

# Learning and Inference in Sparse Coding Models with Langevin Dynamics

**Michael Y.-S. Fang**<sup>1, 2</sup>, **Mayur Mudigonda**<sup>2, 3</sup>, **Ryan Zarcone**<sup>2, 4</sup>, **Amir Khosrowshahi**<sup>2, 6</sup>,  
**Bruno A. Olshausen**<sup>2, 3, 5</sup>,

<sup>1</sup>Department of Physics, University of California Berkeley, Berkeley, CA, USA.

<sup>2</sup>Redwood Center for Theoretical Neuroscience, University of California Berkeley, Berkeley, CA, USA.

<sup>3</sup> Vision Science Graduate Group, School of Optometry, University of California, Berkeley, CA, USA.

<sup>4</sup>Biophysics Graduate Group, University of California Berkeley, Berkeley, CA, USA.

<sup>5</sup>Helen Wills Neuroscience Institute and School of Optometry, University of California Berkeley, Berkeley, CA, USA.

<sup>6</sup> Intel Corporation, Santa Clara, CA, USA

## Abstract

We describe a stochastic, dynamical system capable of inference and learning in a probabilistic latent variable model. The most challenging problem in such models – sampling the posterior distribution over latent variables – is proposed to be solved by harnessing natural sources of stochasticity inherent in electronic and neural systems. We demonstrate this idea for a sparse coding model by deriving a continuous-time equation for inferring its latent variables via Langevin dynamics. The model parameters are learned by simultaneously evolving according to another

continuous-time equation, thus bypassing the need for digital accumulators or a global clock. Moreover we show that Langevin dynamics lead to an efficient procedure for sampling from the posterior distribution in the ‘ $L_0$  sparse’ regime, where latent variables are encouraged to be set to zero as opposed to having a small  $L_1$  norm. This allows the model to properly incorporate the notion of sparsity rather than having to resort to a relaxed version of sparsity to make optimization tractable. Simulations of the proposed dynamical system on both synthetic and natural image datasets demonstrate that the model is capable of probabilistically correct inference, enabling learning of the dictionary as well as parameters of the prior.

## 1 Introduction

Latent variable models such as sparse coding (Olshausen & Field, 1997) and Boltzmann machines (Hinton & Sejnowski, 1983; Ackley et al., 1985) have been shown to be powerful and flexible tools in machine learning. However, training such models properly requires sampling from probability distributions over the latent variables. Typically, instead of sampling, a MAP (maximum *a-posteriori*) estimate or other heuristics are used since most sampling algorithms are laboriously slow and have convergence guarantees only under limited conditions. The time cost in large part comes from simulating stochastic dynamics of state transitions on deterministic, discrete-logic based hardware, requiring random number generation and fine sampling intervals to avoid discretization errors. These limitations have hindered the ability of latent variables models to learn complex structure in data, since adapting the parameters in a more complex, structured model, such as a hierarchical probabilistic model (Lee & Mumford, 2003), necessitates sampling under the posterior distribution.

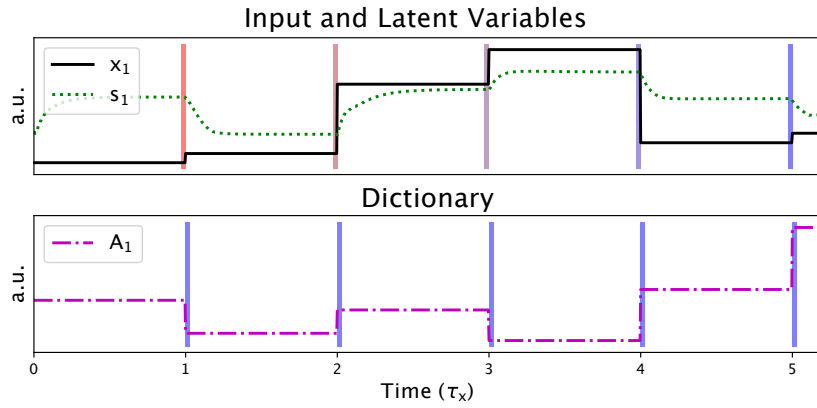
This paper proposes a solution to this problem based on utilizing the intrinsic sources of stochasticity that exist in any physical system. Our central thesis is that rather than forcing a deterministic, discrete-logic based system to simulate stochastic dynamics on continuous variables, a more sensible and efficient solution is to exploit physics to directly implement stochastic, analog computation. In the same way that the analog VLSI retina implements filtering via lateral inhibition in a resistive grid (Mead & Mahowald, 1988) – resulting in orders of magnitude greater computational efficiency than digital simulation – we envision the development of analog circuits that perform the necessary computations and stochastic dynamics for probabilistic inference and learning in complex latent variable models. A recent successful example of this approach is the work of (Borders et al., 2019), who used the intrinsic probabilistic behavior of nanoscale magnetic tunneling junctions to sample from the binary state variables of a Boltzmann machine. Another example is the use of stochastic logic circuits to perform fast Bayesian inference for perception and reasoning tasks (V. Mansinghka & Jonas, 2014; V. K. Mansinghka et al., 2008). Additionally, in neuroscience it has been hypothesized that seemingly random fluctuations in neural activity can be interpreted as a process for sampling from posterior distributions (Hoyer & Hyvärinen, 2003; Berkes et al., 2011; Orbán et al., 2016; Echeveste et al., 2020). Our goal here is to demonstrate, through derivation and simulation of a dynamical system of equations, the viability of such an approach for probabilistic inference and learning in a latent variable model. In an appendix, we point the way to a potential circuit implementation.

