On the Algorithmic Power of Spiking Neural Networks

Chi-Ning Chou[1]
School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA
chiningchou@g.harvard.edu

Kai-Min Chung
Institute of Information Science, Academia Sinica, Taipei, Taiwan
kmchung@iis.sinica.edu.tw

Chi-Jen Lu
Institute of Information Science, Academia Sinica, Taipei, Taiwan
cjlu@iis.sinica.edu.tw

Abstract

Spiking Neural Networks (SNN) are mathematical models in neuroscience to describe the dynamics among a set of neurons that interact with each other by firing instantaneous signals, *a.k.a.*, *spikes*. Interestingly, a recent advance in neuroscience [Barrett-Denève-Machens, NIPS 2013] showed that the neurons' *firing rate*, i.e., the average number of spikes fired per unit of time, can be characterized by the optimal solution of a quadratic program defined by the parameters of the dynamics. This indicated that SNN potentially has the computational power to solve non-trivial quadratic programs. However, the results were justified empirically without rigorous analysis.

We put this into the context of *natural algorithms* and aim to investigate the algorithmic power of SNN. Especially, we emphasize on giving rigorous asymptotic analysis on the performance of SNN in solving optimization problems. To enforce a theoretical study, we first identify a simplified SNN model that is tractable for analysis. Next, we confirm the empirical observation in the work of Barrett et al. by giving an upper bound on the convergence rate of SNN in solving the quadratic program. Further, we observe that in the case where there are infinitely many optimal solutions, SNN tends to converge to the one with smaller ℓ_1 norm. We give an affirmative answer to our finding by showing that SNN can solve the ℓ_1 minimization problem under some regular conditions.

Our main technical insight is a *dual view* of the SNN dynamics, under which SNN can be viewed as a new natural primal-dual algorithm for the ℓ_1 minimization problem. We believe that the dual view is of independent interest and may potentially find interesting interpretation in neuroscience.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Spiking Neural Networks, Natural Algorithms, ℓ_1 Minimization

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.26

Related Version A full version of the paper is available at `https://arxiv.org/abs/1803.10375`.

Acknowledgements The authors would like to thank Tsung-Han Lin, Zhenming Liu, Luca Trevisan, Richard Peng, Yin-Hsun Huang, and Tao Xiao for useful discussions related to this paper. We are also thankful to the anonymous reviewer from ITCS 2019 for useful comments.

[1] Supported by NSF awards CCF 1565264 and CNS 1618026.

10th Innovations in Theoretical Computer Science (ITCS 2019).
Editor: Avrim Blum; Article No. 26; pp. 26:1–26:20

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The theory of *natural algorithms* is a framework that bridges the algorithmic thinking in computer science and the mathematical models in biology. Under this framework, biological systems are viewed as *algorithms* to *efficiently* solve specific *computational problems*. Seminal works such as bird flocking [16, 17], slime systems [51, 65, 10], and evolution [38, 37] successfully provide algorithmic explanations for different natural objects. These works give rigorous theoretical results to confirm empirical observations, shed new light on the biological systems through computational lens, and sometimes lead to new biologically inspired algorithms.

In this work, we investigate *Spiking Neural Networks (SNNs)* as natural algorithms for solving convex optimization problems. SNNs are mathematical models for biological neural networks where a network of neurons transmit information by *firing spikes* through their synaptic connections (i.e., edges between two neurons). Our starting point is a seminal work of Barrett, Denève, and Machens [4], where they showed that the *firing rate* (i.e., the average number of spikes fired by each neuron) of a certain class of *integrate-and-fire* SNNs can be characterized by the optimal solutions of a quadratic program defined by the parameters of SNN. Thus, the SNN can be viewed as a natural algorithm for the corresponding quadratic program. However, no rigorous analysis was given in their work.

We bridge the gap by showing that the firing rate converges to an optimal solution of the corresponding quadratic program with an explicit polynomial bound on the convergent rate. Thus, the SNN indeed gives an *efficient algorithm* for solving the quadratic program. To the best of our knowledge, this is the first result with an explicit bound on the convergent rate. Previous works [58, 59, 63] on related SNN models for optimization problems are either heuristic or only proving convergence results when the time goes to infinity (see Section 1.4 for full discussion on related works).

We take one step further to ask what other optimization problems can SNNs efficiently solve. As our main result, we show that when configured properly, SNNs can solve the ℓ_1 *minimization problem*[2] in polynomial time[3]. Our main technical insight is interpreting the dynamics of SNNs in a *dual space*. In this way, SNNs can be viewed as a new primal-dual algorithm for solving the ℓ_1 *minimization problem*.

In the rest of the introduction, we will first briefly introduce the background of spiking neural networks (SNNs) and formally define the mathematical model we are working on. Next, our results will be presented and compared with other related works. Finally, we wrap up this section with potential future research directions and perspectives.

1.1 Spiking Neural Networks

Spiking neural networks (SNNs) are mathematical models for describing the dynamics of biological neural networks. An SNN consists of neurons, and each of them is associated with an intrinsic electrical charge called *membrane potential*. When the potential of a neuron reaches a certain level, it will fire an instantaneous signal, i.e., *spike*, to other neurons and increase or decrease their potentials.

---

[2] The problem is defined as given $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, and guaranteed that there is a solution to $A\mathbf{x} = \mathbf{b}$. The goal is finding a solution $\mathbf{x}$ with the smallest ℓ_1 norm. See Section 2 for formal definition.

[3] The running time is polynomial in a parameter depending on the inputs. In some cases, this parameter might cause the running to be quasi-polynomial or sub-exponential. See the full version of this paper on arXiv for more details.

Mathematically, the dynamic of neuron's membrane potential in an SNN is typically described by a *differential equation*, and there are many well-studied models such as the *integrate-and-fire model* [35], the *Hodgkin-Huxley model* [27], and their variants [21, 61, 49, 26, 23, 33, 14, 22, 29, 64]. In this work, we focus on the integrate-and-fire model defined as follows. Let n be the number of neurons and $\mathbf{u}(t) \in \mathbb{R}^n$ be the vector of membrane potentials where $\mathbf{u}_i(t)$ is the potential of neuron i at time t for any $i \in [n]$ and $t \geq 0$. The dynamics of $\mathbf{u}(t)$ can be described by the following differential equation: for each $i \in [n]$ and $t \geq 0$

$$\frac{d}{dt}\mathbf{u}_i(t) = \sum_{j\in[n]} -C_{ji}(t)\mathbf{s}_j(t) + \mathbf{I}_i(t) \tag{1}$$

where the initial value of the potentials are set to 0, i.e., $\mathbf{u}_i(0) = 0$ for each $i \in [n]$. There are two terms that determine the dynamics of membrane potentials as shown in (1). The simpler term is the input charging[4] $\mathbf{I}(t) \in \mathbb{R}^n$, which can be thought of as an external effect on each neuron. The other term models the instantaneous spike effect among neurons. Specifically, the $-C_{ji}(t)\mathbf{s}_j(t)$ term models the effect on the potential of neuron i when neuron j fires a spike. Here $C(t) \in \mathbb{R}^{n\times n}$ is the connectivity matrix that encodes the *synapses* between neurons, where $C_{ji}(t)$ describes the connection strength from neuron j to neuron i. $\mathbf{s}(t) \in \mathbb{R}^n$ is the *spike train* which records the spikes of each neuron, and $\mathbf{s}_i(t)$ can be thought of as indicating whether neuron i fires a spike at time t. To sum up, the $-C_{ji}(t)\mathbf{s}(t)$ term decreases[5] the potential of neuron i by $C_{ji}(t^*)$ whenever neuron j fires a spike at time t^*.

The spike train $\mathbf{s}(t)$ is determined by the spike events, which are in turn determined by the spiking rule. A typical spiking rule is the threshold rule. Specifically, let $\eta > 0$ be the spiking threshold, the threshold rule simply says that neuron i fires a spike at time t if and only if $\mathbf{u}_i(t) > \eta$. Next, record the timings when neuron i fires a spike as $0 \leq t_1^{(i)} < t_2^{(i)} < \dots$ and let $k_i(t)$ be the number of spikes within time $[0, t]$. An important statistics of the dynamics is the *firing rate* defined as $\mathbf{x}_i(t) := k_i(t)/t$ for neuron $i \in [n]$ at time t, namely, the average number of spikes of neuron i up to time t. The last thing we need for specifying $\mathbf{s}(t)$ is the *spike shape*, which can be modeled as a function $\delta : \mathbb{R}_{\geq 0} \to \mathbb{R}$. Intuitively, the spike shape describes the effect of a spike, and standard choices of δ could be the Dirac delta function or a pulse function with an exponential tail. Now we can define $\mathbf{s}_i(t) = \sum_{1\leq s\leq k_i(t)} \delta(t - t_s^{(i)})$ to be the *spike train* of neuron i at time t.

We provide the following example to illustrate the SNN dynamics introduced above.

