Local Information with Feedback Perturbation Suffices for
Dictionary Learning in Neural Circuits

Tsung-Han Lin
Intel Labs
Santa Clara, CA, USA
tzung-han.lin@intel.com

Abstract

While the sparse coding principle can successfully model information processing in sensory neural systems, it remains unclear how learning can be accomplished under neural architectural constraints. Feasible learning rules must rely solely on synaptically local information in order to be implemented on spatially distributed neurons. We describe a neural network with spiking neurons that can address the aforementioned fundamental challenge and solve the $\ell_1$-minimizing dictionary learning problem, representing the first model able to do so. Our major innovation is to introduce feedback synapses to create a pathway to turn the seemingly non-local information into local ones. The resulting network encodes the error signal needed for learning as the change of network steady states caused by feedback, and operates akin to the classical stochastic gradient descent method.

1 Introduction

A spiking neural network (SNN) is a computational model with simple neurons as the basic processing units. Different from artificial neural networks, SNNs incorporate the time dimension into computations. The network of neurons operates according to a global reference clock; at a time instance, one or more neurons may send out a 1-bit impulse, the spike, to neighbors through directed connectivities, known as synapses. Neurons form a dynamical system with local state variables and rules that determine when a neuron transmits a spike. The spike rate of a neuron can encode its activation value, borrowing the terminology from artificial neural networks.

SNNs can exploit the temporal ordering of spikes to obtain high computational efficiency, despite that encoding real values as spike rates may appear quite inefficient comparing to the compact binary representations. Consider, for example, a set of competing neurons recurrently connected with inhibitory synapses in Figure 1(a). The winner neuron that has the largest external input will fire at the earliest time, and immediately inhibit the activities of other neurons. This inhibition happens with only a single one-to-many spike communication, in contrast to the all-to-all state exchange and comparison required when neurons only maintain graded activation values. Using the above principle, one can show that a SNN can be configured to efficiently solve the well-known $\ell_1$-minimizing sparse approximation problem [17, 18], which is to determine a sparse subset of features from a feature dictionary to represent a given input, and the features can be viewed as competing neurons that seek to form the best fit of the input data [16].

In this work, we further show that the related dictionary learning problem can be solved in a SNN as well. Dictionary learning was first proposed to model mammalian visual cortex [15], and later found numerous applications in image processing and machine learning [11]. Despite its popularity, it remains unclear how the problem can be solved in a neural architecture. None of the existing learning algorithms are synaptically local: the adaptation of synaptic weights relies on the receptive field information of other neurons, making it impossible to be implemented in a spatially distributed network. As a result, many researchers turn to other less straightforward objective function formulations (e.g., minimizing over long-term average neuron activities [23], or maximizing input-output similarity [9]), or are forced to take approximate gradient directions at the cost of suboptimal results (e.g., simplifying the learning rules to be only Hebbian [8, 19]).
(a) Laterally connected network

(b) Dictionary learning network

Figure 1: The two network topology for sparse coding and dictionary learning. Our main focus is the network in (b). The firing thresholds of the input layer and bias neurons are set to 1.

We solve the dictionary learning problem by introducing feedback synapses. We show that the feedback connections can cause the network steady states to change by an amount identical to the error signal needed for learning, provided that the network synaptic weights satisfy a weight consistency condition. Built on this observation, we develop learning mechanisms that closely resemble the classical stochastic gradient descent, and can perform dictionary learning from a spiking network with randomly initialized synaptic weights.