Beyond the difficulties associated with sampling, learning the parameters of a probabilistic model requires averaging the samples or other quantities computed from them. One direct way of doing this is to accumulate these quantities followed by a parameter update (Fig. 1b). However, this requires a digital accumulator, and the interfacing between analog and digital hardware is often a

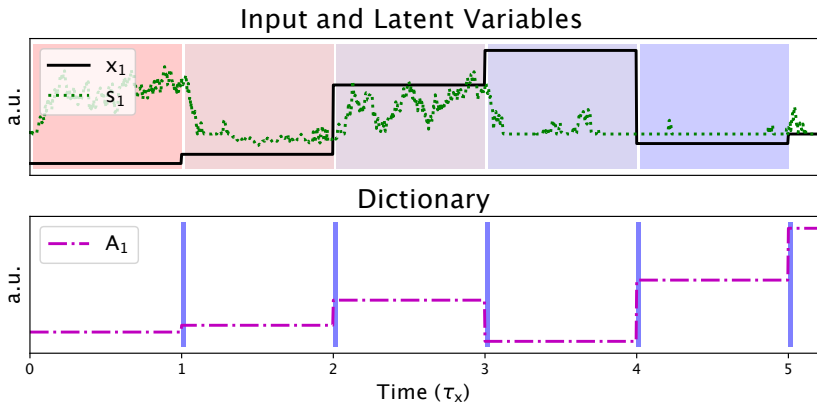
bottleneck for sampling. For example, in recent work by (Roques-Carnes et al., 2019), the limiting component for a photonic sampler was identified as the photodetector. Here we propose a novel, fully analog framework in which the update of parameters occurs *simultaneously* alongside the sampling of latent variables through continuous time dynamics (Fig. 1c). Rather than waiting for the collection of samples for each discrete parameter update, the effective accumulation of samples is achieved by simply having a longer time constant.

To study this analog learning and inference framework we apply it to the sparse coding model, a simple yet expressive probabilistic model with an explicit prior over the latent variables (Tibshirani, 1996; Hastie et al., 2009). The sparse coding model is of interest in both neuroscience and engineering as it provides an account for the neural representation of natural images in visual cortex (Olshausen & Field, 1997) and it has proven useful in computer vision (Wright et al., 2010; Wang et al., 2015) and signal compression (Donoho, 2006). However current implementations of sparse coding are slow due to the optimization required to infer the latent variables for each data sample, and learning is inefficient since only a single such point estimate of the latent variables is used to make a dictionary update (Fig. 1a). In Section 2 we derive a fully continuous-time sparse coding model by making use of fast Langevin dynamics to sample latent variables and slower dynamics to co-evolve the dictionary based on these samples, as in Figure 1c.

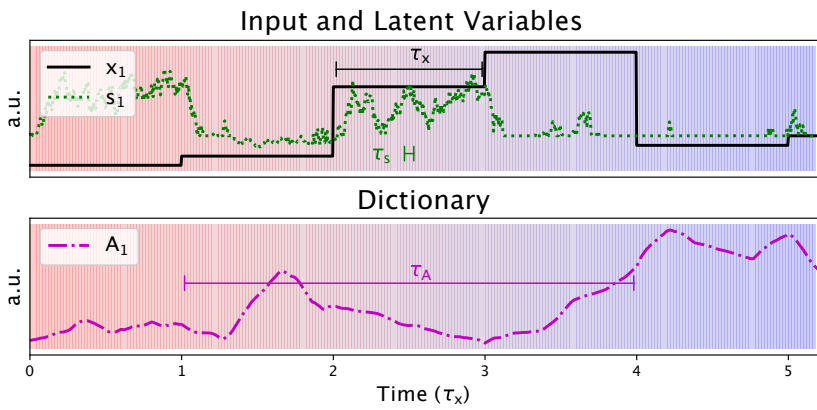
Sampling with Langevin dynamics is well studied both in theory (Bussi & Parrinello, 2007) and in application to Bayesian learning (Welling & Teh, 2011). However, to our knowledge this is the first fully analog approach to simultaneous inference and learning for sparse coding. Prior sampling-based approaches utilized a mixture-of-Gaussians model and employed discrete Gibbs sampling over the mixture variables (Olshausen & Millman, 2000) or a method for preselecting parts of the space to sample via MCMC (Shelton et al., 2011).