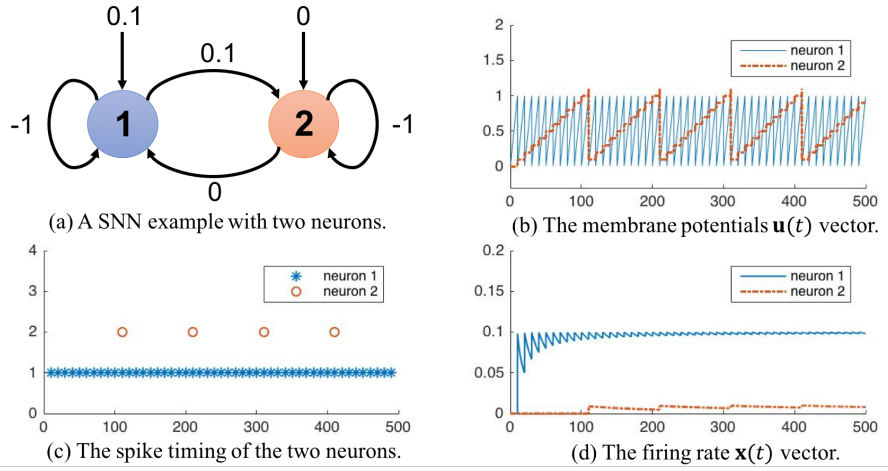
▶ **Example 1.** *Let $n = 2$, $\eta = 1$, and δ be the Dirac delta function such that for any $\epsilon > 0$, $\int_0^\epsilon \delta(t)dt = 1$ and $\delta(t) \geq 0$ for any $t \geq 0$. Let both external charging and connectivity matrix be static, i.e., $\mathbf{I}(t) = \mathbf{I}$ and $C(t) = C$ for any $t \geq 0$, and consider*

$$C = \begin{pmatrix} 1 & 0 \\ -0.1 & 1 \end{pmatrix}, \ \mathbf{I} = \begin{pmatrix} 0.1 \\ 0 \end{pmatrix}, \ \textit{and } \mathbf{u}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

In Figure 1, we simulate this SNN for 500 seconds. We can see that neuron 1 fires a spike every ten seconds while neuron 2 fires a spike every one hundred seconds. As a result, the firing rate of neuron 1 will gradually converge to 0.1 and that of neuron 2 will go to 0.01.

In general, both the input charging vector $\mathbf{I}(t)$ and the connectivity matrix $C(t)$ can evolve over time, in which the change of $\mathbf{I}(t)$ models the variation of the environment and the change of $C_{ji}(t)$ captures the adaptive *learning* behavior of the neurons to the environmental change.

[4] Also known as *input signal* or *input current*.
[5] If $C_{ji}(t^*) < 0$, then the potential of neuron i actually *increases by* $|C_{ji}(t^*)|$.



(a) A SNN example with two neurons.

(b) The membrane potentials $\mathbf{u}(t)$ vector.

(c) The spike timing of the two neurons.

(d) The firing rate $\mathbf{x}(t)$ vector.

Figure 1 The example of SNN with two neurons. In (a), we describe the dynamic of this SNN. Note that the effect of spikes is the negation of the synapse encoded in the connectivity matrix C. In (b), we plot the membrane potential vectors $\mathbf{u}(t)$. In (c), we plot the timings when neurons fire a spike. One can see that neuron 1 fires a spike every ten seconds while neuron 2 fires a spike every one hundred seconds. In (d), we plot the firing rate vector $\mathbf{x}(t)$. One can see that the firing rate of neuron 1 will gradually converge to 0.1 and that of neuron 2 will go to 0.01.

Understanding how synapses evolve over time (i.e., synapse plasticity) is a very important subject in neuroscience. However, in this work, we follow the choice of Barrett et al. [4] and consider *static* SNN dynamics, where both the input charging $\mathbf{I}(t)$ and the synapses $C(t)$ are constants. Although this is a special case compared to the general model in (1), we justify the choice of static SNN by showing that SNN already exhibits non-trivial computational power even in this restricted model.

As in Barrett et al. [4], we focus on static SNN and view it as a natural algorithm for optimization problems. Specifically, given an instance to the optimization problem, the goal is to configure a static SNN (by setting its parameters) so that the firing rate converge to an optimal solution efficiently. In this sense, the result of Barrett et al. [4] can be interpreted as a natural algorithm for certain quadratic programs. In our eyes, the solution being encoded as the firing rate is an interesting and peculiar feature of the SNN dynamics. Also, the dynamics of a static SNN can be viewed as a simple distributed algorithm with a simple communication pattern. Specifically, once the dynamics is set up, each neuron only needs to keep track of its potential and communicate with each other through spikes.

1.2 Our Results

Barrett et al. [4] gave a clean characterization of the firing rates by the network connectivity and input signal. Concretely, they consider *static* SNN where both the connectivity matrix $C \in \mathbb{R}^{n \times n}$ and the external charging $\mathbf{I} \in \mathbb{R}^n$ do not change with time. They argued that the firing rate would converge to the solution of the following quadratic program.

$$\begin{aligned} &\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} && \|C\mathbf{x} - \mathbf{I}\|_2^2 \\ &\text{subject to} && \mathbf{x}_i \geq 0, \ \forall i \in [n]. \end{aligned} \tag{2}$$

They supported this observation by giving simulations on the so called *tightly balanced networks* and yielded pretty accurate predictions in practice. Also, they heuristically explained the reason how they came up with the quadratic program. However, no rigorous theorem had been proved on the convergence of firing rate to the solution of this quadratic program.

To give a theoretical explanation for the discovery of [4], we start with a simpler SNN model to enable the analysis.

The simple SNN model. In the simple SNN model, we make two simplifications on the general model in (1).

First, we pick the shape of spike to be the Dirac delta function. That is, let $\delta(t) = \mathbf{1}_{t=0}$ and thus $\mathbf{s}_i(t) = \mathbf{1}_{\mathbf{u}_i(t)>\eta}$. This simplification saves us from complicated calculation while the Dirac delta function still captures the instantaneous behavior of a spike.

Second, we consider the connectivity matrix C in the form $C = \alpha \cdot A^\top A$ where $\alpha > 0$ is the spiking strength and $A \in \mathbb{R}^{m\times n}$ is the Cholesky decomposition of C. The reason for introducing α is to model the height of the Dirac delta function. Mathematically, it is redundant to have both α and C since the model remains the same when combining α with C. However, as we will see in the next subsection, separating α and C is meaningful as C corresponds to the *input* of the computational problem and α is the parameter that one can choose to configure an SNN to solve the problem.

In this work, we focus on the algorithmic power of SNN in the following sense. Given a problem instance, one configures a SNN and sets the firing rate $\mathbf{x}(t)$ to be the output at time t. We say this SNN solves the problem if $\mathbf{x}(t)$ converges to the solution of the problem.

Simple SNN solves the non-negative least squares. As mentioned, Barrett et al. [4] identified a connection between the firing rate of SNN with integrate-and-fire neurons and a quadratic programming problem (2). They gave empirical evidence for the correctness of this connection, however, no theoretical guarantee had been provided. Our first result confirms their observation by giving the first theoretical analysis. Specifically, when $C = A^\top A$ and $\mathbf{I} = A^\top \mathbf{b}$, the firing rate will converge to the solution of the following *non-negative least squares problem*.

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \|A\mathbf{x} - \mathbf{b}\|_2^2 \\ \text{subject to} \quad & \mathbf{x}_i \geq 0, \ \forall i \in [n]. \end{aligned} \tag{3}$$

▶ **Theorem 1** (informal). *Given $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\epsilon > 0$. Suppose A satisfies some regular conditions*[6]*. Let $\mathbf{x}(t)$ be the firing rate of the simple SNN with $0 < \alpha \leq \alpha(A)$ where $\alpha(A)$ is a function depending on A. When $t \geq \Omega(\frac{\sqrt{n}}{\epsilon\cdot\|\mathbf{b}\|_2})$,*[7] *$\mathbf{x}(t)$ is an ϵ-approximate solution*[8] *for the non-negative least squares problem of $(A, \mathbf{b})$.*

The formal statement and the proof for the theorem are provided in the Section 4 of the full version of this paper. To the best of our knowledge, this is the first[9] theoretical result on the analysis of SNN with an explicit bound on the convergence rate and shows that SNN can be implemented as an efficient algorithm for an optimization problem.

[6] More details about the regular conditions will be discussed in Section 3.3 of the full version.
[7] The $\Omega(\cdot)$ and the $O(\cdot)$ later both hide the dependency on some parameters of A. See Section 3.3 of the full version for more details.
[8] See Definition 3 for the formal definition of ϵ-approximate solution.
[9] See Section 1.4 for comparisons with related works.

Simple SNN solves the ℓ_1 minimization problem. In addition to solving the non-negative least squares problem, as our main result, we also show that the simple SNN is able to solve the ℓ_1 *minimization problem*, which is defined as minimizing the ℓ_1 norm of the solutions of $A\mathbf{x} = \mathbf{b}$. ℓ_1 minimization problem is also known as the *basis pursuit* problem proposed by Chen et al. [18]. The problem is widely used for recovering sparse solution in compressed sensing, signal processing, face recognition etc.