2 Integrate-and-Fire Neuron Model

We first consider a network of $N$ simple integrate-and-fire neurons. Each neuron-$i$, for $i = 1, 2, \ldots, N$, has two internal state variables, the soma current $\mu_i(t)$ and the membrane potential $v_i(t)$, that govern its dynamics. The soma current is determined by two inputs: The first is a constant current $b_i$; the second are the spike trains of all the neighbors neuron-$j$s to which neuron-$i$ is connected. Each spike train is of the form $\sigma_j(t) = \sum_k \delta(t - t_{j,k})$ where $t_{j,k}$ is the time of the $k$-th spike of neuron-$j$ and $\delta(t)$ is the Dirac delta function. The soma current $\mu_i(t)$ is the sum of $b_i$ and the filtered spike trains from its neighbors thus:

$$\mu_i(t) = b_i + \sum_{j \neq i} w_{ij}(\alpha * \sigma_j)(t)$$

where $w_{ij}$ is the synaptic weight from neuron-$j$ to $i$, and $\alpha(t) = \tau_s^{-1} H(t) e^{-t/\tau_s}$ is the filter kernel parameterized by the synaptic decay time constant $\tau_s$; $H(t)$ is the Heaviside function that is 1 when $t \geq 0$ and 0 elsewhere. Note that in a neural architecture, synaptic weights are stored, and hence only available, at the destination neuron. This property is referred to as synaptically local, and constitutes the major challenge for dictionary learning in SNNs.

The soma current is converted to the output spiking activities through the dynamics of membrane potential. This membrane potential is a simple linear integration of the soma current before it reaches the firing threshold:

$$v_i(t) = \int_{t_{i,k}}^t \mu_i(s)ds$$

A spike is generated when the membrane potential exceeds its firing threshold $\theta_i$; at this time, the neuron also immediately resets $v_i$ to 0.

The network of neurons forms a dynamical system where the neurons interact through spikes. An important quantity $\Delta_i$, called the imbalance function, is useful in characterizing the steady states of the system, defined for $t > 0$ as follows,

$$\Delta_i(t) \overset{\text{def}}{=} u_i(t) - \theta_i a_i(t)$$

where $u(t)$ and $a(t)$ are the average soma current and average spike rate, respectively,

$$u_i(t) \overset{\text{def}}{=} \frac{1}{t} \int_0^t \mu_i(s)ds, \quad a_i(t) \overset{\text{def}}{=} \frac{1}{t} \int_0^t \sigma_i(s)ds$$

The imbalance function measures the difference between the average amount of charges accumulated to the membrane potential, equal to $u_i(t)$, and the average amount of charges released through spiking, equal to $\theta_i a_i(t)$.
If the average current converges to a fixed point, one can show that as $t \to \infty$, the imbalance converges towards satisfying the following equilibrium condition \[\text{[18]},\]
\[
\begin{aligned}
\lim_{t \to \infty} \Delta_i(t) &= 0, & \text{if } \lim_{t \to \infty} a_i(t) &> 0 \\
\lim_{t \to \infty} \Delta_i(t) &\leq 0, & \text{if } \lim_{t \to \infty} a_i(t) &= 0
\end{aligned}
\] (2.5)

This result simply suggests that for neurons that have nonzero spike rates at equilibrium, their average outgoing charges must equal the average incoming charges, meaning the imbalance must be 0. On the contrary, if a neuron stops spiking at equilibrium, then the imbalance may either be zero, if it has zero net incoming charges, or a negative value, if it receives more inhibition than excitation. We note that the convergence property of the dynamical system deserves a rigorous treatment (e.g., see \[\text{[17], [18]},\]) although it is not the main focus of this work.

3 Nonnegative Sparse Coding

Solving the sparse coding problem under a given dictionary constitutes an important part of our learning scheme. In this section, we revisit prior results on solving this problem in a SNN \[\text{[17], [18]},\] and we will focus on using nonnegative dictionaries.

Consider the network topology in Figure 1(a). Each neuron-\(i\) receives an input current \(b_i\), and has incoming synapses with weight \(w_{ij}\) from the other \((N-1)\) neurons. Suppose that the synapses are all inhibitory, that is, \(w_{ij} \leq 0\), and hence none of the neurons can spike arbitrarily fast. Using \(\text{(2.5)}\), the equilibrium spike rates \(a_i^* = \lim_{t \to \infty} a_i(t)\) must satisfy
\[
\begin{aligned}
&b_i + \sum_{j \neq i} w_{ij} a_j^* - \theta_i a_i^* = 0, & \text{if } a_i^* > 0 \\
&b_i + \sum_{j \neq i} w_{ij} a_j^* \leq 0, & \text{if } a_i^* = 0
\end{aligned}
\] (3.1)

The above result makes use of the property that the average current will converge to \(\lim_{t \to \infty} u_i(t) = b_i + \sum_{j \neq i} w_{ij} a_j^*\).

