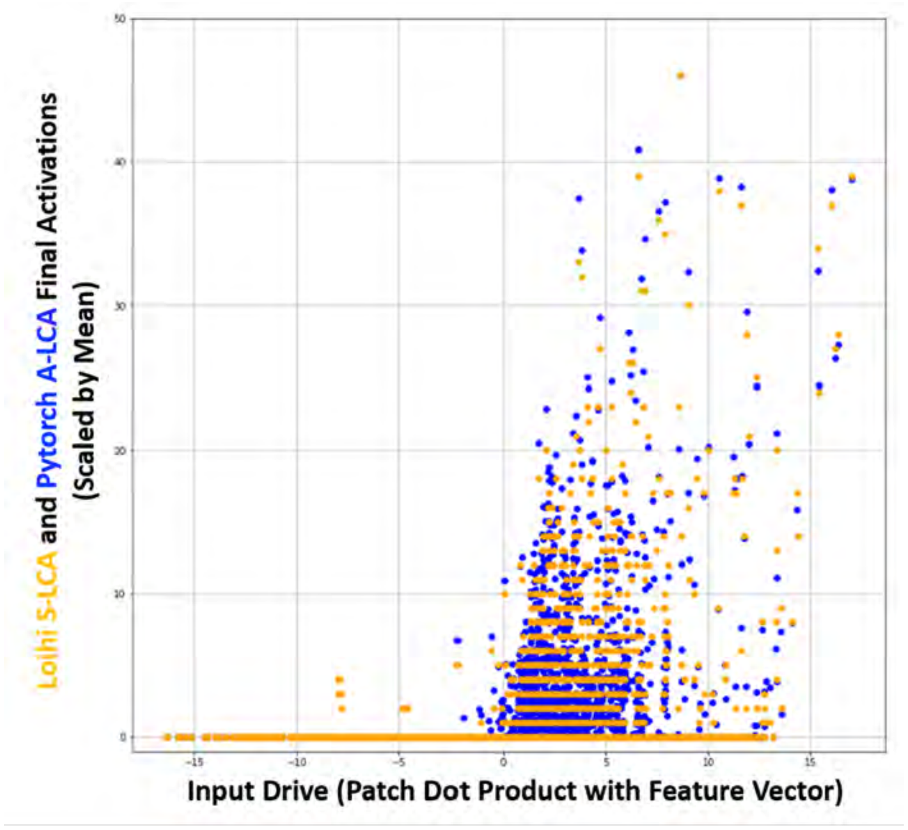


Figure 4.5: **Input drive vs. final activation.** Our S-LCA model produces a similar shape to the A-LCA model. Both models contain a few neurons that became active with features that were negatively aligned with the input, which was not true for the original S-LCA model. The distinct levels of final activity for the S-LCA model demonstrate the bit precision limitation present on the hardware.

Figure 4.5 illustrates the activation of each neuron in the S-LCA model and the A-

LCA model as a function of initial input drive. Both our S-LCA and the A-LCA contain neurons, which are active in the sparse representation despite having negative input drive (i.e., anti-aligned with the stimulus), whereas the previous S-LCA has only the driven neuron active since there was no mechanism for excitatory connections to other neurons. We can also see that our S-LCA provides a reasonable match to A-LCA despite the quantization that takes place on Loihi. Next, we compare the sparse activation of each neuron in our S-LCA directly against that in the A-LCA (Figure 4.6).

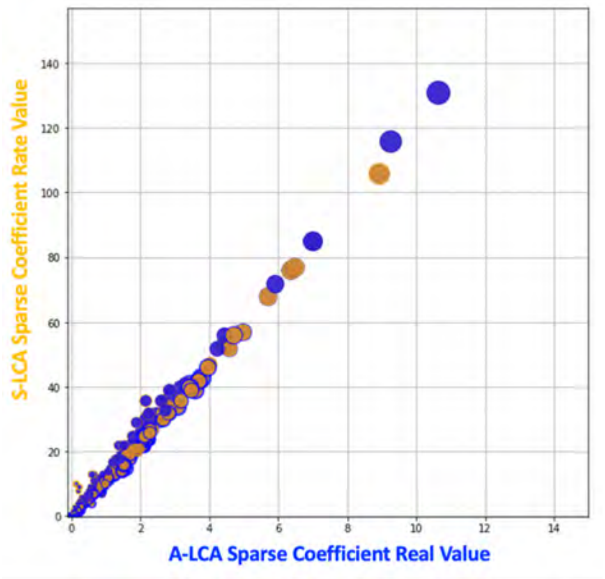


Figure 4.6: **A-LCA vs. S-LCA Matching Coefficient Values** The rate code solution for our S-LCA model is a very close match to the A-LCA solution. Each point represents a single neuron out of the $450 \times 8 \times 8 = 28,800$ neurons in our model. The difference in scale on each axis is due to how the spikes are integrated.

Here, we can see further evidence that our S-LCA performs similarly to A-LCA, as the activations lie close to the diagonal indicating that our S-LCA converged to A-LCA. Finally, we compare our S-LCA model to the A-LCA model by examining the reconstructions of the input image produced by each model from the sparse representation. By comparing the reconstructions visually, we validate that our S-LCA produces a similar sparse representation to the A-LCA. Figure 4.7 confirms that this is the case, as the reconstruction produced by our S-LCA model is very close to that produced by the

A-LCA model. We can also observe that each model is using a very similar number of features to represent each patch.

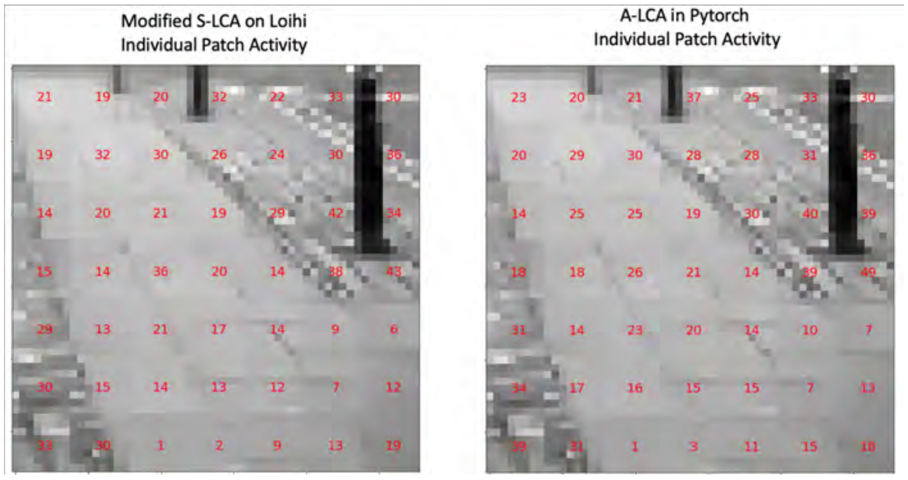


Figure 4.7: **The number of active neurons per patch in our S-LCA modelcompared to that in the A-LCA model.** The number of neurons active per patch is laid over the final sparse reconstructions of the S-LCA (left) and A-LCA (right) models, which illustrates that our S-LCA model closely matches A-LCA both at the image and patch level.

4.7 Discussion and Conclusion

We have extended the earlier S-LCA model that was restricted to inhibitory lateral connections between neurons. Our enhanced S-LCA integrates both excitatory and inhibitory lateral connections, drawing it closer in behavior to A-LCA. Initially, we validated that our S-LCA's behavior aligns with the endpoint observed in A-LCA. Subsequently, we deployed our S-LCA on the Loihi neuromorphic processor, aligning its input and dictionary with an equivalent A-LCA on CPU/GPU setups. Demonstrating for the first time a detailed neuron-by-neuron analysis, we highlighted that the sparse latent representation in our S-LCA mirrors that in A-LCA.

Our study stands as one of the rare instances where a spiking algorithm on contemporary neuromorphic hardware nearly perfectly aligns with its classical counterpart in a practical scenario. Consequently, the efficacy of our S-LCA rivals or surpasses that of A-LCA implementations universally. Our approach of S-LCA on Loihi is almost 40x more energy efficient than its classical counterpart and agrees with similar results using the device in existing literature [14][15] where faster convergence can also been seen. Such advancements pave the way for crafting swift, energy-conserving AI designs in contexts where A-LCA has showcased its merit, such as serving as a resilient foundation for convolutional neural networks [51].

However, this study does have its limitations. We have exclusively evaluated the non-convolutional scenario, adopting a stride equivalent to the patch dimension. While it is improbable that performance between our S-LCA and A-LCA would deviate dramatically in a convolutional framework, further investigation is necessary. Future endeavors might also expand our S-LCA into the spatio-temporal realm, potentially employing video streams or dynamic vision sensor data. This expansion would facilitate the creation and examination of models resonating even more with biological vision processes.

Chapter 5

QUBO

5.1 BSO and Relationship to QUBO

The non convexity of the BSO transfer function allows us to recast the problem into an Ising-model or equivalently a Quadratic Unconstrained Binary Optimization problem, known as QUBO. The problem becomes minimizing a function in the following form [23, 20]

$$H(\boldsymbol{a}; Q, \boldsymbol{h}) = \sum_{i=1}^{n} h_i a_i + \sum_{i<j} Q_{ij} a_i a_j. \tag{5.1}$$

In order to see the relationship to our problem, we write out an expanded version of our optimization function with binary variables:

$$E(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} - \boldsymbol{x}^\top D\boldsymbol{a} + \frac{1}{2}\boldsymbol{a}^\top D^\top D\boldsymbol{a} + \lambda \sum_{i=1}^{p} a_i. \tag{5.2}$$

We can see there are both quadratic and linear terms with respect to $\boldsymbol{a}$. Hence we can formulate our sparse coding problem (2.2) into QUBO form via the transformations [20, 23]:

$$h_i = -D_i^\top \boldsymbol{x} + \lambda + \frac{1}{2} D_i^\top D_i \tag{5.3}$$

$$Q = \frac{1}{2}(D^\top D). \tag{5.4}$$

5.2 D-Wave Quantum Annealer

In classical annealing, a system begins in a randomly selected initial state at a given temperature. This temperature introduces thermal fluctuations, which allow the system to cross over local energy barriers, transitioning into different energy states. The probability of these transitions is determined stochastically by the Boltzmann distribution. This means that while it is possible for the system to move to a higher energy state, it is exponentially more probable for it to transition to a lower one. As the temperature decreases, these annealing systems tend to gravitate towards increasingly lower energy states. This process is iteratively performed with varying random initial conditions. Out of all these iterations, the lowest energy state achieved is taken as the computational result.

Quantum annealing, on the other hand, presents significant differences when compared to its classical counterpart. Instead of initializing in a single, randomly selected state, a quantum annealing system is set in a state that is a quantum superposition of all potential states. To exemplify this, consider the D-Wave quantum annealing machine. Initially, each qubit experiences a transverse magnetic field while no interaction or coupling occurs between these qubits. Thus, a D-Wave machine with N qubits begins in a state representing the superposition of all 2^N possible observable states. One of the primary reasons for the enhanced computational prowess of quantum annealers is this ability to begin in a superposition of all potential states, allowing for a more comprehensive sampling of the entire energy spectrum.

In classical annealing, the temperature is slowly reduced to guide the system towards its lowest energy state. However, in quantum annealing (using the D-Wave as an example), the desired Hamiltonian (specified by the user) is progressively activated while the transverse magnetic field is simultaneously decreased. Instead of overcoming energy barriers by "hopping" over them as in classical systems, quantum annealing leverages quantum tunneling to transition to new energy states. Theoretically, this quantum tunneling enables quantum annealers to bypass getting trapped in local energy minima.

However, there is a practical challenge when using the physical D-Wave device. Given its connectivity constraints, physical qubits need to be "chained" together to achieve the full connectivity demanded by a majority of machine-learning algorithms, and especially fully connected Hopfield networks. This chaining significantly reduces the number of "logical" qubits, placing it at least an order of magnitude below the count of "physical" qubits [23, 22].

5.2.1 Choice of D-Wave Parameters

With newer generations of the D-Wave quantum annealers, more and more features have been added that allow the user a greater control over the anneal process. The specific parameters being used are listed in this section.

One necessary consequence of the minor embeddings are the presence of chains, that is the representation of a logical qubit as a set of physical hardware qubits on the chip. However, after annealing, it is not guaranteed that reading out chained qubits all take the same value (either zero or one), although they technically represent the same qubit. Such a chain is called "broken". To arrive at a value for the logical qubit in eq. (5.1), we used the *majority vote* chain break resolution algorithm [53, 33].

We employ the D-Wave annealer with an annealing time of 100 microseconds, and we query 1000 samples per D-Wave backend call. To compute the chain strength, we employ the uniform torque compensation feature with a UTC prefactor of 0.6. The UTC computation, given a problem QUBO, attempts to compute a chain strength which will minimize chain breaks while not too greatly disrupting the maximum energy scale programmed on the device [13].

5.3 Loihi Neuromorphic Chip Implementation

Intel's Loihi 1 is the first generation neuromorphic computing device that draws inspiration from biology to implement spiking neural networks with neurons as the fundamental processing elements [14]. This section pulls from a previous publication [25].

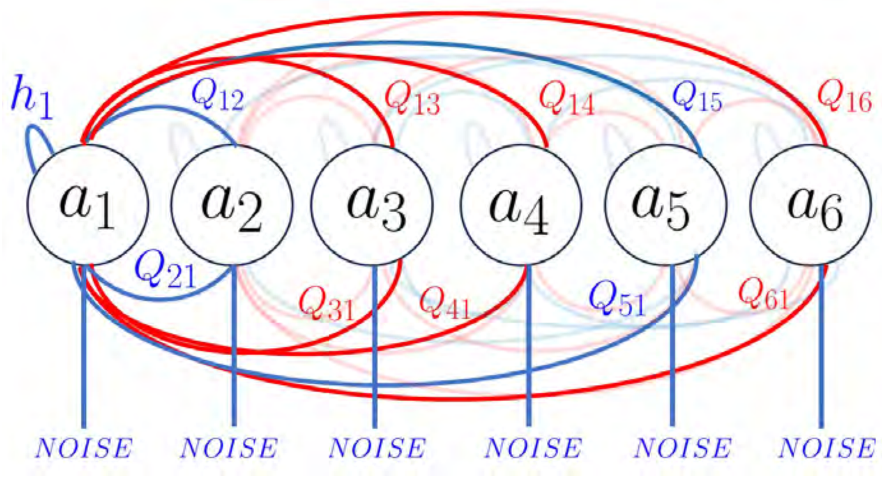


Figure 5.1: **Network connectivity of the variables in eq.** (5.1)**.** Connections include the self interaction terms h_i (symmetric weights proportional to the inner product between features), the inter-neuron connection weights Q_{ij}, and the stochastic noise input. Red is inhibitory connection and blue is excitatory. The network is sampled at different times and activity is measured for an approximate solution.

5.3.1 Overcoming Local Minima on Loihi 1

Compared to a Boltzmann machine [26], spiking networks allow for transitions between extreme objective function variable states (see Figure 5.2). Because of the limited time of activity, or forced refractory period, defined by τ, active neurons are turned off for a predetermined time, and others, which were inhibited by the active neuron now have a chance to activate. These periods allow the network to explore non-locally and facilitate the bypassing of high energy barriers in the optimization landscape [18, 28]. After the refractory period is over, previously active neurons will likely re-fire because they are receiving a strong input and a low-energy state will again be found. Figure 5.2 demonstrates this property through the substantial variation in the energy read outs obtained from Loihi 1 as a function of time. High energy read-outs correspond to refractory periods of neurons active in the ideal solution, and the repeated lowest energy reflects the return to lower energy solution states [14]. For the QuboSolver method run on Loihi 1, a threshold mantissa of 96, weight exponent of 6, and noise mantissa of 0 and exponent of 7 are used. In order to sample each QUBO on Loihi 1, a total of $2,000$ samples are measured; 4 simulation times $(5000, 10000, 15000, 20000)$ are varied over, and 5 different weight matrix scalings $(10, 100, 1000, 10000, 100000)$ are varied, with each parameter combination being sampled 100 times (this gives $4 \cdot 5 \cdot 100 = 2000$ samples per QUBO).

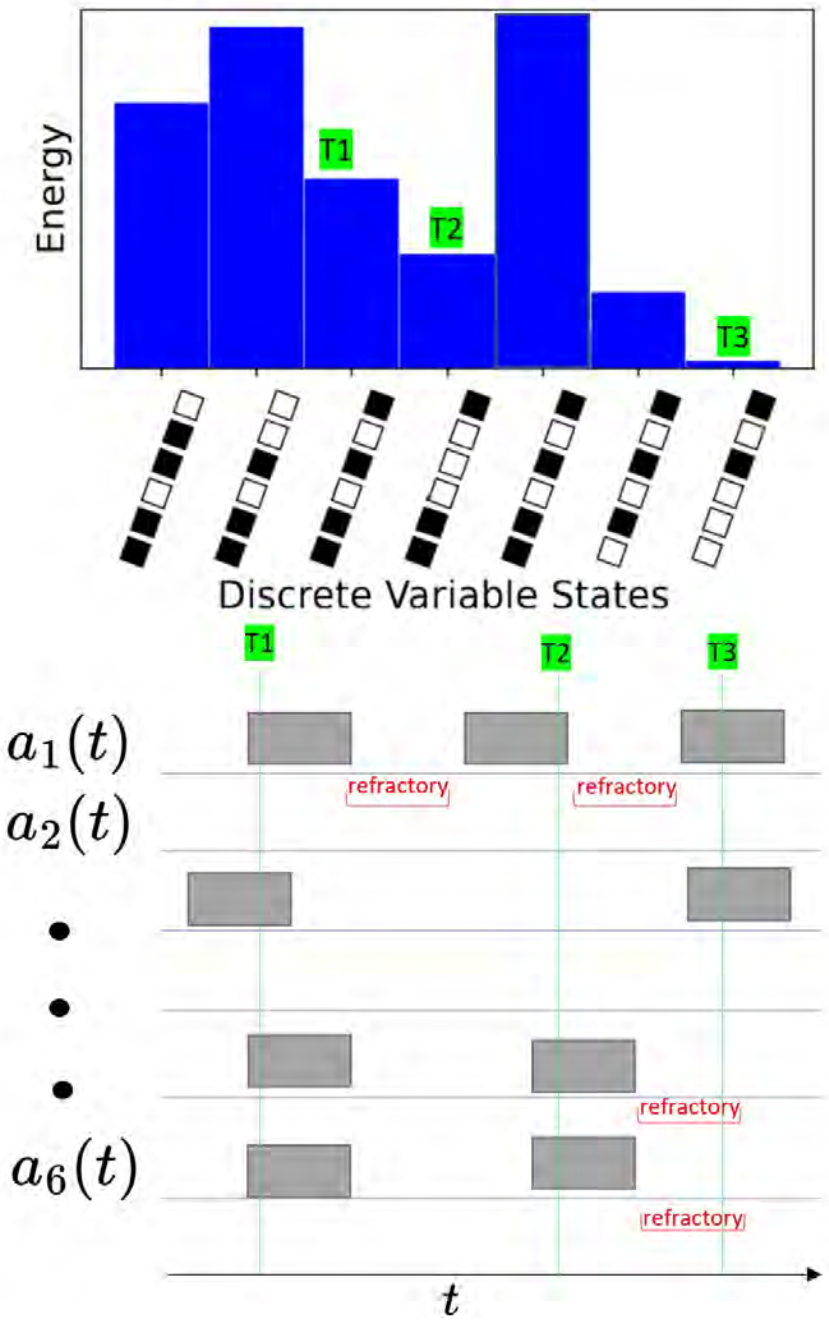


Figure 5.2: **Conceptual diagram of how we expect spike-based dynamics to support the bypassing of high-energy barriers.** Energy, e.g., the objective function evaluation for a set of variable assignments, is given on the y-axis and the x-axis shows variable assignments where ■ denotes +1 and □ denotes 0 (for the chosen number of variables of $n = 6$). In this example, the relatively sparse state of $(0, 0, 0, 1, 0, 1)$ has the lowest overall energy. When the system is sampled at different time periods T1, T2, and T3, we are able to bypass the largest energy barrier because the refractory period automatically shuts off variables 5 and 6 [28].

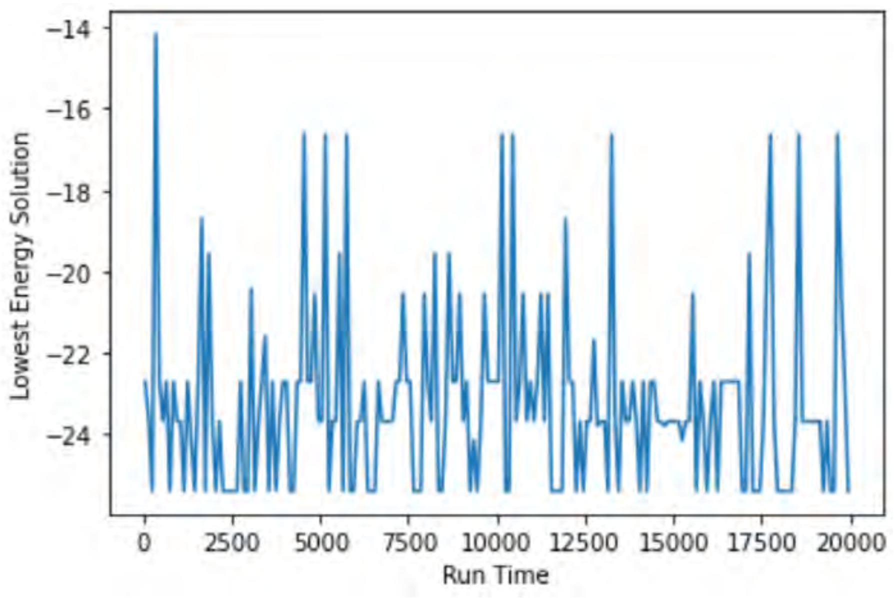


Figure 5.3: **Best solutions as a function of simulation run time.** QUBO energies read out at different simulation times (minimum of 10 readouts per simulation time) from the Loihi 1 neuromorphic processor for a single QUBO patch.

5.4 Un-normalized Dictionary Learning

Sparse coding optimization can be seen as a two-step process, where a dictionary is first learned in an unsupervised way by using a local Hebbian rule. Typically, when learning a basis for solving the convex Lasso problem, the algorithm requires the re-normalization of the columns of the dictionary D after each learning epoch. The normalization is critical for convergence in the Lasso setting because the values of the sparse vector $\boldsymbol{a}$ are allowed to take on any value. Previous work has demonstrated the ability to learn a dictionary in a QUBO regime, but this required the introduction of a new amplification parameter β to the input [23, 22]. Here, we introduce a new learning technique that allows the algorithm to find the optimal norm for features based on a predetermined desired average level of sparsity defined as $\boldsymbol{s} \in (0, 1)$. The dictionary is initialized with features drawn from a normal distribution with random norms below 1 and a small sparsity penalty parameter λ. After solving the binary sparse coding problem for each sample in the training data, the dictionary is updated. If the average sparsity over the training epoch is above the desired level **s**, the penalty parameter λ is increased for the next epoch. Pseudo code for the algorithm is presented below and the learning results are summarized in Figure 5.4. We can see the average neuron activity and reconstruction error converge along with the norms of the learned features.

We applied our technique to a patched version of the standard fashion MNIST (fMNIST) data set [54]. Each 28×28 image was broken up into 16 7×7 patches, and we selected a dictionary of size 64 in order to partition the problem into sub-problems which could be implemented on Loihi 1 (the exact number of variables for the sub-problems, is arbitrary, but fixed). Even with a smaller data structure, it was still necessary to perform our dictionary learning algorithm using the classical simulated annealing approach when solving for our sparse code in Step 6 of Algorithm 2. The NSO parameter λ was increased from 0.1 to 1.4 in increments of 0.1 to adapt to the sparsity of the solution (see the top right plot in Figure 5.4).

Algorithm 2 Dictionary Update Unormalized

```
input: $\boldsymbol{D} \in \mathbb{R}^{m\times n}$, $Train\_data \in \mathbb{R}^{b\times m}$, $\eta \in \mathbb{R}^+$, $\boldsymbol{s} \in (0,1)$, $\lambda > 0$, number of epochs $N$
1 function learn_dictionary(D, a, x, η, s, number of epochs)
2     for epoch $= 1, 2, \ldots, N$
3     $activity\_count = 0$
4     for $i = 1, 2, ..., b$
5         $x = Train\_data[i]$
          Solve for $a$
          $recon = Da$
          $residual = x - recon$
          $\Delta D = residual\ a^T$
          $D = D + \eta\Delta D$
          $activity\_count = activity\_count + sum(a)$
6     if $\frac{activity\_count}{n*b} > \boldsymbol{s}$ then
7         $\lambda = \lambda + 0.1$
8     end
9 return $D$
```

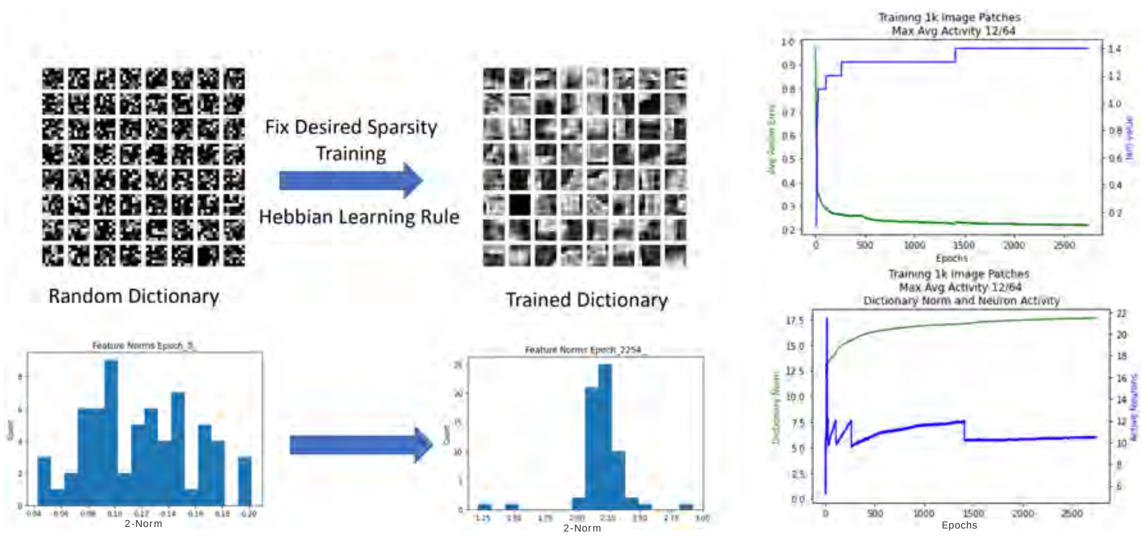


Figure 5.4: **Unormalized dictionary learning**.Randomly initialized dictionary with norms distributed between .01 and .2. After the training algorithm, norms increase and an optimal binary dictionary is learned for a fixed average activity of 12 features.

5.5 Results

Figure 5.4 visualizes the successful implementation of un-normalized dictionary feature learning. Using a local learning rule and a fixed sparsity level, we can see that the algorithm learns a better basis for reconstruction as the average error of the training data decreases over training epochs, and it also converges to the desired average sparsity level.

After successfully training each dictionary with simulated annealing (SA), a total of 16 separate QUBO models are generated. Each QUBO is then sampled using Loihi 1 (see Section 5.3) and D-Wave. The NSO problem was also solved as a non-autonomous LCA system. In order to provide a reasonable comparison against existing classical heuristic algorithms, we also sample each of the 16 QUBO models using simulated annealing. The simulated annealing implementation we use is a D-Wave SDK implementation [10], using 1000 samples per QUBO and all default settings. Using the best solutions (e.g., the computed variable assignments with the lowest energy found among all samples) from

Loihi 1, D-Wave, simulated annealing, and our non-autonomous LCA, we can reconstruct the original image from sampling all 16 QUBOs. These reconstructions are shown in Figure 5.5. Although SA has a lower mean energy, our non-autonomous system LCA and Loihi 1 are able to find reasonable solutions. D-Wave results for forward annealing trail significantly. Similar to previous demonstrations of lower power usage for certain applications [14, 24, 23, 22, 21], Loihi 1 uses an average energy consumption of ~ 0.0192 joules per sample, per QUBO matrix compared to an average energy consumption of ~ 0.115 joules per sample for simulated annealing, $\sim .2$ joules per sample for D-Wave (not including cooling overhead), and .34 joules per solve for our NSO approaches. The simulated annealing and NSO energy consumption was measured using pyRAPL [1] (including RAM power usage). The total power usage was computed by subtracting the idle machine energy consumption (for the same time duration) from the power consumption when simulated annealing was run. The Loihi 1 power consumption was measured using the NxSDK power monitoring function. Solution quality is measured by computing $\boldsymbol{a}^T Q \boldsymbol{a}$ and is an equivalent to the strictly positive objective function after the transformation. In Figure 5.5, we can see the reconstruction, solution energy, and sparsity results for all techniques presented in the manuscript.