Before the discussion on ℓ_1 minimization, let us start with a digression on the *two-sided* simple SNN for the convenience of future analysis.

$$\frac{d}{dt}\mathbf{u}(t) = -\alpha \cdot A^\top A\mathbf{s}(t) + A^\top \mathbf{b}$$

where $\mathbf{s}_i(t) = \mathbf{1}_{\mathbf{u}_i(t)>\eta} - \mathbf{1}_{\mathbf{u}_i(t)<-\eta}$. Note that the two-sided SNN is a special case of the one-sided SNN in the sense that one can use the one-sided SNN to simulate the two-sided SNN as follows. Given a two-sided SNN described above with connectivity matrix $C = A^\top A$ and external charging $\mathbf{I} = A^\top \mathbf{b}$. Let $C' = \begin{pmatrix} A^\top A & -A^\top A \\ -A^\top A & A^\top A \end{pmatrix}$ and $\mathbf{I}' = \begin{pmatrix} A^\top \mathbf{b} \\ -A^\top \mathbf{b} \end{pmatrix}$. Intuitively, this can be thought of as duplicating each neuron and flip its connectivities with other neurons.

To solve the ℓ_1 minimization problem, we simply configure a two-sided SNN as follows. Given an input $(A, \mathbf{b})$, let $C = A^\top A$ and $\mathbf{I} = A^\top \mathbf{b}$. Now, we have the following theorem.

▶ **Theorem 2** (informal). *Given $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\epsilon > 0$. Suppose A satisfies some regular conditions. Let $\mathbf{x}(t)$ be the firing rate of the two-sided simple SNN with $0 < \alpha \leq \alpha(A)$ where $\alpha(A)$ is a function depending on A. When $t \geq \Omega(\frac{n^3}{\epsilon^2})$, $\mathbf{x}(t)$ is an ϵ-approximate solution[10] for the ℓ_1 minimization problem of $(A, \mathbf{b})$.*

See Theorem 7 for the formal statement of this theorem. As we will discuss in the next subsection, under the dual view of the SNN dynamics, the simple two sided SNN can be interpreted as a new natural primal-dual algorithm for the ℓ_1 minimization problem.

1.3 A Dual View of the SNN Dynamics

The main techniques in this work is the discovery of a *dual view* of SNN. Recall that the dynamic of a static SNN can be described by the following differential equation.

$$\frac{d}{dt}\mathbf{u}(t) = -\alpha \cdot C\mathbf{s}(t) + \mathbf{I}$$

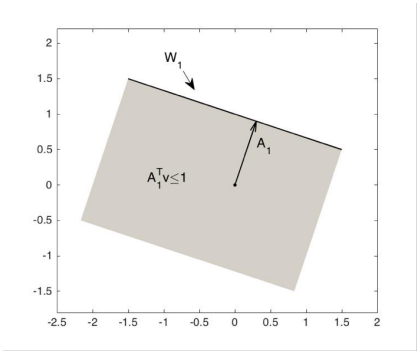
where $\mathbf{u}(0) = \mathbf{0}$ the parameters C and $\mathbf{I}$ can be represented as $C = A^\top A$ and $\mathbf{I} = A^\top \mathbf{b}$ for some $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. For simplicity, we pick the firing threshold $\eta = 1$ here. Let us call the dynamics of $\mathbf{u}(t)$ the *primal SNN*. Now, the *dual SNN*, can be defined as follows.

$$\frac{d}{dt}\mathbf{v}(t) = -\alpha \cdot A\mathbf{s}(t) + \mathbf{b}$$

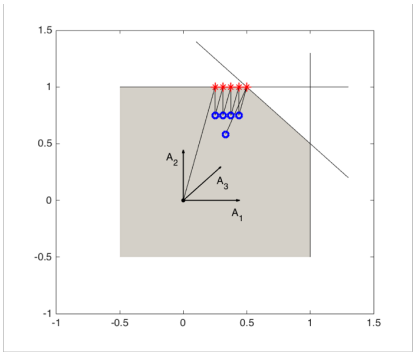
where $\mathbf{v}(0) = \mathbf{0}$ and $\mathbf{s}(t)$ defined as the usual way. At first glance, this merely looks like a simple linear transformation, Nevertheless, the dual SNN provides a nice *geometric view* for the SNN dynamics as follows.

At each update in the dynamics, there are two terms affecting the dual SNN $\mathbf{v}(t)$: the external charging $\mathbf{b} \cdot dt$ and the spiking effect $-\alpha \cdot A\mathbf{s}(t)$. First, the external charging $\mathbf{b} \cdot dt$ can be thought of as a constant force that drags that dual SNN in the direction $\mathbf{b}$.

[10] See Definition 4 for the formal definition of ϵ-approximate solution.



(a) An example of one neuron.



(b) The effect of both the external charging and spikes on dual SNN

Figure 2 These are examples of the geometric interpretation of the dual SNN. In (a), we have one neuron where $A_1 = [\frac{1}{2}\ 1]^\top$. In this case, neuron i would not fire as long as the dual SNN $\mathbf{v}(t)$ stays in the gray area. In (b), we consider a SNN with 3 neurons where $A_1 = [1\ 0]^\top$, $A_2 = [0\ 1]^\top$, and $A_3 = [\frac{2}{3}\ \frac{2}{3}]^\top$. One can see that the effect of spikes on dual SNN is a jump in the direction of the normal vector of the wall(s).

Table 1 Comparison of the geometric view of primal and dual SNNs.

	Primal SNN $\mathbf{u}(t)$	Dual SNN $\mathbf{v}(t)$
Spiking rule	$\mathbf{u}_i(t) > 1$	$A_i^\top \mathbf{v}(t) > 1$
Spiking effect	$-\alpha \cdot A^\top A_i$	$-\alpha \cdot A_i$

To explain the effect of spikes in the dual view, let us start with an geometric view for the spiking rule. Recall that neuron i fires a spike at time t if and only if $\mathbf{u}_i(t) > 1$. In the language of dual SNN, this condition is equivalent to $A_i^\top \mathbf{v}(t) > 1$. Let $W_i = \{\mathbf{v} \in \mathbb{R}^m : \ A_i^\top \mathbf{v} = 1\}$ be the *wall* of neuron i, the above observation is saying that neuron i will fire a spike once it penetrates the wall W_i from the half-space $\{\mathbf{v} \in \mathbb{R}^m : \ A_i^\top \mathbf{v} \leq 1\}$. See Figure 2 for an example. After neuron i fires a spike, the spiking effect on the dual SNN $\mathbf{v}(t)$ would be a $-\alpha \cdot A_i$ term, which corresponds to a jump in the *normal direction* of W_i. See Figure 2 for an example.

The geometric interpretation described above is the main advantage of using dual SNN. Specifically, this gives us a clear picture of how spikes affect the SNN dynamics. Namely, neuron i fires a spike if and only if the dual SNN $\mathbf{v}(t)$ penetrates the wall W_i and then $\mathbf{v}(t)$ jumps back in the normal direction of W_i. Note that this connection would not hold in the primal SNN. In primal SNN $\mathbf{u}(t)$, neuron i fires a spike if and only if $\mathbf{u}_i(t) > 1$ while the effect on $\mathbf{u}(t)$ is moving in the direction $-A^\top A_i$. See Table 1 for a comparison.

Dual view of SNN as a primal-dual algorithm for ℓ_1 minimization problem. First, let us write down the ℓ_1 minimization problem and its dual.

$$\begin{aligned} &\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} && \|\mathbf{x}\|_1 \\ &\text{subject to} && A\mathbf{x} = \mathbf{b}. \end{aligned} \qquad \begin{aligned} &\underset{\mathbf{v}\in\mathbb{R}^m}{\text{maximize}} && \mathbf{b}^\top \mathbf{v} \\ &\text{subject to} && \|A^\top \mathbf{v}\|_\infty \leq 1. \end{aligned}$$

Now we observe that the dual dynamics can be viewed as a variant of the projected gradient descent algorithm to solve the dual program. Before the explanation, recall that for the ℓ_1 minimization problem, we are considering the two-sided SNN for convenience. Indeed, without the spiking term, $\mathbf{v}(t)$ simply moves towards the gradient direction $\mathbf{b}$ of the dual objective function $\mathbf{b}^\top \mathbf{v}$. For the spike term $-\alpha \cdot A\mathbf{s}(t)$, note that $\mathbf{s}_i(t) \neq 0$ (i.e., neuron i fires) if and only if $|A_i^\top \mathbf{v}(t)| = |\mathbf{u}_i(t)| > 1$, which means that $\mathbf{v}(t)$ is outside the feasible polytope $\{\mathbf{v} : \|A^\top \mathbf{v}\|_\infty \leq 1\}$ of the dual program. Therefore, one can view the role of the spike term as *projecting* $\mathbf{v}(t)$ back to the feasible polytope. That is, when the dual SNN $\mathbf{v}(t)$ becomes infeasible, it triggers some spikes, which maintains the dual feasibility and updates the primal solution (the firing rate). To sum up, we can interpret the simple SNN as performing a non-standard projected gradient descent algorithm for the dual program of ℓ_1 minimization in the dual view of SNN.

With this primal-dual view in mind, we analyze the SNN algorithm by combining tools from convex geometry and perturbation theory as well as several non-trivial structural lemmas on the geometry of the dual program of ℓ_1 minimization. One of the key ingredients here is identifying a *potential function* that (i) upper bounds the error of solving ℓ_1 minimization problem and (ii) monotonously converges to 0. More details will be provided in Section 3.

1.4 Related Work

We compare this research with other related works in the following four aspects.

Computational power of SNN. Recognized as the third generation of neural networks [45], the theoretical foundation for the computability of SNN had been built in the pioneering works of Maass et al. [43, 45, 46, 48] in which SNN was shown to be able to simulate standard computational models such as Turing machines, random access machines (RAM), and threshold circuits.