The steady-state condition, \(\text{(3.1)}\), has connections to the nonnegative sparse coding problem,
\[
\mathbf{a}^* = \arg \min_{\mathbf{a} \geq 0} \frac{1}{2} \|\mathbf{x} - \mathbf{D} \mathbf{a}\|^2_2 + \lambda \sum_j \left( \sum_i d_{ij}^2 \right) a_j
\] (3.2)

where \(\mathbf{x} \in \mathbb{R}^M\) is a data sample, \(\mathbf{D} \in \mathbb{R}^{M \times N}\) is a dictionary, and \(\lambda \geq 0\) is the sparse regularization parameter. To see the connection, let \(\mathbf{G} = \mathbf{D}^T \mathbf{D}\) and \(g_{ij}\) be its \((i,j)\)-th entry. The necessary and sufficient optimality condition for \(\text{(3.2)}\) is
\[
\begin{aligned}
&\sum_k d_{ki} x_k - \lambda g_{ii} - \sum_{j \neq i} g_{ij} a_j^* - g_{ii} a_i^* = 0, & \text{if } a_i^* > 0 \\
&\sum_k d_{ki} x_k - \lambda g_{ii} - \sum_{j \neq i} g_{ij} a_j^* \leq 0, & \text{if } a_i^* = 0
\end{aligned}
\] (3.3)

Note the similarity between \(\text{(3.1)}\) and \(\text{(3.3)}\). The correspondence can be established by setting
\[
\begin{aligned}
&b_i = \sum_k d_{ki} x_k - \lambda g_{ii}, \quad w_{ij} = -g_{ij}, \quad \theta_i = g_{ii}
\end{aligned}
\] (3.4)

Indeed, previous work has established that a spiking network configured as above will converge to an equilibrium spike rate identical to the solution of \(\text{(3.2)}\). Note that the dictionary is not encoded explicitly in the network. The input current is configured according to the dictionary projection of the input data, and the synaptic weights represent the correlations between columns of the dictionary.

4 Online Dictionary Learning

We are interested in learning a nonnegative dictionary from \(P\) nonnegative training data samples, \(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_P\). The dictionary learning problem is commonly formulated as,
\[
\min_{\mathbf{D} \geq 0, \|\mathbf{a}_i\|^2_2 = 1, \mathbf{a} \geq 0} \frac{1}{P} \sum_{i=1}^P L (\mathbf{x}_i, \mathbf{D}, \mathbf{a}_i), \quad L (\mathbf{x}, \mathbf{D}, \mathbf{a}) = \frac{1}{2} \|\mathbf{x} - \mathbf{D} \mathbf{a}\|^2_2 + \lambda \|\mathbf{a}\|_1
\] (4.1)
where \(a_i\) is the sparse representation of data \(x_i\). The number of columns, or *atoms*, in the dictionary, \(N\), is a predetermined hyper-parameter. This optimization problem seeks the best performing dictionary for all data samples, minimizing the sum of all sparse coding losses.

### 4.1 A Two-Layer Network for Dictionary Learning

Consider the network topology in Figure 1(b) that consists of two layers of neurons, an input layer of \(M\) neurons at the bottom and a sparse-code layer of \(N\) neurons on top. There are four groups of synaptic weights: the excitatory feedforward and feedback synapses, \(F \in \mathbb{R}^{N \times M}\), \(\gamma B \in \mathbb{R}^{M \times N}\) where \(\gamma\) is a scalar in \([0, 1]\), and \(F, B \geq 0\); the inhibitory lateral and bias synapses, \(W \in \mathbb{R}^{N \times N}\), \(L \in \mathbb{R}^{N \times 1}\), and \(W, L \leq 0\).