(a) Discrete Dictionary Update with MAP



(b) Discrete Dictionary Update with Sampling



(c) Continuous, Simultaneous Dictionary Update with Sampling

Figure 1: (caption on following page)

Figure 1: Illustration of three approaches to learning latent variable models. a) In the standard approach, data  $\mathbf{x}$  is presented at regular intervals (upper plot, black trace). A MAP estimate of latent variables  $\mathbf{s}$  is calculated via gradient descent or other iterative algorithm (green trace). The resulting estimate is used for a discrete update to the dictionary  $A$  (lower plot). The blue vertical bars illustrate the computational inefficiency where only a single point estimate of the coefficients is used to make a dictionary update. b) In a sampling-based approach, for each data interval multiple samples from the posterior are averaged for a dictionary update. The colored regions in the top panel show that many samples are collected to approximate the posterior distribution. However, the discrete dictionary updates (at corresponding vertical bands) make a fully analog implementation difficult. c) Rather than waiting for the accumulation of samples, the dictionary  $A$  is updated simultaneously alongside the latent variables  $\mathbf{s}$ . The slow timescale of the dictionary compared to the latent variables  $\tau_A \gg \tau_s$  allows for effective averaging. (Learning rates shown are purely for illustrative purposes.)

An additional advantage of Langevin dynamics is that it leads us to a simple procedure for sampling from the posterior when using an ‘ $L_0$  sparse’ prior that explicitly encourages latent variables to be set to zero rather than simply taking on small values (also known as a ‘spike and slab’ prior). Normally such priors are avoided as finding the optimal sparse representation of a signal requires solving a combinatorial search problem. Instead, sparsity is enforced by imposing an  $L_1$  cost function on the latent variables, which is used as a proxy for  $L_0$  since it allows for convex optimization. However, in probabilistic terms, the  $L_1$  cost corresponds to a Laplacian prior which only weakly captures the notion of sparsity. We show in Section 3 how Langevin sparse

coding releases us from this restriction. By simple thresholding of a continuous variable undergoing Langevin dynamics, we obtain samples from the posterior using an ‘ $L_0$  sparse’ prior.

In Section 4.1, we demonstrate the efficacy of this model for correct inference and learning using a synthetic dataset. Furthermore in Section 4.2 we demonstrate that this approach allows for learning the size of the dictionary, which was attempted in previous work using variational approximation of the posterior (Berkes et al., 2008). Then in Section 4.3, we fit our  $L_0$ -sparse coding model to the Van Hateren dataset of natural images. In addition to learning the dictionary elements, we provide an estimate for the sparsity of natural images.

To summarize, the main contributions presented are:

1. A theoretical formulation of simultaneous dynamics for sampling from latent variables and learning model parameters.
2. Langevin Sparse Coding (LSC), a continuous-time, probabilistic model for simultaneous inference and learning in a sparse coding model.
3. An efficient procedure for sampling from the posterior with an ‘ $L_0$  sparse’ prior.
4. Learning not only the dictionary for representing natural images but also other parameters of the model such as the sparsity level and size of the dictionary.

## 2 Langevin Sparse Coding

Sparse coding is a simple yet efficient algorithm for learning structure in data by finding a ‘dictionary’ to describe patterns contained in the data. While it is formulated as a probabilistic latent-variable model, it is often approximated in practice by finding point estimates for the latent variables rather

than sampling from their posterior distribution. As a result, it is difficult to make rigorous claims about the relation between the learned dictionary and the statistics of the data, and it is problematic to adapt other parameters of the model such as the degree of sparsity or overcompleteness of the dictionary. More broadly, it has hindered the advancement of sparse coding into a more powerful generative modeling framework – for example, by incorporating hierarchical structure – since there is no principled way to learn the parameters of such models without sampling from the posterior.

In this section, we introduce Langevin Sparse Coding (LSC), which efficiently samples the latent-variables of a sparse coding model and allows simultaneous, continuous updates of dictionary elements along with the latent variables. This last property is important in making the LSC framework amenable for fully analog implementation. We begin with a review of the canonical approach of Discrete Sparse Coding (DSC). Next, we introduce simultaneous-update sparse coding (SSC) in which dictionary updates are made continuously and concurrent with the dynamics of the coefficients. Finally, we present LSC where we demonstrate that the inherent noise to analog systems can be used to perform sampling.