However, this line of works focused on the universality of the computational power and did not consider the efficiency of SNN in solving specific computational problems. In recent years, a line of exciting research have reported the efficiency of SNN in solving specific computational problems such as sparse coding [68, 62, 63], dictionary learning [36], pattern recognition [19, 32, 6], and quadratic programming [4]. These works indicated the advantage of SNN in handling *sparsity* as well as being *energy efficient* and inspired real-world applications [5]. However, to the best of our knowledge, no theoretical guarantee on the efficiency of SNN had been provided. For instance, Tang et al. [62, 63] only proved the *convergence in the limit* result for SNN solving sparse coding problem instead of giving an explicit convergence rate analysis. The main contribution in this work is giving a rigorous guarantee on the convergence rate of the computational power of SNN.

The number of spikes versus the timing of spikes. In this work, we mainly focus on the firing rate of SNN. That is, we only study the computational power with respect to the *number* of spikes. Another important property of SNN is the *timing* of spikes.

The power of the timing of spikes had been reported since the 90s from some experimental evidences indicating that neural systems might use the timing of spikes to encode information [1, 28, 54]. From then on, a bunch of works have been focused on the aspect of time as a basis of information coding both from theoretical [52, 45, 48, 66] and experimental [25, 7, 34] sides. It is generally believed that the timing of spikes is more powerful then the firing rate [67, 56, 53]. Other than the capacity of encoding information, the timing of spikes has also been studied in the context of computational power [67, 44, 45, 24] and learning [12, 3, 60]. See the survey by Paugam et al. [53] for a thorough discussion.

While the timing of spikes is conceived as an important source of the power of SNN, in this work we simply focus on the firing rate and already yield some non-trivial findings in terms of the computational power. We believe that our work is still in the very beginning stage of the study of the computational power of SNN. Investigating how does the timing of spikes play a role is an interesting and important future direction. Immediate open questions here would be how could the timing of spikes fit into our study? What's the dual view of the timing of spikes? Can the timing of spikes solve the optimization problems more efficiently? Can the timing of spikes solve more difficult problems?

SNN with randomness. While most of the literature focused on deterministic SNN, there is also an active line of works studying the SNN model with randomness[11] [2, 57, 20, 15, 30, 47, 31, 40, 41, 42, 39].

Buesing et al. [15] used *noisy SNN* to implement MCMC sampling and Jonke et al. [30, 47, 31] further instantiated the idea to attack **NP**-hard problems such as *traveling salesman problem (TSP)* and *constraint satisfaction problem (CSP)*. Concretely, their noisy SNN has a randomized spiking rule and the firing pattern would form a distribution over the solution space whereas the closer a solution is to the optimal solution, the higher the probability it is sampled. They got nice experimental performance in terms of solving empirical instance approximately. They also pointed out that their noisy SNN has the potential to be implemented energy-efficiently in practice.

Lynch, Musco, and Parter [41] studied the *stochastic SNNs* with a focus on the Winner-Take-All (WTA) problem. Their sequence of works [40, 41, 42, 39] gave the first asymptotic analysis for stochastic SNN in solving WTA, similarity testing, and neural coding. They view SNNs as distributed algorithms and derived computational tradeoff in running time and network size.

In this work, we consider the SNN model without randomness and thus is incomparable with the above SNN models with randomness. It is an interesting direction to apply the dual view of deterministic SNN to SNN with randomness.

Locally competitive algorithms. Inspired by the dynamics of biological neural networks, Ruzell et al. designed the *locally competitive algorithms* (LCA) [55] for solving the Lasso (least absolute shrinkage and selection operator) optimization problem[12], which is widely used in statistical modeling. Roughly speaking, LCA is also a dynamics among a set of *artificial neurons* which continuously signal their potential values (or a function of the values) to their neighboring neurons. There are two main differences between SNN and LCA. First, the neuron in SNN fires discrete spikes while the artificial neuron in LCA produces continuous signal. Next, the neurons' potentials in LCA will converge to a fixed value, which is the output of the algorithm. In contrast, in SNN, only the neurons' firing rates may converge instead of their potentials.

Nevertheless, there is a *spikified* version of LCA introduced by Shapero et al. [58, 59] called *spike LCA (S-LCA)* in which the continuous signals are replaced with discrete spikes. S-LCA is almost the same as the SNN we are considering except a shrinkage term[13]. Recently, Tang et al. [63] showed that the firing rate of S-LCA indeed converges to a variant of

[11] SNN model with noise is also known as stochastic SNN or noisy SNN depending on how the randomness involves in the model.

[12] Note that Lasso is equivalent to the Basis Pursuit De-Noising (BPDN) program under certain parameters transformation.

[13] That is, the potential of each neuron will drop with rate proportional to the current potential value.

Lasso problem[14] in the limit. These works also experimentally demonstrated the efficient convergence of S-LCA and its advantage of fast identifying sparse solutions with potentially competitive practical performance to other Lasso algorithms (e.g., FISTA [5]). However, there is no proof of convergence rate, and thus no explicit complexity bound of S-LCA.

1.5 Future Works and Perspectives

In this work, we give a theoretical study on the algorithmic power of SNN. Specifically, we focus on the firing rate of SNN and confirm an empirical analysis by Barrett et al. [4] with a convergence theorem (i.e., Theorem 1). Furthermore, we discover a dual view of SNN and show that SNN is able to solve the ℓ_1 minimization problem (i.e., Theorem 2). In the following, we give interpretations to our results and point out future research directions.

First, how to interpret the dual dynamics of SNN? In this work, we discover the dual SNN based on mathematical convenience. Is there any biological interpretation?

Second, push further the analysis of simple SNN. We believe the parameters we get in the main theorems are not optimal. Is it possible to further sharpen the upper bound? We think this is both theoretically and practically interesting because both non-negative least squares and ℓ_1 minimization are important problems that have been well-studied studied in the literature. Comparing the running time complexity or parallel time complexity of SNN algorithm with other algorithms could also be of theoretical interest and might inspire new algorithm with better complexity. Also, for practical purpose, having better parameters would give more confidence in implementing SNN as a natural algorithm.

Third, further investigate the potential of SNN dynamics as natural algorithms. The question is two-folded: (i) What algorithms can SNN implement? (ii) What computational problems can SNN solve? It seems that SNN is good at dealing with sparsity. Could it be helpful in related computational tasks such as fast Fourier transform (FFT) or sparse matrix-vector multiplication? It is interesting to identify optimization problems and class of instances where SNN algorithm can outperform other algorithms.

Finally, explore the practical advantage of SNN dynamics as natural algorithms. The potential practical time efficiency, energy efficiency, and simplicity for hardware implementation have been suggested in several works [50, 8, 9]. It would be exciting to see whether SNN has nice performance on practical applications such as compressed sensing, Lasso, and etc.

Organization. The rest of the paper is organized as follows. Preliminaries are provided in Section 2. In Section 3, we formally present the dual view of SNN and give a proof sketch for the convergence theorem for ℓ_1 minimization problem. The full proofs for Theorem 1 and Theorem 2 are provided in the full version of this paper available on arXiv.

2 Preliminaries

In Section 2.1, we build up some notations for the rest of the paper. In Section 2.2, we define two optimization problems and the corresponding convergence guarantees.

2.1 Notations

For any $n \in \mathbb{N}$, denote $[n] = \{1, 2, \ldots, n\}$ and $[\pm n] = \{\pm 1, \pm 2, \ldots, \pm n\}$. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ be two vectors. $|\mathbf{x}| \in \mathbb{R}^n$ denotes the entry-wise absolute value of $\mathbf{x}$, i.e., $|\mathbf{x}|_i = |\mathbf{x}_i|$ for any $i \in [n]$. $\mathbf{x} \preceq \mathbf{y}$ refers to entry-wise comparison, i.e., $\mathbf{x}_i \leq \mathbf{y}_i \ \forall i \in [n]$.

[14] In this variant, all the entries in matrix A is non-negative.

Let A be an $m \times n$ real matrix. For any $i \in [n]$, denote the ith column of A as A_i and its negation to be A_{-i}, i.e., $A_{-i} = -A_i$. When A is positive semidefinite, we define the A-norm of a vector $\mathbf{x} \in \mathbb{R}^n$ to be $\|\mathbf{x}\|_A := \sqrt{\mathbf{x}^\top A\mathbf{x}}$. Let $A^\dagger$ to be the pseudo-inverse of A. Define the maximum eigenvalue of A as $\lambda_{\max}(A) := \max_{\mathbf{x}\in\mathbb{R}^n:\ \|\mathbf{x}\|_2=1} \|\mathbf{x}\|_A$, the minimum non-zero eigenvalue of A to be $\lambda_{\min}(A) := 1/(\max_{\mathbf{x}\in\mathbb{R}^n:\ \|\mathbf{x}\|_2=1} \|\mathbf{x}\|_{A^\dagger})$, and the condition number of A to be $\kappa(A) := \lambda_{\max}(A)/\lambda_{\min}(A)$. If we do not specified, the following $\lambda_{\max}, \lambda_{\min}$, and κ are the eigenvalues and condition number of the connectivity matrix $C = A^\top A$. For any $\mathbf{b} \in \mathbb{R}^m$, we denote $\mathbf{b}_A$ to be the projection of $\mathbf{b}$ on the range space of A.