In the case of \(\gamma = 0\), Figure 1(b) is an instantiation of Figure 1(a), where the constant current inputs are replaced by spike trains of identical averages. To see this, note that the feedback synapses are removed in this setting, and the input and bias neurons will spike at a constant rate \(x_1, x_2, \ldots, x_M\) and \(\lambda\), as they are only driven by constant external inputs. We can similarly establish the correspondence between the equilibrium at the sparse-code layer, as in (3.1), and the optimality condition for sparse coding in (3.3), by configuring the network as follows,

\[
f_{ij} = d_{ji}, \quad w_{ij} = -g_{ij}, \quad \theta_i = g_{ii}, \quad l_i = -\theta_i
\]

\(\theta_i\) being the firing threshold of neuron-\(i\) in the sparse-code layer. From (4.2), we can see that dictionary learning in this network means adapting the feedforward weights towards the optimal dictionary, and the lateral weights towards the correlations between the optimal dictionary atoms. In addition, learning proceeds in an online manner, where data samples are given sequentially by swapping inputs, and the dictionary is updated as soon as a new data sample is available.

We derive learning mechanisms that resemble the classical online stochastic gradient descent [11, Sec 5.5], consisting of two iterative steps. The first step computes the optimal sparse code with respect to the current dictionary, and the second step updates the dictionary by estimating the gradient from a single training sample, giving the following update sequence

\[
D_{i+1} = \Pi_C[D_i - \eta \nabla_D L(x_i, D_i; \mathbf{a}_i^*)]
\]

\[
= \Pi_C[D_i - \eta(D_i \mathbf{a}_i^* - x_i) \mathbf{a}_i^{*T}]
\]

where the projection operator \(\Pi_C\) projects to the positive quadrant and renormalizes each atom in the updated dictionary, and \(\eta\) is the learning rate.

We operate the spiking network with two stages to mimic the above two iterative steps. In the first stage, called the feedforward stage, we set \(\gamma = 0\) and feed the training sample \(x_i\) to the input layer neurons. From the discussions above, the optimal sparse code can be found as the equilibrium spike rates at the sparse-code layer. The main challenges lie in the second stage where the dictionary needs to be updated using synaptically local mechanisms, whereas the information needed appears to be non-local for the following two reasons: 1) **Reconstruction error cannot be locally computed at the sparse-code layer.** The gradient consists of a reconstruction error term, which is crucial to determining the best way to adapt the dictionary. Unfortunately, computing \(\mathbf{D}\mathbf{a}_i^*\) requires the full knowledge of \(\mathbf{D}\), but only one column of the dictionary is local to a sparse-code neuron. 2) **Atom correlations are non-local to compute.** The lateral synaptic weights should be updated to capture the new correlations between the updated atoms. Again, computing a correlation requires knowledge of two atoms, while only one of them is accessible by a sparse-code neuron. In the next section, we show how feedback synapses can be exploited to address these two fundamental challenges.

### 4.2 Synaptically Local Learning

**Reconstruction with Feedback Synapses.** In the second stage of learning, called feedback stage, we set \(\gamma\) to a nonzero value to engage the feedback synapses, moving the network towards a new steady state. Interestingly, there exists a condition that if satisfied, engaging the feedback synapses will only perturb the equilibrium spike rates at the input layer, while leaving the sparse-code layer untouched. We call this condition...
feedback consistency,

\[ \text{FB} = \mathbf{H}, \quad \text{where} \quad \mathbf{H} = \begin{bmatrix} \theta_1 & -w_{12} & \cdots & -w_{1N} \\ -w_{21} & \theta_2 & \cdots & -w_{2N} \\ \vdots & \ddots & \ddots & \vdots \\ -w_{N1} & -w_{N2} & \cdots & \theta_N \end{bmatrix} \] (4.4)

Note that \( \mathbf{H} \) is composed of the lateral weights \( w_{ij} \) and firing thresholds \( \theta_i \).