## 2.1 Probabilistic Model

Sparse coding assumes that the data,  $\mathbf{x} \in \mathbb{R}^D$ , are described as a linear combination of elements from a dictionary  $A \in \mathbb{R}^{D \times K}$  with additive Gaussian noise  $\mathbf{n} \in \mathbb{R}^D$ :

$$\mathbf{x} = A \mathbf{s} + \mathbf{n} \tag{1}$$

where  $n_i \stackrel{iid}{\sim} N(0, \sigma^2)$ . The coefficients  $\mathbf{s} \in \mathbb{R}^K$  are latent variables that are assumed to be sparsely distributed, so that any given datapoint should be well approximated using a small number of columns of the dictionary. Sparsity is enforced by the choice of prior, typically chosen to be



factorial:

$$p_s(\mathbf{s}) = \prod_{i=1}^K p_s(s_i) \quad (2)$$

$$p_s(s_i) \propto \exp(-\lambda C(s_i)) \quad (3)$$

where the form of  $C$  is chosen so that  $p_s(s_i)$  is peaked at  $s_i = 0$  and with heavy tails away from zero. (Note that non-factorial priors are also possible, see e.g., (Garrigues & Olshausen, 2010).)

The posterior over the latent variables in this model may be written in exponential form,

$$p(\mathbf{s} \mid \mathbf{x}, A) \propto \exp(-E(A, \mathbf{s}, \mathbf{x})), \quad (4)$$

with the energy function  $E(A, \mathbf{s}, \mathbf{x})$  given by

$$E(A, \mathbf{s}, \mathbf{x}) = \frac{\|\mathbf{x} - A\mathbf{s}\|_2^2}{2\sigma^2} + \lambda \sum_i C(s_i). \quad (5)$$

Thus inferring a good (highly probable) interpretation of a given data sample,  $\mathbf{x}$ , corresponds to finding a set of latent variables,  $\mathbf{s}$ , with low energy,  $E$ .

The goal of learning in this model is to find a dictionary,  $A$ , that provides the best fit to the data. This is accomplished by solving for the maximum likelihood estimator (MLE) of the dictionary

$$A^* = \arg \max_A \langle \log p(\mathbf{x} \mid A) \rangle_{\mathbf{x} \sim \mathcal{D}} \quad (6)$$

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

$$\nabla_A \langle \log p(\mathbf{x} \mid A) \rangle_{\mathbf{x} \sim \mathcal{D}} = \left\langle \left\langle -\nabla_A E(A, \mathbf{s}, \mathbf{x}) \right\rangle_{\mathbf{s} \mid \mathbf{x}} \right\rangle_{\mathbf{x} \sim \mathcal{D}} \quad (7)$$

$$= \left\langle \left\langle (\mathbf{x} - A\mathbf{s}) \mathbf{s}^T \right\rangle_{\mathbf{s} \mid \mathbf{x}} \right\rangle_{\mathbf{x} \sim \mathcal{D}} \quad (8)$$

where  $\langle \cdot \rangle_{\mathbf{s} \mid \mathbf{x}}$  denotes expectation with respect to the posterior distribution  $p(\mathbf{s} \mid \mathbf{x}, A)$  (see (Lewicki & Olshausen, 1999) for a derivation). Thus, adapting the dictionary to the data requires, for each data

sample  $\mathbf{x}$ , sampling from the posterior over  $\mathbf{s}$  and computing the correlation between the residual,  $\mathbf{x} - A \mathbf{s}$ , and  $\mathbf{s}$ . The dictionary  $A$  would then be incrementally updated according to this correlation (eq. 8). Equilibrium is reached when  $\langle \langle \hat{\mathbf{x}}(\mathbf{s}) \mathbf{s}^T \rangle_{\mathbf{s}|\mathbf{x}} \rangle_{\mathbf{x} \sim \mathcal{D}} = \langle \mathbf{x} \langle \mathbf{s}^T \rangle_{\mathbf{s}|\mathbf{x}} \rangle_{\mathbf{x} \sim \mathcal{D}}$ , with  $\hat{\mathbf{x}}(\mathbf{s}) = A \mathbf{s}$ .

Beyond learning the dictionary, one can adapt other parameters of the model such as  $\sigma$  and  $\lambda$  also via gradient descent. The gradients for these parameters are as follows:

$$\nabla_{\sigma} \langle \log p(\mathbf{x}|A) \rangle_{\mathbf{x} \sim \mathcal{D}} \propto \frac{1}{D} \left\langle \langle |\mathbf{x} - A \mathbf{s}|^2 \rangle_{\mathbf{s}|\mathbf{x}} \right\rangle_{\mathbf{x} \sim \mathcal{D}} - \sigma^2 \quad (9)$$

$$\nabla_{\lambda} \langle \log p(\mathbf{x}|A) \rangle_{\mathbf{x} \sim \mathcal{D}} \propto \frac{1}{K} \left\langle \left\langle \sum_i^K C(s_i) \right\rangle_{\mathbf{s}|\mathbf{x}} \right\rangle_{\mathbf{x} \sim \mathcal{D}} - \langle C(s) \rangle_{p_{\mathbf{s}}(\mathbf{s})} \quad (10)$$

Adapting these parameters similarly requires computing averages under the posterior distribution for each data sample. Note that when the sparse coding model objective is formulated purely in terms of its energy function (eq. 5) – which is typically the case – then there is no principled way to adapt these parameters to the data. The probabilistic framework makes it possible, so long as it is tractable to sample from the posterior distribution.