[1]`https://pyrapl.readthedocs.io/en/latest/`

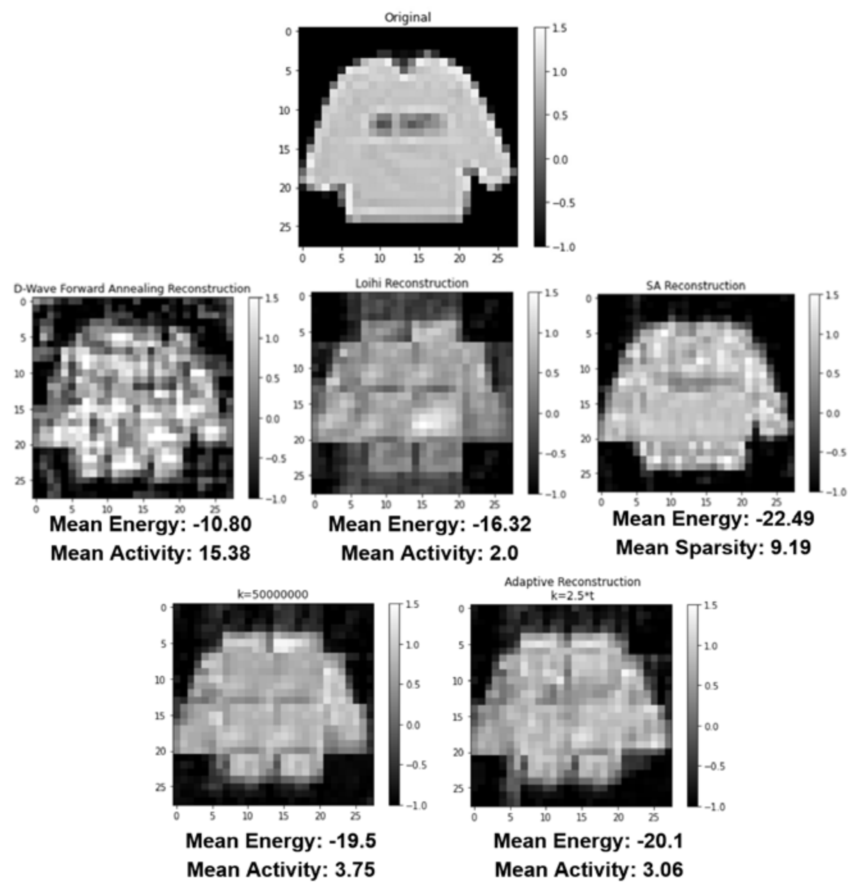


Figure 5.5: **Reconstructions from classical SA, Loihi 1, D-Wave, and the two separate BSO approximations.** Full image consists of 16 separate QUBO solves and the mean energies and sparsity levels are displayed. The sparsity levels are the mean (across the 16 QUBO models) number of variables in the lowest energy state which were in the state of +1.

Chapter 6

Related Published Papers

As we inch closer to the physical limitations of traditional computational hardware, delving into innovative computational platforms becomes crucial for the continual progress of artificial intelligence. The following five papers laid the foundation for the topic of this dissertation and demonstrate the progress made comparing quantum and neuromorphic hardware for solving the sparse coding problem. All current publications in the original forms are available at the end of the manuscript.

6.1 Machine Learning in a Post Moore's Law World: Quantum vs. Neuromorphic Substrates

Here we initiated the first comparison between novel hardware options—the D-Wave quantum annealer and the Intel Loihi 1 spiking processor—applying them to a uniform machine learning problem. To ensure a fair and valid comparison, we opt for the Fashion MNIST dataset, subjected to dimensionality reduction via sparse principal component analysis, while maintaining constant classification performance and a graph-based clustering metric. This approach facilitates a direct mapping of the problem onto both hardware types.

Our analysis spans various metrics, including power consumption, reconstruction quality, and classification accuracy. When confronted with the same meticulously constructed challenge, the two substrates exhibit comparable performance, but ultimately solve a different problem. Loihi 1 solves NSO with slope ReLU found in Chapter 4 and D-Wave solves BSO. The initial findings indicate neuromorphic and quantum systems are at early stages of development, but hold potential as viable solutions for certain classically formidable problems, such as sparse coding, by capitalizing on the unique attributes of each substrate.

6.2 Alien vs. Predator: Brain Inspired Sparse Coding Optimization on Neuromorphic and Quantum Devices

In this work we extend the previous findings by appropriately tuning and learning dictionaries for the different substrates to improve the overall performance. The Henze-Penrose statistic as a measure of classification problem difficulty is demonstrated to show the utility of dimensionally reduced Fashion-MNIST dataset. Additionally, we generate a second dataset with inverted signs and append it to the original, aiming to create a scenario where each class possesses a mean zero distribution. This setup results in data that is not readily separable by linear methods. We introduce an early-stage normalization technique tailored for Loihi, accompanied by an exploration of optimal parameter settings and unsupervised dictionary learning, applicable across all three data variations. Figure 6.1 shows the time evolution of the objective function and the motivation for early normalization.

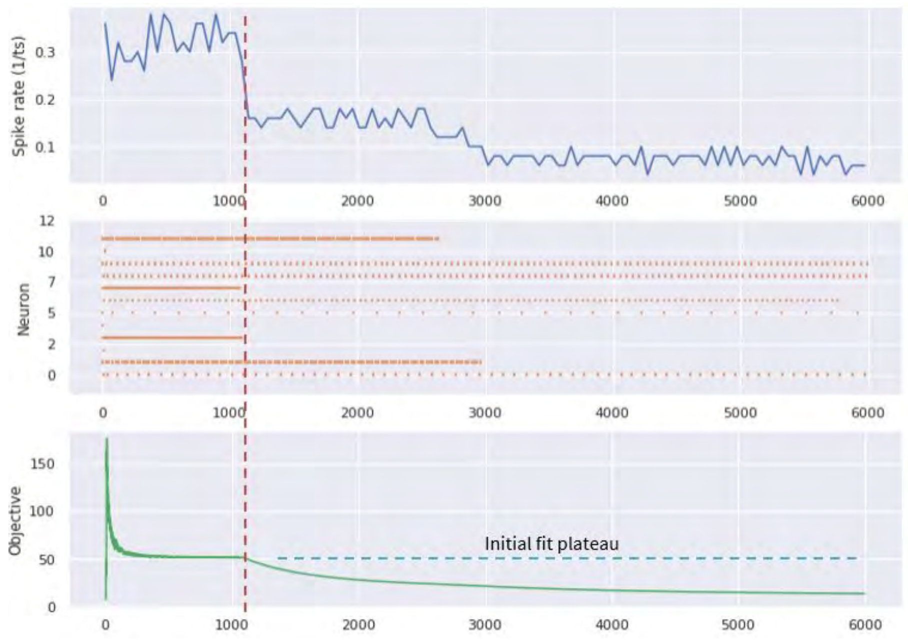


Figure 6.1: **Simulation time steps against spike rate, neuron activation, and objective function.** Red dotted line represents where most significant neuron competition has already occurred and activity is dampened out. Top Panel: Average spike rate of neurons in network. Middle Panel: A spike raster plot of how often active neurons are firing over the simulation. Lower Panel: The value of the objective function and the long regularization time after the initial fit plateau. The vast majority of the reduction in the loss function occurs early in the simulation.

6.3 Fast Post-Hoc Normalization for Brain Inspired Sparse Coding on a Neuromorphic Device

Here we extend the normalization technique and compare with solutions derived classically through the greedy orthogonal matching pursuit (OMP) algorithm executed on a standard digital processor. A thorough analysis of optimal parameter selection, reconstruction errors, and unsupervised dictionary learning for both Loihi and its classical counterpart are presented. By increasing the sparsity parameter λ, and tuning to the same sparsity level as the final solutions from the full simulation allow for an over $50\times$ speed up with almost identical solutions. Figure 6.2 demonstrates tuning the full simulation for reconstruction error and then finding the appropriate λ increase for the post-hoc approach to match sparsity levels.

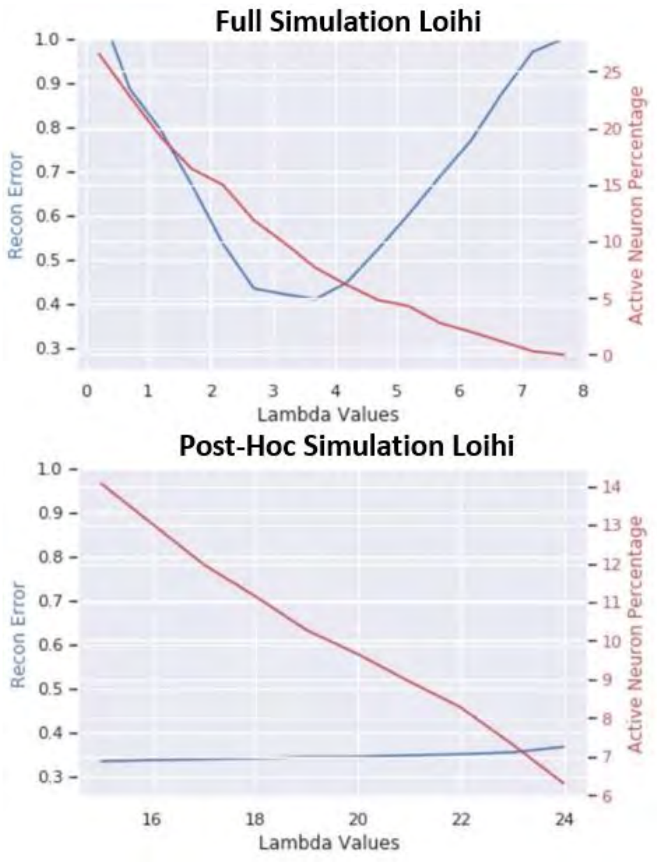


Figure 6.2: **Optimal sparsity penalty λ selection.** Each graph has reconstruction error/loss in blue and active feature percentage in red. There is a clear optimal sparsity level dictated by full simulation Loihi (Top), and post-hoc normalization technique on Loihi (Bottom) roughly exhibits a monotonic relationship. We can impose the same level of sparsity as the full time simulation Loihi on the post hoc approach by drastically increasing the penalty term λ.

6.4 Apples-to-spikes: The First Detailed Comparison of LASSO Solutions Generated by a Spiking Neuromorphic Processor

Prior research applied a spiking version of Locally Competitive Algorithm (S-LCA) on the Loihi neuromorphic processor, maintaining lateral connections solely in the inhibitory domain, contrasting the analog LCA (A-LCA) which incorporates both excitatory and inhibitory connections. Without lateral excitatory interactions, the S-LCA implementation on Loihi was able to deduce sparse representations for image patches that approached a global minimum, though a detailed analysis of the specific neural activations (i.e., the solution) was not conducted.

In this study, we initially establish that the limitations imposed on lateral connections in the prior S-LCA implementation were overly restrictive. Subsequently, we introduce an enhanced version of S-LCA that integrates both excitatory and inhibitory lateral connections. We executed this advanced S-LCA on the Loihi processor, demonstrating that the resultant sparse latent representations more accurately mirrored those determined by A-LCA. More precisely, this research conducts the inaugural comparison of individual neuron activations between S-LCA and A-LCA, illustrating that the final solution from our S-LCA closely aligns with that of A-LCA. To the best of our knowledge, this research represents one of the rare instances where a spiking algorithm, when implemented on contemporary neuromorphic hardware for a practical task, showcases a performance nearly indistinguishable from its non-spiking analog. Much of this paper can be found in Chapter 4.

6.5 Sampling Binary Sparse Coding QUBO Models Using a Spiking Neuromorphic Processor

In this work, we address the problem using an L2 norm for reconstruction error minimization and an L0 (or equivalently, L1) norm to impose sparsity on the binary vector, resulting in a Quadratic Unconstrained Binary Optimization (QUBO) problem, a typically NP-hard challenge. Our contributions are twofold. Initially, we introduce an approach for unsupervised and unnormalized dictionary feature learning, aimed at optimally aligning with the data while adhering to a predetermined level of sparsity. Subsequently, we solve the binary sparse coding problem utilizing the Loihi 1 neuromorphic chip, leveraging stochastic neural networks to navigate the non-convex energy landscape. We evaluate our solutions in comparison to the traditional heuristic method of simulated annealing. Our results indicate that neuromorphic computing is a viable option for generating low-energy solutions in binary sparse coding QUBO models. Although Loihi 1 demonstrates proficiency in producing highly sparse solutions for QUBO models, there is a necessity for enhancements in the implementation to achieve competitiveness with simulated annealing. Many of the ideas and figures in Chapter 5 are from this paper.

Chapter 7

Future Work

7.1 Quantum Evolution Monte Carlo

In order to improve the results found for D-Wave, we will use the annealer in connection with so-called Monte Carlo chain of reverse anneals, in which the best solution of any anneal is encoded as the initial state of the next anneal. This process works on the *logical* problem, meaning after unembedding of all chained qubits.

To be precise, a sequence of reverse anneals are chained together in a Monte Carlo-like process, where each subsequent round of reverse anneals is initialized with a classical state that is defined by the best solution found at the last set of reverse anneals. This chain of reverse anneals is initialized with the best solution found from a 100 microsecond forward anneal with 1000 samples. Each reverse annealing step in the chain uses `reinitialize_state=True` when executing on the D-Wave quantum annealers, which re-initializes the state of the reverse anneal after each anneal-readout cycle. This technique has been used many times in other contexts and is referred to as both "iterative reverse annealing" [55, 3, 2] and "quantum evolution Monte Carlo" (QEMC) [31, 30, 32, 29, 37].

Note that the D-Wave quantum annealer feature $h - gain$, which specifies a time dependent multiplicative term on all linear terms for all time points over the course of the anneal, can also be used in order to encode classical states into the anneal [44], and thereby also allows an iterative h-gain state encoding technique, similar to this reverse annealing technique, to be used on [43].

For all experiments involving reverse annealing, the reverse annealing schedule used was given by $\{[0,1],[10,s],[90,s],[100,1]\}$, where each pair defines a point in time (from the start to the end of the anneal process) and an anneal fraction s. The anneal fraction is the normalized time used to control how the quantum Hamiltonian is moved from the initial superposition of states to the problem QUBO during the anneal [11]. The anneal schedule is constructed by linear interpolation between those four points. The reverse anneal schedule we use is symmetric with a pause of 80 microseconds. It has an increasing and decreasing ramp on either side of a pause of duration 10 microseconds. We vary the anneal fraction s at which the pause occurs.

Moreover, we employed D-Wave with flag *reduce_intersample_correlation* enabled for all experiments, which adds a pause in-between each anneal in order to reduce correlations in the data (those correlations may exist in time due to the spin bath polarization effect, see [1]). Both parameters *readout_thermalization* and *programming_thermalization* were set to 0 microseconds. The reverse annealing specific parameter *reinitialize_state* was enabled for all reverse annealing executions, causing the annealer to reapply the initial classical state after each anneal readout cycle [12].

7.2 Neuromorphic Warm Starting

Motivated by the results found using QEMC, we implement a warm starting technique on Loihi 2 in order to improve solution quality. For a proper comparison, we ran the QUBO solver for 100 iterations and used the previous solution as the initial state for the next run while simultaneously randomly selecting refractory periods for each run.

7.3 Preliminary Results

Figure 7.1 demonstrates the different results for the s parameter in QEMC and the convergence to the exact solver solution when tuned correctly. In the same figure, we can see the Loihi 2 also benefits from the warm starting technique after several iterations. Figure 7.2 shows the improvement of both reconstructions over the same set of QUBOs presented in Chapter 5. We can also see Loihi 2 is able to provide a better initial solution than both D-Wave and Loihi 1.

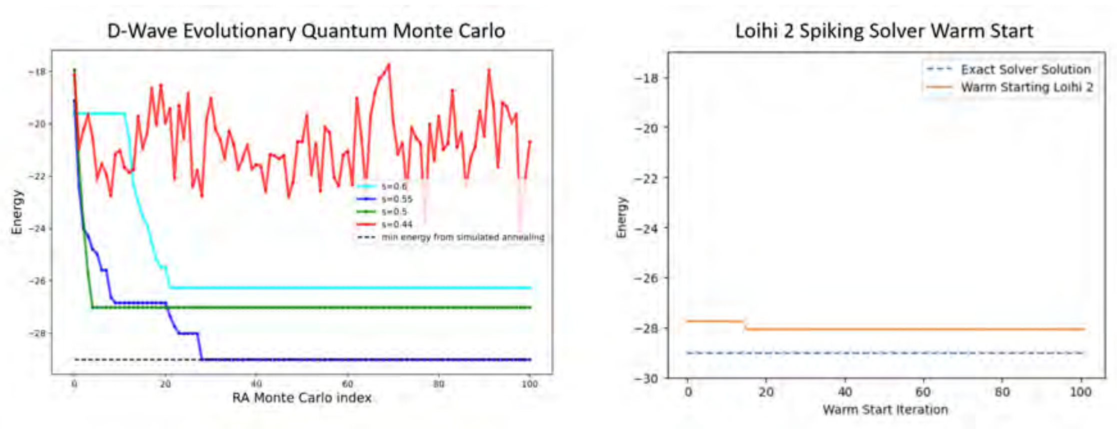


Figure 7.1: **QEMC and Loihi 2 warm starting over 100 iterations for both techniques.** Loihi 2 initially finds a good solution and sees a slight improvement over trials. After tuning the parameters, QEMC eventually obtains the same solution as the exact solver.

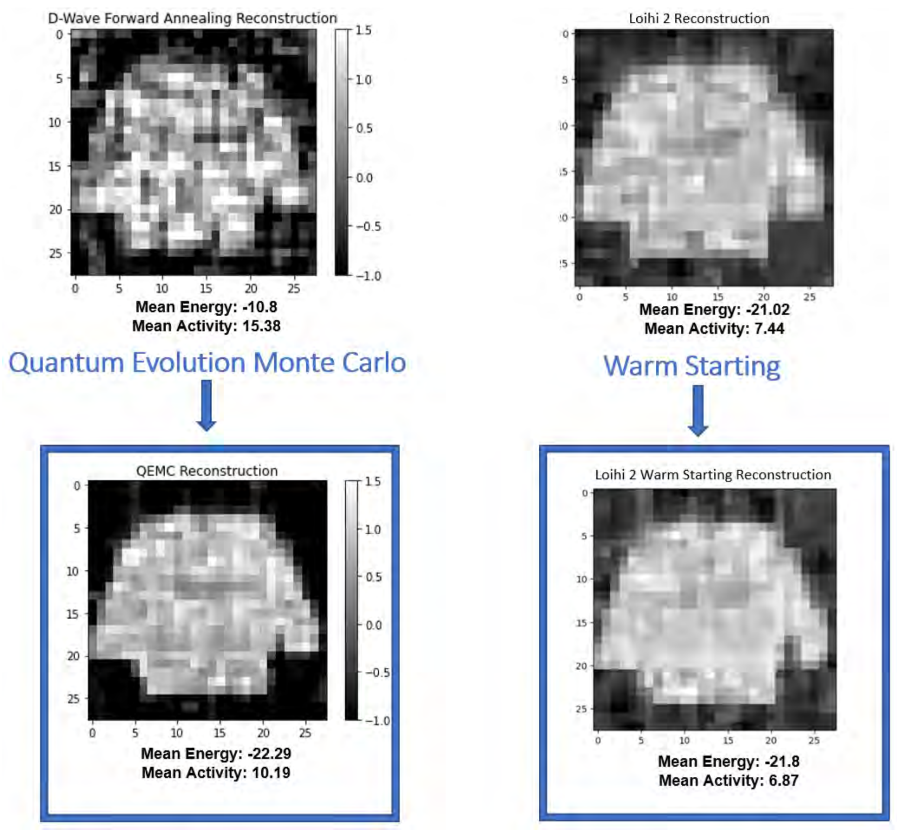


Figure 7.2: **QEMC and Loihi 2 warm starting results over the same 16 patches from Chapter 5.** The D-Wave results are drastically improved. Loihi 2 is much better than Loihi 1 and also benefits from the warm starting approach.

Chapter 8

Conclusion

In this dissertation, we have embarked on a comprehensive exploration of the intricate dynamics and properties of recurrent neural networks, particularly focusing on dynamical and gradient systems, to address constrained sparse optimization problems. The focal point of our investigation has been Hopfield networks, distinguished by the interaction and self-organization of neurons into configurations that represent the minimizer of the problem at hand. By delving deep into the mathematical underpinnings of these networks, we have strived to bridge the gaps between theoretical neuroscience, artificial intelligence, and neural computations, bringing forth a clearer understanding of their convergence behaviors in various contexts. This journey of exploration and analysis has been complemented by practical implementations on classical computers, spiking neural networks, and quantum hardware, ensuring a holistic and grounded perspective on the subject matter.

Dynamical systems, characterized by state variables and evolution rules, exhibit a plethora of behaviors ranging from stability and periodicity to chaos, contingent upon their parameters and initial conditions. Gradient systems, a subset of dynamical systems, navigate the trajectory of the system based on the gradient of a potential function, inherently seeking local minima and thereby playing a crucial role in optimization problems.

Within this spectrum, the Hopfield network stands out as a fully connected, recurrent neural network that doubles as a gradient system, directing configurations toward lower energy states and forming an energy landscape dotted with attractors and valleys. This unique capability positions it as an associative memory, retrieving stored patterns even from noisy or incomplete inputs.

Zooming in on a specific variant of the Hopfield network, the Locally Competitive Algorithm (LCA), we tackled the sparse coding problem, aiming to reconstruct input signals from a sparse linear combination of features in an overcomplete dictionary. This problem not only bears significant relevance in signal processing but also finds parallels in the neural activities of the V1 layer of the mammalian visual cortex, presenting a fascinating intersection of computational neuroscience and machine learning. In LCA, neurons are fully interconnected, competing through excitation and inhibition to converge to a sparse representation of the input.

Our investigation did not stop at the theoretical and computational analysis of these networks; we took it a step further by addressing the nuances of activation functions and their implications on convergence behavior. Previous work has predominantly centered on the computational facets of LCA for non-negative sparse optimization with unit ReLU activations, leaving gaps in understanding, especially regarding other activation functions. This dissertation fills these gaps, providing a comprehensive analysis of convergence for generic ReLU and Rectified Sigmoid activation functions, and extending the exploration to binary sparse optimization problems.

Armed with this understanding, we introduced non-autonomous systems with time-varying sigmoid activations, converging toward step functions to address binary sparse optimization problems, a domain where traditional gradient system approaches falter due to non-convex energy landscapes. Verifying our theoretical insights, we conducted numerical experiments on classical computers, further solidifying the foundation of our analysis.

The journey did not end there; we embraced the emerging realms of quantum annealing and spiking neuromorphic computing, demonstrating the relevance and applicability of our findings in these non-von Neumann architectures. By mapping the continuous domain networks to their spiking counterparts and reformulating binary sparse optimization problems into the form of QUBOs for quantum annealers, we established a bridge between classical and modern computational paradigms, showcasing the versatility and potential of our approaches. The ability of spiking neuromrophic processors to solve both the spike rate approximation of LCA with ReLU activations, as well as QUBOs directly, at significantly lower power than both classical and quantum annealing computation, provides enticing evidence for the benefits brought by brain inspired devices.

Appendix A

Appendix

A full derivation of (2.9) follows:

$$\begin{aligned}
\partial_{u_i}\widetilde{E}(\boldsymbol{u}) &= \partial_{a_i}E(\boldsymbol{a})\frac{\partial a_i}{\partial u_i} \\
&= \partial_{a_i}\left[\frac{1}{2}(\sum_{j=1}^{m} x_j^2 - 2\sum_{j=1}^{m}(x_j\sum_{i=1}^{p} D_{ji}a_i) + \sum_{j=1}^{m}(\sum_{i=1}^{p} D_{ji}a_i)^2) + \lambda\sum_{i=1}^{p} a_i\right]\frac{\partial a_i}{\partial u_i} \\
&= \left[-\sum_{j=1}^{m} x_j D_{ji} + \sum_{j=1}^{m}(\sum_{i=1}^{p} D_{ji}a_i)D_{ji} + \lambda\right]\frac{\partial a_i}{\partial u_i} \\
&= \left[-\boldsymbol{x}^\top D_{[:,i]} + D_{[:,i]}^\top D\boldsymbol{a} + \lambda\right]\frac{\partial a_i}{\partial u_i} \\
&= \left[-\boldsymbol{x}^\top D_{[:,i]} + D_{[:,i]}^\top D\boldsymbol{a} + \lambda\right]\sigma'(u_i) \\
&= \left[-\boldsymbol{x}^\top D_{[:,i]} + D_{[:,i]}^\top(\sum_{k\neq i}^{p} a_k D_{[:,k]}) + D_{[:,i]}^\top D_{[:,i]}a_i + \lambda\right]\sigma'(u_i). \qquad \text{(A.1)}
\end{aligned}$$

Sampling binary sparse coding QUBO models using a spiking neuromorphic processor

Kyle Henke
khenke@lanl.gov
Los Alamos National Laboratory, CCS-3 Information Sciences
Los Alamos, New Mexico, USA

Elijah Pelofske
epelofske@lanl.gov
Los Alamos National Laboratory, CCS-3 Information Sciences
Los Alamos, New Mexico, USA

Georg Hahn
ghahn@hsph.harvard.edu
Harvard University, T.H. Chan School of Public Health
Boston, Massachusetts, USA

Garrett T. Kenyon
gkenyon@lanl.gov
Los Alamos National Laboratory, CCS-3 Information Sciences
Los Alamos, New Mexico, USA

ABSTRACT

We consider the problem of computing a sparse binary representation of an image. To be precise, given an image and an overcomplete, non-orthonormal basis, we aim to find a sparse binary vector indicating the minimal set of basis vectors that when added together best reconstruct the given input. We formulate this problem with an L_2 loss on the reconstruction error, and an L_0 (or, equivalently, an L_1) loss on the binary vector enforcing sparsity. This yields a so-called Quadratic Unconstrained Binary Optimization (QUBO) problem, whose solution is generally NP-hard to find. The contribution of this work is twofold. First, the method of unsupervised and unnormalized dictionary feature learning for a desired sparsity level to best match the data is presented. Second, the binary sparse coding problem is then solved on the Loihi 1 neuromorphic chip by the use of stochastic networks of neurons to traverse the non-convex energy landscape. The solutions are benchmarked against the classical heuristic simulated annealing. We demonstrate neuromorphic computing is suitable for sampling low energy solutions of binary sparse coding QUBO models, and although Loihi 1 is capable of sampling very sparse solutions of the QUBO models, there needs to be improvement in the implementation in order to be competitive with simulated annealing.

KEYWORDS

neuromorphic computing, sparse coding, computer vision, spiking neural networks, unsupervised machine learning, QUBO, Loihi 1

ACM Reference Format:
Kyle Henke, Elijah Pelofske, Georg Hahn, and Garrett T. Kenyon. 2023. Sampling binary sparse coding QUBO models using a spiking neuromorphic processor. In *International Conference on Neuromorphic Systems (ICONS 2023).* ACM, New York, NY, USA, 5 pages.

ICONS '23, August 1–3, 2023, Santa Fe, NM, USA
© 2023 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0175-7/23/08.
https://doi.org/10.1145/3589737.3606003

1 INTRODUCTION

We are interested in the computation of a sparse binary reconstruction of an image. This task plays a role whenever an image of interest is not directly observable and instead must reconstructed from a limited sample or projection using compressive sensing. Sparse binary reconstruction is of interest in, for instance, the fields of radioastronomy and molecular imaging, as well as image compression [11, 15]. Sparse binary coding falls into the class of Quadratic Unconstrained Binary Optimization (QUBO). QUBO models are challenging computational problems that are difficult to solve exactly using classical algorithms due to exponential run time complexity, in general. QUBO models are a specific type of discrete combinatorial optimization problems, and in general it is of considerable interest to be able to compute optimal solutions of QUBO models more efficiently than existing methods. Networks of spiking neurons with noise have been shown to offer new opportunities for solving these problems. By programming the constraints into the architecture of a network of spiking neurons and controlling the frequency of network states during the resulting stochastic dynamics of the network, the exploration of complicated energy (e.g., objective function) landscapes describing our problem of interest can be performed in practical time.

Mathematically, given a signal $\boldsymbol{x} \in \mathrm{R}^m$ and an overcomplete and non-orthonormal basis of $n > m$ vectors $\boldsymbol{D} = \{D_1, \ldots, D_n\}$, we aim to infer a sparse representation of the input using few elements from the dictionary. Here, an overcomplete set is defined as one that contains more functions than needed for a basis. All basis matrices as well as the image $\boldsymbol{x}$ are assumed to be of equal dimensions. The task is to find the minimal set of non-zero activation coefficients $\boldsymbol{a}$ that accurately reconstruct the given input signal $\boldsymbol{x}$, where $\boldsymbol{a} \in \mathrm{B}^n$ is a binary vector of length n for $\mathrm{B} = \{0, 1\}$. We can express the computation of a sparse binary representation of the image $\boldsymbol{x}$ using the basis $\boldsymbol{D}$ as the minimization of the energy function

$$E(\boldsymbol{x}, \boldsymbol{a}) = \min_{\boldsymbol{a}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{D}\boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_0 \tag{1}$$

where $\|\cdot\|_2$ is the Euclidean norm and $\|\cdot\|_0$ denotes the number of nonzero elements. The parameter $\lambda > 0$ is a Lasso-type parameter [14] controlling the sparseness of the solution. A large value of λ results in a more sparse solution to eq. (1), while smaller values yield denser solutions. Therefore, the parameter λ allows one to effectively balance the reconstruction error (the L_2 norm) and the number of non-zero activation coefficients (the L_0 norm). Since eq. (1) belongs to the class of 0-1 integer programming problems, finding a sparse representation falls into an NP-hard complexity class. The objective function of eq. (1) is non-convex and typically contains multiple local minima.