2.2 Optimization problems

In this subsection, we are going to introduce two optimization problems: *non-negative least squares* and ℓ_1 *minimization*.

2.2.1 Non-negative least squares

▶ **Problem 1** (non-negative least squares). *Let $m, n \in \mathbb{N}$. Given $A \in \mathbb{R}^{m\times n}$ and vector $\mathbf{b} \in \mathbb{R}^m$, find $\mathbf{x} \in \mathbb{R}^n$ that minimizes $\|\mathbf{b} - A\mathbf{x}\|_2^2/2$ subject to $\mathbf{x}_i \geq 0$ for all $i \in [n]$.*

▶ Remark. Recall that the least squares problem is defined as finding $\mathbf{x}$ that minimize $\|\mathbf{b} - A\mathbf{x}\|_2$. That is, the non-negative least squares is a restricted version of the least squares problem. Nevertheless, one can use a non-negative least squares solver to solve the least squares problem by setting $A' = \begin{pmatrix} A^\top A & -A^\top A \\ -A^\top A & A^\top A \end{pmatrix}$ and $\mathbf{b}' = \begin{pmatrix} \mathbf{b} \\ -\mathbf{b} \end{pmatrix}$ where $(A, \mathbf{b})$ is the instance of least squares and $(A', \mathbf{b}')$ is the instance of non-negative least squares.

The SNN algorithm might not solve the non-negative least squares problem exactly and thus we define the following notion of solving the non-negative least squares problem *approximately*.

▶ **Definition 3** (ϵ-approximate solution to non-negative least squares). Let $m, n \in \mathbb{N}$ and $\epsilon > 0$. Given $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. We say $\mathbf{x}$ is an ϵ-approximate solution to the non-negative least squares problem of $(A, \mathbf{b})$ if $\|A\mathbf{x} - A\mathbf{x}^*\|_2 \leq \epsilon\|\mathbf{b}\|_2$ where $\mathbf{x}^*$ is an optimal solution.

2.2.2 ℓ_1 minimization

▶ **Problem 2** (ℓ_1 minimization). *Let $m, n \in \mathbb{N}$. Given $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$ such that there exists a solution to $A\mathbf{x} = \mathbf{b}$. The goal of ℓ_1 minimization is to solve the following optimization problem.*

$$\begin{aligned} &\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} && \|\mathbf{x}\|_1 \\ &\text{subject to} && A\mathbf{x} = \mathbf{b}. \end{aligned}$$

Similarly, we do not expect SNN algorithm to solve the ℓ_1 minimization exactly. Thus, we define the notion of solving the ℓ_1 minimization problem *approximately* as follows.

▶ **Definition 4** (ϵ-approximate solution to ℓ_1 minimization). Let $m, n \in \mathbb{N}$ and $\epsilon > 0$. Given $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Let $\mathbf{OPT}^{\ell_1}$ denote the optimal value of the ℓ_1 minimization problem of $(A, \mathbf{b})$. We say $\mathbf{x} \in \mathbb{R}^n$ is an ϵ-approximate solution of the ℓ_1 minimization problem of $(A, \mathbf{b})$ if $\|\mathbf{b} - A\mathbf{x}\|_2 \leq \epsilon \cdot \|\mathbf{b}\|_2$ and $\|\mathbf{x}\|_1 - \mathbf{OPT}^{\ell_1} \leq \epsilon \cdot \mathbf{OPT}^{\ell_1}$.

2.3 Karush-Kuhn-Tucker conditions

Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient conditions for the optimality of optimization problems under some regular assumptions. Consider the following optimization program.

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad \forall i = 1, 2, \dots m, \\ & h_j(\mathbf{x}) = 0, \quad \forall j = 1, 2, \dots, k, \end{aligned} \tag{4}$$

where $f, g_1, \dots, g_m, h_1, \dots, h_k$ are convex and differentiable. Let $\mathbf{v} \in \mathbb{R}^m$ and $\boldsymbol{\mu} \in \mathbb{R}^k$ be the dual variables. KKT conditions give necessary and sufficient conditions for $(\mathbf{x}, \mathbf{v}, \boldsymbol{\mu})$ be a pair of primal and dual optimal solutions.

▶ **Theorem 5** (KKT conditions, Chapter 5.5.3 in [13]). *$(\mathbf{x}, \mathbf{v}, \boldsymbol{\mu})$ are a pair of primal and dual optimal solutions for (4) if and only if the following conditions hold.*

- *$\mathbf{x}$ is primal feasible, i.e., $g_i(\mathbf{x}) \leq$ and $h_j(\mathbf{x}) = 0$ for all $i \in [m]$ and $j \in [k]$.*
- *$(\mathbf{v}, \boldsymbol{\mu})$ is dual feasible, i.e., $\mathbf{v}_i \geq 0$ for all $i \in [m]$.*
- *The Lagrange multiplier vanishes, i.e., $\nabla f(\mathbf{x}) + \sum_{i\in[m]} \mathbf{v}_i \nabla g_i(\mathbf{x}) + \sum_{j\in[k]} \boldsymbol{\mu}_j \nabla h_j(\mathbf{x}) = 0$.*
- *$(\mathbf{x}, \mathbf{v}, \boldsymbol{\mu})$ satisfy complementary slackness, i.e., $\mathbf{v}_i f_i(\mathbf{x}) \geq 0$ for all $i \in [m]$.*

2.4 Perturbation theory

Perturbation theory, sometimes known as sensitivity analysis, for optimization problems concerns the situation where the optimization program is perturbed and the goal is to give a good estimation for the optimal solution. See a nice survey by Bonnans and Shapiro [11]. In the following we state a special case for convex optimization program with strong duality.

▶ **Theorem 6** (perturbation, Chapter 5.6 in [13][15]). *Given the following two optimization programs where the strong duality holds and there exists feasible dual solution.*

$$\begin{aligned} \underset{\mathbf{x}}{\textit{minimize}} \quad & f(\mathbf{x}) \\ \textit{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad \forall i = 1, 2, \dots, m, \\ & h_j(\mathbf{x}) = 0, \quad \forall j = 1, 2, \dots, k. \end{aligned} \tag{5}$$

$$\begin{aligned} \underset{\mathbf{x}}{\textit{minimize}} \quad & f(\mathbf{x}) \\ \textit{subject to} \quad & g_i(\mathbf{x}) \leq \mathbf{a}_i, \quad \forall i = 1, 2, \dots, m, \\ & h_j(\mathbf{x}) = \mathbf{b}_j, \quad \forall j = 1, 2, \dots, k. \end{aligned} \tag{6}$$

Let $\mathbf{OPT}^{original}$ be the optimal value of the original program (5) and $\mathbf{OPT}^{perturbed}$ be the optimal value of the perturbed program (6). Let $(\mathbf{v}^*, \boldsymbol{\mu}^*) \in \mathbb{R}^m \times \mathbb{R}^k$ be the optimal dual solution of the perturbed program (6). We have

$$\mathbf{OPT}^{original} \geq \mathbf{OPT}^{perturbed} + \mathbf{a}^\top \mathbf{v}^* + \mathbf{b}^\top \boldsymbol{\mu}^*.$$

[15] Note that we switch the original and perturbed programs in the statement in [13].

3 A simple SNN algorithm for ℓ_1 minimization

In this section, we focus on the discovery of the dual view of simple SNN and how it can be viewed as a *primal-dual algorithm* for solving the ℓ_1 minimization problem.

Recall that for the ℓ_1 minimization problem, we are working on the *two-sided* simple SNN for the convenience of future analysis. That is,

$$\frac{d}{dt}\mathbf{u}(t) = -\alpha \cdot A^\top A\mathbf{s}(t) + A^\top \mathbf{b},$$

where $\mathbf{s}_i(t) = \mathbf{1}_{\mathbf{u}_i(t)>\eta} - \mathbf{1}_{\mathbf{u}_i(t)<-\eta}$. To solve the ℓ_1 minimization problem, we configure a two-sided simple SNN as follows. Given an input $(A, \mathbf{b})$, let $C = A^\top A$ and $\mathbf{I} = A^\top \mathbf{b}$. However, currently it is unclear how does the above simple SNN dynamics relate to the ℓ_1 minimization problem.

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \|\mathbf{x}\|_1 \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b}. \end{aligned} \tag{7}$$

Interesting, the connection between simple SNN and the ℓ_1 minimization problem happens in the *dual program* of the ℓ_1 minimization problem. Before we formally explain this connection, let us write down the dual program of (7).

$$\begin{aligned} \underset{\mathbf{v}\in\mathbb{R}^m}{\text{maximize}} \quad & \mathbf{b}^\top \mathbf{v} \\ \text{subject to} \quad & \|A^\top \mathbf{v}\|_\infty \le 1. \end{aligned} \tag{8}$$

Let us try to make some geometric observations on (8). First, the objective of the dual program is to maximize the inner product with $\mathbf{b}$, which is quite related to the external charging of SNN since we take $\mathbf{I} = A^\top \mathbf{b}$. Next, the feasible region of the dual program is a polytope (or a polyhedron) defined by the intersection of half-spaces $\{\mathbf{v} \in \mathbb{R}^m : A_i^\top \mathbf{v} \le 1\}$ and $\{\mathbf{v} \in \mathbb{R}^m : -A_i^\top \mathbf{v} \le 1\}$ for each $i \in [n]$ where A_i denotes the i^{th} column of A.