To see this, let \( y \in \mathbb{R}^M \), \( z \in \mathbb{R}^N \) be the equilibrium spike rates at the input and sparse-code layer, respectively. The equilibrium spike rates at the input layer can be easily derived. Given that the input neurons do not receive any inhibition, their imbalance functions must be zero at equilibrium, and hence their spike rates are,

\[ y^{(1)} = x \]
\[ y^{(2)} = (1 - \gamma)x + \gamma \mathbf{B}z^{(2)} \] (4.5)

with superscripts denoting the particular learning stage that the equilibrium spike rates belong to.

For the sparse-code layer, note that the equilibrium spike rate must satisfy the equilibrium condition in (2.5). This allows us to examine the relationships between \( z^{(1)} \) and \( z^{(2)} \). Let \( \mathbf{e} \in \mathbb{R}^N \) be the imbalance of the sparse-code layer neurons at equilibrium, \( \mathbf{e} = \lim_{t \to \infty} \Delta_i(t) \), we can write the imbalance during the feedforward and feedback stage at equilibrium as

\[ \mathbf{e}^{(1)} = \mathbf{F}x + \lambda \mathbf{L} - \mathbf{H}z^{(1)} \] (4.6)
\[ \mathbf{e}^{(2)} = \mathbf{F}y^{(2)} + (1 - \gamma)\lambda \mathbf{L} - \mathbf{H}z^{(2)} \] (4.7)

Substituting (4.5) into (4.7),

\[ \mathbf{e}^{(2)} = (1 - \gamma)\mathbf{F}x + \gamma \mathbf{F} \mathbf{B}z^{(2)} + (1 - \gamma)\lambda \mathbf{L} - \mathbf{H}z^{(2)} \] (4.8)

Now, suppose that the feedback weights satisfy feedback consistency, \( \text{FB} = \mathbf{H} \), \( \mathbf{e}^{(2)} \) can be further reduced,

\[ \mathbf{e}^{(2)} = (1 - \gamma) \left( \mathbf{F}x + \lambda \mathbf{L} - \mathbf{H}z^{(2)} \right) \] (4.9)

Note the similarity between (4.6) and (4.9), and \( 1 - \gamma > 0 \). This suggests that a feasible \( z^{(1)} \) that satisfies (2.5) must also be a feasible \( z^{(2)} \), and vice versa. In other words, if the feedforward-only network possesses a unique equilibrium spike rate, then the sparse-code layer spike rates must remain unaltered between the two learning stages, \( z^{(2)} = z^{(1)} \).

With this result, we turn our attention to the amount of spike rate changes at the input layer in a feedback-consistent network.

\[ y^{(2)} - y^{(1)} = \gamma (\mathbf{B}z^{(1)} - x) \] (4.10)

As the sparse code computed in the feedforward stage is preserved in the feedback stage, the change amounts to a reconstruction error, in that the reconstruction is formed by the feedback weights \( \mathbf{B} \) as the dictionary. Suppose for now that the feedforward and feedback weights are symmetric (except for a scalar factor \( \gamma \)), consistent, and equal to an underlying dictionary \( \mathbf{D} \), that is, \( \mathbf{B} = \mathbf{F}^T = \mathbf{D} \) and \( \mathbf{H} = \mathbf{D}^T \mathbf{D} \), which is also the ideal situation that learning should achieve. The reconstruction errors needed in the gradient calculations become locally available as the change of input layer spike rates. This leads to the following synaptically local learning rules that update the weights along the desired gradient direction in (4.3),

\[ f_{ij} \leftarrow f_{ij} + \eta_f z_i^{(2)} (y_j^{(1)} - y_j^{(2)}) - \lambda_f f_{ij} \]
\[ b_{ij} \leftarrow b_{ij} + \eta_b (y_i^{(1)} - y_i^{(2)}) z_j^{(2)} - \lambda_b b_{ij} \] (4.11)