We investigate a spiking neuromorphic processor to solve the binary sparse representation problem given by the objective function in eq. (1). Neuromorphic computing is a proposed computing model inspired by the human brain, which is able to complete learning tasks better than classical von Neumann computers [3, 12, 13].

2 METHODS

2.1 Transformation relations

The problem being solved has to be given as a QUBO problem. In this formulation, the observable states of any neuron is 0 and 1. We start by reformulating eq. (1) in QUBO form. To this end, we observe that for $\boldsymbol{a} \in \mathbb{B}^n$,

$$E(\boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{x} - D\boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{a}\|_0 = \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{x} - \boldsymbol{x}^\top D\boldsymbol{a} + \frac{1}{2}\boldsymbol{a}^\top D^\top D\boldsymbol{a} + \lambda\sum_{i=1}^{n} a_i.$$

As expected, multiplying out eq. (1) yields a quadratic form in $\boldsymbol{a}$, meaning that we can recast our objective function as a QUBO problem. For this we define the following two transformations:

$$h_i = -D_i^\top\boldsymbol{x} + \lambda + \frac{1}{2}D_i^\top D_i, \qquad Q = \frac{1}{2}(D^\top D). \tag{2}$$

Using eq. (2), we can rewrite eq. (1) as a QUBO, given by

$$H(\boldsymbol{h}, Q, \boldsymbol{a}) = \sum_{i=1} h_i a_i + \sum_{i<j} Q_{ij} a_i a_j, \tag{3}$$

which is now in suitable form to be solved on Intel's Loihi neuromorphic chip [5, 8]. Network connectivity mapping can be seen in Figure 2, where a_i denote the neurons, h_i are the self interactions on the neurons, and Q_{ij} are the inter-neuron connection weights.

2.2 Loihi neuromorphic chip implementation

Intel's Loihi 1 is the first generation neuromorphic computing device that draws inspiration from biology to implement spiking neural networks with neurons as the fundamental processing elements [2].

2.2.1 Overcoming local minima on Loihi 1. Compared to a Boltzmann machine [9], spiking networks allow for transitions between extreme objective function variable states (see Figure 3). Because of the limited time of activity, or forced refractory period, defined by τ, active neurons are turned off for a determined time and others who were inhibited by the active neuron now have a chance to activate. These periods allow the network to explore non-locally and facilitate the bypassing of high energy barriers in the optimization landscape [4, 10]. After the refractory period is over, previously active neurons will likely re-fire because they are receiving a strong input and a low-energy state will again be found. Figure 3 demonstrates this property through the substantial variation in the energy reads obtained from Loihi 1 as a function of time. High energy read outs correspond to refractory periods of neurons active in the ideal solution, and the repeated lowest energy reflects the return to lower energy solution states [2]. For the QuboSolver method ran on Loihi 1, a threshold mantissa of 96, weight exponent of 6, and noise mantissa of 0 and exponent of 7 are used. In order to sample each QUBO on Loihi 1, a total of 2, 000 samples are measured; 4 simulation times (5, 000, 10, 000, 15, 000, 20, 000) are varied over, and 5 different weight matrix scalings (10, 100, 1000, 10000, 100000) are varied, with each parameter combination being sampled 100 times (this gives $4 \cdot 5 \cdot 100 = 2000$ samples per QUBO).

2.2.2 Un-normalized Dictionary Learning. Sparse coding optimization can be seen as a two step process where a dictionary is first learned in an unsupervised way by using a local Hebbian rule. Typically, when learning a basis for solving the convex Lasso problem, the algorithm requires the re-normalization of the columns of the dictionary D after each learning epoch. The normalization is critical for convergence in the Lasso setting because the values of the sparse vector $\boldsymbol{a}$ are allowed to take on any value. Previous work has demonstrated the ability to learn a dictionary in a QUBO regime, but this required the introduction of a new amplification parameter β to the input [5, 8]. Here, we introduce a new learning technique that allows the algorithm to find the optimal norm for features based up on a predetermined desired average level of sparsity defined as $\boldsymbol{s} \in (0, 1)$. The dictionary is initialized with features drawn from a normal distribution with random norms below 1 and a small sparsity penalty parameter λ. After solving the binary sparse coding problem for each sample in the training data, the dictionary is updated. If the average sparsity over the training epoch is above the desired level s, the penalty parameter λ is increased for the next epoch. Pseudo code for the algorithm is presented below and the learning results are summarized in Figure 1. We can see the average neuron activity and reconstruction error converge along with the norms of the learned features.

We applied our technique to a patched version of the standard fashion MNIST (fMNIST) data set [16]. Each 28x28 image was broken up into 16 7x7 patches and we selected a dictionary of size 64 in order to partition the problem into sub-problems which could be implemented on Loihi 1 (the exact number of variables for the sub-problems is arbitrary but fixed). Even with a smaller data structure, it was still necessary to perform our dictionary learning algorithm using the classical simulated annealing approach when solving for our sparse code in step 6 of Algorithm 1. The Lasso parameter λ was increased from 0.1 to 1.4 in increments of 0.1 to adapt to the sparsity of the solution (see the top right plot in Figure 1).

3 RESULTS

Figure 1 visualizes the successful implementation of un-normalized dictionary feature learning. Using a local learning rule and a fixed sparsity level, we can see that the algorithm learns a better basis for reconstruction as the average error of the training data decreases over training epochs and it also converges to the desired average sparsity level.

After successfully training each dictionary with simulated annealing (SA), a total of 16 separate QUBO models are generated.

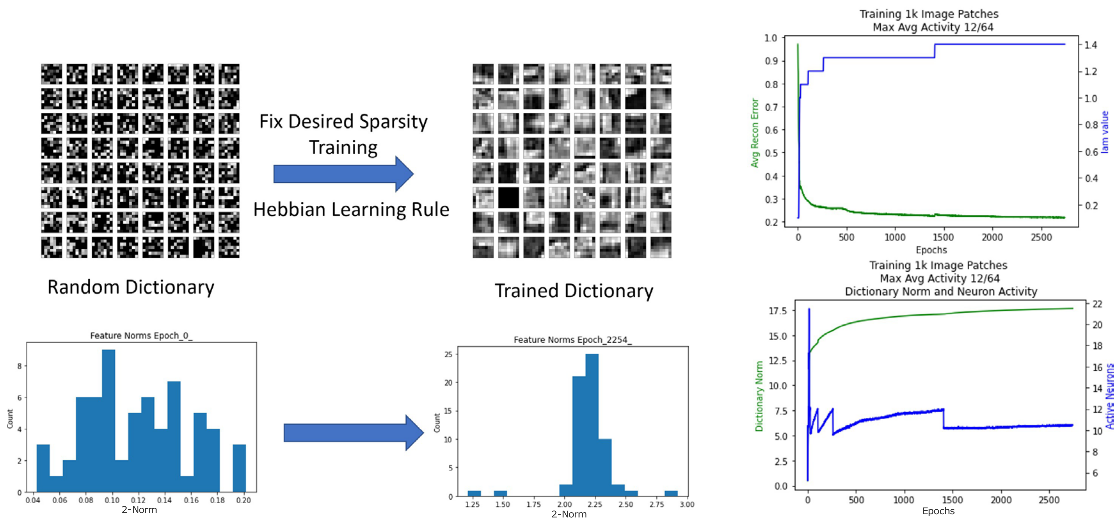


Figure 1: Randomly initialized dictionary with norms distributed between .01 **and** .2**. After the training algorithm, norms increase and an optimal binary dictionary is learned for a fixed average activity of** 12 **features.**

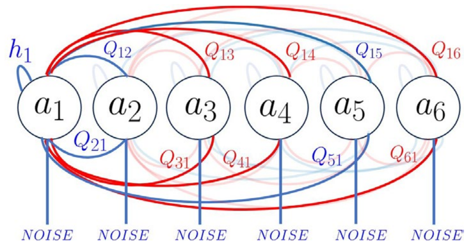


Figure 2: Network connectivity of the variables in eq. (3)**. Connections include the self interaction terms h_i (symmetric weights proportional to the inner product between features), the inter-neuron connection weights Q_{ij} , and the stochastic noise input. Red is inhibitory connection and blue is excitatory. Network is sampled at different times and activity is measured for solution.**

Algorithm 1: Dictionary Update

```
input: $D \in \mathbb{R}^{m \times n}$, $Train\_data \in \mathbb{R}^{b \times m}$, $\eta \in \mathbb{R}^+$, $s \in (0,1)$,
       $\lambda > 0$, number of epochs $N$
function learn_dictionary($D, a, x, \eta, s$, number of epochs)
    for epoch = $1, 2, \ldots, N$
        $activity\_count = 0$;
        for $i = 1, 2, \ldots, b$
            $x = Train\_data[i]$
            Solve for $a$
            $recon = Da$
            $residual = x - recon$
            $\Delta D = residual\ a^T$
            $D = D + \eta \Delta D$
            $activity\_count = activity\_count + sum(a)$
        end
        if $\frac{activity\_count}{n*b} > s$ then
            $\lambda = \lambda + 0.1$
        end
    end
end
return $D$
```

Each QUBO is then sampled using Loihi 1 (see Section 2.2). In order to provide a reasonable comparison against existing classical heuristic algorithms, we also sample each of the 16 QUBO models using simulated annealing. The simulated annealing implementation we use is a D-Wave SDK implementation [1], using 1000 samples per QUBO and all default settings. Using the best solutions (e.g., the computed variable assignments with the lowest energy found among all samples) from both Loihi 1 and simulated annealing, we can reconstruct the original image from sampling all 16 QUBOs. These reconstructions are shown in Figure 4. Although SA has a lower mean energy, Loihi 1 is able to find reasonable solutions at much lower average sparsity levels. Similar to previous demonstrations of lower power usage for certain applications [3, 5–8], Loihi 1 uses an average power consumption of ~ 0.0192 joules per sample, per QUBO matrix compared to an average power consumption of ~ 0.115 joules per sample per QUBO matrix for simulated annealing.

The simulated annealing power consumption was measured using pyRAPL [1] (including RAM power usage). The total power usage was computed by subtracting the idle machine power consumption (for the same time duration) from the power consumption when simulated annealing was run. The Loihi 1 power consumption was measured using the nxsdk power monitoring function.

[1]https://pyrapl.readthedocs.io/en/latest/

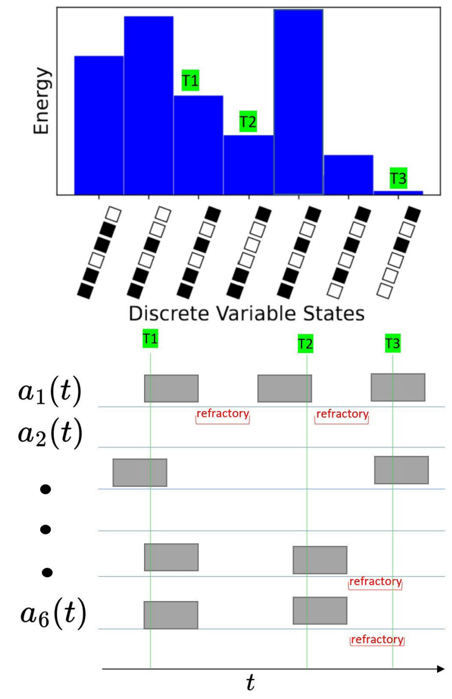


Figure 3: Conceptual diagram of how we expect spike-based dynamics support the bypassing of high-energy barriers. Energy, e.g., the objective function evaluation for a set of variable assignments, is given on the y-axis and the x-axis shows variable assignments where ■ denotes $+1$ and □ denotes 0 (for the chosen number of variables of $n = 6$). In this example, the relatively sparse state of $(0, 0, 0, 1, 0, 1)$ has the lowest overall energy. When the system is sampled at different time periods T1, T2, and T3, we are able to bypass the largest energy barrier because the refractory period automatically shuts off variables 5 and 6 [10].

4 DISCUSSION AND CONCLUSION

In this work, we derived a technique for learning an unmormalized dictionary for binary sparse coding in an unsupervised manner when given a desired sparsity level. The trained dictionary was then used for solving the binary sparse coding problem in the form of a QUBO using the Loihi 1 spiking neuromorphic processor and compared against simulated annealing. Measurements taken from Loihi 1 demonstrate the use of refractory periods and stochasticity allow the spiking processors to overcome large energy barriers in the non-convex landscape. The solutions from Loihi 1 are not of the same quality compared with simulated annealing, but it is interesting to note that the solutions are considerably sparser, and use less energy to compute each sample compared to simulated annealing.

Future work could include comparing the results on Loihi 2, the second generation of Intel's spiking processor. Using an iterative warm start approach with Loihi, where the best solution found at each iteration is used to initialize the system at the next iteration, similar to an iterative warm start algorithm in classical optimization, could improve the total space explored and thus the likelihood of finding a global minimum.

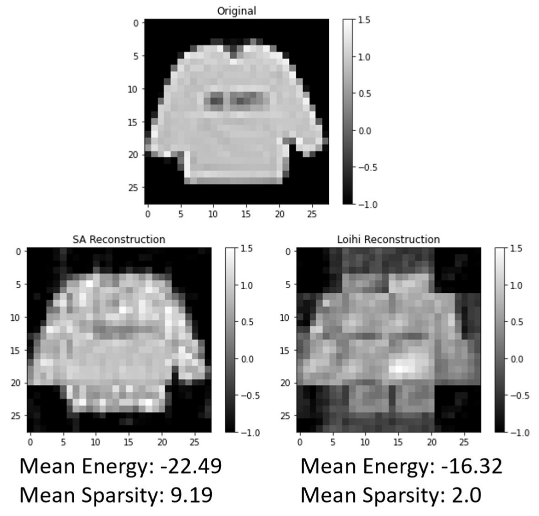


Figure 4: Reconstructions from classical SA and Loihi 1. Full image consists of 16 separate QUBO solves and the mean energies and sparsity levels are displayed. The sparsity levels are the mean (across the 16 QUBO models) number of variables in the lowest energy state which were in the state of $+1$.

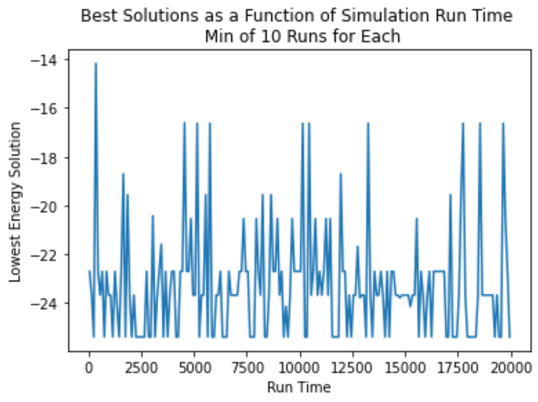


Figure 5: QUBO energies read out at different simulation times (minimum of 10 readouts per simulation time) from the Loihi 1 neuromorphic processor for a single QUBO patch.

5 ACKNOWLEDGEMENTS

This work was supported by the U.S. Department of Energy through the Los Alamos National Laboratory. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (with Contract No. 89233218CNA000001). We gratefully acknowledge support from the Advanced Scientific Computing Research (ASCR) program office in the Department of Energy's (DOE) Office of Science, award #77902 along with funding from the NNSA's Advanced Simulation and Computing Beyond Moore's Law Program at Los Alamos National Laboratory. This work has been assigned the technical report number LA-UR-23-25877.

REFERENCES

[1] D-Wave. 2022. dwave-simulated-annealing. https://github.com/dwavesystems/dwave-neal.

[2] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. 2021. Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook. *Proc. IEEE* 109, 5 (2021), 911–934. https://doi.org/10.1109/JPROC.2021.3067593

[3] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. 2021. Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook. *Proc. IEEE* 109, 5 (2021), 911–934. https://doi.org/10.1109/JPROC.2021.3067593

[4] Gabriel A. Fonseca Guerra and Steve B. Furber. 2017. Using Stochastic Spiking Neural Networks on SpiNNaker to Solve Constraint Satisfaction Problems. *Frontiers in Neuroscience* 11 (2017), 1–13. https://doi.org/10.3389/fnins.2017.00714

[5] Kyle Henke, Garrett T. Kenyon, and Ben Migliori. 2020. Machine Learning in a Post Moore's Law World: Quantum vs. Neuromorphic Substrates. In *2020 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*. Institute of Electrical and Electronics Engineers (IEEE), Piscataway, NJ, 74–77. https://doi.org/10.1109/SSIAI49293.2020.9094596

[6] Kyle Henke, Garrett T. Kenyon, and Ben Migliori. 2022. Fast Post-Hoc Normalization for Brain Inspired Sparse Coding on a Neuromorphic Device. *IEEE Transactions on Parallel and Distributed Systems* 33, 2 (2022), 302–309. https://doi.org/10.1109/TPDS.2021.3068777

[7] Kyle Henke, Garrett T. Kenyon, and Ben Migliori. 2022. Fast Post-Hoc Normalization for Brain Inspired Sparse Coding on a Neuromorphic Device. *IEEE Transactions on Parallel and Distributed Systems* 33, 2 (2022), 302–309. https://doi.org/10.1109/TPDS.2021.3068777

[8] Kyle Henke, Ben Migliori, and Garrett T. Kenyon. 2020. Alien vs. Predator: Brain Inspired Sparse Coding Optimization on Neuromorphic and Quantum Devices. In *2020 International Conference on Rebooting Computing (ICRC)*. Institute of Electrical and Electronics Engineers (IEEE), Piscataway, NJ, 26–33. https://doi.org/10.1109/ICRC2020.2020.00015

[9] Geoffrey E. Hinton. 2007. Boltzmann Machines. https://www.cs.toronto.edu/~hinton/csc321/readings/boltz321.pdf.

[10] Zeno Jonke, Stefan Habenschuss, and Wolfgang Maass. 2016. Solving Constraint Satisfaction Problems with Networks of Spiking Neurons. *Front Neurosci* 10, 118 (2016), 1–16. https://doi.org/10.3389/fnins.2016.00118

[11] Rahul Mohideen, Pascal Peter, and Joachim Weickert. 2021. A systematic evaluation of coding strategies for sparse binary images. *Signal Processing Image Communication* 99 (2021), 116424.

[12] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. 2019. Towards spike-based machine intelligence with neuromorphic computing. *Nature* 575, 7784 (2019), 607–617.

[13] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Prasanna Date, and Bill Kay. 2022. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science* 2, 1 (2022), 10–19. https://doi.org/10.1038/s43588-021-00184-y

[14] R. Tibshirani. 1996. Regression Shrinkage and Selection Via the Lasso. *J Roy Stat Soc B Met* 58, 1 (1996), 267–288. https://doi.org/10.1111/j.1467-9868.2011.00771.x

[15] M. Ting, R. Raich, and A. Hero. 2006. Sparse Image Reconstruction using Sparse Priors. In *International Conference on Image Processing, Atlanta, GA, USA*. Institute of Electrical and Electronics Engineers (IEEE), Piscataway, NJ, 1261–1264. https://doi.org/10.1109/ICIP.2006.312574

[16] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. http://arxiv.org/abs/1708.07747

Apples-to-spikes: The first detailed comparison of LASSO solutions generated by a spiking neuromorphic processor

Kyle Henke*
khenke@lanl.gov
Los Alamos National Laboratory
Los Alamos, New Mexico, USA

Michael Teti*
mteti@lanl.gov
Los Alamos National Laboratory
Los Alamos, New Mexico, USA

Garrett T. Kenyon
gkenyon@lanl.gov
Los Alamos National Laboratory
Los Alamos, New Mexico, USA

Ben Migliori
ben.migliori@lanl.gov
Los Alamos National Laboratory
Los Alamos, New Mexico, USA

Gerd J. Kunde*
g.j.kunde@lanl.gov
Los Alamos National Laboratory
Los Alamos, New Mexico, USA

ABSTRACT

The Locally Competitive Algorithm (LCA) is a model of simple cells in the primary visual cortex, based on convex sparse coding via recurrent lateral competition between neighboring neurons. Previous work implemented spiking LCA (S-LCA) on the Loihi neuromorphic processor in which the lateral connections were constrained to be inhibitory, unlike non-spiking, analog LCA (A-LCA) where both excitatory and inhibitory connections are present. In the absence of lateral excitation, an implementation of S-LCA on the Loihi neuromorphic processor inferred sparse representations of image patches that were close to the global minimum, but an examination of the individual neural activations (i.e. solution) was not performed. In this work, we first prove that the constraints placed on the lateral connections in the previous S-LCA implementation were unnecessarily restrictive, and we develop an S-LCA implementation with both excitatory and inhibitory lateral connections. We implemented this improved S-LCA with both inhibitory and excitatory lateral connections on Loihi and show that the resulting sparse latent representations were much closer to those inferred by A-LCA. Specifically, we perform the first comparison of individual neuron activations between S-LCA and A-LCA and show that the final solution of our S-LCA converges to that of A-LCA. To date, this work provides one of the only instances in which a spiking algorithm implemented on modern neuromorphic hardware and performing a realistic task has exhibited such close behavior to its non-spiking counterpart.

KEYWORDS

neuromorphic computing, sparse coding, computer vision, spiking neural networks

ACM Reference Format:
Kyle Henke, Michael Teti, Garrett T. Kenyon, Ben Migliori, and Gerd J. Kunde. 2022. Apples-to-spikes: The first detailed comparison of LASSO solutions generated by a spiking neuromorphic processor. In *International Conference on Neuromorphic Systems (ICONS 2022), July 27–29, 2022, Knoxville, TN, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3546790.3546811

*Equal contribution

ICONS 2022, July 27–29, 2022, Knoxville, TN, USA
2022. ACM ISBN 978-1-4503-9789-6/22/07.
https://doi.org/10.1145/3546790.3546811

1 INTRODUCTION

Computational neuroscience is often focused on understanding how complex phenomenon can emerge from networks of neurons. Often, this is achieved by formulating models that best represent the underlying physics governing communication within large neuronal networks. For application purposes, it is also crucial to have rigorous mathematical foundations to provide theoretical guidance for convergence and performance of artificial systems. Here, we focus on sparse coding models, which have been shown to approximate the response characteristics and receptive field statistics of neurons in the primary visual cortex (V1) [10, 12]. In the sparse coding problem, the goal is to obtain a faithful but efficient (i.e. sparse) representation of a given input. A representation which satisfies these criteria is found by minimizing an energy function consisting of a term which represents the error between the input and its reconstruction (which is computed from the representation), plus a term which represents how sparse the representation is. When the reconstruction error is measured with the l_2 norm and the sparsity is measured with the l_1 norm, the problem is equivalent to LASSO (i.e. l_1-penalized regression) [16] and has a global minimum. Rozell et al. [12] developed a recurrent network termed the Locally Competitive Algorithm (LCA), which minimizes the sparse coding energy function by simulating the feature-specific, local lateral competition that is observed in V1 [2]. LCA can be expressed in terms of a governing dynamical system of equations for which there exist a Lyapunov function with a fixed point attractor whose minima correspond to the global minima of a LASSO optimization problem.

Biologically plausible sparse coding models, such as LCA, are of great interest to the neuromorphic community because they are able to model key characteristics of biological sensory processing [17, 18] while remaining useful in machine learning applications [9, 15]. Since spiking implementations on neuromorphic hardware often differ greatly from their non-spiking counterparts on classical computing hardware, it is necessary to have a deep understanding of how or if they differ before they can be widely used in potentially critical applications. First, [13] showed that the sparse coding objective function converged to that of LASSO in a simulated spiking neural network (SNN) composed of integrate-and-fire neurons. Fair et al. then demonstrated that a spiking LCA implementation on the TrueNorth neuromorphic system [1] exhibited close dynamics to non-spiking LCA implemented on classical computing hardware, but they considered a model with a very constrained input, dictionary, and dynamics [5]. Most recently, [3] implemented LCA on the Loihi neuromorphic processor using leaky-integrate-and-fire (LIF) neurons with infinite time constants (i.e. non-leaky), but their model only contained inhibitory lateral connections, unlike non-spiking LCA which has both excitatory and inhibitory lateral connections. They demonstrated that a sparse coding objective function is monotonically decreasing when using the vector of average firing rates as a measure of neural activity. However, no coefficient-by-coefficient comparison between LASSO and spiking LCA has been performed to date, leaving open the question as to how closely spiking LCA implemented on neuromorphic hardware approximates LASSO on more complex, realistic problems.

In this work, we extend the previous spiking implementation of LCA [3], which we refer to as S-LCA, by developing a modified S-LCA with both excitatory and inhibitory lateral connections (Figure 1), which we prove should converge to non-spiking, analog LCA (A-LCA). We then performed the first neuron-by-neuron comparison between S-LCA implemented on modern neuromorphic hardware, namely Intel's Loihi, and A-LCA implemented on classical computing hardware (i.e. LASSO). We show that our S-LCA implementation exhibits a very close match to A-LCA, both at the individual neuron level and the system level, and it is a better match than the previous S-LCA implementation due primarily to the incorporation of excitatory lateral connections.

2 BACKGROUND

2.1 Sparse Coding

Sparse coding is a signal processing technique which models cortical processing of lower-dimensional sensory inputs into a sparse, higher-dimensional space. Sparse coding models have been shown to approximate the receptive fields and response characteristics of V1 simple cells [10, 17]. [12] has shown that sparse coding optimization problems can be solved using the dynamics of fully recurrent neural networks with fixed point attractors when lateral inhibition is incorporated. This biologically plausible implementation (from experimental observations of similar connectivity, c.f. rat whisker barrel cortex) encourages sparse solutions by allowing neurons to compete with each other for shared representation of the input, in a model known as a locally competitive algorithm (LCA). When neural activation is penalized through inhibition in the cost function, the resulting dynamical system emerges as an all-to-all connected Hopfield network [7]. The connections represented through a symmetric weight matrix satisfy a Lyapunov condition which guarantees the convergence to a fixed point of lowest energy, the optimal sparse solution.

In this mathematical notation, our input signal x lives in R^m and the dictionary D has $p > m$ basis vectors also in R^m. We then want to approximate x as Da where $a \in R^p$. D is overcomplete or redundant and an infinite number of solutions to the minimization problem become possible. Hence, a λ sparsity penalty is introduced to represent a uniform applied inhibitory field and create a unique solution. Thus, we are minimizing the reconstruction distance in

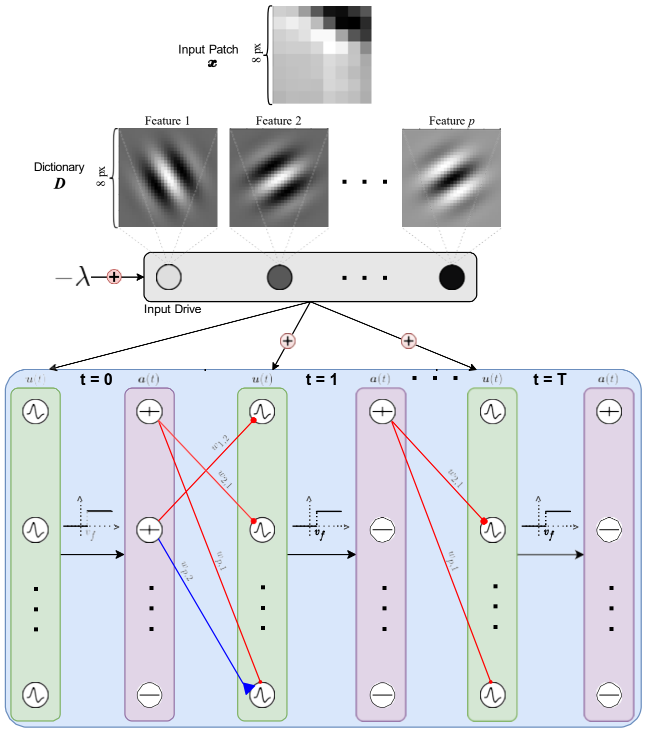


Figure 1: Our S-LCA implementation on Loihi. A general depiction of the S-LCA algorithm as implemented on Loihi using a single 8 × 8 patch as input. The input drive, which is the dot product between each neuron's feature vector and the input patch, is computed and used to initialize the membrane potentials after subtracting the A-LCA trade-off parameter λ. At each timestep, each neuron's membrane potential is charged up (or down) by the input drive and compared to a spiking threshold v_f. Any neuron whose membrane potential is greater than v_f will "spike", and thus inhibit (red) or excite (blue) neurons whose features overlap with its own, depending on whether the features are aligned or anti-aligned, respectively. The previous S-LCA implementation only contained inhibitory (red) lateral connections. The membrane potential is reset to zero after every spike. After T iterations, typically only a few neurons remain active. The average firing rate of each active neuron in the S-LCA model is computed over the last 1,000 timesteps for comparison with the A-LCA model. Our comparisons are performed on a 56 × 56 pixel image, but we use 8 × 8 features and a stride of 8, which is the same process depicted here but with 7 × 7 = 49 patches.

l_2 subject to an l_1 constraint on the parameters a in the following standard LASSO set up:

$$E(a(t)) = \frac{1}{2}||x - Da(t)||_2^2 + \lambda||(a(t))||_1 \quad (1)$$

2.2 Derivation of Dynamical System

We can derive a system of differential equations proportional to, an hence with the same fixed point attractors, as the classical LASSO problem. Given $D \in \mathrm{R}^{mxp}$ and $x \in \mathrm{R}^m$ with $\lambda > 0$, solve:

$$
\begin{aligned}
E(a(t)) &= \frac{1}{2}||x - Da(t)||_2^2 + \int_0^\infty (u(t) - T_\lambda(u(t)))da(t) \\
&= \frac{1}{2}(x - Da(t))^T(x - Da(t)) + \int_0^\infty (u(t) - T_\lambda(u(t)))da(t) \\
&= \frac{1}{2}(x^T x - x^T Da(t) - a(t)^T D^T x + a(t)^T D^T Da(t)) \\
&\quad + \int_0^\infty (u(t) - T_\lambda(u(t)))da(t) \\
&= \frac{1}{2}(x^T x - 2x^T Da(t) + a(t)^T D^T Da(t) \\
&\quad + \int_0^\infty (u(t) - T_\lambda(u(t)))da(t)
\end{aligned} \tag{2}
$$

We take the partial derivative of $E(a(t))$ with respect to $a(t)$:

$$\frac{\partial E(a(t))}{\partial a(t)} = -x^T D + D^T Da(t) + u(t) - T_\lambda(u(t)) \tag{3}$$

Now we can define our gradient system as:

$$\dot{u}(t) \propto -\frac{\partial E(a(t))}{\partial a(t)} \tag{4}$$

$$
\begin{aligned}
\dot{u}(t) &= \frac{1}{\tau}(x^T D - D^T Da(t) - u(t) + T_\lambda(u(t)) \\
&= \frac{1}{\tau}(x^T D - D^T Da(t) - u(t) + a(t))
\end{aligned} \tag{5}
$$

Here we assume the existence of an input/output transfer function $a(t) = T_\lambda(u(t))$ with threshold λ. The neuron activation is represented by the thresholding function $T = T_\lambda$ and describes the non-linear activity of how and when signals are sent to the rest of the network. To solve a LASSO sparse coding problem, we use a soft-threshold function $a_i = T_\lambda(u_i)$ whose value is $u_i - \lambda$ when $u_i > \lambda$ and 0 otherwise. We can then define the vector function which applies the same scalar function T to each of the input vectors components as $T : R^p \rightarrow R^p$.