## 2.2 Discrete Sparse Coding

In practice, the expectation over the data in (8) is approximated via stochastic gradient descent (SGD). For a batch of data of size  $N$ ,  $\{\mathbf{x}_n\}_{n=1 \dots N}$ , the update rule is

$$\Delta A = \eta \frac{1}{N} \sum_{n=1}^N \langle (\mathbf{x}_n - A \mathbf{s}_n) \mathbf{s}_n^T \rangle_{\mathbf{s}_n|\mathbf{x}_n} \quad (11)$$

where  $\eta$  specifies the learning rate. However, the expectation over  $\mathbf{s}_n$  is usually considered intractable and so in practice it is approximated by the maximum *a posteriori* (MAP) estimator of  $\mathbf{s}_n$

$$\mathbf{s}_n^* = \arg \min_{\mathbf{s}_n} E(A, \mathbf{s}_n, \mathbf{x}_n). \quad (12)$$

Solving via gradient descent yields the iterative update equation

$$\Delta \mathbf{s}_n \propto -\nabla_{\mathbf{s}} E(A, \mathbf{s}_n, \mathbf{x}_n) \quad (13)$$

$$= -\frac{1}{\sigma^2} A^T (\mathbf{x}_n - A \mathbf{s}_n) - \lambda C'(\mathbf{s}_n) \quad (14)$$

where  $C'$  is the derivative of cost function  $C$  above (5) and operates elementwise on  $\mathbf{s}_n$ . For each  $\mathbf{x}_n$ , equation (14) is iteratively evaluated until it converges to a solution. In order to make this a convex optimization, the cost function  $C$  is typically taken to be the  $L_1$  norm, corresponding to a Laplacian prior  $p_s(\mathbf{s})$ . Gradient descent does not generally constitute the most efficient method for finding the MAP estimate, but we use it here as a step towards the development of LSC below.

The price we pay for approximating the expectation  $\langle \cdot \rangle_{\mathbf{s}_n | \mathbf{x}_n}$  in equation 11 with a single MAP estimate is that it now becomes necessary to normalize the dictionary elements  $A = (\mathbf{A}_1, \dots, \mathbf{A}_K)$  after each update via

$$\mathbf{A}_i \leftarrow \frac{\mathbf{A}_i}{\|\mathbf{A}_i\|_2} \equiv \hat{\mathbf{A}}_i. \quad (15)$$

This is necessary because the MAP estimator  $\mathbf{s}^*$  will consistently underestimate  $\mathbf{s}$  such that it is biased toward zero (due to the sparse prior). As a result, each  $\mathbf{A}_i$  will grow without bound unless normalized. (As we shall see below, this no longer becomes necessary when we sample from the posterior.)

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

$$E(A, S, X) \equiv \sum_{n=1}^N E(A, \mathbf{s}_n, \mathbf{x}_n) \quad (16)$$

$$= \frac{\|AS - X\|_{2,2}^2}{2\sigma^2} + \lambda \|S\|_{1,1}. \quad (17)$$

We have defined batch matrices  $S \in \mathbb{R}^{K \times N}$  and  $X \in \mathbb{R}^{D \times N}$ . Above,  $\|\cdot\|_{p,q}$  refer to the  $L_{(p,q)}$  matrix norm, defined by

$$\|A\|_{p,q} = \left( \sum_j \left( \sum_i |a_{ij}|^p \right)^{\frac{q}{p}} \right)^{\frac{1}{q}} \quad (18)$$

With the batch energy defined, the update rules are

$$S \leftarrow S - \eta_S \nabla_S E(A, S, X) \quad (19)$$

$$A \leftarrow A - \eta_A \nabla_A E(A, S, X) \quad (20)$$

$$A \leftarrow \text{Norm}(A) \quad (21)$$

where the  $\text{Norm}()$  operation corresponds to the normalization of equation 15.

To coordinate the updates of  $S$  and  $A$ , a nested loop must be used (Alg. 1). The inner loop approximates the MAP estimator  $S^*$  while the outer loop finds the MLE of  $A$ .

---

**Algorithm 1** Algorithm for discrete sparse coding (DSC). Note line 6 was included purely to emphasize  $S^*$  as a MAP estimate.