It will be convenient to introduce the following notation before we move on. For $i \in [n]$, let $A_{-i} = -A_i$. Let $[\pm n] = \{\pm 1, \pm 2, \ldots, \pm n\}$. Thus, the feasible polytope of the dual program is defined by the intersection of half-spaces defined by $A_j^\top \mathbf{v} \le 1$ for all $j \in [\pm n]$. We call this polytope the *dual polytope*[16]. Moreover, for each $j \in [\pm n]$, we call the hyperplane $\{\mathbf{v} : A_j^\top \mathbf{v} = 1\}$ the *wall* W_j of the dual polytope. See Figure 3 for examples.

Now, the key observation is that by a linear transformation, the dynamics of simple SNN has a natural interpretation in the dual space. We call it the *dual SNN* defined as follows.

3.1 Dual SNN

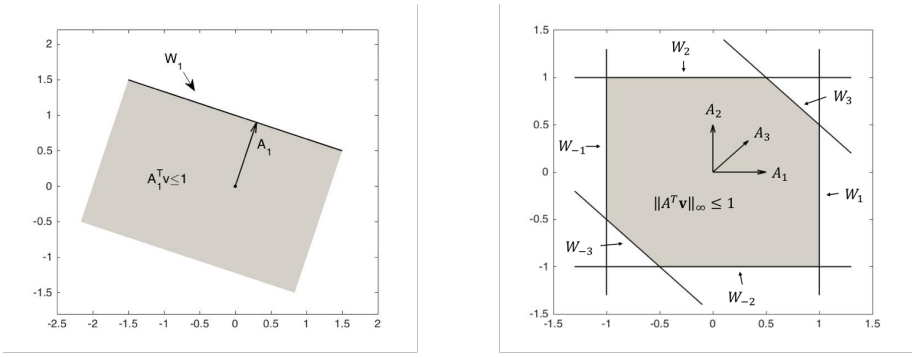
We first recall the simple SNN dynamics which we call the *primal SNN* from now on. For convenience, we set the threshold parameter $\eta = 1$. For any $t \ge 0$,

$$\mathbf{u}(t + dt) = \mathbf{u}(t) - \alpha \cdot A^\top A \cdot \mathbf{s}(t) + A^\top \mathbf{b} \cdot dt. \tag{9}$$

Now, we define the dual SNN $\mathbf{v}(t) \in \mathbb{R}^m$ as follows. Let $\mathbf{v}(0) = \mathbf{0}$ and for each $t \ge 0$, define

$$\mathbf{v}(t + dt) = \mathbf{v}(t) - \alpha \cdot A\mathbf{s}(t) + \mathbf{b} \cdot dt. \tag{10}$$

[16] In the case where the feasible region of the dual program is not bounded, it is a dual *polyhedron*. For the convenience of the presentation, we usually assume the feasible region is bounded.



(a) An example of one neuron.

(b) An example of three neuron.

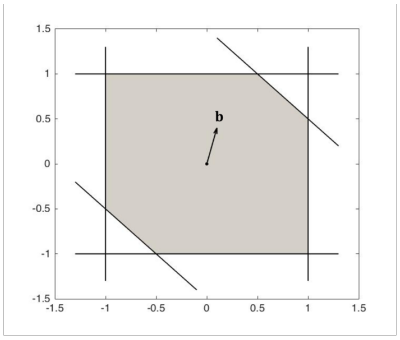
Figure 3 This is examples of the geometric interpretation of the dual program of ℓ_1 minimization problem. In (a), we have $n = 1$ where $A_1 = [\frac{1}{3}\ 1]^\top$. In this case, the gray area, i.e., the feasible region of the dual program, is unbounded. In (b), we have $n = 3$ where $A_1 = [1\ 0]^\top$, $A_2 = [0\ 1]^\top$, and $A_3 = [\frac{2}{3}\ \frac{2}{3}]^\top$. In this case, the gray area is bounded and thus called dual polytope.

Let us make some remarks about the connection between the primal and dual SNNs. First, it can be immediately seen that $\mathbf{u}(t) = A^\top \mathbf{v}(t)$ for each $t \in \mathbb{N}$ from (9) and (10). That is, given $\mathbf{v}(t)$, it is easy to get $\mathbf{u}(t)$ by multiplying $\mathbf{u}(t)$ with $A^\top$ on the left. It turns out that the other direction also holds. For each $t \in \mathbb{N}$, we have $\mathbf{v}(t) = (A^\top)^\dagger \mathbf{u}(t)$, where $(A^\top)^\dagger$ is the pseudo-inverse of $A^\top$. The reason is because the primal SNN $\mathbf{u}(t)$ lies in the column space of A. Thus, the two dynamics are in fact *isomorphic* to each other.

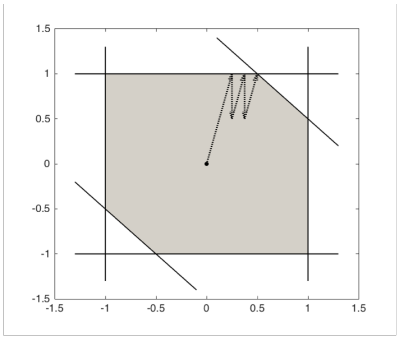
Now let us understand the dynamics of dual SNN in the dual space $\mathbb{R}^m$. At each timestep, there are two terms, i.e., the external charging $\mathbf{b} \cdot dt$ and the spiking effect $-\alpha A\mathbf{s}(t)$, that affect the dual SNN $\mathbf{v}(t)$. The external charging can be thought of as a constant force that drags that dual SNN in the direction $\mathbf{b}$. See Figure 4. This coincides with the objective function of the dual program (8) and thus the external charging can then be viewed as taking a *gradient* step towards solving (8).

Nevertheless, to solve (8), one need to make sure the solution $\mathbf{v}$ is feasible, i.e., $\mathbf{v}$ should lie in the dual polytope. Interestingly, this is exactly what the spike is doing! Recall that neuron i fires a spike if $|u_i(t)| > 1$ (recall that we set $\eta = 1$), which corresponds to $|A_i^\top \mathbf{v}(t)| > 1$ in the dual space. Thus, the spike term has the following nice geometric interpretation: if $\mathbf{v}(t)$ "exceeds" the wall W_j for some $j \in [\pm n]$, then neuron $|j|$ fires a spike and $\mathbf{v}(t)$ is "bounced back" in the normal direction of the wall W_j in the sense that $\mathbf{v}(t)$ is subtracted by $\alpha \cdot A_j$. See Figure 4 for example.

Therefore, one can view the dual SNN as performing a variant of projected gradient descent algorithm for the dual program of ℓ_1 minimization problem. Specifically, to maintain the feasibility, the vector is not projected back to the feasible region as usual, but is "bounced back" in the normal direction of the wall W_j corresponding to the violated constraint $A_j^\top \mathbf{v} \leq 1$. An advantage of this variant is that the "bounced back" operation is simply subtraction of $\alpha \cdot A_j$, which is significantly more efficient than the orthogonal projection back to the feasible region. On the other hand, note that the dynamics might not exactly converge to the optimal dual solution $\mathbf{v}^{\mathbf{OPT}}$. Intuitively, the best we can hope for is that $\mathbf{v}(t)$ will converge to a small neighboring region of $\mathbf{v}^{\mathbf{OPT}}$(assuming the spiking strength α is sufficiently small). The above intuition of viewing dual SNN as a projected gradient descent algorithm for the dual program of the ℓ_1-minimization problem will be formally proved in the full version of this paper.



(a) The effect of external charging on dual SNN.



(b) The effect of both the external charging and spikes on dual SNN.

Figure 4 This is examples of the geometric interpretation of the dual We consider the same matrix A as in Figure 3 and $\mathbf{b} = [0.1\ 0.4]^\top$. In (a), one can see that the external charging $\mathbf{b}$ points the direction that dual SNN is moving. In (b), one can see that the effect of spikes on dual SNN is a jump in the direction of the normal vector of the wall.

The primal-dual connection. So far we have informally seen that the dual SNN can be viewed as solving the dual program of the ℓ_1-minimization problem. However, this does not immediately give us a reason why the firing rate would converge to the solution of the primal program. It turns out that there is a beautiful connection between the dual SNN and firing rate through the *Karush-Kuhn-Tucker (KKT) conditions* (see Section 2.3) and perturbation theory (see Section 2.4).

We now discuss some intuitions about how the dual solution translates to the primal solution. To jump into the core idea, let us consider an ideal scenario where the dual SNN $\mathbf{v}(t)$ is already very close to the optimal dual solution $\mathbf{v}^{\mathbf{OPT}}$ for the dual program of the ℓ_1 minimization problem. Since $\mathbf{v}^{\mathbf{OPT}}$ is the optimal solution and thus it must lie on the boundary of the dual polytope. Let $\Gamma \subseteq [\pm n]$ be the set of walls that $\mathbf{v}^{\mathbf{OPT}}$ touches. That is, $j \in \Gamma$ if and only if $A_j^\top \mathbf{v}^{\mathbf{OPT}} = 1$. Now, let $\mathbf{x}^{\mathbf{OPT}}$ denote the optimal primal solution of the ℓ_1 minimization problem. Observe that by the complementary slackness in the KKT conditions, for each $i \in [n]$, we have $\mathbf{x}_i^{\mathbf{OPT}} > 0$ (resp. $\mathbf{x}_i^{\mathbf{OPT}} < 0$) if $i \in \Gamma$ (resp. $-i \in \Gamma$) and $\mathbf{x}_i^{\mathbf{OPT}} = 0$ if $i, -i \notin \Gamma$. To summary, this is saying that Γ contains the coordinates that are non-zero in the primal optimal solution $\mathbf{x}^{\mathbf{OPT}}$. See Figure 5 for an example.