The sparse coding problem can be described in neurophysiological terms by letting the p dictionary elements represent p neurons and their respective receptive fields. The input stimulus received by each neuron is equivalent to the inner product between the input signal and the feature it represents, notated as $b_i = x^T D_i$. The constant bias drive b_i increases (or decreases) the membrane potential of the neuron-i represented as u_i. When u_i is above the λ, neuron-i will then send inhibitory or excitatory signals to the other $p - 1$ neurons equal to the product of the activation coefficient a_i and the connection weight $w_{i,j} = -D^T D_j$.

Using the LaSalle invariance principle [14][12], Rozell and others were able to prove the above system of equations possesses three distinct characteristics.

(1) If C is the set of optimal LASSO solutions and $F = T^{-1}(C)$ is C's inverse mapping under the thresholding function T, then any arbitrary initial condition $\dot{u}(0)$ will always converge to the set F
(2) In the limit $\lim_{t\rightarrow\infty} E(a(t)) = E^*$, the value of the objective function E will approach the optimal value
(3) If the optimal solution a^* is unique, then $\dot{u}(t) \rightarrow \dot{u}^*$ and $T(u(t)) \rightarrow T(u(t)^*) = a^*$ as $t \rightarrow \infty$

2.3 S-LCA and Convergence to A-LCA

Here we give an overview of the the convergence proof provided by Tang et al. [14] for a LASSO problem with strictly positive connectivity weights $w_{i,j}$ and extend the result into a regime where both positive and negative weights are present.

First, we define the only independent variable in our spiking network as the soma currents $\mu_i(t)$ for the p neurons which receive a constant input bias $b_i = D^T x$ and maintain an internal electric potential $v_i(t)$. When an electric potential reaches a firing threshold v_f at a time $t = k$, the corresponding neuron simultaneously fires a spike to either inhibit or excite the other $p - 1$ neurons and resets its potential to v_r. Let $a = e^{-t}$ and define the soma currents of the other neurons to change in the following manner:

$$\mu_j(t) = \mu_j(t) - w_{ji}\, a(t - t_{i,k}) \tag{6}$$

Now, define $\sigma_i(t) = \sum_k \delta(t - t_{i,k})$ as the sum of Dirac delta functions δ whenever the neuron spikes over the simulation time. This leads to the final defining equations of soma currents:

$$\mu_i(t) = b_i - \sum_{j \neq i} w_{ij}(a * \sigma_j)(t) \tag{7}$$

$$\dot{\mu}_i(t) = b_i - \mu_i(t) - \sum_{j \neq i} w_{ij}\sigma_j(t) \tag{8}$$

The instantaneous spike rate $a_i(t)$ and average soma current $u_i(t)$ are defined as:

$$a_i(t) = \frac{1}{t - t_0}\int_{t_0}^{t} \sigma_i(s)ds \tag{9}$$

$$u_i(t) = \frac{1}{t - t_0}\int_{t_0}^{t} b_i - \sum_{i \neq j} w_{i,j}(a_u * \sigma_j)(s)ds \tag{10}$$

Leading to the spiking analog of differential equation 3 as:

$$\dot{u}_i = b_i - u_i - \sum_{j \neq i} w_{ij}a_j(t) - \frac{(u_i(t) - u_i(t_0))}{t - t_0} \tag{11}$$

2.4 S-LCA With Excitatory Connections

Here we make a distinction and extend the previous work. Originally, only inhibitory connections were allowed in order to ensure the soma current magnitudes and corresponding average potentials are bounded. For a strictly inhibitory network, the max bound on current is defined as $B_+ = max_i\, b_i$ since the largest value obtainable in equation 12 requires zero inhibition from other neurons. Moreover, [14] also showed there is a lower bound and the existence of some $R > 0$ such that $t_{i,k+1} - t_{i,k} \geq 1/R$ for all $i = 1, 2, ..., n$ and $k \geq 0$ whenever two spike times exist. We can leverage this knowledge to show the soma currents of our updated model are also bounded above and below. First let $A > max_i \sum_j |w_{i,j}|_1$ and $B = max_j |b_j|_1$ since we know the inner product of features and biases are bounded. Using the fact $(a * \sigma_j)(t) \leq \sum_{l=0}^{\infty} e^{-\frac{l}{R}} < \infty$, can show:

$$
\begin{aligned}
\|\mu_i(t)\|_1 &= \left\| b_i - \sum_{j \neq i} w_{ij}(a * \sigma_j)(t) \right\|_1 \\
&\le |b_i| + \sum_{j \neq i} \|w_{ij}\| \, \|(a * \sigma_j)(t)\|_1 \\
&\le \max_j \|b_j\| + \sum_{j \neq i} \|w_{ij}\| \, \|(a * \sigma_j)(t)\|_1 \\
&\le B + nA \|(a * \sigma_j)(t)\|_1 \\
&\le B + nA \left\| \sum_{l=0}^{\infty} e^{-\frac{l}{R}} \right\|_1 < \infty
\end{aligned} \tag{12}
$$

Implying the soma currents are bounded from above and below. Equipped with this knowledge, we can follow the proof by [14] and state $u(t) = [u_1(t), u_2(t), ..., u_p(t)]^T$ has at least one limit point $u^* \in \mathbb{R}^p$ such that $u(t_k) \to u^*$ as the sequence of t_ks $\to \infty$ when $k \to \infty$ from the Bolzano-Weirstrass theorem.

This implies:

$$\lim_{t\to\infty} \dot{u}_i(t) = \lim_{t\to\infty} \frac{1}{t - t_0}(\mu_i - u_i) = 0 \tag{13}$$

Hence $T(u(t_k)) \to T(u^*) = a^*$, we can conclude the system converges to the same limit found in A-LCA:

$$0 = b - u^* - (D^T D - I)a^* \tag{14}$$

2.5 Unsupervised Dictionary Learning

The second part of the optimization process involves learning the best dictionary D for the given data set. Random features were first selected for the dictionary and a stochastic gradient descent algorithm with local Hebbian Learning rule was used to update the feature vectors of any active neurons so as to slightly improve the sparse reconstruction.

2.6 L2 Differentiation

First, lets look at the objective function for a sparse coding problem. $x \in \mathbb{R}^m$ is the input $D \in \mathbb{R}^{mxp}$ is our dictionary and $a \in \mathbb{R}^p$ is the sparse code.

$$E(a(t)) = \frac{1}{2}||x - Da(t)||_2^2 + \lambda||a(t)||_1$$

We expand on the reconstruction error term for purposes of gradient descent because the sparsity penalty drops after differentiation wrt D.

$$
\begin{aligned}
E(a(t))^* &= \frac{1}{2}||x - Da(t)||_2^2 \\
&= \frac{1}{2}(x - Da(t))^T(x - Da(t)) \\
&= \frac{1}{2}(x^T x - x^T Da(t) - (Da(t))^T x + (Da(t))^T Da(t)) \\
&= \frac{1}{2}\Big(\sum_{i=1}^{m} x_i^2 - 2\sum_{i=1}^{m}(x_i \sum_{j=1}^{p} D_{ij} a_j(t)) \\
&\quad + \sum_{i=1}^{m}(\sum_{j=1}^{p} D_{ij} a_j(t))^2\Big)
\end{aligned} \tag{15}
$$

Now we can differentiate E^* with respect to D_{yz} to see how each individual dictionary element changes.

$$
\begin{aligned}
\frac{\partial E^*}{\partial D_{yz}} &= \frac{1}{2}(-2x_y a_z(t) + 2(\sum_{j=1}^{p} D_{yj} a_j) a_z(t)) \\
&= (\sum_{j=1}^{p} D_{yj} a_j(t) - x_y) a_z(t) \\
&= -r_y a_z(t)
\end{aligned}
$$

Where r_y represents the yth component of the residual. We can then expand into matrix form:

$$\frac{\partial E}{\partial D} = -\begin{pmatrix} r_1 a_1(t) & r_1 a_2(t) & \dots & r_1 a_p(t) \\ r_2 a_1(t) & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ r_m a_1 & & & r_m a_p \end{pmatrix} \tag{16}$$

$$= -ra^T(t) \tag{17}$$

$$= -(x - Da(t))a(t)^T \tag{18}$$

The Hebbian learning algorithm [6] given a single input, x, is summarized in Algorithm 1. In practice, a mini-batch of input samples are used for each update instead of a single input sample. Since our gradient system is proportional to the derivative of the energy wrt D of the LASSO problem, we know the learning process will descend the gradient of our neurophysiological representation.

Algorithm 1 Dictionary Update

```
Input: $D \in \mathbb{R}^{m\times p}$, $a \in \mathbb{R}^p$, $x \in \mathbb{R}^m$, $\eta \in \mathbb{R}^+$
Output: $D \in \mathbb{R}^{m\times p}$
 1: function update_dictionary($D, a, x, \eta$)
 2:     $recon = Da$
 3:     $residual = x - recon$
 4:     $\Delta D = residual\ a^T$
 5:     $D = D + \eta\Delta D$
 6:     for $i = 1, 2, ..., p$ do
 7:         $D_i = D_i / \mathrm{norm}(D_i, 2)$
 8:     end for
 9: return $D$
10: end function
```

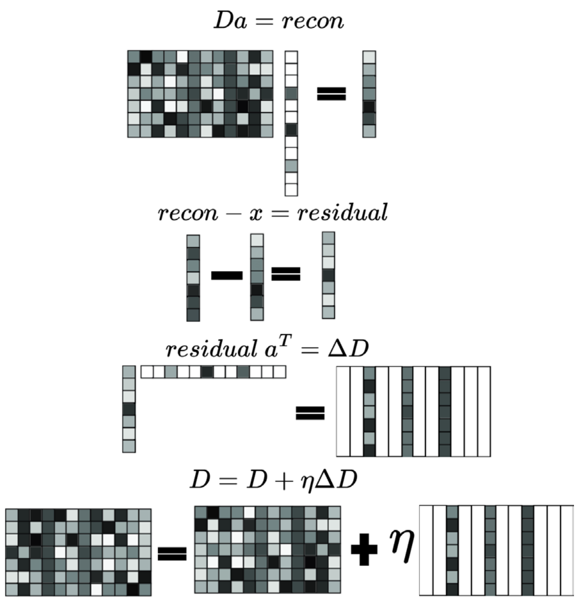


Figure 2: Sparse dictionary learning. Sparse coding is often combined with dictionary learning in an alternating fashion. After computing $a(t)$ by minimizing Equation 1 with fixed D, D is then updated to minimize Equation 1 with fixed $a(t)$ (Algorithm 1). At each update, the dictionary only changes in the directions of the active neurons.

3 METHODS

3.1 A-LCA Implementation

To compare S-LCA on Loihi to non-spiking LCA [12], we implement a single LCA layer in PyTorch [11] using the LCA-PyTorch package. Specifically, we create a convolutional LCA layer with valid padding, 450 features of size 8×8, a stride of 8, and a rectified soft threshold. With this model, we trained a dictionary for 5,000 updates (Algorithm 1) on 50,000 grayscale images of size 56×56 selected from the COCO dataset [8] with $\lambda = 0.5$. The dictionary (Figure 3) was then used in both this non-spiking A-LCA model and the spiking S-LCA model in our comparisons on held out images from our COCO set. For our comparisons to S-LCA on Loihi, we use $\lambda = 0.73$ in this A-LCA to match the activation sparsity.

3.2 S-LCA Loihi Implementation and Modifications

The previous S-LCA implementation on Loihi that used only inhibitory lateral connections [3][4] was structured the following way:

Neurons in the spiking network are driven by a respective bias current b (not a spiking input) that is calculated once, at the beginning of a run, as the dot product of the dictionary element and the respective patch and is scaled then scaled to the available bit space. The weights in [3][4] are chosen to be positive definite and made to work via the construction of an expanded dictionary twice the size of the original, consisting of strictly positive dictionary

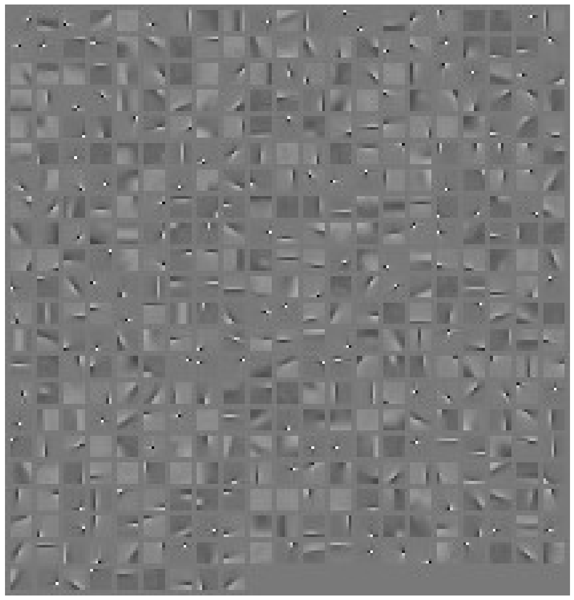


Figure 3: The dictionary used by both the S-LCA and A-LCA models in our experiments. The dictionary (D) is composed of 450 features of size 8×8.

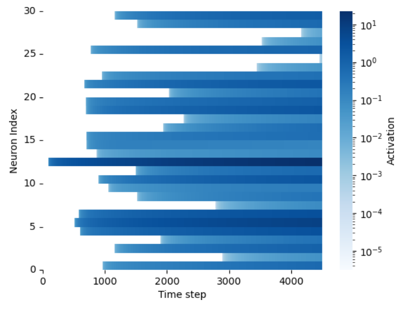
elements in the top half of the dictionary and inverted negative elements in the lower half. This S-LCA implementation converged towards a minimum to the LASSO sparse coding objective function in which the feature vectors lacked negative sub-units. The lack of anti-aligned sub-units prohibited more biologically realistic environments where neurons can also excite one other.

Here we demonstrate that the addition of these excitetory sub-units, in combination with the inhibitory sub-units, gives rise to a dynamical spiking system that behaves more closely to a conventional non-spiking A-LCA model (Fig. 4). Specifically, when features contain both excitatory and inhibitory sub-units, both positive and negative lateral connections arise naturally via taking the transpose of the dictionary doted with its self. A given spiking neuron will now inhibit neurons with similar explanations of the same patch (positive inner product) but will excited neurons with dissimilar explanations (negative inner product). In addition, we re-implemented the ranges of biases, weights and activations such that there were no longer sign flips (integer overflow) due to the limited bit ranges on Loihi.

4 RESULTS

After initializing both the S-LCA and A-LCA models with the dictionary learned in Section 3.1 (Figure 3), both models were run on their respective hardware using the same test image with the parameters outlined in Sections 3.2 and 3.1.

We show that our S-LCA exhibits closer dynamics to A-LCA than previous implementations of S-LCA by allowing only one neuron in each model to receive an input drive while all other neurons received no input drive. Since earlier S-LCA architectures contained no excitatory lateral connections, we hypothesized that only the neuron receiving input drive would be active in those



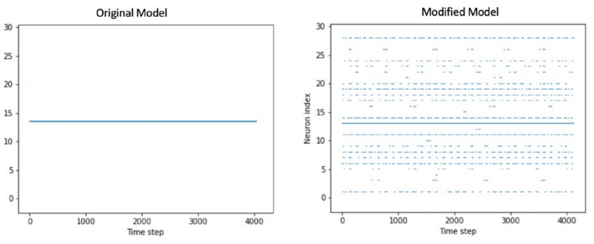


Figure 4: Our modified S-LCA model contains excitatory connections like in A-LCA. Activation when only the neuron best aligned with the input patch receives bias drive; all other biases were set to zero. A-LCA model (top) and our implementation of S-LCA on Loihi (bottom right) exhibit activity for neurons with zero input drive, while the previous S-LCA implementation on Loihi (bottom left) [3] doesn't excite activity of other neurons (bottom left), confirming an absence of excitatory connections.

models. In contrast, our S-LCA and A-LCA contain excitatory lateral connections, which should raise the membrane potential of some of the other neurons above threshold even without input drive. In Figure 4, we confirm this, as both A-LCA (top) and our S-LCA (bottom right) have multiple neurons active, whereas the previous S-LCA (bottom left) only has one active neuron (the only one with a non-zero input drive). In both A-LCA and our S-LCA, the same neurons appear to be active at qualitatively similar activity levels as the system converges.

Figure 5 illustrates the activation of each neuron in the S-LCA model and the A-LCA model as a function of initial input drive. Both our S-LCA and the A-LCA contain neurons which are active in the sparse representation despite having negative input drive (i.e. anti-aligned with the stimulus), whereas the previous S-LCA has only the driven neuron active since there was no mechanism for excitatory connections to other neurons. We can also see that our S-LCA provides a reasonable match to A-LCA despite the quantization that takes place on Loihi. Next, we compare the sparse activation of each neuron in our S-LCA directly against that in the A-LCA (Figure 6). Here, we can see further evidence that our S-LCA performs very close to A-LCA, as the activations lie close to the diagonal indicating that our S-LCA converged to A-LCA.

Finally, we compare our S-LCA model to the A-LCA model by examining the reconstructions of the input image produced by each model from the sparse representation. By comparing the reconstructions visually, we validate that our S-LCA produces a similar

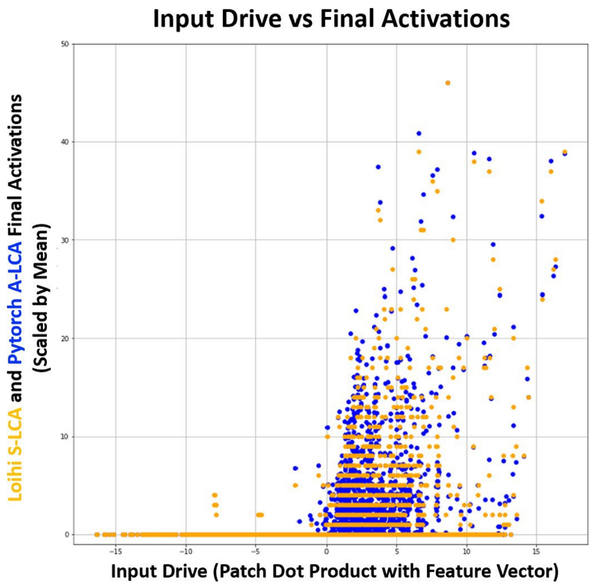


Figure 5: Input drive vs. final activation. Our S-LCA model produces a similar shape to the A-LCA model. Both models contain a few neurons that became active with features that were negatively aligned with the input, which was not true for the original S-LCA model. The distinct levels of final activity for the S-LCA model demonstrate the bit precision limitation present on the hardware.

sparse representation to the A-LCA. Figure 7 confirms that this is the case, as the reconstruction produced by our S-LCA model is very close to that produced by the A-LCA model. We can also observe that each model is using a very similar number of features to represent each patch.

5 DISCUSSION

In this work, we improved upon the previous S-LCA model, which only allowed inhibitory lateral connections between neurons, by developing an S-LCA model with both excitatory and inhibitory lateral connections that more closely matches A-LCA. Specifically, we first prove that our S-LCA system converges to the same limit found in A-LCA. Next, we implemented our S-LCA on the Loihi neuromorphic processor and initialized it with the same input and dictionary as a comparable A-LCA implemented on CPU/GPU hardware. We then performed the first neuron-by-neuron comparison between S-LCA and A-LCA and show that the sparse latent representation in our S-LCA converges to that of A-LCA.

This work is one of a few examples in which a spiking algorithm implemented on modern neuromorphic hardware exhibits almost an exact match to the comparable classical implementation under a realistic task. As a result, the performance of our S-LCA implementation meets or exceeds that of A-LCA implementations across the board, as S-LCA implementations on Loihi have already been shown to require much less power and time to converge [4][3]. This opens the door for the development of fast, low-power AI models in applications where A-LCA has already proven to be valuable, for

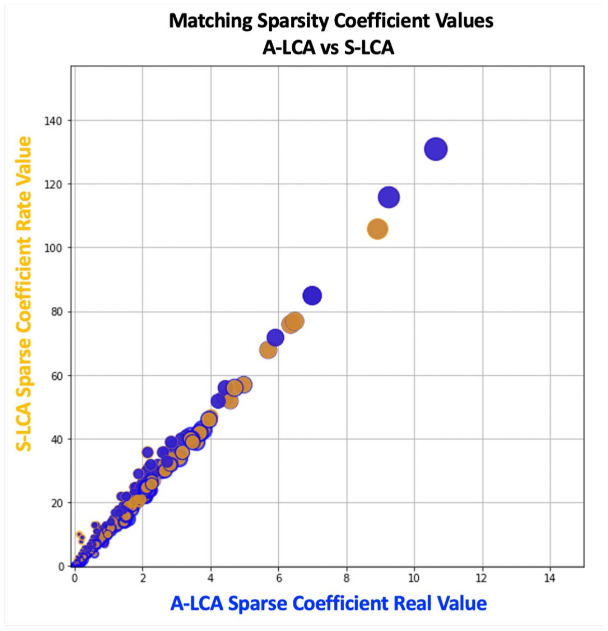


Figure 6: The rate code solution for our S-LCA model is a very close match to the A-LCA solution. Each point represents a single neuron out of the $450 \times 8 \times 8 = 28,800$ neurons in our model. The difference in scale on each axis is due to how the spikes are integrated.

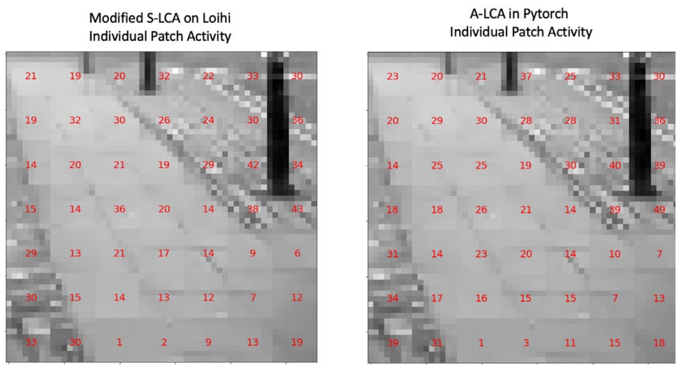


Figure 7: The number of active neurons per patch in our S-LCA model is very close to that in the A-LCA model. The number of neurons active per patch is laid over the final sparse reconstructions of the S-LCA (left) and A-LCA (right) models, which illustrates that our S-LCA model closely matches A-LCA both at the image and patch level.

example as a robust frontend for convolutional neural networks [15].

One limitation of this work is that we have only considered the non-convolutional case by using a stride equal to the patch size. Although it is unlikely that our S-LCA and A-LCA will perform drastically different in the convolutional setting, future work will need to verify this. In addition, future work can extend our S-LCA implementation to the spatio-temporal domain, perhaps by using video inputs or those from a dynamic vision sensor. This will allow us to develop and test models that are even closer to biological visual processing.

6 CONCLUSION

We developed an improved spiking LCA algorithm with both inhibitory and excitatory lateral connections, contrary to the previous spiking LCA implementation which only included inhibitory lateral connections. We then implemented our spiking LCA model on a modern neuromorphic processor, namely Intel's Loihi, and we performed the first comparison of individual activations between spiking LCA on neuromorphic hardware and non-spiking LCA on CPU/GPU hardware. We show that our spiking LCA implementation exhibits a closer match to the non-spiking LCA than the previous spiking implementation, both qualitatively and quantitatively. In addition, our LCA implementation provides a deeper insight into how non-spiking LCA and spiking-LCA are related when instantiated on neuromorphic substrates, while providing one of the few examples in which a spiking algorithm implemented on neuromorphic hardware performs as well as or better than the classical implementation across the board.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the Advanced Scientific Computing Research (ASCR) program office in the Department of Energy's (DOE) Office of Science, award #77902 along with funding from LANL ASC Beyond Moores's Law program.

REFERENCES

[1] A Cassidy, Jun Sawada, P Merolla, J Arthur, R Alvarez-lcaze, Filipp Akopyan, B Jackson, and D Modha. 2016. TrueNorth: A high-performance, low-power neurosynaptic processor for multi-sensory perception, action, and cognition. In *Proceedings of the Government Microcircuits Applications & Critical Technology Conference, Orlando, FL, USA*. 14–17.
[2] Selmaan N Chettih and Christopher D Harvey. 2019. Single-neuron perturbations reveal feature-specific competition in V1. *Nature* 567, 7748 (2019), 334–340.
[3] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro* 38, 1 (2018), 82–99.
[4] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh Risbud. 2021. Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook. *Proc. IEEE* PP (04 2021), 1–24. https://doi.org/10.1109/JPROC.2021.3067593
[5] Kaitlin L Fair, Daniel R Mendat, Andreas G Andreou, Christopher J Rozell, Justin Romberg, and David V Anderson. 2019. Sparse coding using the locally competitive algorithm on the TrueNorth neurosynaptic system. *Frontiers in Neuroscience* (2019), 754.
[6] Donald Olding Hebb. 2005. *The organization of behavior: A neuropsychological theory*. Psychology Press.
[7] John J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79, 8 (1982), 2554–2558.
[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
[9] Sheng Y Lundquist. 2020. *Exploring the Potential of Sparse Coding for Machine Learning*. Ph.D. Dissertation. Portland State University.
[10] Bruno A Olshausen and David J Field. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381, 6583 (1996), 607–609.
[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

[12] Christopher J Rozell, Don H Johnson, Richard G Baraniuk, and Bruno A Olshausen. 2008. Sparse coding via thresholding and local competition in neural circuits. *Neural computation* 20, 10 (2008), 2526–2563.

[13] Samuel Shapero, Mengchen Zhu, Jennifer Hasler, and Christopher Rozell. 2014. Optimal sparse approximation with integrate and fire neurons. *International journal of neural systems* 24, 05 (2014), 1440001.

[14] Ping Tak Peter Tang, Tsung-Han Lin, and Mike Davies. 2017. Sparse coding by spiking neural networks: Convergence theory and computational results. *arXiv preprint arXiv:1705.05475* (2017).

[15] Michael Teti, Garrett Kenyon, Ben Migliori, and Juston Moore. 2022. LCANets: Lateral Competition Improves Robustness Against Corruption and Attack. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 21232–21252. https://proceedings.mlr.press/v162/teti22a.html

[16] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288.

[17] Mengchen Zhu and Christopher J Rozell. 2013. Visual nonclassical receptive field effects emerge from sparse coding in a dynamical system. *PLoS computational biology* 9, 8 (2013), e1003191.

[18] Joel Zylberberg, Jason Timothy Murphy, and Michael Robert DeWeese. 2011. A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields. *PLoS computational biology* 7, 10 (2011), e1002250.