---

```

1: for  $k \leftarrow 1$  to  $N_A$  do
2:    $X \leftarrow \text{SAMPLEBATCH}()$ 
3:   for  $n \leftarrow 1$  to  $N_s$  do
4:      $S \leftarrow S - \eta_S \cdot \nabla_S E(A, S, X)$ 
5:   end for
6:    $S^* \leftarrow S$ 
7:    $A \leftarrow A - \eta_A \cdot \nabla_A E(A, S^*, X)$ 
8:    $A \leftarrow \text{Norm}(A)$ 
9: end for

```

---

A closely related cousin of DSC, the Locally Competitive Algorithm (LCA) (Rozell et al., 2008), computes the MAP estimate by following dynamics that descend the energy  $E$  in a more efficient manner. Instead of doing direct gradient-descent (eq. 14),  $\mathbf{s}$  is taken to be a monotonically increasing, nonlinear function of another variable  $\mathbf{u}$  that follows the gradient with respect to  $\mathbf{s}$ :

$$\Delta \mathbf{u}_n \propto -\nabla_{\mathbf{s}} E(A, \mathbf{s}_n, \mathbf{x}_n) \quad (22)$$

$$\mathbf{s}_n = g(\mathbf{u}_n) \quad (23)$$

where  $g$  operates elementwise on  $\mathbf{u}$  and is determined by the choice of cost function  $C$ . For an L1 cost,  $g$  is a signed Relu function with threshold  $\lambda$ :

$$g(u_i) = \begin{cases} 0 & |u_i| < \lambda \\ \text{sign}(u_i)(|u_i| - \lambda) & |u_i| \geq \lambda \end{cases} \quad (24)$$

Other than this difference in the dynamics for MAP inference, which falls purely within the inner loop (line 4) of Algorithm 1, both DSC and LCA update the dictionary based on a single MAP estimate and thus suffer the same inefficiency as depicted in Figure 1a.

### 2.3 Simultaneous (Update) Sparse Coding - SSC

We note that the DSC algorithm above requires the alternating update of the dictionary elements and coefficients. Typically, this necessitates a digital clock for synchronization and is a major challenge towards fully analog implementation. In this subsection, we present an asynchronous framework – Simultaneous-Update Sparse Coding (SSC) – where both the dictionary and coefficients are updated simultaneously.

Rather than updating the dictionary  $A$  at the end of the loop when  $S$  has converged to the MAP estimator  $S^*$ , SSC updates  $A$  continuously and concurrent with  $S$ . In search of dynamics amenable

to analog computation, we take the step sizes to be infinitesimally small, and arrive at the following set of differential equations.

$$\tau_S \dot{S} = -\nabla_S E(A, S, X(t)) \quad (25)$$

$$\tau_A \dot{A} = -\nabla_A E(A, S, X(t)) \quad (26)$$

while still enforcing the normalization constraint on  $A$  (eq. 15). Here, we take  $X(t)$  to be updated synchronously at regular intervals of  $\tau_X$ . At each update, a new batch of samples is drawn.

To compare SSC and DSC, consider the following simulation for SSC using the Euler Method.

---

**Algorithm 2** Euler Method simulation of SSC with stepsize of  $\Delta t$  and regular interval input of  $X$

---

```

1: for  $t \leftarrow 1$  to  $t_{\max}/\Delta t$  do

2:    $dS \leftarrow \frac{\partial E}{\partial S}(A, S, X(t))$ 

3:    $dA \leftarrow \frac{\partial E}{\partial A}(A, S, X(t))$ 

4:    $S \leftarrow S - \frac{\Delta t}{\tau_S} \cdot dS$ 

5:    $A \leftarrow A - \frac{\Delta t}{\tau_A} \cdot dA$ 

6:    $A \leftarrow \text{Norm}(A)$ 

7: end for
```

---

Comparing Algorithm 1 and Algorithm 2, the timescales  $\tau$  can be related to the learning rates,  $\eta$ , and the number of iterations  $N_S$ . We stress an important difference between the two is that SSC is fully described through a set of coupled differential equations and requires no control structure (i.e. a nested for loop). This is especially desirable for analog implementation as a global clock is no longer necessary. Furthermore, there is no longer need for synchronous, regular input of the data  $X$ . While not explored here, dynamic input such as videos can be naturally processed without any

frame-by-frame synchronization.

## 2.4 Sampling via Langevin Dynamics

Consider a time-varying system described by coordinates  $\mathbf{u}(t)$  with energy  $E(\mathbf{u})$ . It can be modeled by Langevin dynamics according to the following stochastic differential equation:

$$\dot{\mathbf{u}} = -\nabla E(\mathbf{u}) + \sqrt{2T}\xi(t), \quad (27)$$

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

$$p^{(\infty)}(\mathbf{u}) \propto e^{-E(\mathbf{u})/T} \quad (28)$$

This relation suggests that we change the dynamics of SSC (25) by injecting noise to  $\dot{S}$ :

$$\tau_S \dot{S} = -\nabla_S E(A, S, X) + \sqrt{2T\tau_S}\xi(t) \quad (29)$$

Note that under the scaling of  $t \rightarrow t/\tau_S$ , we have  $\langle \xi(t/\tau_S)\xi(t'/\tau_S)^T \rangle = \mathbf{I}\delta(\tau_S^{-1}(t - t')) = \tau_S \mathbf{I}\delta(t - t') = \langle \sqrt{\tau_S}\xi(t)\sqrt{\tau_S}\xi(t')^T \rangle$ . This necessitates the somewhat unexpected scaling factor of  $\tau_S$ .