With this observation, once the dual SNN $\mathbf{v}(t)$ is very close to the optimal dual solution $\mathbf{v}^{\mathbf{OPT}}$ and stays nearby, only those neurons correspond to Γ would fire spikes. In other words, the firing rate of the non-zero coordinates in the primal optimal solution $\mathbf{x}^{\mathbf{OPT}}$ will remain non-zero due to the spikes while the other coordinates will gradually go to zero.

At this point, we have seen that (i) the dual SNN can be viewed as a projected gradient descent algorithm for the dual program of ℓ_1 minimization problem and (ii) the dual solution (resp. dual SNN) and primal solution (resp. firing rate) have a natural connection through the KKT conditions. Now, let us formally state the main theorem of this section about simple SNN solving ℓ_1 minimization problem.

▶ **Theorem 7.** *Given $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ where all the row of A has unit norm. Let $\gamma(A)$ be the niceness parameter*[17] *of A. Suppose $\gamma(A) > 0$ and there exists a solution to $A\mathbf{x} = \mathbf{b}$. There exists a polynomial $\alpha(\cdot)$ such that for any $t \geq 0$, let $\mathbf{x}(t)$ be the firing rate*

[17] The niceness parameter is formally defined in Definition 4 of the full version of this paper.

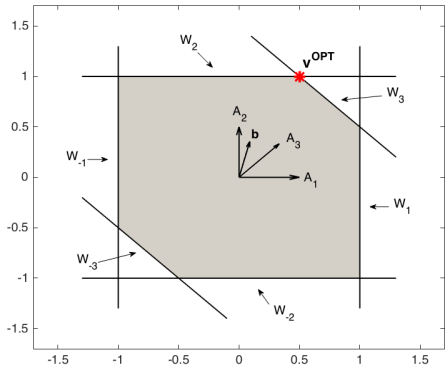


Figure 5 This is an example based on Figure 3 and Figure 4. In this example, $A_1 = [1\ 0]^\top$, $A_2 = [0\ 1]^\top$, $A_3 = [\frac{2}{3}\ \frac{2}{3}]^\top$, and $\mathbf{b} = [0.1\ 0.4]^\top$. The optimal dual solution is $\mathbf{v^{OPT}} = [\frac{1}{2}\ 1]^\top$ as shown in the figure. Thus, by the above definition we have $\Gamma = \{2, 3\}$. By the KKT conditions, we then know that only the 2^{nd} and 3^{rd} coordinate of the optimal primal solution are non-zero. Indeed, the optimal primal solution is $\mathbf{x^{OPT}} = [0\ \frac{3}{10}\ \frac{3}{20}]^\top$.

of the simple SNN with $C = A^\top A$, $\mathbf{I} = A^\top \mathbf{b}$, $\eta = 1$, $0 < \alpha \leq \alpha(\frac{\gamma(A)}{n \cdot \lambda_{\max}})$. *Let* $\mathbf{OPT}^{\ell_1}$ *be the optimal value of the* ℓ_1 *minimization problem. For any* $\epsilon > 0$, *when* $t \geq \Omega(\frac{m^2 \cdot n \cdot \|\mathbf{b}\|_2^2}{\epsilon^2 \cdot \lambda_{\min} \cdot \mathbf{OPT}^{\ell_1}})$, *then* $\mathbf{x}(t)$ *is an* ϵ*-approximate solution to the* ℓ_1 *minimization problem for* $(A, \mathbf{b})$.

Two remarks on the statement of Theorem 7. First, we consider the *continuous SNN* instead of the discrete SNN, which is of interest for simulation on classical computer. In discrete SNN, the *step size* is some non-negligible $\Delta t > 0$ instead of dt. The main reason for considering continuous SNN is that this significantly simplify the proof by avoiding a huge amount of nasty calculations. We suspect that the proof idea would hold for discrete SNN with discretization parameter $\Delta t \leq \Delta t(\frac{\gamma(A)}{n \cdot \lambda_{\max}})$ for some polynomial $\Delta t(\cdot)$. Second, the parameters in Theorem 7 have not been optimized and we believe all the dependencies can be improved. Since the parameters highly affect the efficiency of SNN as an algorithm for ℓ_1 minimization problem, we pose it as an interesting open problem to study what are the best dependencies one can get.

3.2 Overview of the proof for Theorem 7

The proof for Theorem 7 consists of two main steps as mentioned in the previous subsection. The first step argues that the dual SNN $\mathbf{v}(t)$ would converge to the neighborhood of the optimal dual solution $\mathbf{v^{OPT}}$. The second step is connecting the dual solution (i.e., the dual SNN) to the primal solution (i.e., the firing rate). In the subsection, we sketch the proof for Theorem 7 while some lemmas and definitions might not appear in this conference version and can found in the full version of this paper.

In the first step, we try to identify a *potential function*[18] that captures how close is $\mathbf{v}(t)$ to the optimal dual solution $\mathbf{v^{OPT}}$. It turns out that this is not an easy task since the effect of spikes makes the behavior of dual SNN very non-monotone. We conquer the difficulty via a technique that we call *ideal coupling* (see Definition 6 and Figure 7 in the full version).

[18] Potential function is widely used in the analysis of many gradient-descent based algorithm. The difficulty lies in the search of a good potential function for the algorithm.

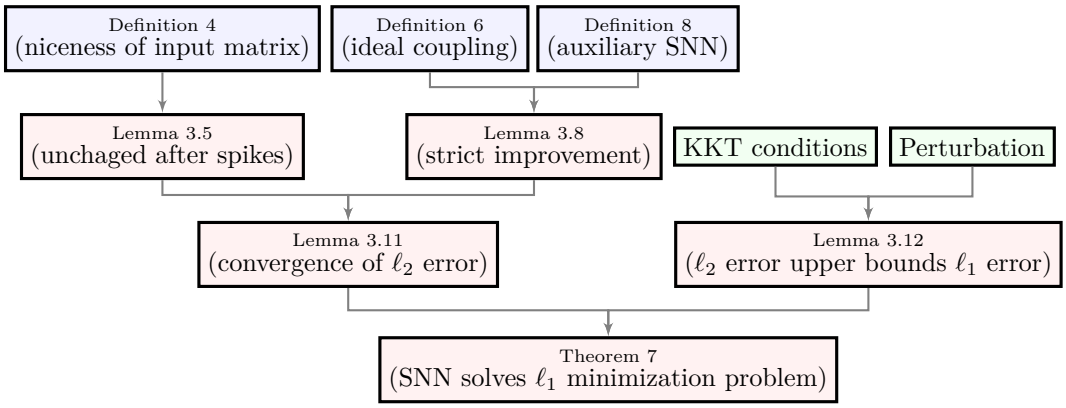


Figure 6 Overview of the proof for Theorem 7. The missing lemmas and definitions can be found in the full version of this paper.

The idea is associating the dual SNN $\mathbf{v}(t)$ with an *ideal SNN* $\mathbf{v}^{\text{ideal}}(t)$ for every $t \geq 0$ such that the ideal SNN would have *smoother* behavior comparing to the spiking phenomenon in the dual SNN. We will formally define the ideal SNN in Section 3.4 of the full version. There are two advantages of using ideal SNN instead of handling dual SNN directly: (i) Ideal SNN is smoother than dual SNN in the sense that it would not change after spikes (see Lemma 3.5 in the full version). Further, by introducing some auxiliary processes (i.e., the auxiliary SNNs defined in Definition 8 in the full version), we are able to identify a potential function that is strictly improving at any moment and measures how well the dual SNN has been solving the ℓ_1 minimization problem (see Lemma 3.8 in the full version). (ii) ideal SNN is naturally associated with an *ideal solution* (defined in Definition 7 in the full version) which is easier to analyze than the firing rate. Using these good properties of ideal SNN, we can prove in Lemma 3.11 (in the full version) that the ℓ_2 residual error of the ideal solution will converge to 0.

After we are able to show the convergence of the ℓ_2 residual error in Lemma 3.11 (in the full version), we move to the second step where the goal is showing that the ℓ_1 norm of the solution is also small. We look at the KKT conditions of the ℓ_1 minimization problem and observe that the primal and dual solutions of SNN satisfy the KKT conditions of a *perturbed* program of the ℓ_1 minimization problem. Finally, combine tools from perturbation theory, we can upper bound the ℓ_1 error of the ideal solution by its ℓ_2 residual error in Lemma 3.12 (in the full version).

Theorem 7 then follows from Lemma 3.11 and Lemma 3.12 (in the full version) with some special cares on how to transform everything for ideal solution to the firing rate. See Figure 6 for an overall structure of the proof for Theorem 7.