\( \eta_f \) and \( \eta_b \) being the learning rates. Note that a weight decay term is included at the end to prevent the weights from growing too large, with \( \lambda_f \) and \( \lambda_b \) being the regularization coefficients. This is where our algorithm departs from the classical stochastic gradient descent, as renormalizing the atoms in the feedback weights is non-local. In addition, we truncate the weight values when they go below zero to ensure their nonnegativity.
In the case of asymmetric weights, we can still adopt the learning rules above. Initially, the weight updates may not be able to improve the dictionary, given that the reconstruction in the feedback stage is formed using a dictionary quite different from the encoding dictionary $F$. However, over many updates, the weights will gradually become symmetric, since the learning rules adjust both feedforward and feedback weights in the same direction, and their initial differences will diminish with the decay term. When the two weights become sufficiently aligned, the learning rules will likely find a descending direction, despite not the steepest, towards the optimal dictionary. Perfect symmetry is not necessarily for learning to work.

**Maintaining Feedback Consistency.** Feedback consistency is the key property behind the rationale of the learning mechanism above. Suppose that a network is initialized to be feedback-consistent, after an update to its feedforward and feedback weights, one must adjust the lateral weights and firing thresholds accordingly to restore the consistency. Unfortunately, direct computations, $H \leftarrow FB$, are not synaptically local. The sparse-code layer neurons, who can modify $H$, do not have access to the feedback weights, which are local to the input layer neurons.

To avoid non-local computations, we instead have the sparse-code layer neurons minimize the following inconsistency loss $L_c$ given $P_s$ training samples, again using stochastic gradient descent

$$\min_H \frac{1}{P_s} \sum_{i=1}^{P_s} L_c(H, z_i^{(2)}) \quad , \quad L_c(H, z) = \frac{1}{2} \| (H - FB)z \|^2$$

The key observation is that the inconsistency loss can be measured from the difference in equilibrium spike rates of sparse-code neurons between the two learning stages. Their relationship can be easily shown by reorganizing (4.6) and (4.8),

$$\gamma(H - FB)z^{(2)} = -e^{(2)} + (1 - \gamma)e^{(1)} - (1 - \gamma)H \left( z^{(2)} - z^{(1)} \right)$$  \hspace{1cm} (4.13)

We can then derive the gradient to minimize $L_c$

$$\nabla_H L_c(H, z_i^{(2)}) = (H - FB)z^{(2)} \left( z^{(2)} \right)^T$$

$$= \frac{1}{\gamma} \left( -e^{(2)} + (1 - \gamma)e^{(1)} - (1 - \gamma)H \left( z^{(2)} - z^{(1)} \right) \right) \left( z^{(2)} \right)^T$$ \hspace{1cm} (4.14)

Note that the gradient above can be computed with synaptically local information. Suppose for now that $F$ and $B$ are fixed during which the sub-problem is being solved, we then can use the rule $H \leftarrow H - \eta_h \nabla_H L_c - \lambda_h H$ to update both lateral weights and firing thresholds. With a sufficiently large $P_s$, feedback consistency can be restored.

We can further relax the assumption that $F$ and $B$ are fixed during which $H$ is adjusted, by using a much faster learning rate, $\eta_h \gg \eta_f$ and $\eta_h \gg \eta_b$. In other words, $F$ and $B$ are approximately constant when the network is solving (4.12). All learning rules then can be activated and learn simultaneously when a new training sample is presented. The network eventually will learn an underlying dictionary $D = F^T \approx B$, and the optimal lateral weights $H \approx D^T D$.