Following the above dynamics, for fixed  $A$  and input  $X$ ,  $S$  will sample from the posterior distribution,

$$p_{S|X}(S(t)|X, A) \propto e^{-E(A, S, X)/T}. \quad (30)$$

This is a remarkable result: *By simply injecting noise into the continuous-time dynamics normally used for MAP inference in sparse coding, we obtain a dynamical system that naturally samples from the desired posterior distribution (eq. 4).* With  $T = 0$ , we recover the SSC dynamics above (eqs. 25-26) where  $S$  converges to the MAP estimate.

A useful property of (29) is that the equilibrium distribution is independent of the time constant  $\tau_S$ . By taking  $\tau_A \gg \tau_S$ , the assumption that  $A$  is fixed with respect to the dynamics of  $S$  can be upheld. Conversely, because  $S$  evolves much faster than  $A$ , the dynamics of  $A$  are well approximated by

$$\tau_A \dot{A} = -\langle \nabla_A E(A, S, X) \rangle_{S|A, X}. \quad (31)$$

This is the exact mean gradient that we originally sought to calculate (eq. 7).

In summary, we have derived a new method for inference and learning in a sparse coding model, Langevin Sparse Coding (LSC), as specified by the continuous, coupled dynamics of equations 29 and 31, that achieves the desired property illustrated in Figure 1c. Importantly, our aim doing this is not simply to produce another MCMC algorithm, but rather to move toward a physical realization that naturally implements these dynamics (an example of which is described in Appendix C).

### 3 ‘ $L_0$ Sparse’ Prior

Since the goal of sparse coding is to represent each data item using a small number of non-zero latent variables, the prior should ideally have a sharp peak at zero in order to encourage many latent variables to be set to zero. In this case, the cost term  $C$  within the energy function (5) would resemble an  $L_0$  cost that rewards coefficients for being strictly zero (as opposed to being non-zero and merely small in amplitude). However such cost functions are not used in practice because they are not amenable to gradient-based or convex optimization methods for computing the MAP estimate. Instead, the  $L_1$  cost is usually adopted as a proxy for  $L_0$  as it has been shown to yield equivalent solutions under certain conditions (Tropp, 2006). However from the perspective of a probabilistic model, the  $L_1$  cost corresponds to a Laplacian prior that only weakly expresses the



notion of sparsity. In fact, the Laplacian is the *maximum entropy* distribution for a real-valued variable of a given mean absolute value. Here we show that the use of ‘ $L_0$  Sparse’ priors becomes tractable in our sampling-based setting, and we develop a modified LSC formulation that enables efficient sampling from the posterior.

Consider the following prior consisting of a mixture of a delta-function and Laplacian distribution (also known as a ‘spike and slab’ prior (Mitchell & Beauchamp, 1988)):

$$p_0(s) = \pi \lambda e^{-\lambda s} + (1 - \pi) \delta(s). \quad (32)$$

With  $\pi$  as the probability of being ‘active’,  $1 - \pi$  quantifies the  $L_0$  sparsity, or how likely  $s$  is to be zero. When  $s$  is in the active state it is exponentially distributed with mean  $1/\lambda$  (see right panel of figure 2). Note that here and in what follows we will assume the latent variables to be non-negative as opposed to allowing them to go positive or negative as is typically the case in sparse coding models.

To develop an efficient sampling strategy, we first define auxiliary variables  $\mathbf{u}$  such that each  $u_i$  independently follows an exponential distribution:

$$p_U(u_i) = \lambda e^{-\lambda u_i}. \quad (33)$$

We then take the latent variables  $\mathbf{s}$  to be given by  $s_i = f(u_i)$  where  $f$  is a biased ReLU function:

$$s_i = f(u_i) = \begin{cases} 0 & u_i < u_0 \\ u_i - u_0 & u_i \geq u_0 \end{cases} \quad (34)$$

for some positive  $u_0$ . We can show that  $s_i$  is then distributed according to the prior  $p_0(s)$  by