Fast Post-Hoc Normalization for Brain Inspired Sparse Coding on a Neuromorphic Device

Kyle Henke, Garrett T. Kenyon, and Ben Migliori

Abstract—Exploration of novel computational platforms is critical for the advancement of artificial intelligence as we approach the physical limitations of traditional hardware. Biologically accurate, energy efficient neuromorphic systems are particularly promising for enabling future breakthroughs because of their ability to process information in parallel and to scale using extremely low power. Sparse coding is a signal processing technique which has been known to model the information encoding in the primary visual cortex. When sparse solutions are solved using local neuron competition along with the unsupervised dictionary learning that mimics cortical development, we can build an end to end, hardware to software, brain inspired solution to a machine learning problem. In this article, we perform a detailed comparison of sparse coding solutions generated classically by orthogonal matching pursuit (OMP) implemented on a conventional digital processor with spike-based solutions obtained using the Intel Loihi neuromorphic processor. A novel "post-hoc" normalization technique to shorten simulation time for Loihi is presented along with analysis of optimal parameter selection, reconstruction errors, and unsupervised dictionary learning for Loihi approaches and their classical counterparts. Preliminary results show that both the Loihi full simulation approach and the post-hoc normalization approach are well suited to neuromorphic processors and operate in a size, weight and power regime that is not accessible by classical approaches. Ultimately, the use of this normalization technique allows for faster and, often, better solutions than demonstrated previously.

Index Terms—Neuromorphic computing, machine learning, artificial intelligence, neurocomputers, computer vision, signal processing

1 Introduction

The impending end of Moore's Law has created a need for new computational substrates if scientists are going to continue making progress in the pursuit of artificial general intelligence (AGI). In this work, we draw inspiration from neuromorphic computing and "wet" neuroscience as a potential solution to modern computing limitations with the end goal of AGI in mind. Specifically, we focus on the encoding of low-dimensional inputs into a sparse, efficient, high-dimensional space. This process is known to account for experimental neurobiological measurements [8], [12] and to support efficient neuromorphic computation [3].

Sparse coding is a signal processing technique which models neurobiological sensory input encoding in living systems. Part of solving the sparse coding problem involves the unsupervised learning of a dictionary. This dictionary, technically an overcomplete spanning set, can be done in a biologically inspired manner using only local information available at the synapse. Given a dictionary, the algorithm must choose which elements to apply to reconstruct an input with the fewest possible active dictionary coefficients. This approach is particularly interesting for our study as it enables an end-to-end neuromorphic approach, where both the hardware and algorithm are operating in a bioinspired manner.

Here, the focus is on a class of algorithms based on Hopfield networks, which are fully recurrent dynamical neural circuits governed by fixed point attractors [1]. Specifically, sparse attractor networks in which a uniform applied field is used to globally suppress activity, encouraging solutions consisting of a minimum number of active elements are considered and possess several properties that make them ideal for comparing digital and neuromorphic processors [2]. First, sparse attractor networks compute solutions to difficult optimization problems by settling into low-energy states that are embedded in complex energy landscapes containing multiple local minima. Second, by exploiting local learning rules to sculpt the energy landscape to better model the input data, such networks are naturally self-organizing and unsupervised learning emerges. Optimal solutions to sparse coding are NP-hard [2], but many approaches achieve adequate solutions. The bioinspired locally competitive algorithm (LCA) [13] implemented by Intel on Loihi [3] is an excellent example of such an approach. The full simulation provided from Intel is shown to allow for unsupervised dictionary learning and construction of sparse codes. However, the overall stimulated neural activity shows regions of rapid decay and slow decay that are not leveraged for any purpose in the Intel implementation. Here, we utilize a "post-hoc" normalization step that terminates the simulation at the end of the rapid decay segment and gives solutions faster and of lower reconstruction error than the full simulation, albeit at the cost of lower sparsity.

In this paper, we will first proceed by describing the low energy Loihi chip and the sparse coding problem in more

• *The authors are with the Computer, Computational, and Statistical Sciences (CCS-3), Los Alamos National Laboratory, NM 87545 USA. E-mail: {khenke, gkenyon, ben.migliori}@lanl.gov.*

Manuscript received 2 Sept. 2020; revised 27 Feb. 2021; accepted 8 Mar. 2021. Date of publication 24 Mar. 2021; date of current version 18 Aug. 2021. (Corresponding author: Kyle Henke.) Recommended for acceptance by S. Pakin, C. Teuscher, and C. Schuman. Digital Object Identifier no. 10.1109/TPDS.2021.3068777

detail. Subsequent text will explain the classical orthogonal matching pursuit (OMP) solution and our new post-hoc normalization technique to extend Loihi's capabilities. Finally, comparisons using similar optimal sparsity levels suggest OMP still beats the neuromorphic approaches in terms of reconstruction error and compute time, but at significantly higher power consumption.

Notably, our contribution enables researchers to choose where in the parameter space of speed and sparsity they wish to operate; this has implications for practical application of neuromorphic sparse coding.

2 Previous Work

Intel produced significant work on implementing the spiking LCA sparse coding/LASSO model onto their novel Loihi chip. We will give a brief overview of their device for better understanding of the implementation, but encourage readers to see [3] for more details.

The Loihi neuromorphic computing device implements spiking neural networks with neurons realized in hardware as the basic processing elements. Loihi, like its predecessors SpiNNaker[4] and TrueNorth [5], represents information as single-bit impulses, or spikes, transmitted at specific times and directed towards specific targets through programmable connections known as synapses. As a result, time and parallelism are explicitly incorporated into the representation and the network operates as a dynamical system communicating through these spikes. An implementation of spike timing dependent plasticity makes Loihi capable of online learning, in addition to being capable of inference [3].

In the general spiking neural network described on Loihi, spike trains are formulated as a sequence of Dirac delta functions of the form $\sigma(t) = \sum_k \delta(t - t_k)$ where t_k is the time of the kth input spike. Each neural unit on the device implements an asynchronous discrete-time implementation of Leaky Integrate and Fire (LIF) neurons with internal state variables consisting of a $u_i =$ *synaptic response current* and a resulting $v_i =$ *membrane potential* for each neuron $i = 1, 2, \ldots, n$ [3]. The system evolves in time and propagates information through the specified network graph with timing and patterns of neural activity defining the computational tasks. The entire network relationship can be summarized by the following equation for the synaptic response current for each neuron:

$$u_i(t) = \sum_{i \neq j} w_{i,j}(\alpha_u * \sigma_j)(t) + b_i, \tag{1}$$

where $\alpha_u(t) = \frac{1}{\tau_u}\exp(-t/\tau_u)H(t)$ is the synaptic filter impulse response with $H(t)$ as the unit step function and b_i is a constant bias. Here w_{ij} is the synaptic weight from neuron-j to i and τ_u is a time constant. We can then describe the membrane potential $v_i(t)$ by the following dynamical system differential equation:

$$\dot{v}_i(t) = -\frac{1}{\tau_v} v_i(t) + u_i(t) - \theta_i\sigma_i(t), \tag{2}$$

where $v_i =$ *membrane potential*, τ_v is a second time constant capturing the leakage of potential out of each neuron, and θ_i is the firing threshold for each v_i. Note all v_i's are initialized with values less than their respective θ_i and are reset to 0 after the spikes occur.

Plugging Equation (1) into Equation (2), we obtain the differential equation for the *ith* neuron's membrane potential as

$$\dot{v}_i(t) = -\frac{1}{\tau_v} v_i(t) + \sum_{i \neq j} w_{i,j}(\alpha_u * \sigma_j)(t) + b_i - \theta_i\sigma_i \tag{3}$$

Intel implemented a convolutional Spiking Locally Competitive Algorithm (S-LCA) problem on Loihi and defined the solutions as the stable, converged average spike rates of the general solution to the above system of differential equations. A more detailed explanation can be found in Section 3.3, but most notably for our work, they observed a rapid decrease in the cost function after only a few simulation steps, implying the neurons with the highest initial excitation are more likely to spike early in time and will immediately out-compete and inhibit other neurons. However, this observation was not applied to the Intel S-LCA solution.

3 Methods

3.1 Mean Zero Fashion-MNIST Data Set

Fashion-MNIST dataset [6] is a 28×28 greyscale labelled image dataset with ten classes. Fashion-MNIST is significantly more difficult than the classical MNIST challenge but is still tractable for most modern machine learning algorithms. However, the individual images in Fashion-MNIST are still too large (784 dimensions) to fit on many novel computing substrates studied by the authors (i.e., the D-Wave quantum annealer) and thus sparse PCA was used to obtain a reduced dimensional representation. Although Loihi is capable of scaling to handle datasets of this dimension easily, we purposefully studied the minimal feasible problem such that this study may be compared with others in our research series. To determine the sparse PCA coding, the Henze-Penrose (HP) [6] statistic for estimating class separability was used to estimate the minimum dimensionality for the Fashion-MNIST data set that does not substantially degrade classification performance [7]. The data set was reduced via sparse PCA and the HP statistic was calculated for each reduction. When ΔHP begins to increase rapidly (the "HP Rollover Point") it indicates the dataset compression is causing large changes in cluster overlap. Using the elbow criteria heuristic, the critical point for Fashion-MNIST was found to be 32 dimensions. To confirm that a 32-dimensional fashion MNIST contained a classification challenge of similarly difficulty to the uncompressed representation, we trained SVMs to classify both original and compressed datasets. The RMS change in the confusion matrix (where 0 is no accuracy, and 1.0 is perfect accuracy) between the 784-dimension and 32-dimension representation was .007. The SVM and HP metrics together demonstrate that neither the problem difficulty nor the classification accuracy significantly changed under compression [7]. The sPCA vectors were then used to reconstruct reduced dimensional images. Each image was divided into an array of 4 × 4 non-overlapping patches, with each patch 7 × 7 pixels in extent. Each patch was independently sparse coded. The resulting sparse reconstructions of each patch could be reassembled for comparison with the original image to verify that the sparse encoding was reasonable.

3.2 Sparse Coding

In mathematical notation, we are trying to solve the two step optimization problem

$$E(\tilde{I}, \tilde{a}) = \min_{\tilde{a}, \mathbf{F}} \frac{1}{2} \|\tilde{I} - \mathbf{F}\tilde{a}\|_2^2 + \lambda \|\tilde{a}\|_1; \qquad (4)$$

where $\tilde{I}$ is the input we want to reconstruct, $\tilde{a}$ is the sparse vector being solved for, and $\mathbf{F}$ is the set of basis functions (or features/neurons in our interpretation) used for the reconstruction. When the dictionary $\mathbf{F}$ is overcomplete, meaning there are more features than the length of the input, an infinite number of solutions to the minimization problem become possible. Therefor, a λ sparsity penalty is introduced into the cost function to represent the uniform applied field and create a unique solution. Additionally, when the dictionary $\mathbf{F}$ is of larger size, the process more closely resembles the human V1 receptive field [8] and we are able to stay closer to biology. For the interested reader, we recommend [8], [13]

3.3 Orthogonal Matching Pursuit (OMP)

Orthogonal matching pursuit is a classical iterative greedy algorithm used for solving the sparse coding problem for final solution $\tilde{a}$. At each step of the algorithm, the feature in the dictionary $\mathbf{F}$ which is most correlated with the current error vector, or residual, is selected and placed into the basis set of features for reconstructing the input image $\tilde{I}$. Next, the algorithm updates the residual, or error vector, by projecting the input image $\tilde{I}$ onto the linear subspace spanned by the features that have already been selected in previous iterations. Since the residual or error vector at each step of OMP are orthogonal to all of the features previously selected, no feature is selected twice and the subset of our dictionary $\mathbf{F}$ used for final reconstruction grows at each step. This process continues until some stopping criteria is reached. In this work, we iterate until enough features are selected to match the optimal sparsity level found for the separate Loihi techniques. Once the basis vectors are selected, the least squares problem of the selected columns is solved for non-zero coefficients of the sparse solution.

The OMP algorithm can be precisely stated with the following steps.

- Step 1: normalized the features in the dictionary $\mathbf{F}$ so that $\|\mathbf{F}_i\|_2 = 1$ for $p = 1; 2; \ldots; n$ where n is the number of features in the dictionary.
- Step 2: Select k as the number of nonzero coefficients we want in the solution vector.
- Step 3: Initialize residual vector $\tilde{r}_0 = \tilde{I}$ and pre-allocate the basis set for reconstruction A as a matrix of zeros of size $m \times k$ where m is the length of the input image $\tilde{I}$. Set iteration counter $i = 1$.
- Step 4: Find the feature $\mathbf{F}_p$ that has the maximum absolute value of the inner product with current residual $\tilde{r}_{i-1}$ i.e., solves
$$\max_{i} \mathbf{F}_p{}^T \tilde{r}_{i-1}:$$
- Step 5: Set column $A_i = \mathbf{F}_p$ from step 4.
- Step 6: Create projection operator onto the linear space spanned by A as $P_i = A(A^T A)^{-1} A^T$
  Update $\tilde{r}_i = (I - P_i)\tilde{I}$ where I is the Identity matrix.
- Step 7: If $i = k$, solve
$$\min_{\tilde{s}} \|A\tilde{s} - \tilde{I}\|_2^2:$$
  Set coefficients of the solution to the above minimization problem $\tilde{s}$ as non-zero coefficients in final corresponding sparse solution $\tilde{a}$. Here A is of rank $k \ll n$ and we select the k corresponding features of $\mathbf{F}$ that were selected during Step 4 to be active by the magnitude of the weights in $\tilde{s}$.
  Else, set $i = i + 1$ and return to Step 4.

3.4 Locally Competitive Algorithm (LCA) Implementation for Finding Sparse Representations of Data

In the human brain, individual neurons respond to specific stimulus at varying degrees of initial activation. After a period of time, the final encoding of the input is represented by only a few of the neurons which best characterize the data.

3.4.1 *Lateral Inhibition*

Previous work [13] has shown that *sparse coding* optimization problems can be solved using the dynamics of neural networks incorporating lateral inhibition. This biologically plausible implementation, known as a locally competitive algorithm, encourages sparse solutions by allowing neurons to compete with each other for fractional representation of the input. When used with a loss function that penalizes neural activation, the resulting dynamical system will evolve to a sparse solution. However, such dynamical systems are susceptible to local minima. They also require inter-layer connectivity, either directly or through regularizations.

3.4.2 *Neuromorphic Hardware for Sparse Attractor Networks*

The neuromorphic implementation of sparse coding sends binary signals as spikes in response to current flowing from signals sent by neighboring neurons or inputs which can either excite or inhibit one another while also decaying according to a leak of potential over time when there is no input. The neuromorphic implementation injects current weighted according to a non-orthonormal basis ($\mathbf{F}^T\tilde{I}$) into a network of neurons and reads outputs as spike rates $\tilde{a}$. The sparse coding loss function (Equation (4)) can be approximated as

$$E(t, \tilde{a}) = \frac{1}{2}\|\tilde{I} - \mathbf{F}\tilde{a}\|^2 + \int (\tilde{v} - T_\lambda(\tilde{v})); \qquad (5)$$

where we assume the existence of an input/output transfer function a ¼ T,\ðvÞ with threshold ,\, the details of which are determined by the nature of the leaky integrate-and-fire process. We use this substitution in order to write the dynamics in terms of the membrane potential v.

EðI; aÞ ¼ 1/2 ðI − FaÞ^T ðI − FaÞ þ Z ð v − T,\ðvÞÞ Z (6)

¼ 1/2 ðĨ^T Ĩ − 2ðĨ^T Fã Þ þ ã^T F^T Fã Þ ð v − TðvÞÞ: ,\ (7)

Taking the gradient of the cost function directly above with respect to the sparse vector a, substituting in T,\ðvÞ ¼ a, we arrive at

E/a ¼ −I^T F þ F^TFTðvÞ þ v − T ð vÞ: ,\ ,\ (8)

Taking the opposite (negative) direction of this gradient, we obtain the following set of coupled differential equations defining a non-linear dynamical system for the membrane potential with t as the appropriate time constants from Equations (1) and (2)

ṽ ¼ 1/t ð-ṽ þ F^T Ĩ − F^T F • T,\ðvÞþ T,\ðvÞÞ (9)

¼ 1/t ð-ṽ þ F^T Ĩ − ðF^T F − IÞ• T,\ðṽÞÞ: (10)

Although the precise form of T,\ðvÞ is unspecified, we nonetheless anticipate the network of leaky integrate-and-fire neurons implemented in neuromorphic hardware will tend to a state of activity that minimizes a sparse reconstruction objective function of the above form.

The -v term acts as the decay piece of the system, slowly decreasing the potential of each neuron over time. If an active neuron is not continuously excited, it will rapidly fall below firing threshold and deactivate because of inhibition and leaking potential. The input stimulus F^T I term charges up each of the neurons, exciting neurons whose features best match the input, and is the reformulation of the constant bias b from the vectorized Equation (1). ðF^TF − IÞ• T,\ðvÞ is the inhibitory signal corresponding to the original off diagonal weights w_{ij} also from Equation (1), forcing neurons which explain a similar component of the data to compete by inhibiting one another (Fig. 1), making our formulation a special case of the general form of network dynamics found in Equation (3). This competition continues until the equation converges to a stable fixed point sparse representation of average spike rates of the neurons, and this fixed point has been shown to be identical to the solution of the optimization problem [3], [13].

3.5 Post-Hoc Normalization

As the Loihi system evolves through time, the spiking neural network provides a rapid decrease in error in typically the first 100 timesteps (Fig. 2). During this period, the most important neurons immediately dampen out other active, but less important features. The remainder of the 6,000 steps of the simulation time are used to slowly achieve an initial fit plateau and then to effectively "normalize" the rates of the neurons corresponding to the most important features in the final solution. This happens because the final solution represents the dot product between the spike rate over simulation time and the corresponding feature vectors associated with active neurons. As a simulation runs longer, quiescent neurons have their contribution effectively diluted by slow but active neurons. This behaves similarly to feature normalization and allows the final result to scale to the inputs.

In an attempt to speed up the time to solution, we implement a *post-hoc* normalization technique where the simulation is ended after 100 steps and rates are immediately normalized. As (Fig. 8) and Table 1 suggests, these solutions are often times even better in terms of final reconstruction error than their full simulation counter parts and also activate the same features (Fig. 7). Although *post-hoc* normalization does not yield completely binary solution vectors, we believe the early normalization provides a better approximation because the rates of those neurons left over are much more uniform than if left to longer regularization.

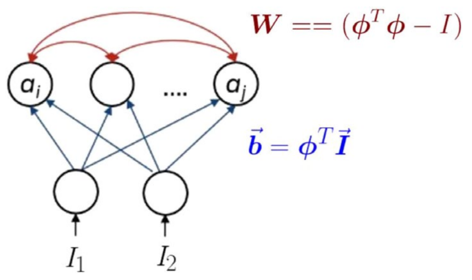


Fig. 1. Example of LCA structure in a network of V1 Neurons. Input image components I_1 and I_2 are fed into a layer of neurons which represent the features, or columns of **F**. Blue lines represent the connection strength between each $\mathbf{F}_i$ and the input image found by the vectors inner product. Each $\mathbf{F}_i$ competes with all other neurons, represented by the red connections and their respective inner products, to find which sparse combinations of features best represent the input as the system evolves.

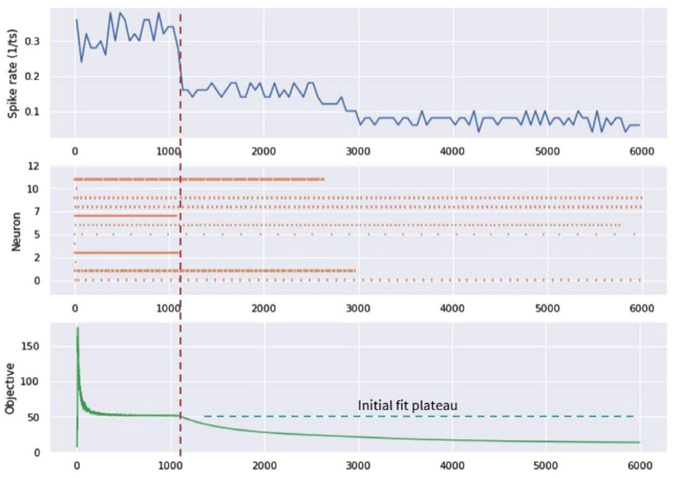


Fig. 2. Simulation time steps are on the x-axis. Red dotted line represents where most significant neuron competition has already occurred and activity is dampened out. Top Panel: Average spike rate of neurons in network. Middle Panel: A spike raster plot of how often active neurons are firing over the simulation. Lower Panel: The value of the objective function and the long regularization time after the initial fit plateau. The vast majority of the reduction in the loss function occurs early in the simulation.

TABLE 1
Table of Compute Times, Power Consumption, and Average Reconstructions

	Full Simulation Loihi	Post hoc Loihi	OMP
Avg. Compute Time (sec)	13.77 (1.697)	.316 (.03)	.007 (.006)
Power (W)	1.07	1.23	18.71
Energy (mJ)	101059.8	172.5	17610
Percent Active Features	9.22 (1.87)	9.33 (2.38)	9.37 (.01)
Avg. Reconstruction Error	.393 (.131)	.369 (.1574)	.178 (.08)

Standard deviations are in () for available data. There is significantly less power consumption for Loihi and the post hoc approach allows for faster, more energy efficient solutions to better approxiate OMP.

More importantly, this post-hoc normalized representation is closer to a true ℓ_0 norm. Take the following thought experiment: in a particularly lucky initialization, the only active neurons in a Loihi simulation are those corresponding to the optimal sparse solution. Those neurons all fire in a balanced way throughout the simulation, such that at the end, each has fired an equal number of spikes. When the rate/dictionary dot product is then taken, the end result is identical within a scaling factor to that achieved if the first spike of each neuron was taken and the simulation then stopped. This thought experiment is, in fact, a binary coding; however, Loihi does not natively achieve this type of coding. By limiting our sampling to a region of simulation in which inhibiting (i.e., active neurons) have not fully quieted non-active neurons but the loss has dropped to the initial plateau, we achieve a representation in which each active feature is either present or not without the dilution discussed above. Thus it is a closer approximation to a true ℓ_0 norm, but must be normalized to the correct magnitude to compute the reconstruction and compare it to the input. As the normalization is a computationally simple step on conventional hardware, we move it outside of the neuromorphic system and apply it post-hoc. This results in both a speed-up and a sparse code in which the simulation cannot dilute certain components away.

3.6 Unsupervised Dictionary Learning

The second part of the optimization process involves learning the best dictionary $\mathbf{F}$ for the given data set. Random features were first selected for the dictionary and a stochastic gradient descent algorithm with local Hebbian Learning rule was deployed using the both Loihi approaches and similar sparsity levels using classical Orthogonal Matching Pursuit [15]. The learning algorithm can be summarized as follows:

- Step 1: Select a random set of length m image patches I_p for $p = 1, 2, \ldots, n$. These patches will make up the initial features of $\mathbf{F}$. Here $n >> m$ to satisfy overcomplete requirement. Select subset of all patches for training.
- Step 2: Select mini-batch of training data and set $D\mathbf{F}_{Total} = 0$
- Step 3: Select image I from mini-batch.
- Step 4: Solve sparse coding problem for a using OMP or with LCA using a Loihi technique.
- Step 5: Calculate error of reconstruction

$$r = I - \mathbf{F}a.$$

- Step 6: Take outer product of error with sparse solution to find error in direction of active features. This matrix will have zeros down the columns of the inactive features.

$$D\mathbf{F} = r a^T.$$

- Step 7: $D\mathbf{F}_{Total} = D\mathbf{F}_{Total} + D\mathbf{F}$ Return to Step 3 until done with mini-batch
- Step 8: Update dictionary $\mathbf{F}$ with chosen learning rate h

$$\mathbf{F}_1 = \mathbf{F} + hD\mathbf{F}_{Total}.$$

- Step 9: If $\|\mathbf{F}_1 - \mathbf{F}\| <$ tolerance, END
- Step 10: $\mathbf{F} = \mathbf{F}_1$
- Step 11 (Optional): Normalize columns of $\mathbf{F}$
- Step 12: Return to Step 2

3.7 Input Amplification and Optimal Sparsity

As previous work has shown [14], in order to successfully reconstruct the Fashion-MNIST dataset using overlaid dictionary elements, a b parameter must be introduced into the cost function to amplify the input away from unit norm and allow multiple features to be utilized. This creates a modified cost function used in our scoring,

$$E(I, a) = \min_{a\mathbf{F}} \frac{1}{2} \|bI - \mathbf{F}a\|^2 + \lambda \|a\|_1. \tag{11}$$

The selection of the appropriate b parameter is dependent on specific architectures.

4 Results

Final sparse representations are found through a sequential process by first tuning the device towards the particular problem, and then training an optimal dictionary for the dimensionally reduced Fashion-MNIST data set.

4.1 Beta and Lambda Tuning for Loihi

For selecting the input amplification b term and the sparsity penalty λ we performed hyperparameter optimization and recorded the reconstruction error, computed as $\|I - \mathbf{F}a\|$ for a subset of images. Figs. 3 and 4 show the full simulation Loihi requires an input amplification of at least 8 in order to achieve reasonable reconstructions, and the post-hoc normalization doesn't require the amplification, but does benefit significantly.

4.2 Unsupervised Dictionary Learning With Sleep

The unsupervised learning algorithm from methods section F was implemented for the two Loihi techniques and with OMP using the same sparsity levels. OMP gave clean

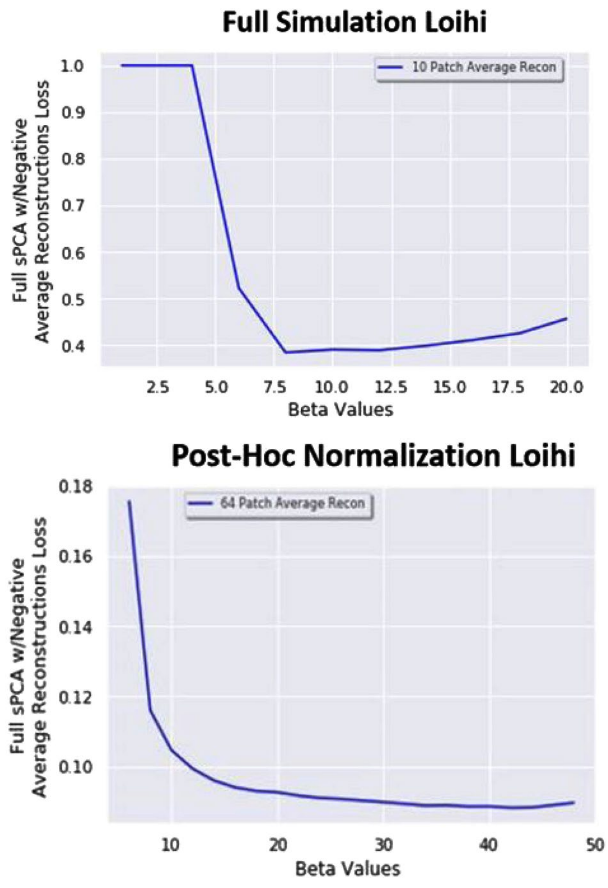


Fig. 3. Reconstruction error/loss plotted as a function of input amplification b. These values indicate how far away from unit norm the input must be for best reconstructions and are a unique characteristic of the distinct architectures. Full Loihi simulation on our problem has a more clear necessity to be above unit norm (Top), while the post-hoc approach starts with relatively good reconstructions and gets consistent improvement the higher the input is amplified (Bottom).

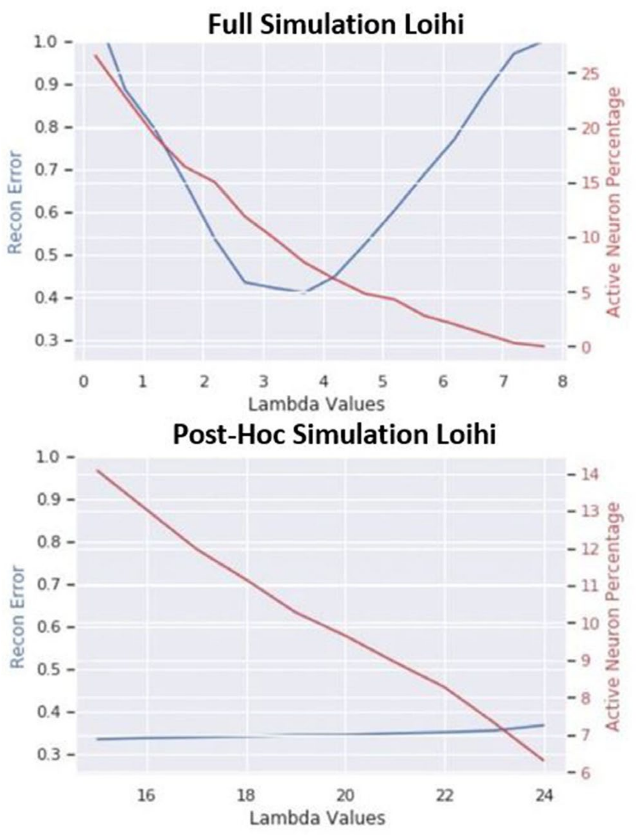


Fig. 4. Optimal sparsity penalty ,\ selection. Each graph has reconstruction error/loss in blue and active feature percentage in red. There is a clear optimal sparsity level dictated by full simulation Loihi (Top), and post-hoc normalization technique on Loihi (Bottom) roughly exhibits a monotonic relationship. We can impose the same level of sparsity as the full time simulation Loihi on the post hoc approach by drastically increasing the penalty term ,\.

reductions in overall reconstruction error, but the Loihi approaches encounter various problems.