## 5 Numerical Simulations

We examined the proposed learning algorithm using three standard datasets in image processing, machine learning, and computational neuroscience. **Dataset A.** Randomly sampled $8 \times 8$ patches from the grayscale Lena image to learn 256 atoms. **Dataset B.** $28 \times 28$ MNIST images \[10\] to learn 512 atoms. **Dataset C.** Randomly sampled $16 \times 16$ patches from whitened natural scenes \[15\] to learn 1024 atoms. For Dataset A and C, the patches are further subtracted by the means, normalized, and split into positive and negative channels to create nonnegative inputs \[8\]. The spiking networks are ran with a time step of 1/32. For each input, the feedforward stage is ran from $t = 0$ to $t = 20$ and the feedback stage is ran from $t = 20$ to $t = 40$, and the spike rates are measured simply as the total number of spikes within the time window of 20. We deliberately chose a short time window (the spike rates only have a precision of 0.05) to demonstrate the fast convergence of spike patterns; a more accurate equilibrium spike rate may be obtained if one is willing to use a larger window...
starting at some $t > 0$. The synaptic weights are randomly initialized to be asymmetric and inconsistent, with the lateral weights set to be sufficiently strong so that the spike rates will not diverge in the feedback stage. For the learning rates, we set $\eta_f = \eta_b$ and $\eta_h = 32 \eta_f$.

**Learning Dynamics.** Figure 2 shows the spike patterns before and after learning in both layers. Before learning, we see both sparse-code and input layer neurons exhibit perturbed spike rates in the feedback stage, as predicted in the earlier section. The perturbation in sparse-code neurons is caused by the inconsistency between randomly initialized synaptic weights, while the perturbation in input neurons is additionally due to the large reconstruction errors. After learning, the spike patterns become much steadier as the network learns to maintain weight consistency and minimize reconstruction error. Figure 3 shows the scatter plot of the learned lateral weights and firing thresholds, $H$, versus their desired values, $FB$. It can be seen that after learning, the network is able to maintain feedback consistency.

**Comparison with Stochastic Gradient Descent.** Dictionary learning is a notorious non-convex optimization problem. Here we demonstrate the proposed algorithm can indeed find a good local minimum. We compare the convergence behavior with stochastic gradient descent (SGD) with batch size of 1, to which our algorithm closely resembles. For SGD, we use the same learning rate as the spiking network, $\eta = \eta_f$, and explore two nearby learning rates $\eta = 2\eta_f$ and $\eta = 0.5\eta_f$. Additionally, we experiment initializing the spiking network weights to be symmetric and consistent to understand the impact of random initialization. The weight decay rates are chosen so that the firing thresholds, which correspond to the squared norms of atoms, converge
to a dynamic equilibrium around 1 to ensure a fair comparison. For each dataset, a separate test set of 10,000 samples is extracted, whose objective function value is used as the quality measure for the learned dictionaries.

Figure 4 shows that our SNN algorithm can obtain a solution of similar, if not better, objective function values to SGD consistently across the datasets. Surprisingly, the SNN algorithm can even reach better solutions with fewer training samples, while SGD can be stuck at a poor local minimum especially when the dictionary is large. This can be attributed to the dynamic adaptation of firing thresholds that mitigates the issue in SGD that some atoms can be rarely activated and remain unlearned. In SNN, if an atom is not activated over many training samples, its firing threshold decays, which makes it more likely to be activated for the next sample. Further, we observe that random weight initialization in SNN only causes slightly slower convergence, and eventually can find solutions of very similar objective function values.

### 6 Discussion

**Feedback Perturbation and Spike-Driven Learning.** Our learning mechanism can be viewed as using the feedback connections to test the optimality of synaptic weights. As we have shown, an optimal network should receive little perturbation from feedback, and the derived learning rules correspond to local and greedy approaches to reduce the amount of drift in spike patterns. Although our learning rules are based on spike rates, this idea certainly can be realized in a spike-driven manner to enable rapid correction of network dynamics. In particular, spike timing dependent plasticity (STDP) is an ideal candidate to implement the feedforward and feedback learning rules. The learning rules in (4.11) share the same form with differential Hebbian and anti-Hebbian plasticity, whose link to STDP has been shown [20]. On the other hand, the connection between our lateral learning rule and spike ordering based learning is less clear. It can be seen that the rule is driven by shifts in postsynaptic spike rates, but a feasible mechanism to capture the exact weight dependency remains an open problem.