The full proof for Theorem 7 and other technical details are all provided in the full version of this paper.

References

1 Moshe Abeles. *Corticonics: Neural circuits of the cerebral cortex*. Cambridge University Press, 1991.

2 Christina Allen and Charles F Stevens. An evaluation of causes for unreliability of synaptic transmission. *Proceedings of the National Academy of Sciences*, 91(22):10380–10383, 1994.

3 Arunava Banerjee. Learning Precise Spike Train–to–Spike Train Transformations in Multilayer Feedforward Neuronal Networks. *Neural computation*, 28(5):826–848, 2016.

4 David G Barrett, Sophie Denève, and Christian K Machens. Firing rate predictions in optimal balanced networks. In *Advances in Neural Information Processing Systems*, pages 1538–1546, 2013.

5 Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

6 Yoshua Bengio, Thomas Mesnard, Asja Fischer, Saizheng Zhang, and Yuhuai Wu. STDP-compatible approximation of backpropagation in an energy-based model. *Neural computation*, 29(3):555–577, 2017.

7 William Bialek, Fred Rieke, RR De Ruyter Van Steveninck, and David Warland. Reading a neural code. *Science*, 252(5014):1854–1857, 1991.

8 Jonathan Binas, Giacomo Indiveri, and Michael Pfeiffer. Spiking Analog VLSI Neuron Assemblies as Constraint Satisfaction Problem Solvers. *arXiv preprint arXiv:1511.00540*, 2015.

9 Anmol Biswas, Sidharth Prasad, Sandip Lashkare, and Udayan Ganguly. A simple and efficient SNN and its performance & robustness evaluation method to enable hardware implementation. *arXiv preprint arXiv:1612.02233*, 2016.

10 Vincenzo Bonifaci, Kurt Mehlhorn, and Girish Varma. Physarum can compute shortest paths. *Journal of Theoretical Biology*, 309:121–133, 2012.

11 J Frédéric Bonnans and Alexander Shapiro. Optimization problems with perturbations: A guided tour. *SIAM review*, 40(2):228–264, 1998.

12 Olaf Booij and Hieu tat Nguyen. A gradient descent rule for spiking neurons emitting multiple spikes. *Information Processing Letters*, 95(6):552–558, 2005.

13 Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

14 Nicolas Brunel and Peter E Latham. Firing rate of the noisy quadratic integrate-and-fire neuron. *Neural Computation*, 15(10):2281–2306, 2003.

15 Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol*, 7(11):e1002211, 2011.

16 Bernard Chazelle. Natural algorithms. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 422–431. Society for Industrial and Applied Mathematics, 2009.

17 Bernard Chazelle. Natural algorithms and influence systems. *Communications of the ACM*, 55(12):101–110, 2012.

18 Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

19 Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

20 A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292–303, 2008.

21 Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466, 1961.

22 Nicolas Fourcaud-Trocmé, David Hansel, Carl Van Vreeswijk, and Nicolas Brunel. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *Journal of Neuroscience*, 23(37):11628–11640, 2003.

23 Wulfram Gerstner. Time structure of the activity in neural network models. *Physical review E*, 51(1):738, 1995.

24 Tim Gollisch and Markus Meister. Rapid neural coding in the retina with relative spike latencies. *science*, 319(5866):1108–1111, 2008.

25 Walter Heiligenberg. *Neural nets in electric fish*. MIT press Cambridge, MA, 1991.

26 James L Hindmarsh and RM Rose. A model of neuronal bursting using three coupled first order differential equations. *Proc. R. Soc. Lond. B*, 221(1222):87–102, 1984.

27 Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

28 John J Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535):33, 1995.

29 Eugene M Izhikevich et al. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

30 Zeno Jonke, Stefan Habenschuss, and Wolfgang Maass. A theoretical basis for efficient computations with noisy spiking neurons. *arXiv preprint arXiv:1412.5862*, 2014.

31 Zeno Jonke, Stefan Habenschuss, and Wolfgang Maass. Solving constraint satisfaction problems with networks of spiking neurons. *Frontiers in neuroscience*, 10, 2016.

32 Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, and Timothée Masquelier. Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. *Neurocomputing*, 205:382–392, 2016.

33 Werner M Kistler, Wulfram Gerstner, and J Leo van Hemmen. Reduction of the Hodgkin-Huxley equations to a single-variable threshold model. *Neural computation*, 9(5):1015–1045, 1997.

34 Nobuyuki Kuwabara and Nobuo Suga. Delay lines and amplitude selectivity are created in subthalamic auditory nuclei: the brachium of the inferior colliculus of the mustached bat. *Journal of neurophysiology*, 69(5):1713–1724, 1993.

35 Louis Lapicque. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Gen*, 9(1):620–635, 1907.

36 Tsung-Han Lin and Ping Tak Peter Tang. Dictionary Learning by Dynamical Neural Networks. *arXiv preprint arXiv:1805.08952*, 2018.

37 Adi Livnat and Christos Papadimitriou. Sex as an algorithm: the theory of evolution under the lens of computation. *Communications of the ACM*, 59(11):84–93, 2016.

38 Adi Livnat, Christos Papadimitriou, Aviad Rubinstein, Gregory Valiant, and Andrew Wan. Satisfiability and evolution. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 524–530. IEEE, 2014.

39 Nancy Lynch and Cameron Musco. A Basic Compositional Model for Spiking Neural Networks. *arXiv preprint arXiv:1808.03884*, 2018.

40 Nancy Lynch, Cameron Musco, and Merav Parter. Spiking Neural Networks: An Algorithmic Perspective. In *Workshop on Biological Distributed Algorithms (BDA), July 28th, 2017, Washington DC, USA*, 2017.

41 Nancy A. Lynch, Cameron Musco, and Merav Parter. Computational Tradeoffs in Biological Neural Networks: Self-Stabilizing Winner-Take-All Networks. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 15:1–15:44, 2017. doi:10.4230/LIPIcs.ITCS.2017.15.

42 Nancy A. Lynch, Cameron Musco, and Merav Parter. Neuro-RAM Unit with Applications to Similarity Testing and Compression in Spiking Neural Networks. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 33:1–33:16, 2017. doi:10.4230/LIPIcs.DISC.2017.33.

43 Wolfgang Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural computation*, 8(1):1–40, 1996.

44 Wolfgang Maass. Fast sigmoidal networks via spiking neurons. *Neural Computation*, 9(2):279–304, 1997.

45 Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

46 Wolfgang Maass. Computing with spiking neurons. *Pulsed neural networks*, 85, 1999.

47 Wolfgang Maass. To Spike or Not to Spike: That Is the Question. *Proceedings of the IEEE*, 103(12):2219–2224, 2015.

48 Wolfgang Maass and Christopher M Bishop. *Pulsed neural networks*. MIT press, 2001.

49 Catherine Morris and Harold Lecar. Voltage oscillations in the barnacle giant muscle fiber. *Biophysical journal*, 35(1):193–213, 1981.

50 Hesham Mostafa, Lorenz K Müller, and Giacomo Indiveri. An event-based architecture for solving constraint satisfaction problems. *Nature communications*, 6, 2015.

51 Toshiyuki Nakagaki, Hiroyasu Yamada, and Ágota Tóth. Intelligence: Maze-solving by an amoeboid organism. *Nature*, 407(6803):470–470, 2000.

52 Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.

53 Hélene Paugam-Moisy and Sander Bohte. Computing with spiking neuron networks. In *Handbook of natural computing*, pages 335–376. Springer, 2012.

54 Fred Rieke and David Warland. *Spikes: exploring the neural code*. MIT press, 1999.

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

56 Rufin Van Rullen and Simon J Thorpe. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural computation*, 13(6):1255–1283, 2001.

57 Michael N Shadlen and William T Newsome. Noise, neural codes and cortical organization. *Current opinion in neurobiology*, 4(4):569–579, 1994.

58 Samuel Shapero, Christopher Rozell, and Paul Hasler. Configurable hardware integrate and fire neurons for sparse approximation. *Neural Networks*, 45:134–143, 2013.

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

60 Sumit Bam Shrestha and Qing Song. Robust learning in SpikeProp. *Neural Networks*, 86:54–68, 2017.

61 Richard B Stein. A theoretical analysis of neuronal variability. *Biophysical Journal*, 5(2):173, 1965.

62 Ping Tak Peter Tang. Convergence of LCA Flows to (C) LASSO Solutions. *arXiv preprint arXiv:1603.01644*, 2016.

63 Ping Tak Peter Tang, Tsung-Han Lin, and Mike Davies. Sparse Coding by Spiking Neural Networks: Convergence Theory and Computational Results. *arXiv preprint arXiv:1705.05475*, 2017.

64 Wondimu Teka, Toma M Marinov, and Fidel Santamaria. Neuronal spike timing adaptation described with a fractional leaky integrate-and-fire model. *PLoS computational biology*, 10(3):e1003526, 2014.

65 Atsushi Tero, Ryo Kobayashi, and Toshiyuki Nakagaki. A mathematical model for adaptive transport network in path finding by true slime mold. *Journal of theoretical biology*, 244(4):553–564, 2007.

66 Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-based strategies for rapid processing. *Neural networks*, 14(6-7):715–725, 2001.

67 Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *nature*, 381(6582):520, 1996.

68 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. *PLoS computational biology*, 7(10):e1002250, 2011.