Previous work on Loihi [16] shows introducing periodic normally distributed noisy examples (or sleep) into a learning algorithm allows for better convergence towards an optimal solution. With this motivation in mind, we initialized the full simulation Loihi algorithm with random features and introduced noisy data at set intervals as an amendment to the learning algorithm presented. Two separate runs were performed where Step 9 (feature normalization) was either performed, or skipped. Fig. 5, shows convergence of both techniques towards an optimal dictionary, but eventually start overfitting. To alleviate this result, we chose to stop the algorithm when the reconstruction error for a mini-batch was larger than the previous 3 batches. While the unnormalized approach results in slightly better reconstructions on the pure Loihi implementations, the average reconstruction on Loihi was significantly better when the trained OMP dictionary was fed to the device. The same algorithm was implemented with the post-hoc approach, but no learning was present, even though the reconstructions were consistently lower than their full simulation counterpart. A comparison of dictionary elements optimized for OMP are shown in Fig. 6 where we can see cleaner edges present after training.

4.3 Reconstructions and Performance

OMP dictionary learning gave best reconstructions for all techniques and was used for final comparisons. Table 1 shows each of the approaches used the same sparsity level. Post hoc is able to give better reconstruction almost 50 times faster and at significantly less energy. OMP uses more power, but also gives the best reconstructions in less time. See Fig. 8 for repatched results.

5 Discussion

The LCA implementation for solving the sparse coding problem was successfully run on the Loihi spiking neuromorphic chip and compared to the results from Orthogonal Matching Pursuit after hyperparameter optimization. Loihi is able to use weighted amounts of each feature for reconstruction, but because the solutions are spike rate limited, they do not have as much resolution as the continuous coefficients of the OMP solutions given the fixed length of the simulation. However, the post-hoc method is driven by our desire to converge as quickly as possible. Thus, it has even less ability to tune neural coefficients to create an accurate reconstruction as (see Fig. 2) the simulation is terminated before a subset of neurons can be inhibited and turned off.

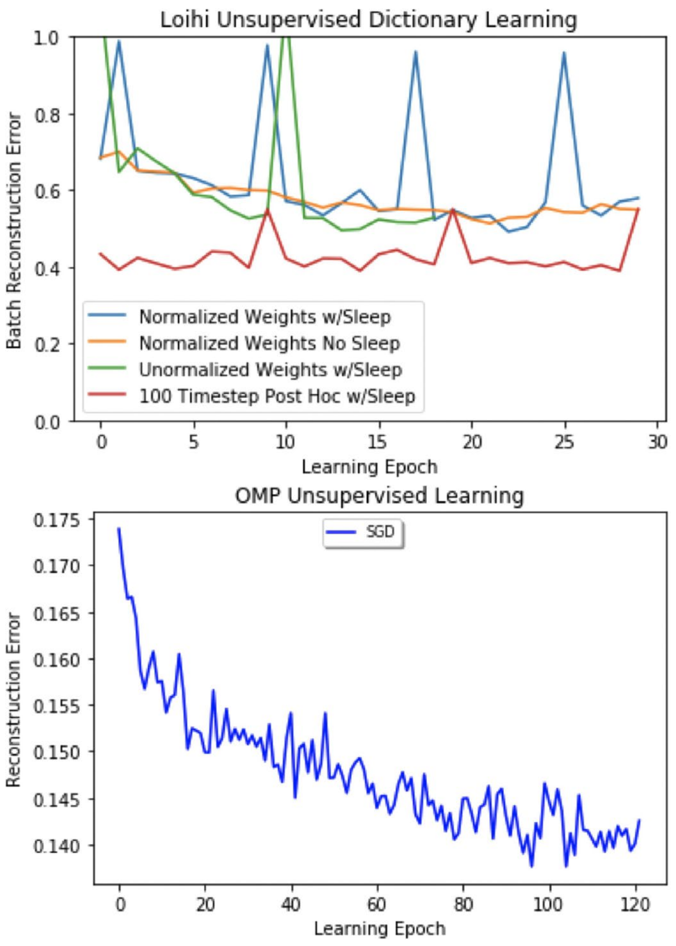


Fig. 5. Stochastic gradient descent with local Hebbian rule dictionary learning results. Random features are fed into the algorithm and solutions are found using OMP (Bottom) and Loihi simulation (Top). Orange is no noise and no sleep, blue is the result when dictionary features are normalized after every epoch with periodic noise, green line represents when features are not normalized, periodic noise is introduced with stopping criteria, and red is the post hoc unormalized with noise results. All full time simulation Loihi approaches achieve initial learning before overfitting with the un-normalized noisy approach providing best solution. OMP shows more consistent learning and much lower average reconstruction error per batch.

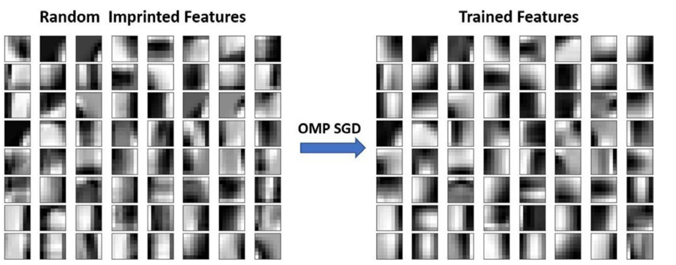


Fig. 6. Random patches selected for dictionary and their final form after the OMP unsupervised dictionary training.

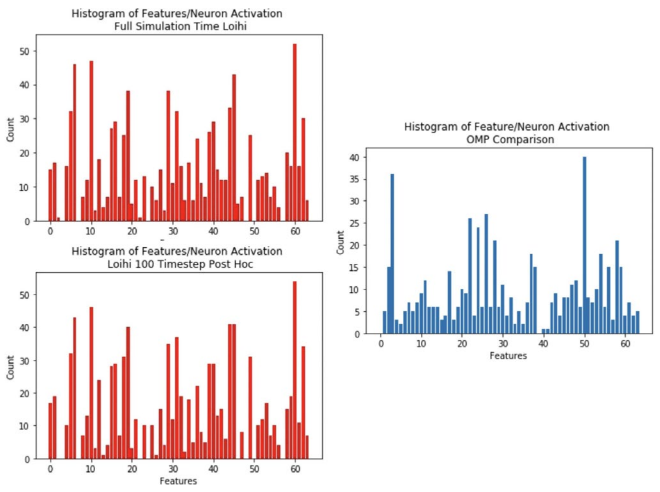


Fig. 7. (Top Left) Feature/Neuron activation for a 10 image subset solved by full time simulation Loihi. (Bottom Left) Post hoc feature/neuron activation counts are very similar to full simulation. (Right) OMP shows a different distribution of utilized features showing evidence the continuous coefficients result in different activity.

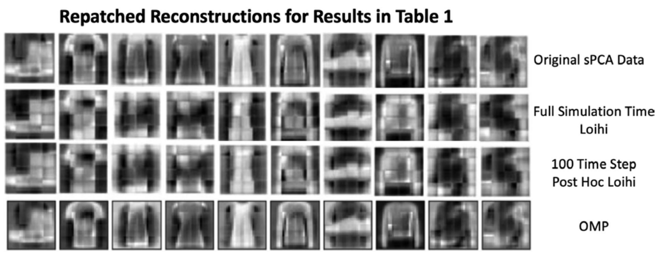

Repatched Reconstructions for Results in Table 1
Original sPCA Data
Full Simulation Time Loihi
100 Time Step Post Hoc Loihi
OMP

Fig. 8. Reconstruction comparison between Loihi techniques and similar sparsity levels with OMP. Average error for 10 images.

In the full length Loihi simulation, that subset becomes quiescent, and the longer run times perform a similar normalization as our "post-hoc" method. This simulation-length normalization is achieved in Loihi as a consequence of the method of reconstruction; the reconstruction is given as the dot product between the dictionary and the spike rates. By allowing certain neurons to become quiescent and then continuing to run the simulation, the average spike rate of some neurons is smoothly driven down. However, our "post-hoc" normalization step achieves a similar result in a fraction of the time, while utilizing contributions from all active neurons (i.e., dictionary elements) in a more uniform manner. To compare the performance, we constrain the OMP method, the Loihi method, and the post-hoc Loihi method to all have similar sparsity (approximately 9.3 percent percent activation). When this comparison is performed, the post-hoc method results in a statistically lower reconstruction error than the full simulation in less than 5 percent of the time. This result will encourage the use of post hoc solution when an optimal dictionary has already been trained because reconstructions are better and feature activations are relatively the same.

To perform an adequate comparison, the continuous OMP implementation was forced to have the same sparsity as the other two methods but had different feature activations. As a result of the continuous coefficients available to OMP, this method resulted in the lowest reconstruction error; however, our analysis of power consumption (Table 1) shows that this technique consumes significantly more power and energy.

One of the important observations from these experiments relates to the dictionary learning phase. In the existing Loihi full length simulation method, dictionary learning can take place. It is also successful in the OMP methods. However, in the studies we present here, the use of post-hoc normalization appears to place the system very quickly into a local minima for a particular dictionary. This can be seen in Fig. 5, where the post-hoc reconstruction error remains fixed throughout the epochs studies. Interestingly, when probed with random dictionary elements, the same phenomena is observed. This imposes an effective limitation on post-hoc normalization in that it makes the best of a given dictionary and is not yet amenable to further learning. We believe this is an artifact of the way in which batches and post-hoc normalization are computed, and warrants further study. Because we were able to achieve our initial objective of significant reconstruction speedup without further learning, we did not pursue this further within the scope of this paper. It should be noted that this could be an area of further optimization for the technique.

6 Conclusion

Our exploration of biologically inspired, energy efficient neuromorphic systems, for the purposes of continuing the advancement of machine learning past the limitations of classical approaches, has illuminated some of the potentially breakthrough advantages of novel computing substrates such as Loihi. By demonstrating significant speedup for a pre-defined dictionary at similar or better accuracy, our work indicates yet another route to algorithmically improve overall system throughput without any hardware changes. The application of post-hoc normalization allows computation of sparse representations in less than 5 percent of the full simulation time once the lambda ,\ (penalty) is tuned to provide the desired sparsity found in the full simulation. This has significant implications for the practical use of neuromorphic methods such as Loihi in resource-constrained environments where available power and time for computation are finite. Unlike traditional sparse coding, manipulations to the cost function are required for optimal reconstruction when using full length simulation Loihi and our novel post-hoc normalization as described in the text. By forcing the sparsity of OMP to match that of what was optimal for different Loihi techniques, we were able to generate classical comparisons of this method, albeit at higher power cost. Although OMP out-performed the Loihi approaches in terms of raw reconstruction, the Loihi techniques are rate coded solutions with fewer degrees of freedom and provide competitive results at dramatically lower power. In addition, the approximately 50 times speed up of our post-hoc reconstruction compared with the full length Loihi simulation provides an avenue for the neuromorphic device to better compete with traditional techniques in terms of raw computation time. Future work will include classical and quantum annealing LCA comparisons for the true binary sparse coding approximation of the post-hoc solutions, classical LCA to compare with the full simulation, and classification scores for all methods to demonstrate impact on downstream machine learning techniques.

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational properties," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554–2558, 1982.

[2] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, pp. 227–234, Apr. 1995.

[3] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan./Feb. 2018.

[4] M. M. Khan *et al.*, "SpiNNaker: Mapping neural networks onto a massively-parallel chip multiprocessor," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2008, pp. 2849–2856.

[5] A. Cassidy *et al.*, "TrueNorth: A high-performance, low-power neurosynaptic processor for multi-sensory perception, action, and cognition," *Comput. Sci.*, 2016.

[6] N. Henze and M. D. Penrose, "On the multivariate runs test," *Ann. Statist.*, vol. 27, no. 1, pp. 290–298, 1999.

[7] Waagen *et al.*, "Fashion MNIST charts for LANL discussions," Air Force Research Laboratory (AFRL), 2019.

[8] B. Olshausen and D. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.

[9] K. Boahen, "A neuromorph's prospectus," *Comput. Sci. Eng.*, vol. 19, pp. 14–28, Mar. 2017. [Online]. Available: https://aip.scitation.org/doi/abs/10.1109/MCSE.2017.33?journalCode=csx

[10] V. Kornijcuk *et al.*, "Leaky integrate-and-fire neuron circuit based on floating-gate integrator," *Front. Neurosci.*, vol. 23, May 2016, Art. no. 212.

[11] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*

[12] M. Zhu and C. J. Rozell, "Visual nonclassical receptive field effects emerge from sparse coding in a dynamical system," *PLoS Comput. Biol.*, vol. 9, no. 8, 2013, Art. no. e1003191.

[13] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding viathresholding and local competition in neural circuits," *Neural Comput.*, vol. 20, pp. 2526–2563, Oct. 2008.

[14] K. Henke, G. T. Kenyon, and B. Migliori, "Machine learning in a post moore's law world: Quantum vs. neuromorphic substrates," in *Proc. IEEE Southwest Symp. Image Anal. Interpretation*, 2020, pp. 74–77.

[15] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. New York, NY, USA: Wiley, 1949.

[16] Y. Watkins, E. Kim, A. Sornborger, and G. Kenyon, "Using sinusoidally-modulated noise as a surrogate for slow-wave sleep to accomplish stable unsupervised dictionary learning in a spike-based sparse coding model," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 1482–1487.

[17] T. E. Potok *et al.*, "A study of complex deep learning networks on high-performance, neuromorphic, and quantum computers," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, 2018, Art. no. 19.

Alien vs. Predator: Brain Inspired Sparse Coding Optimization on Neuromorphic and Quantum Devices

Kyle Henke
Computer, Computational,
and Statistical Sciences (CCS-3)
Los Alamos National Laboratory
Los Alamos, United States
Email: khenke@lanl.gov

Ben Migliori
Computer, Computational,
and Statistical Sciences (CCS-7)
Los Alamos National Laboratory
Los Alamos, United States
Email: ben.migliori@lanl.gov

Garrett T. Kenyon
Computer, Computational,
and Statistical Sciences (CCS-3)
Los Alamos National Laboratory
Los Alamos, United States
Email: gkenyon@lanl.gov

2020 International Conference on Rebooting Computing (ICRC) | 978-1-6654-1975-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICRC2020.2020.00015

***Abstract*—Machine Learning has achieved immense progress by exploiting CPUs and GPUs on classical computing hardware. However, the inevitable end of Moore's Law on these devices requires the adaptation and exploration of novel computational platforms in order to continue these advancements. Biologically accurate, energy efficient neuromorphic systems and fully entangled quantum systems are particularly promising arenas for enabling future advances. In this work, we perform a detailed comparison on a level playing field between these two novel substrates by applying them to an identical challenge.**

We solve the sparse coding problem using the biologically inspired Locally Competitive Algorithm (LCA) on the D-Wave quantum annealer and Intel Loihi neuromorphic spiking processor. The Fashion-MNIST data set was chosen and dimensionally-reduced by sparse Principal Component Analysis (sPCA). A sign flipped second data set was created and appended to the original in order to give each class a mean zero distribution, effectively creating an environment where the data could not be linearly separated. An early in time normalization technique for Loihi is presented along with analysis of optimal parameter selection and unsupervised dictionary learning for all three variations. Studies are ongoing, but preliminary results suggest each computational substrate requires casting the NP-Hard optimization problem in a slightly different manner to best capture the individual strengths, and the new Loihi method allows for more realistic comparison between the two.

I. Introduction

Throughout the scientific community there is growing concern of our increasingly rapid approach towards the theoretical limits of classical computation, better known as the end of Moore's Law. In spite of this knowledge, reliance on machine learning and autonomous products has exponentially grown, and hence, the exploration of novel computational platforms which can overcome projected deficiencies is needed. The incredible efficiency of the human mind and powerful optimization potential of entangled quantum systems provides two avenues where projected limitations may be alleviated.

Additionally, the importance of unsupervised learning has grown in the AI community as a means of advancing the field out of the narrow, or weak, AI regime and into a strong AI, or general intelligence setting. Here, we draw inspiration from the a human brain's unmatched ability to generalize information by solving the sparse coding problem using the biologically accurate Locally Competitive Algorithm (LCA) and performing unsupervised dictionary learning using a local Hebbian rule on the D-Wave quantum annealing device and the Loihi spiking neuromorphic processor.

a) Analog vs. Digital: Classical computing can be divided into two categories, depending on whether the underlying circuits are digital or analog. Digital logic gates are universal, allowing the construction of computing architectures capable of running any valid program. Analog computers, conversely, use the dynamical evolution of a physical system to perform a given computation. While analog computers can be extremely fast and power efficient, the noise associated with the evolution of such systems causes difficulty when programming and often represents a major limitation.

Neuromorphic processors, drawing inspiration from biological brains, comprise a class of ultra low-power analog devices that are capable of self-organizing in response structured input. In this sense, neuromorphic processors are able to "program" themselves, potentially alleviating a major limitation of analog computing devices. As we require computers to exhibit greater autonomy and intelligence, and the focus of computing applications shifts toward machine learning and machine intelligence, analog neuromorphic processors are likely to play an increasingly prominent role [1].

The question of analog vs. digital systems maps into the quantum computing regime as well. Like their classical digital counterparts, quantum logic gates can, in principle, enable the construction of computers capable of running any valid program. Also like their classical counterparts, quantum analog computers, such as quantum annealing machines, are physical systems in which the dynamical evolution of the system performs the desired computation. Both analog and gate-based approaches to quantum computing seek to exploit quantum entanglement, superposition, and other quantum effects to solve problems that would otherwise be intractable using a purely classical approach, a goal known as quantum supremacy [2]. In this work, we focus on analog quantum computing entirely.

b) Quantum Annealing: Quantum annealing refers to a class of quantum analog computers that "compute" by settling into a low-energy state of a particular Hamiltonian which describes the quantum system. The design of the first commercially available quantum annealing machines, produced by D-Wave [3], illustrates the general concept. D-Wave computers are constructed from superconducting quantum interference devices (SQUIDS), with each SQUID subject to a local magnetic field and coupled to neighboring SQUIDS by adjustable links comprised of Josephson Junctions. Each SQUID represents a binary qubit that has two possible observable states, 0 and 1. In general, each qubit can exist in an entangled superposition of both states. The user programs a D-Wave computer by specifying the values of the pairwise coupling coefficients between qubits along with the local magnetic field applied individually to each qubit.

In a classical annealing process, the system is prepared in a randomly chosen initial state at a finite temperature. Thermal fluctuations cause classical annealing systems to jump over local energy barriers and into new energy states, with transitions between states driven stochastically according to a Boltzmann distribution. Jumps to higher energy states are possible but exponentially less likely than transitions to lower energy states. As the temperature is lowered, classical annealing systems tend to settle into progressively lower energy states. The annealing process is repeated multiple times using different randomly chosen initial conditions, with the lowest final energy state achieved across all annealing runs representing the answer to the computation.

The implementation of quantum annealing differs from classical annealing in several important respects. In quantum annealing, the system is not prepared in a single randomly chosen state as with a classical annealing process. Rather, a quantum annealing machine is prepared in an initial state that consists of a quantum superposition of all possible states. Using the D-Wave quantum annealing machine as a concrete example, a transverse magnetic field is applied to each qubit in the absence of any coupling between qubits. A D-Wave computer consisting of N qubits is thus prepared in an initial state that represents the superposition of all 2^N possible observable states. In part, the computing power of quantum annealing machines derives from the ability to more effectively sample the entire energy landscape.

Whereas classical annealing involves slowly lowering the temperature of the system, quantum annealing is implemented on the D-Wave by gradually turning on the user specified Hamiltonian while the transverse magnetic field is gradually turned off. Rather than jumping over local energy barriers, in quantum annealing a transition to new energy states is accomplished via quantum tunneling. In theory, quantum tunneling allows quantum annealers to avoid getting trapped in local minima. In practice, it is unlikely that any existing quantum annealing process can maintain quantum coherence over sufficiently large spatial and temporal regimes to achieve pure quantum annealing, but this limitation can be partially alleviated by running the anneal multiple times and sampling from the underlying Boltzmann distribution where the global minimum should eventually be present and hence observed.

c) Neuromorphic Computing: Inspired by biology, Intel's Loihi neuromorphic computing device implements spiking neural networks with neurons as the basic processing elements. Loihi, like its predecessors SpiNNaker [4] and TrueNorth [5], represents information as single-bit impulses, or spikes, transmitted at specific times and directed towards specific targets through connections known as synapses. Effectively, time and parallelism are explicitly incorporated into the representation and the network operates as a dynamical system communicating through these spikes. Because Loihi has spike timing dependent plasticity, it is not an inference-only device, but can also be used for online learning [6].

Users specify input as a sequence of Dirac delta functions or bias currents of the form $\sigma(t) = \sum_k \delta(t - t_k)$ where t_k is the time of the k-th input spike. Each neural unit on the device implements an asynchronous discrete-time implementation of Leaky Integrate and Fire (LIF) neuron with internal state variables consisting of a *synaptic response current* and a resulting *membrane potential* [7]. The system evolves in time and propagates information through the defined network graph with timing and patterns of neural activity defining the computational tasks.

The Loihi neuromorphic device we consider employs digital interconnections between spiking neurons to implement sparse attractor-based neural networks. A single Loihi chip has a manycore mesh comprising 128 neuromorphic cores, three embedded x86 processor cores, and off-chip communication interfaces that connect the mesh in four directions to other chips. An asynchronous network-on-chip (NoC) communicates between cores in the form of packetized messages. The NoC writes, reads requests, reads response messages for core management and x86-to-x86 messaging, spike messages for SNN computation, and provides barrier messages for time coordination between cores. All messages are collected by an external host CPU or on-chip by the x86 cores. Each neuromorphic core contains 1,024 basic spiking neural units grouped into sets of trees creating the neurons. In total, the basic architecture allows for 4096 on-chip cores and up to 16,384 chips if the messages between chips are formulated in a hierarchical manner to allow off chip communication over a second-level network [6].

II. Related Work

An earlier comparison of quantum annealing and neuromorphic architectures sampled from a distribution defined by a Limited Boltzmann Machine whose inter- and intra-layer connectivity was constrained by the topology of the D-Wave while their neuromorphic implementation demonstrated a low-power implementation using memristive interconnects [19]. Here, we use only the lowest energy solution returned by the D-Wave as an estimate of the optimal binary sparse representation of the data and seek to solve the same optimization problem in neuromorphic hardware. Other work has shown [16] it is possible to compare performance between multiple non-von-Neumann substrates on the same task. A crucial result of that work was the introduction of β parameter in the objective function to overcome representational constraints imposed by the use of normalized weights in combination with binary activations. In this current work, we optimize the cost function for both the neuromorphic and quantum devices in terms of both the β and λ parameters, employ a more powerful patch based representation of the data, and learn dictionary weights.

III. Methods

A. Mean Zero Fashion-MNIST Data Set

One of the primary challenges with comparing quantum and neuromorphic systems is the fundamental difference in data-handling capacity of the two methods. Here, we introduce a dataset to which both methods can be equally applied.

Fashion-MNIST dataset [8] is a 28×28 greyscale labelled image dataset with ten classes. Fashion-MNIST is significantly more difficult than the classical MNIST challenge but is still tractable for most modern machine learning algorithms. However, the individual images in Fashion-MNIST are still far too large (784 dimensions) to fit on the D-Wave annealer. The Henze-Penrose (HP) [9] statistic for estimating class separability was used to estimate the minimum dimensionality for the Fashion-MNIST data set that does not substantially degrade classification performance [10]. The data set was reduced via sparse Principal Component Analysis (sPCA) and the HP statistic was calculated for each reduction. To compute the HP statistic, first the minimum spanning tree between classes is calculated using the Euclidean distance between sPCA representations, then the number of transitions between the classes in this spanning tree is calculated and represented by the symbol S_{FR}. For a two class system we can express this statistic in the equation below:

$$\mathrm{H}_{xy} = 1 - S_{FR}\frac{n_x + n_y}{2n_{xy}} \tag{1}$$

where n_x and n_y are the number of nodes in the tree for class x and y respectively. When ΔHP begins to increase rapidly (the "HP Rollover Point") it indicates that dataset compression is causing large changes in cluster overlap. The critical point for Fashion-MNIST was found to be 32 dimensions. To confirm that a 32-dimensional fashion MNIST contained a classification challenge of similarly difficulty to the uncompressed representation, we trained linear Support Vector Machines (SVMs) to classify both original and compressed datasets. The RMS change in the confusion matrix (where 0 is no accuracy, and 1.0 is perfect accuracy) between the 784-dimension and 32-dimension representation was .007. The SVM and HP metrics together demonstrate that neither the problem difficulty nor the classification accuracy significantly changed under compression [10]. To remove the linear separability present in the Fashion-MNIST dataset, we appended a sPCA coefficient-flipped 2nd data set so that all classes were of mean zero. The

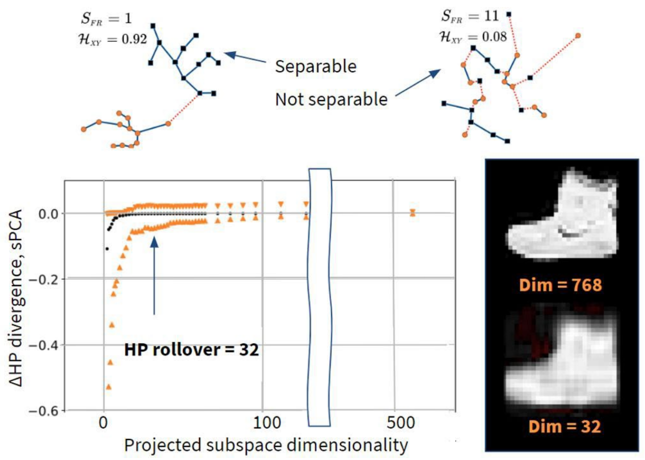


Fig. 1: An example of the HP analysis for two classes and for the dataset under discussion. Upper panel: An example of two minimal spanning trees for separable and non-separable data, showing (left) single transitions between clusters and (right) many transitions between classes in the non-seperable data. Lower panel: HP statistic as a function of dimensionality. Triangles show bounds and circles show the mean. As the dimensionality is reduced, the HP statistic eventually drops rapidly, indicating the rollover point. Right inset: Reconstruction examples of Fashion-MNIST at original dimensionality (768) and at the HP rollover point (32)

sPCA vectors were then used to reconstruct reduced dimensional images. Each image was divided into an array of 4×4 non-overlapping patches, with each patch 7×7 pixels in extent. Each patch was independently sparse coded. The resulting sparse reconstructions of each patch could be reassembled for comparison with the original image to verify that the sparse encoding was reasonable. The compressed/augmented Fashion-MNIST dataset was now not linearly separable and non-liner classification techniques averaged a corresponding 8-10 percent drop in accuracy.

By establishing this challenge with a dimensionality that would be dramatically smaller in dimension than the limited number of qubits on the D-Wave annealer, we enable a closer comparison of the two methods without making extrapolations required when separate datasets are used.

B. Sparse Coding:

Neural networks represent an increasingly important class of algorithms in which exact solutions are not necessary and good solutions are often good enough. Here, the focus is on a class of algorithms based on Hopfield networks, which are fully recurrent dynamical neural circuits governed by fixed point attractors [11]. Specifically, sparse attractor networks in which a uniform applied field is used to globally suppress activity, encouraging solutions consisting of a minimum number of active elements are considered [12]. Sparse attractor networks possess several properties that make them ideal for comparing quantum and neuromorphic processors. First, sparse attractor networks compute solutions to difficult optimization problems by settling into low-energy states that are embedded in complex energy landscapes containing multiple local minima. Second, by exploiting local learning rules to sculpt the energy landscape to better model the input data, such networks are naturally self-organizing.

C. Locally Competitive Algorithm (LCA) Implementation for Finding Sparse Representations of Data

a) Quantum Annealing for Sparse Attractor Networks: Given an overcomplete, non-orthonormal basis $\{\varphi_i\}$, inferring a sparse representation involves finding the minimal set of non-zero activation coefficients $\boldsymbol{a}$ that accurately reconstruct a given input signal $\boldsymbol{I}$, corresponding to a minimum of the following energy function:

$$E(\boldsymbol{I}, \boldsymbol{a}) = \min_{\{\boldsymbol{a}\}}[\frac{1}{2}||\boldsymbol{I} - \boldsymbol{\varphi a}||^2 + \lambda||\boldsymbol{a}||_0] \quad (2)$$

where λ is a trade-off parameter that determines the balance between reconstruction error and the number of non-zero activation coefficients. A larger λ will result in a more sparse solution to Eq. (7). This energy function is non-convex and contains multiple local minima, so that finding a sparse representation falls into an NP-hard complexity class of decision problems [13] [14].

b) Lateral inhibition: Previous work [15] has shown that *sparse coding* optimization problems can be solved using the dynamics of neural networks incorporating lateral inhibition. This biologically plausible implementation, known as a locally competitive algorithm (LCA), encourages sparse solutions by allowing neurons to compete with each other for fractional representation of the input. When used with a loss function that penalizes neural activation, the resulting dynamical system will evolve to a sparse solution. However, such dynamical systems are susceptible to local minima. They also require inter-layer connectivity, either directly or through regularizations.

c) Transformation relations: In a quantum annealing system, each neuron is mapped to a binary qubit. Because the observable states of any qubit/neuron are 0 and 1, each qubit/neuron is treated as a "quantum object" that either fires a spike (1) or is silent (0). Because each qubit/neuron is a quantum object, the state of any qubit/neuron is described in general by a superposition of 1 and 0, in which the qubit/neuron is both active and non-active at the same time, a logical impossibility for any classical system. If this quantum superposition is maintained, it is the characteristic which should allow the D-Wave to explore the entire energy landscape at once.