In autoencoder learning, [11,14] similarly explored using feedback synapses for gradient computations. However, the lack of lateral connectivities in an autoencoder makes it difficult to handle potential reverberation, and time delays are needed to separate the activities of the input and sparse-code (or hidden) layers. In contrast, our learning mechanism is based on the steady states of two network configurations. This strategy is actually a form of contrastive Hebbian learning [14] in that the feedback synapses serve to bring the network from its “free state” to a “clamped state”.

**Practical Value.** The proposed algorithm shows that the dictionary learning problem can be solved with fine-grained parallelism. The synaptically local property means the computations can be fully distributed to
individual neurons, eliminating the bottlenecks central unit. The parallelism is best exploited by mapping the spiking network to a VLSI architecture, e.g., [13], where each neuron can be implemented as a processing element. Existing dictionary learning algorithms, e.g., [11, 12], can be accelerated by exploiting data parallelism, while it is less clear how to parallelize them within a single training sample to further reduce computation latency.

Our learning rules can be applied to related sparse coding models, such as reweighted $\ell_1$ minimization [6] and Elastic Net [22] (see [5, 15] for the respective dynamical system formulations). It can also be extended to be a parallel solver for convolutional sparse coding [21, 2]. Although the weight sharing property in a convolutional model is fundamentally “non-local”, this limitation may be overcome by clever memory lookup methods, as is commonly done in the computation of convolutional neural networks.

Acknowledgments

The author thanks Peter Tang, Javier Turek, Narayan Srinivasa and Stephen Tarsa for insightful discussion and feedback on the manuscript, and Hong Wang for encouragement and support.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[2] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *CVPR*, pages 391–398, 2013.

[3] C. S. N. Brito and W. Gerstner. Nonlinear hebbian learning as a unifying principle in receptive field formation. *PLoS Comput Biol*, 12(9):1–24, 2016.

[4] K. S. Burbank. Mirrored stdp implements autoencoder learning in a network of spiking neurons. *PLoS Comput Biol*, 11(12):e1004566, 2015.

[5] A. S. Charles, P. Garrigues, and C. J. Rozell. A common network architecture efficiently implements a variety of sparsity-based inference problems. *Neural computation*, 24(12):3317–3339, 2012.

[6] P. Garrigues and B. A. Olshausen. Group sparse coding with a laplacian scale mixture prior. In *Advances in neural information processing systems*, pages 676–684, 2010.

[7] G. E. Hinton and J. L. McClelland. Learning representations by recirculation. In *Neural information processing systems*, pages 358–366, 1988.

[8] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[9] T. Hu, C. Pehlevan, and D. B. Chklovskii. A hebbian/anti-hebbian network for online sparse dictionary learning derived from symmetric matrix factorization. In *2014 48th Asilomar Conference on Signals, Systems and Computers*, pages 613–619. IEEE, 2014.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[12] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[13] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[14] J. R. Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models: Proceedings of the 1990 summer school*, pages 10–17, 1990.
[15] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:13, 1996.

[16] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563, 2008.

[17] S. Shapero, M. Zhu, J. Hasler, and C. Rozell. Optimal sparse approximation with integrate and fire neurons. *International journal of neural systems*, 24(05):1440001, 2014.

[18] P. T. P. Tang, T.-H. Lin, and M. Davies. Sparse coding by spiking neural networks: Convergence theory and computational results. *ArXiv e-prints*, 2017, 1705.05475.

[19] P. Vertechi, W. Brendel, and C. K. Machens. Unsupervised learning of an efficient short-term memory network. In *Advances in Neural Information Processing Systems*, pages 3653–3661, 2014.

[20] X. Xie and H. S. Seung. Spike-based learning rules and stabilization of persistent neural activity. In *Advances in Neural Information Processing Systems*, pages 199–208, 2000.

[21] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *CVPR*, pages 2528–2535. IEEE, 2010.

[22] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. Royal Statist. Soc B.*, 67:301–320, 2005.

[23] J. Zylberberg, J. T. Murphy, and M. R. 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 Comput Biol*, 7(10):e1002250, 2011.