The D-Wave 2000Q [3] [13] [14] searches for optimal solutions to a (discrete) Ising system consisting of N_q binary variables described by the following classical Hamiltonian:

$$H(\boldsymbol{h}, \boldsymbol{Q}, \boldsymbol{a}) = \sum_i^{N_q} h_i a_i + \sum_{i<j}^{N_q} Q_{ij} a_i a_j \quad (3)$$

with binary activation coefficients $a_i = \{0, 1\} \forall i \in (1, 2, 3, ..., N_q)$. This objective function defines a Quadratic Unconstrained Binary Optimization (QUBO) problem. We cast our sparse coding problem, Eq. (7), into QUBO form, Eq. (3), by the transformations [13] [14]:

$$\begin{aligned} h_i &= (-\boldsymbol{\varphi}^T\boldsymbol{I} + (\lambda + \frac{1}{2}))_i \\ Q_{ij} &= (\boldsymbol{\varphi}^T\boldsymbol{\varphi})_{ij}. \end{aligned} \quad (4)$$

In Eq. (4), the bias term $\boldsymbol{h}$ in the Ising model is proportional to the weighted input $\boldsymbol{\varphi}^T\boldsymbol{I}$ while the coupling term $\boldsymbol{Q}$ corresponds to lateral competition (see also [15]) between qubits given by the interaction matrix $\boldsymbol{\varphi}^T\boldsymbol{\varphi}$. Note that the sparsity trade-off parameter λ appears as a uniform applied magnetic field that encourages all qubits to be in the $a_i = 0$ state [13] [14].

d) D-Wave 2000Q hardware : The D-Wave 2000Q [3] consists of 2000 qubits and 5600 couplers arranged into 12x12 unit cells, forming a Chimera structure with dimensions 12x12x8. Sparse interactions between qubits are restricted to the 16 connections within a unit cell and the 16 connections between nearest-neighboring unit cells [3] [13] [14] (see Fig. 2). One qubit can therefore interact with at most 6 other qubits.

e) Embedding technique: Despite the sparsity of physical connections on the D-Wave, it is nonetheless possible to construct graphs with arbitrarily dense connectivity by employing "embedding" techniques. Embedding works by chaining together physical qubits so as to extend the effective connectivity but at the cost of reducing the total number of available logical qubits. The D-Wave API provides a heuristic algorithm that searches for an optimal embedding that minimizes the number of physical qubits that are chained together (see Fig. 3 for an example).

The exact mapping of a spin glass problem onto the physical D-Wave 2000Q chimera, including defects, can typically contain approximately $N_q \sim 1750$ spins (qubits) with > 4000

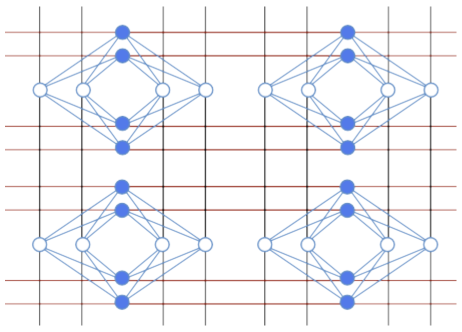


Fig. 2: A subset of the D-Wave consisting of 32 qubits arranged into 4 unit cells. Vertical (horizontal) orientations drawn as blue (white) circles. Interactions occur through the 16 bipartite interactions (blue edges) within a unit cell. Nearest neighboring bipartite interactions between each pair of nearest neighboring unit cells are characterized as black and red edges.

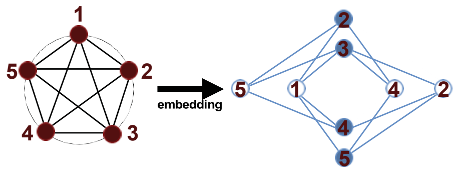


Fig. 3: Embedding example. 5 fully-connected particles (left, red filled circles connected by 10 black edges) are mapped onto a single unit cell using 8 qubits (right, 8 blue circles) and all 16 physical connections (blue edges).

local spin-spin interactions. In contrast, embedding an arbitrary QUBO problem onto the same 2000Q chimera typically allows no more than $N_q \sim 64$ nodes (logical qubits) but these nodes may be fully connected. Thus, embedding effectively trades qubits for connectivity and is in and of itself an NP hard optimization problem.

f) Neuromorphic Hardware for Sparse Attractor Networks: The neuromorphic implementation of sparse coding sends binary signals as spikes in response to current flowing from signals sent by neighboring neurons or inputs which can either excite or inhibit one another while also decaying according to a leak of potential over time when there is no input. The neuromorphic implementation injects current weighted according to a non-orthonormal basis ($\boldsymbol{\varphi}^T\boldsymbol{I}$) into a network of neurons and reads outputs as spike rates $\boldsymbol{a}$. The sparse coding loss function can be approximated as (Equation 5)

$$E(\boldsymbol{I}, \boldsymbol{a}) = \min_{\{\boldsymbol{a}\}}[\frac{1}{2}||\boldsymbol{I} - \boldsymbol{\varphi a}||^2 + \gamma(\boldsymbol{u} - T_\lambda(\boldsymbol{u}))] \qquad (5)$$

where we assume the existence of an input/output transfer function $\boldsymbol{a} = T_\lambda(\boldsymbol{u})$ with threshold λ, the details of which are determined by the nature of the leaky integrate-and-fire process.

Taking the negative gradient of the cost function (Equation 5) with respect to the sparse vector $\boldsymbol{a}$, we obtain the following set of coupled differential equations defining a non-linear dynamical system:

$$\dot{\boldsymbol{u}} = \frac{1}{\tau}(-\boldsymbol{u} + \boldsymbol{\varphi}^T\boldsymbol{I} - \boldsymbol{\varphi}^T\boldsymbol{\varphi} \cdot \boldsymbol{a} + T_\lambda(\boldsymbol{u})) \qquad (6)$$

Although the precise form of $T_\lambda(\boldsymbol{u})$ is unspecified, we nonetheless anticipate that network of leaky integrate-and-fire neurons implemented in neuromorphic hardware will tend to a state of activity that minimizes a sparse reconstruction objective function of the above form. We will later discuss a normalization technique which can be argued gives an approximation of the $||\boldsymbol{a}||_0$ used in the D-Wave cost function.

The $-\boldsymbol{u}$ term acts as the decay piece of the system, slowly decreasing the potential of each neuron over time. If an

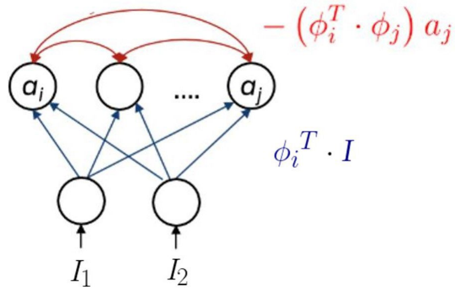


Fig. 4: Example of LCA structure in a network of V1 Neurons. Input image components I_1 and I_2 are fed into a layer of neurons which represent the features, or columns of $\boldsymbol{\varphi}$. Blue lines represent the connection strength between each $\boldsymbol{\varphi}_i$ and the input image found by the vectors inner product. Each $\boldsymbol{\varphi}_i$ competes with all other neurons, represented by the red connections and their respective inner products, to find which sparse combinations of features best represent the input as the system evolves.

active neuron is not continuously excited, it will rapidly fall below firing threshold and deactivate because of inhibition and leaking potential. The $\boldsymbol{\varphi}^T\boldsymbol{I}$ term charges up each of the neurons, exciting neurons whose features best match the input. $\boldsymbol{\varphi}^T\boldsymbol{\varphi} \cdot \boldsymbol{a}$ is the inhibitory signal, forcing neurons which explain a similar component of the data to compete by inhibiting one another. This competition continues until the equation converges to a stable fixed point sparse representation of average spike rates of the neurons, and this fixed point has been shown to be identical to the solution of the optimization problem [6] [15].

g) Post-Hoc Normalization: As the Loihi system evolves through time, the spiking neural network provides a rapid decrease in error in the first 100 or so timesteps (figure [5]). During this period, the most important neurons immediately dampen out other active but less important features. The remainder of the 6000 steps of the simulation time are used to slowly achieve an initial fit plateau and then to normalize the rates of the most important features in the final solution. This happens because the final solution represents the dot product between the spike rate over simulation time and the corresponding feature vectors associated with active neurons. As a simulation runs longer, quiescent neurons have their contribution effectively diluted by slow but active neurons. This behaves similarly to feature normalization, and does not have an analogue in the quantum system under comparison.

In an attempt to speed up the time to solution and produce a more equal comparison, we implement a $post-hoc$ normalization technique where the simulation is ended after 100 steps and rates are immediately normalized.

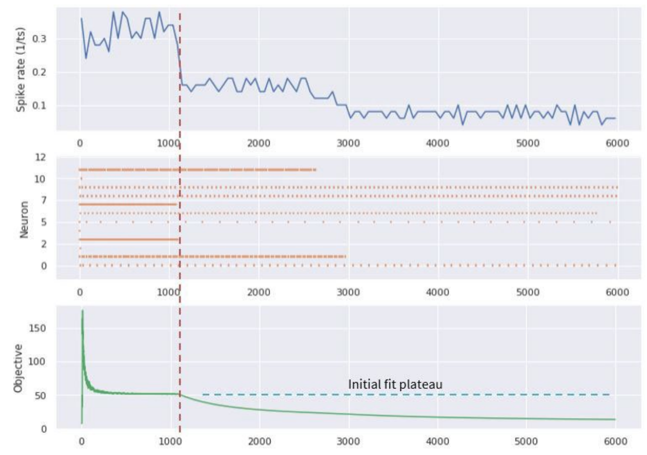


Fig. 5: Simulation time steps are on the x-axis. Red dotted line represents where most significant neuron competition has already occurred and activity is dampened out. Top Panel: Average spike rate of neurons in network. Middle Panel: A spike raster plot of how often active neurons are firing over the simulation. Lower Panel: The value of the objective function and the long regularization time after the initial fit plateau. The vast majority of the reduction in the loss function occurs early in the simulation.

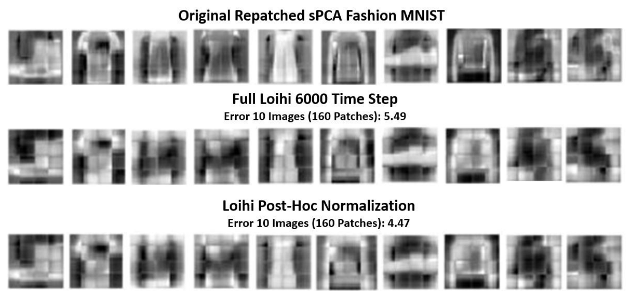

Original Repatched sPCA Fashion MNIST
Full Loihi 6000 Time Step
Error 10 Images (160 Patches): 5.49
Loihi Post-Hoc Normalization
Error 10 Images (160 Patches): 4.47

Fig. 6: Repatched image space reconstructions of the original sPCA (Top), full Loihi simulation sparse coefficients (Middle), and post-hoc normalized sparse coefficients (Bottom) for 10 images containing 16 patches each. Average reconstruction error for post-hoc solutions are better, but solutions are less sparse (see figure [8]).

IV. Results

A. Input Amplification and Optimal Sparsity

In order to successfully reconstruct our input signals, a β parameter must be introduced into the cost function to amplify the input away from unit norm and allow multiple features to be utilized. This creates a new cost functions [16],

$$E(\boldsymbol{I}, \boldsymbol{a}) = \min_{\{\boldsymbol{a}\}}[\frac{1}{2}||\beta\boldsymbol{I} - \boldsymbol{\varphi a}||_2 + \lambda||\boldsymbol{a}||_{0,p}] \tag{7}$$

where the penalty term is a 0 norm for D-wave and an effective standard p norm for Loihi.

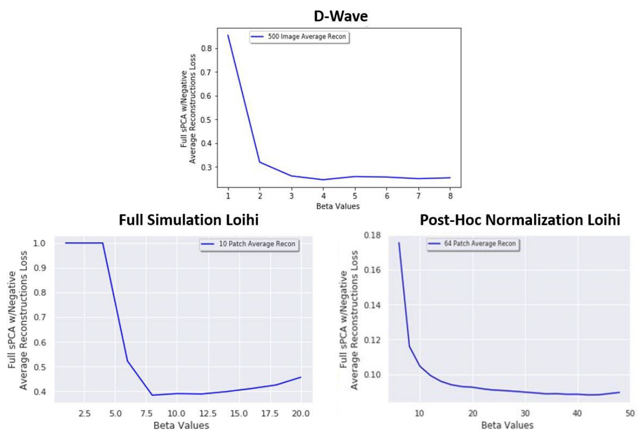


Fig. 7: Reconstruction error/loss plotted as a function of input amplification β. These values indicate how far away from unit norm the input must be for best reconstructions and are a unique characteristic of the distinct architectures. Top: D-Wave average for 500 images. Bottom Left: Full Loihi average 10 images. Bottom Right: post-hoc normalized Loihi for 64 images.

Figure [7] suggests the different substrates and techniques require varying levels of optimal input amplification β values, but all eventually reach some plateau where reconstruction error doesn't improve any further. In addition, a clear relationship between the optimal input amplification β parameter and the sparsity penalty λ value can also be observed. Intuitively, a larger amplification in the input should require a stronger lambda value in order to keep the same level of sparsity required for optimal reconstruction. In contrast to traditional sparse coding where λ and reconstruction error have a monotonically increasing relationship because of access to continuous coefficients in the sparse solution vector $\boldsymbol{a}$, there are clear points of minimum optimal sparsity penalties λ for the D-Wave and full Loihi simulations. The post-hoc normalization version of Loihi does not provide such a clear distinction of optimal λ, and resembles what is typically seen in traditional sparse coding, figure [8].

B. Dictionary Optimizations

Before final solutions were gathered, dictionary optimization was performed on the two substrates and different solution techniques. Random features were first selected for the dictionary and a stochastic gradient descent algorithm with Hebbian Learning rule was deployed using the classical Orthogonal Matching Pursuit (OMP) algorithm to lower the average reconstruction error/loss [17]. After the dictionary was trained, the same algorithm was run using the solutions from the different machines. Figure [9], shows convergence to a better binary sparse coding dictionary on the D-Wave but no change in the reconstruction error for the Loihi techniques. The lack of learning suggests the dictionary was already opti-

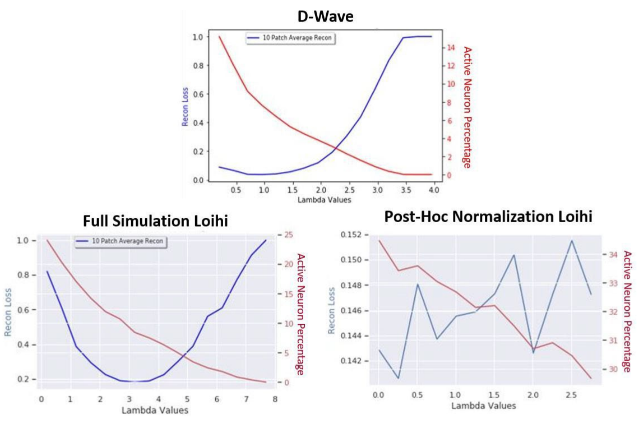


Fig. 8: Optimal sparsity penalty λ selection. Each graph has reconstruction error/loss in blue and active neuron/feature percentage in red. There is a clear optimal sparsity level dictated by λ for the D-Wave (Top) and full simulation Loihi (Bottom Left), but the post-hoc normalization technique on Loihi (Bottom Right) roughly exhibits a monotonic relationship.

mized for an L-p penalty, and thus, no learning could actually occur providing more evidence the neuromorphic processor is not solving a true binary sparse coding problem.We then initialized Loihi with the same original random features used to start the OMP training and clear learning was achieved before eventual overfitting on the full simulation time. To alleviate the overfitting, we first implemented a sleep schedule by introducing normally distributed data periodically [18] and did not normalize the learning features. As can be seen in figure [9], the learning improved until the the norms of the features began to blow up creating errors even higher than without noise and normalization. Hence a stopping criteria was utilized when the errors started to grow again. Although better learning was achieved, the final dictionary did not provide better results than when the OMP dictionary was fed onto the device. Learning was not seen in the post-hoc normalization approach with both the OMP dictionary or the random features and we believe this is a result of having the ability to use weighted amounts of all the features at all times.

V. Discussion

We successfully ran an LCA implementation for solving the same sparse coding problem on two fundamentally different devices, the D-Wave quantum annealing machine and the Loihi neuromorphic processor. In this work, we identified the relationship between the optimal λ, which sets the threshold of how sparse our solutions would be, β, which adjusts the scale of the input image and how these parameters behaved as a function of the specific substrates. The D-Wave produces true binary sparse coding. Loihi, on the other hand, produced an approximation of binary sparse coding whose fidelity was inversely dependent on the elapsed simulation time. To

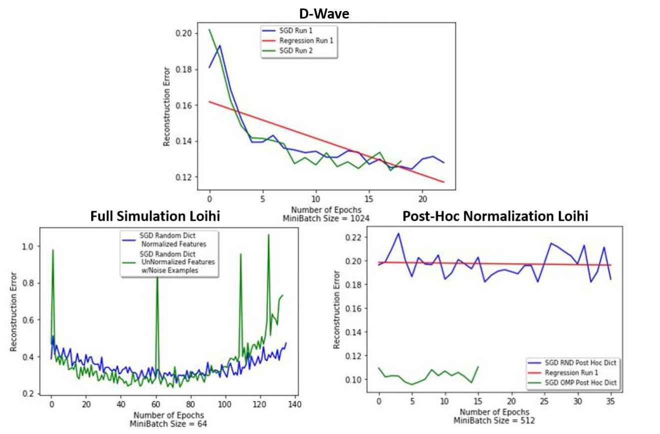


Fig. 9: Stochastic gradient descent with local Hebbian rule dictionary learning results. Top: Two runs of the algorithm successfully learn a similar binary dictionary when the OMP dictionary is fed initially into the D-Wave. Bottom Left: Random features are fed into algorithm and solutions are found using Full Simulation Loihi. Blue is result when dictionary features are normalized after every epoch, and green line represents when features are not normalized and periodic noise is introduced. Both approaches achieve initial learning before overfitting with the un-normalized noisy approach providing best solution. Bottom Right: post-hoc normalization does not show any evidence of learning with random features (blue) and OMP trained dictionary (green).

compensate for this, and to achieve a better comparison with the D-Wave, we terminated the neuromorphic solution early and utilized post-hoc normalization to adjust the overall scale of the reconstruction.

As figure [6] suggests, these solutions are often times even better in terms of final reconstruction error than their full simulation counter parts. Although post-hoc normalization does not yield completely binary solution vectors, we believe stopping the simulation early provides a better approximation of the desired binary solution because the rates of those neurons left over are much more uniform than if left to longer regularization. More importantly, this post-hoc normalized representation is closer to a true P_0 norm. By limiting our sampling to a region of simulation in which inhibiting (i.e., active neurons) have not fully quieted non-active neurons, we achieve a representation in which each active feature is either present or not without the dilution discussed above.

For D-Wave, the requirement of a lower lambda value for optimal results coupled with a lower input amplification β suggest only a few features are needed for the optimal reconstruction. This requirement appears to be the result of using all or non binary features for the final reconstruction. In contrast, the Loihi neuromorphic processor is able to use weighted amounts of each feature for reconstruction. When comparing the post-hoc normalization parameters with the traditional long simulation times for Loihi, we believe there is not an optimal lambda value for the post-hoc version because the initial number of firing neurons is much greater, and the more neurons available to use, the better the reconstruction will be. In other words, for the post-hoc solutions to be of the lowest reconstruction error, the λ value must be as small as possible so that portions of all features can be used. Alternative approaches for implementing LCA on Loihi which force solutions to be more binary would be a valuable extension of this research. One possible path is to set a single spike rate for every active neuron, while a second option would include a rate threshold which dampens activation to 0 if the rate is below said value or 1 if above. These additions would encourage a more sparse neural representation, but it is likely they would be less biologically-relevant and has thus been less explored in the neuromorphic community.

VI. Conclusion

Our exploration of biologically inspired, energy efficient neuromorphic systems and fully entangled quantum system, for the purposes of continuing the advancement of machine learning past the limitations of classical approaches, has illuminated many similarities between the two fundamentally different substrates. We successfully learned optimal dictionaries in an unsupervised manner for each device. Unlike traditional sparse coding, manipulations to the cost function of each implementation are required for optimal reconstruction. Each of the platforms and respective techniques require amplification β of input, while penalty parameter λ tuning for quantum annealing and full-simulation-length neuromorphic methods are necessary. The requirement of λ and β parameter tuning for non-continuous sparse coefficient solutions suggests there is similar structure between the full Loihi and D-Wave implementations. This characteristic ,coupled with the more binary-like and faster post-hoc approach, result in a strengthened linkage between these emerging non-von Neumann substrates and create a space where they can be more directly compared. Future work will include classical high performance computing, single spike rate Loihi, and binary rate thresholded Loihi sparse solutions. These approaches, along with their corresponding classification scores, should be a more direct comparison with the D-Wave quantum annealer.

References

[1] K. Boahen, *A neuromorph's prospectus*,Computing in Science Engineering, vol. 19. pp.14-28, Mar. 2017, https://aip.scitation.org/doi/abs/10.1109/MCSE.2017.33?journalCode=csx

[2] John Preskill *Quantum computing and the entanglement frontier*, arXiv, 2012

[3] DWAVE, 2016. [Online]. Available: http://www.dwavesys.com/

[4] M.M. Khan et al,*SpiNNaker: Mapping Neural Networks onto a Massively-Parallel Chip Multiprocessor*, 2008 International Joint Conference on Neural Networks (IJCNN 2008)

[5] A. Cassidy et al.,*TrueNorth: A High-Performance, Low-Power Neurosynaptic Processor for Multi-Sensory Perception, Action, and Cognition,* Computer Science. 2016

[6] M. Davies et al., Loihi: A Neuromorphic Manycore Processor with On-Chip Learning, in IEEE Micro, vol. 38, no. 1, pp. 82-99, January/February 2018.

[7] V. Kornijcuk et al., Leaky Integrate-and-Fire Neuron Circuit Based on Floating-Gate Integrator, Frontiers in Neuroscience, 23 May 2016

[8] H. Xiao,K. Rasul,and R.Vollgraf, *Fashion-MNIST:a Novel Image Dataset for Benchmarking Machine Learning Algorithms*, arXiv:1708.07747, [Online]. Available: https://github.com/zalandoresearch/Fashion-MNIST

[9] N. Henze and M. D. Penrose, *On the multivariate runs test*, Ann. Statist., vol. 27, no. 1, pp. 290-298, 1999

[10] Waagen et al, *Fashion MNIST Charts for LANL Discussions,* Air Force Research Laboratory (AFRL) 2019

[11] Hopfield, J. J., *Neural networks and physical systems with emergent collective computational properties* Proc. Nat. Acad. Sci. (USA) 79, 2554-2558. (1982)

[12] B.K. Natarajan, Sparse Approximate Solutions to Linear Systems, SIAM Journal on Computing, April 1995

[13] N.T.T. Nguyen and G. T. Kenyon, Solving sparse representation for object classification using quantum d-wave 2x machine, Proceedings of The First International Workshop on Post Moore's Era Supercomputing, J. S. Vetter and S. Matsuoka, Eds. Future Technologies Group Technical Report FTGTR-2016-11, November 2016, pp. 43–44.

[14] Nguyen, Nga T.T. and Kenyon, Garrett T. *Image Classification Using Quantum Inference on the,D-Wave 2X*,2018 IEEE International Conference on Rebooting Computing (ICRC), IEEE, 2018,

[15] C.J. Rozell,D.H. Johnson, R.G. Baraniuk, and B.A. Olshausen, *Sparse coding viathresholding and local competition in neural circuits*, Neural Computation, vol. 20, pp.2526–2563, Oct. 2008. [Online]. Available: http://ieeexplore.ieee.org/document/6796039/

[16] K. Henke, G. T. Kenyon and B. Migliori, *Machine Learning in a Post Moore's Law World: Quantum vs. Neuromorphic Substrates* 2020 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), Albuquerque, NM, USA, 2020, pp. 74-77, doi: 10.1109/SSIAI49293.2020.9094596.

[17] Hebb DO,*The organization of behavior: a neuropsychological theory.* Wiley, New York, 1949

[18] Watkins, Yijing and Kim, Edward and Sornborger, Andrew and Kenyon, Garrett *Using Sinusoidally-Modulated Noise as a Surrogate for Slow-Wave Sleep to Accomplish Stable Unsupervised Dictionary Learning in a Spike-Based Sparse Coding Model*,2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2020

[19] Potok, Thomas E. and Schuman, Catherine and Young, Steven and Patton, Robert and Spedalieri, Federico and Liu, Jeremy and Yao, Ke-Thia and Rose, Garrett and Chakma, Gangotree,*A Study of Complex Deep Learning Networks on High-Performance, Neuromorphic, and Quantum Computers*, Association for Computing Machinery, 2018

Machine Learning in a Post Moore's Law World: Quantum vs. Neuromorphic Substrates

Kyle Henke
Computer, Computational, and Statistical Sciences (CCS-3)
Los Alamos National Laboratory
Los Alamos, United States
Email: khenke@lanl.gov

Garrett T. Kenyon
Computer, Computational, and Statistical Sciences (CCS-3)
Los Alamos National Laboratory
Los Alamos, United States
Email: gkenyon@lanl.gov

Ben Migliori
Computer, Computational, and Statistical Sciences (CCS-7)
Los Alamos National Laboratory
Los Alamos, United States
Email: ben.migliori@lanl.gov

***Abstract*—Although machine learning currently relies on conventional computer architectures, the looming end of Moore's Law necessitates exploration of novel computational platforms. Neuromorphic and quantum systems are a natural path to pursue; biological neurons are incredibly efficient, and quantum mechanics provides theoretical foundations for fast solutions to optimization problems. Here, we make the first comparison of emerging hardware (D-Wave quantum annealer and Intel Loihi spiking processor) on an identically-posed machine learning problem. We implement the bioinspired Locally Competitive Algorithm (LCA) for solving sparse coding on the different substrates. To make the comparison valid, our dataset of choice (Fashion MNIST) is dimensionally-reduced via sparse principal component analysis, under the constraint that both classification performance and a graph-based clustering metric remain unchanged. This enables the problem to be mapped identically to both devices. An analysis of several metrics, including power consumption, reconstruction, and classification accuracy are presented. When given the same specifically-constructed challenge, both substrates perform similarly. Our results suggest while neuromorphic and quantum systems are still in their infancy, they present a possible route to address certain types of classically challenging problems, such as sparse coding, in a way that leverages the unique aspects of the substrates.**

***Index Terms*—quantum annealing; neuromorphic computing; sparse coding**

I. Introduction

Quantum annealing systems and neuromorphic spiking processors are fundamentally different computational substrates, each designed to perform a specific non-Von Neumann task. Typically, users attempt to maximize the usage of each substrate for experimentation; because of the dramatic difference in computational capacity, this has prevented a comparison between the two techniques on level footing. Here, we are able to cast a Locally Competitive Algorithm (LCA) [1] onto both pieces of hardware and solve the same NP-Hard sparse coding problem for the same benchmark dataset. The lowest common computational constraint is the physical restriction of the D-Wave quantum annealer, which has significantly fewer degrees of freedom than typically used in machine learning challenges. We reduce a common dataset (Fashion-MNIST [2]) using the Henze-Penrose statistic for class separability and support vector machines (SVMs) as a metric for dimensionality reduction with constant "problem difficulty". While such a challenge does not maximize computational throughput on either system, it allows direct comparison of results between the two.

A. Neuromorphic Spiking Processors

Intel's Loihi neuromorphic computing device implements spiking neural networks inspired by biology. Like SpiNNaker [3] and TrueNorth [4], Loihi represents information as single-bit impulses, transmitted at specific times and to specific targets. Thus time is explicitly incorporated in the representation, as is massive parallelism. Loihi has programmable synaptic learning rules (i.e. spike timing dependent plasticity) and is thus not an inference-only device.

Users specify input as a sequence of delta functions or bias currents to a set of target neurons. Each neural unit on the device implements an asynchronous discrete-time implementation of Leaky Integrate and Fire (LIF) neuron [5]; this system evolves in time and propagates information through the defined network graph. The timing and patterns of neural activity define the computational tasks.

B. Quantum Annealing

In a classical annealing process, the system is prepared in a randomly chosen initial state at a finite temperature. Thermal fluctuations cause classical annealing systems to jump-over local energy barriers and into new energy states, with transitions between states driven stochastically according to a Boltzmann distribution. Jumps to higher energy states are possible but exponentially less likely than transitions to lower energy states. As the temperature is lowered, classical annealing systems tend to settle into progressively lower energy states. The annealing process is repeated multiple times using different randomly chosen initial conditions, with the lowest final energy state achieved across all annealing runs representing the answer to the computation.

The implementation of quantum annealing differs from classical annealing in several important respects. In quantum annealing, the system is not prepared in a single randomly chosen state as with a classical annealing process. Rather, a quantum annealing machine is prepared in an initial state that consists of a quantum superposition of all possible states.

Using the D-Wave quantum annealing machine as a concrete example, a transverse magnetic field is applied to each qubit in the absence of any coupling between qubits. A D-Wave computer consisting of N qubits is thus prepared in an initial state that represents the superposition of all 2^N possible observable states. In part, the computing power of quantum annealing machines derives from the ability to initialize the system in a superposition of all possible states, which in turn enables a quantum annealing machine to more effectively sample the entire energy landscape.

Whereas classical annealing involves slowly lowering the temperature of the system, quantum annealing is implemented on the D-Wave by gradually turning on the user specified Hamiltonian while the transverse magnetic field is gradually turned off. Rather than jumping over local energy barriers, in quantum annealing a transition to new energy states is accomplished via quantum tunneling. In theory, quantum tunneling allows quantum annealers to avoid getting trapped in local minima.

Because of the connectivity limitations of the physical D-Wave device, physical qubits must be "chained together" to enable full connectivity as required by most machine-learning algorithms; this reduces that number of "logical" qubits to at least an order of magnitude below the number of "physical qubits".

C. Sparse PCA Fashion MNIST

We utilize the Fashion-MNIST dataset [2], a 28 by 28 greyscale labelled image dataset with ten classes. Fashion-MNIST is significantly more challenging than the classical MNIST challenge, but is still tractable for many modern machine learning algorithms. However, it is still far too large (784 dimensions) to fit on the D-Wave annealer. The Henze-Penrose (HP) statistic for estimating class separability was used to select the appropriate dimensionality for the fashion MNIST data set. First, the minimum spanning tree between classes is calculated and then S_{FR} is found as the number of transitions between the classes. For a two class system we can express this statistic in the equation below:

$$H_{xy} = 1 - S_{FR}\frac{n_x + n_y}{2n_{xy}} \quad (1)$$

where n_x and n_y are the number of nodes in the tree for class x and y respectively. The data set was reduced via sparse PCA and the HP statistic was calculated for each reduction. When ΔHP begins to increase rapidly (the "HP Rollover Point") it indicates that dataset compression is causing large changes in clusterability. For our data, the critical point was found to be 32 dimensions. To confirm that a 32-dimensional Fashion MNIST contained a similar clustering challenge to the uncompressed representation, we trained SVMs to classify both original and compressed datasets. The RMS change in the confusion matrix between the 784-dimension and 32-dimension representation was .007 (where 0 is no accuracy, and 1.0 is perfect accuracy). The SVM and HP metrics together demonstrate that neither the problem difficulty nor the classification accuracy significantly changed under compression [6].

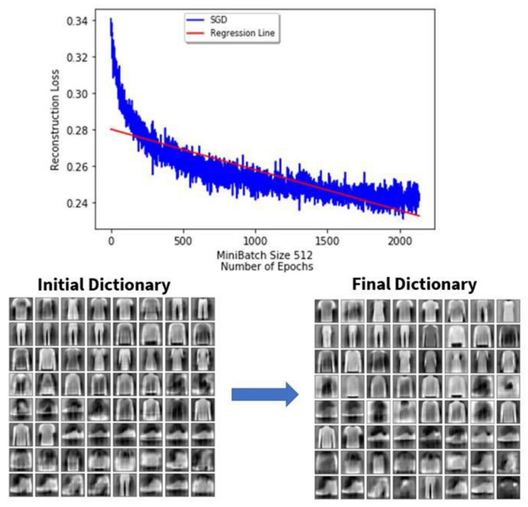


Fig. 1: Orthogonal Matching Pursuit (OMP) Stochastic Gradient Descent Training Results

D. The Sparse Coding Problem

Given an overcomplete, non-orthonormal basis $\{\varphi_i\}$, inferring a sparse representation involves finding the minimal set of binary activation coefficients $\boldsymbol{a}$ that accurately reconstruct a given input signal $\boldsymbol{I}$, corresponding to a minimum of the following energy function:

$$E(\bar{\boldsymbol{I}}, \bar{\boldsymbol{a}}) = \min_{\{\boldsymbol{a}\}}\left[\frac{1}{2}||\bar{\boldsymbol{I}} - \boldsymbol{\varphi a}||^2 + \lambda||\bar{\boldsymbol{a}}||_0\right] \quad (2)$$

where λ is a trade-off parameter that determines the balance between reconstruction error and the number of non-zero activation coefficients. A larger λ will result in a more sparse solution to Eq. (2). This energy function is non-convex and contains multiple local minima, so finding a sparse representation falls into an NP-hard complexity class of decision problems [7].

E. Dictionary Optimization

Before we gathered results from the various substrates, an optimal dictionary φ was trained using stochastic gradient descent (SGD). For this task, we invoking a local weighted Hebbian learning rule to move the dictionary in the direction of the most active features after the classical Orthogonal Matching Pursuit (OMP) algorithm was used to find the sparse representations of each image in the mini-batch (Figure 1).

II. Sparse Coding Implementations on Quantum and Neuromorphic Hardware

A. Sparse Coding as Lateral Inhibition and Competition

Previous work [1] has shown that *sparse coding* optimization problems can be solved using the dynamics of neural networks incorporating lateral inhibition, a biologically plausible implementation of a sparse solver referred to as a locally competitive algorithm (LCA). A sparse solution is found by allowing the resulting dynamical system to evolve to a minimum energy configuration.

In the neuromorphic implementation of sparse coding, neurons send binary signals as spikes in response to current flowing from signals sent by neighboring neurons or inputs which can either excite or inhibit one another. The neuromorphic implementation injects current weighted according to a non-orthonormal basis ($\boldsymbol{\varphi}^T \boldsymbol{x}$) into a network of neurons, and reads outputs as spike rates $\boldsymbol{a}$. The sparse coding loss function (Equation 2) may then be mapped to a dynamical systems model through substitution [9]:

$$\dot{\boldsymbol{u}} = \frac{1}{\tau}(\boldsymbol{\varphi}^T \boldsymbol{x} - \boldsymbol{u} - (\boldsymbol{\varphi}^T \boldsymbol{\varphi} - \boldsymbol{I}) \cdot \boldsymbol{a}) \tag{3}$$

The $-\boldsymbol{u}$ term acts as the decay piece of the system, slowly decreasing the potential of each neuron over time. If an active neuron is not continuously excited, it will rapidly fall below firing threshold and deactivate. The $\boldsymbol{\varphi}^T \boldsymbol{x}$ term charges up each of the neurons, exciting neurons whose features best match the input. $(\boldsymbol{\varphi}^T \boldsymbol{\varphi} - \boldsymbol{I}) \cdot \boldsymbol{a}$ is the inhibitory signal, forcing neurons which explain a similar component of the data to compete by inhibiting one another. This competition continues until the equation converges to a stable sparse representation [1].

Normalization and sparsity: The spiking output of Loihi represents the output of the LCA process as a vector *veca*, which is projected onto the normalized non-orthonormal basis φ to convert the sparse code into a dense representation. As Loihi uses spike rates, the number of spikes per basis neuron is converted into a rate vector that is used in this projection. Thus for Loihi, time acts as a regularizer; should a basis vector be overrepresented, the neuron responsible will be inhibited and stop firing. As the simulation continues, the effect of that vector is then reduced as the number of spikes per unit time decreases. However, we are interested in a binary £$_0$ representation, as expected for true sparse coding. To approximate this, we interrupt the Loihi simulation after the first set of spikes occur. In this representation, each basis vector is represented in binary fashion, as would be required for £$_0$. However, as the basis vectors sum, this results in the reconstruction norm blowing up.

To counteract this effect, we introduce a $\beta \in \mathbb{Z}$ factor as a coefficient of the input I^- in Equation 3. This term allows approximately *beta* overlapping basis vectors to be summed in a binary fashion without causing a large penalty in the reconstruction loss (Equation 2). We tune β as a hyperparameter and choose a value based on convergence of the reconstruction loss. As will be shown in the next section, a similar process is required for sparse coding with quantum annealing.

B. *Quantum Annealing for Sparse Attractor Networks*

Transformation relations: In a quantum annealing system, each neuron is mapped to a binary qubit, with a state described in general by a superposition of 1 and 0. It is this quantum superposition that allows the D-Wave to explore the entire energy landscape at once.

The D-Wave 2000Q[10] finds optimal solutions to a (discrete) Ising system consisting of N_q binary variables described by the following classical Hamiltonian:

$$H(\boldsymbol{h}, \boldsymbol{Q}, \boldsymbol{a}) = \sum_{i}^{N_q} h_i a_i + \sum_{i<j}^{N_q} Q_{ij} a_i a_j \tag{4}$$

with binary activation coefficients $a_i = \{0, 1\} \; \forall i \in (1, 2, 3, \ldots, N_q)$. This objective function defines a Quadratic Unconstrained Binary Optimization (QUBO) problem. We cast our sparse coding problem, Eq. (2), into QUBO form, Eq. (4), by the transformations [11]:

$$h_i = (-\boldsymbol{\varphi}^T \boldsymbol{I} + (\lambda + \frac{1}{2}))_i, \quad Q_{ij} = (\boldsymbol{\varphi}^T \boldsymbol{\varphi})_{ij} \tag{5}$$

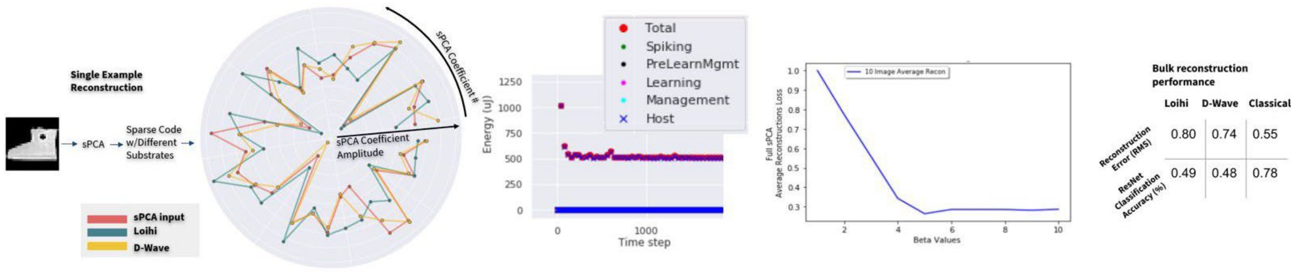
In Eq. (5), the bias term $\boldsymbol{h}$ in the Ising model is proportional to the weighted input $\boldsymbol{\varphi}^T \boldsymbol{I}$ while the coupling term $\boldsymbol{Q}$ corresponds to lateral competition (see also [1]) between qubits given by the interaction matrix $\boldsymbol{\varphi}^T \boldsymbol{\varphi}$. Note that the sparsity trade-off parameter λ appears as a uniform applied magnetic field that encourages all qubits to be in the $a_i = 0$ state[10].

D-Wave 2000Q Hardware and Embedding: Embedding an arbitrary QUBO problem onto the 2000Q chimera typically allows no more than $N_q \sim 64$ nodes (logical qubits) but these nodes may be fully connected. Thus, embedding effectively trades qubits for connectivity, and is in and of itself an NP hard optimization problem. In addition, the more overcomplete our dictionary φ, the better the overall reconstruction are, but comes at the cost of exponentially growing embedding times (see Fig 2). Since we have 32 sparse PCA coefficients ($\sim N_q/2$), each Hamiltonian satisfies the overcomplete requirement, but when all 64 logical qubits are used, the full time to solution can take up to 4 minutes per image.

Input Amplification for D-Wave Reconstruction: D-Wave sums features of the given dictionary to recreate input data. Since our coefficients in the sparse representation are binary, the norm of the resulting output explodes as it does in the neuromorphic case, forcing an all zero solution. Hence we introduce a parameter $\beta \in \mathbb{Z}$ into the cost function (Equation 2, as a coefficient of I^-) and tune it until the reconstruction error is minimized and plateaus. The final reconstruction is then normalized back to the same length as the original input.

III. Results

Sparse reconstructions for the same subset of Fashion MNIST sPCA images were found with D-Wave, Loihi, and a commercial optimization algorithm (GUROBI). The reconstructions were then classified using ResNets [11]. These representations, and the metrics for each, are shown in Figure 3. The reconstruction is represented as a polar plot showing the projection of the sparse code into the basis set. Although the power consumption of the D-Wave is not readily available, we show that power consumption of Loihi remains constant at $\sim$1.2W during computation. An example of a β tuning curve, as described in the implementation section, is also shown.



	Loihi	D-Wave	Classical
Reconstruction Error (RMS)	0.80	0.74	0.55
ResNet Classification Accuracy (%)	0.49	0.48	0.78

Fig. 2: (left) Single image process to final reconstruction in sPCA space example. This example is characteristic for randomly selected reconstructions. Reconstructed sPCA coefficient amplitude is shown on the radius, and sPCA coefficient number on the angular axis. (middle left) Power consumption remains constant during simulation on the Loihi processor. (middle right) Example of beta tuning to allow for binary reconstruction without normalization. Note that there is a clear plateau. (right) Reconstruction error for all images and ResNet Classification performance.

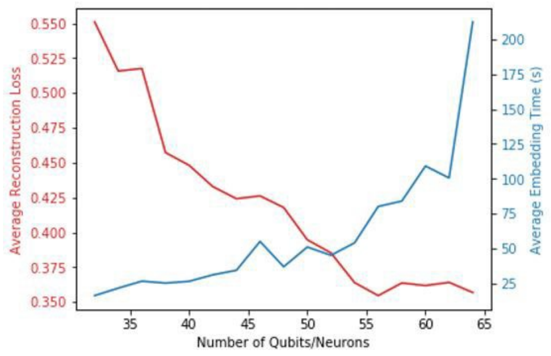


Fig. 3: Example of how reconstruction loss decreases as problem gets more over-complete on D-Wave at the cost of exponentially growing embedding times for 10 randomly sampled images.

The primary result, shown in the table, is that the bulk reconstruction performance is similar across all examples between both computing substrates. Further, the classification performance of a SVMs and ResNets operating on the sparse codes are also comparable. These results show that the resulting sparse codes are of equal quality, and allow metrics to be contrasted and compared for the two systems.

IV. Conclusion and Future Work

We have demonstrated hard problems can be equally implemented on neuromorphic and quantum systems using calibrated dimensionality reduction. This result represents one of the only apples-to-apples comparisons of such systems, and shows that bio-inspired methods and quantum methods may possess similar underlying structure. By utilizing very small datasets as proxies for larger, harder datasets, this work also provides guidance for the application of larger quantum systems in the future. Although the research presented here did not fully utilize the Loihi chip, there is reason to believe that scaling within the D-Wave environment may be significantly more effective than scaling within Loihi. As larger and more connected D-Wave chips are produced, the LCA problem demonstrated here will better match the hardware and may be accomplished with fewer reads of the annealing device and with greater likelihood of observing the ground state. Loihi presents significant size, weight, and power benefits, requiring only a few watts per processor. This effectively gives Loihi the ability to be implemented in small devices at the edge, as opposed to the infrastructure-like nature of the D-Wave.

The best classical algorithm tested (GUROBI) outperformed these emerging substrates in terms of Root Mean Squared Error reconstruction loss and final classification, but is difficult to implement and requires its own tuning process. For future work, lower dimensional patch-based approaches would allow for more overcomplete problems and likely lead to better performances from the emerging substrates while stressing classical methods further. Until such problems are specifically valuable or the novel substrates scale up, it remains important to consider classical methods in difficult optimization problems.

References

[1] C.J. Rozell,D.H. Johnson, R.G. Baraniuk, and B.A. Olshausen, *Sparse coding viathresholding and local competition in neural circuits*, Neural Computation, vol. 20, pp.2526–2563, Oct. 2008. [Online]. Available: http://ieeexplore.ieee.org/document/6796039/
[2] H. Xiao,K. Rasul,and R.Vollgraf, *Fashion-MNIST:a Novel Image Dataset for Benchmarking Machine Learning Algorithms*, arXiv:1708.07747, [Online]. Available: https://github.com/zalandoresearch/fashion-mnist
[3] M.M. Khan et al,*SpiNNaker: Mapping Neural Networks onto a Massively-Parallel Chip Multiprocessor*, 2008 International Joint Conference on Neural Networks (IJCNN 2008)
[4] A. Cassidy et al.,*TrueNorth: A High-Performance, Low-Power Neurosynaptic Processor for Multi-Sensory Perception, Action, and Cognition,* Computer Science. 2016
[5] V. Kornijcuk et al., Leaky Integrate-and-Fire Neuron Circuit Based on Floating-Gate Integrator, Frontiers in Neuroscience, 23 May 2016
[6] Waagen et al, Fashion MNIST Charts for LANL Discussions, Air Force Research Laboratory (AFRL) 2019
[7] B.K. Natarajan, Sparse Approximate Solutions to Linear Systems, SIAM Journal on Computing, April 1995
[8] M. Davies et al., Loihi: A Neuromorphic Manycore Processor with On-Chip Learning, in IEEE Micro, vol. 38, no. 1, pp. 82-99, January/February 2018.
[9] DWAVE, 2016. [Online]. Available: http://www.dwavesys.com/
[10] N.T.T. Nguyen and G. T. Kenyon, Solving sparse representation for object classification using quantum d-wave 2x machine, in Proceedings of The First International Workshop on Post Moore's Era Supercomputing, J. S. Vetter and S. Matsuoka, Eds. Future Technologies Group Technical Report FTGTR-2016-11, November 2016, pp. 43–44.
[11] K.He, X. Zhang, S. Ren and J. Sun, Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778.

Bibliography

[1] Tameem Albash and Jeffrey Marshall. “Comparing Relaxation Mechanisms in Quantum and Classical Transverse-Field Annealing”. In: *Phys Rev Applied* 15.1 (2021). doi: 10.1103/physrevapplied.15.014029.

[2] Shunta Arai, Masayuki Ohzeki, and Kazuyuki Tanaka. “Mean field analysis of reverse annealing for code-division multiple-access multiuser detection”. In: *Phys Rev Research* 3 (3 2021), p. 033006. doi: 10.1103/PhysRevResearch.3.033006.

[3] Yuki Bando et al. “Breakdown of the Weak-Coupling Limit in Quantum Annealing”. In: *Phys Rev Applied* 17 (5 2022), p. 054033. doi: 10.1103/PhysRevApplied.17.054033.

[4] Ronny Bergmann and Roland Herzog. “Intrinsic Formulation of KKT Conditions and Constraint Qualifications on Smooth Manifolds”. In: *SIAM Journal on Optimization* 29.4 (Jan. 2019), pp. 2423–2444. doi: 10.1137/18m1181602. url: https://doi.org/10.11372F18m1181602.

[5] Katrina Biele and Greg Oman. “A short proof of the Bolzano-Weierstrass Theorem”. In: *The College Mathematics Journal* ? (Jan. 2018), ?

[6] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[7] A Cassidy et al. “TrueNorth: A high-performance, low-power neurosynaptic processor for multi-sensory perception, action, and cognition”. In: *Proceedings of the Government Microcircuits Applications & Critical Technology Conference, Orlando, FL, USA*. 2016, pp. 14–17.

[8] Selmaan N Chettih and Christopher D Harvey. “Single-neuron perturbations reveal feature-specific competition in V1”. In: *Nature* 567.7748 (2019), pp. 334–340.

[9] Frank H Clarke. “Generalized Gradients of Lipschitz Functionals”. In: *Advances in Mathematics* 40.1 (1981), pp. 52–67.

[10] D-Wave. *dwave-simulated-annealing*. https://github.com/dwavesystems/dwave-neal. 2022.

[11] D-Wave Systems. *D-Wave Ocean Software Documentation: Annealing Implementation and Controls*. https://docs.dwavesys.com/docs/latest/c_qpu_annealing.html. 2023.

[12] D-Wave Systems. *D-Wave Ocean Software Documentation: Solver Parameters*. https://docs.dwavesys.com/docs/latest/c_solver_parameters.html. 2023.

[13] D-Wave Systems. *D-Wave Ocean Software Documentation: Uniform Torque Compensation.* https://docs.ocean.dwavesys.com/projects/system/en/stable/reference/generated/dwave.embedding.chain_strength.uniform_torque_compensation.html. 2018.

[14] Mike Davies et al. "Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook". In: *Proceedings of the IEEE* 109.5 (2021), pp. 911–934. doi: 10.1109/JPROC.2021.3067593.

[15] Mike Davies et al. "Loihi: A neuromorphic manycore processor with on-chip learning". In: *Ieee Micro* 38.1 (2018), pp. 82–99.

[16] Kaitlin L Fair et al. "Sparse coding using the locally competitive algorithm on the TrueNorth neurosynaptic system". In: *Frontiers in Neuroscience* (2019), p. 754.

[17] A. F. Filippov. *Differential Equations with Discontinuous Righthand Sides.* Dordrecht: Kluwer Aacademic Publishers, 1988.

[18] Gabriel A. Fonseca Guerra and Steve B. Furber. "Using Stochastic Spiking Neural Networks on SpiNNaker to Solve Constraint Satisfaction Problems". In: *Frontiers in Neuroscience* 11 (2017), pp. 1–13. issn: 1662-453X. doi: 10.3389/fnins.2017.00714. url: https://www.frontiersin.org/articles/10.3389/fnins.2017.00714.

[19] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

[20] Kyle Henke, Garrett Kenyon, and Ben Migliori. "Machine Learning in a Post Moore's Law World: Quantum vs. Neuromorphic Substrates". In: Mar. 2020, pp. 74–77. doi: 10.1109/SSIAI49293.2020.9094596.

[21] Kyle Henke, Garrett T. Kenyon, and Ben Migliori. "Fast Post-Hoc Normalization for Brain Inspired Sparse Coding on a Neuromorphic Device". In: *IEEE Transactions on Parallel and Distributed Systems* 33.2 (2022), pp. 302–309. doi: 10.1109/TPDS.2021.3068777.

[22] Kyle Henke, Garrett T. Kenyon, and Ben Migliori. "Machine Learning in a Post Moore's Law World: Quantum vs. Neuromorphic Substrates". In: *2020 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI).* Piscataway, NJ: Institute of Electrical and Electronics Engineers (IEEE), 2020, pp. 74–77. doi: 10.1109/SSIAI49293.2020.9094596.

[23] Kyle Henke, Ben Migliori, and Garrett T. Kenyon. "Alien vs. Predator: Brain Inspired Sparse Coding Optimization on Neuromorphic and Quantum Devices". In: *2020 International Conference on Rebooting Computing (ICRC).* 2020, pp. 26–33. doi: 10.1109/ICRC2020.2020.00015.

[24] Kyle Henke et al. "Apples-to-Spikes: The First Detailed Comparison of LASSO Solutions Generated by a Spiking Neuromorphic Processor". In: *Proceedings of the International Conference on Neuromorphic Systems 2022.* ICONS '22. Knoxville, TN, USA: Association for Computing Machinery, 2022. isbn: 9781450397896. doi: 10.1145/3546790.3546811. url: https://doi.org/10.1145/3546790.3546811.

[25] Kyle Henke et al. *Sampling binary sparse coding QUBO models using a spiking neuromorphic processor*. 2023. arXiv: 2306.01940 [cs.NE].

[26] Geoffrey E. Hinton. *Boltzmann Machines*. https://www.cs.toronto.edu/~hinton/csc321/readings/boltz321.pdf. 2007.

[27] 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.

[28] Zeno Jonke, Stefan Habenschuss, and Wolfgang Maass. "Solving Constraint Satisfaction Problems with Networks of Spiking Neurons". In: *Front Neurosci* 10.118 (2016), pp. 1–16. doi: 10.3389/fnins.2016.00118.

[29] Paul Kairys et al. "Simulating the Shastry-Sutherland Ising Model Using Quantum Annealing". In: *PRX Quantum* 1 (2 2020), p. 020320. doi: 10.1103/PRXQuantum.1.020320.

[30] Andrew D. King et al. "Observation of topological phenomena in a programmable lattice of 1,800 qubits". In: *Nature* 560.7719 (2018), pp. 456–460. doi: 10.1038/s41586-018-0410-x.

[31] Andrew D. King et al. "Quantum Annealing Simulation of Out-of-Equilibrium Magnetization in a Spin-Chain Compound". In: *PRX Quantum* 2.3 (2021). doi: 10.1103/prxquantum.2.030317.

[32] Andrew D. King et al. "Scaling advantage over path-integral Monte Carlo in quantum simulation of geometrically frustrated magnets". In: *Nature Communications* 12.1 (2021). doi: 10.1038/s41467-021-20901-5.

[33] Ami S. Koshikawa et al. "Benchmark Test of Black-box Optimization Using D-Wave Quantum Annealer". In: *Journal of the Physical Society of Japan* 90.6 (2021), p. 064001. doi: 10.7566/JPSJ.90.064001.

[34] M. de Le ón and P.R. Rodrigues. *Methods of Differential Geometry in Analytical Mechanics*. ISSN. Elsevier Science, 2011. isbn: 9780080872698. url: https://books.google.com/books?id=5pCfP8CiSzAC.

[35] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[36] Luciano Lopez and Luca Dieci. "A Survey of Numerical Methods for IVPs of ODEs with Discontinuous Right-Hand Side". In: *JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS* 236 (Oct. 2012), pp. 967–3991. doi: 10.1016/j.cam.2012.02.011.

[37] Alejandro Lopez-Bezanilla et al. *Kagome qubit ice*. arXiv:2301.01853. 2023.

[38] Sheng Y Lundquist. "Exploring the Potential of Sparse Coding for Machine Learning". PhD thesis. Portland State University, 2020.

[39] F. J. Murray and K. S. Miller. *Existence Theorems for Ordinary Differential Equations*. Dover Publications, 2007.

[40] R. K. Nagle, E. B. Saff, and A. D. Snider. *Fundamentals of Differential Equations and Boundary Value Problems*. Addison-Wesley, 2012.

[41] 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.

[42] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019), pp. 8026–8037.

[43] Elijah Pelofske, Georg Hahn, and Hristo Djidjev. *Initial state encoding via reverse quantum annealing and h-gain features*. 2023. arXiv: 2303.13748 [quant-ph].

[44] Elijah Pelofske, Georg Hahn, and Hristo N. Djidjev. "Advanced anneal paths for improved quantum annealing". In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Oct. 2020. doi: 10.1109/qce49297.2020.00040. url: https://doi.org/10.1109%2Fqce49297.2020.00040.

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

[46] Joseph La Salle and Solomon Lefschetz. *Stability by Liapunov's direct method: With applications*. Acad. Press, 1973.

[47] M. Sandberg. "Convergence of forward Euler method for non-convex differential inclusions". In: *SIAM J. Numer. Anal.* 47.1 (2008), pp. 308–320.

[48] Samuel Shapero et al. "Optimal sparse approximation with integrate and fire neurons". In: *International journal of neural systems* 24.05 (2014), p. 1440001.

[49] Ping Tak Peter Tang. "Convergence of LCA Flows to (C)LASSO Solutions". In: *arXiv preprint arXiv:1603.01644* (2016).

[50] Ping Tak Peter Tang, Tsung-Han Lin, and Mike Davies. "Sparse coding by spiking neural networks: Convergence theory and computational results". In: *arXiv preprint arXiv:1705.05475* (2017).

[51] Michael Teti et al. "LCANets: Lateral Competition Improves Robustness Against Corruption and Attack". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 21232–21252. url: https://proceedings.mlr.press/v162/teti22a.html.

[52] 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. issn: 00359246. url: http://www.jstor.org/stable/2346178 (visited on 04/12/2023).

[53] Davide Venturelli et al. "Quantum Optimization of Fully Connected Spin Glasses". In: *Phys Rev X* 5 (3 2015), p. 031040. doi: 10.1103/PhysRevX.5.031040.

[54] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. url: http://arxiv.org/abs/1708.07747.

[55] Yu Yamashiro et al. “Dynamics of reverse annealing for the fully connected p-spin model”. In: *Phys Rev A* 100 (5 2019), p. 052321. doi: 10.1103/PhysRevA.100.052321.

[56] Mengchen Zhu and Christopher J Rozell. “Visual nonclassical receptive field effects emerge from sparse coding in a dynamical system”. In: *PLoS computational biology* 9.8 (2013), e1003191.

[57] Joel Zylberberg, Jason Timothy Murphy, and Michael Robert DeWeese. “A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields”. In: *PLoS computational biology* 7.10 (2011), e1002250.