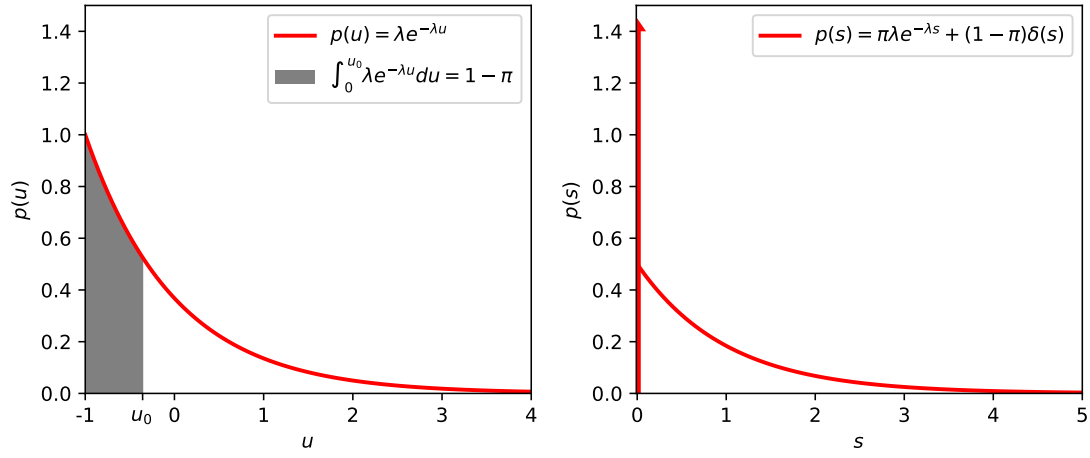


Figure 2: ‘ $L_0$  Sparse’ prior. Left panel shows the exponential distribution  $p(u)$ . With the change of variable  $s = f(u)$  via the application of a soft-thresholding function, we obtain the desired L0-like distribution  $p_0(s)$  shown in the right panel (shown for the region  $s \geq 0$ ). The threshold parameter  $u_0$  is chosen so that the probability weight of the delta function,  $1 - \pi$ , is equal to the shaded region in the left panel. These plots show the resulting distributions for  $\lambda = 1, \pi = 0.5$ .

marginalizing the joint distribution  $p(s, u)$  over  $u$  as follows:

$$\begin{aligned}
p_S(s) &= \int_{-\infty}^{\infty} p(s|u) p_U(u) du \\
&= \int_0^{u_0} \delta(s) p_U(u) du + \int_{u_0}^{\infty} \delta(s - (u - u_0)) p_U(u) du \\
&= \delta(s) \int_0^{u_0} p_U(u) du + p_U(s + u_0) \\
&= \delta(s) [1 - e^{-\lambda u_0}] + \lambda e^{-\lambda s} e^{-\lambda u_0} \\
&= [1 - \pi] \delta(s) + \pi \lambda e^{-\lambda s} \equiv p_0(s)
\end{aligned} \tag{35}$$

with  $\pi = e^{-\lambda u_0}$ . The relation between  $p(u)$ ,  $u_0$  and  $p(s)$  is illustrated in Figure 2.

To derive the Langevin dynamics for sampling from the posterior using the  $L_0$ -sparse prior above, we first re-write the energy function in terms of  $\mathbf{u}$ :

$$E(A, \mathbf{u}, \mathbf{x}) = \frac{1}{2} \frac{\|\mathbf{x} - A f(|\mathbf{u}|)\|_2^2}{\sigma^2} + \lambda \|\mathbf{u}\|_1. \tag{36}$$

We then let  $\mathbf{u}$  follow Langevin dynamics governed by this energy function. Note that we can allow the  $u_i$  to move freely between positive and negative values and then use only their absolute value in evaluating the energy. This essentially reflects the dynamics about the origin which avoids the problems associated with having an infinite energy barrier at  $u_i = 0$ . Letting  $|\mathbf{u}|$  denote the elementwise absolute value of  $\mathbf{u}$ , the distribution of  $|\mathbf{u}|$  will converge to

$$p(|\mathbf{u}| | \mathbf{x}) \propto \exp \left( -\|A f(|\mathbf{u}|) - \mathbf{x}\|_2^2 / \sigma^2 - \lambda \|\mathbf{u}\|_1 \right) \tag{37}$$

$$\propto p(\mathbf{x} | f(|\mathbf{u}|)) p_U(|\mathbf{u}|) \tag{38}$$

$$= p(\mathbf{x} | \mathbf{s}) p_0(\mathbf{s}) \tag{39}$$

Thus we obtain a second remarkable result: *By following Langevin dynamics on the energy in (36) with  $\mathbf{s} = f(|\mathbf{u}|)$ , we obtain samples from the posterior  $p(\mathbf{s} | \mathbf{x})$  given by combining the likelihood*

with the  $L_0$ -sparse prior  $p_0(\mathbf{s})$ . This is significant, because a MAP-estimate based approach would be impossible with such a prior since the posterior will always have its maximum at  $\mathbf{s} = 0$  regardless of the likelihood.

Applying the LSC equations (29, 31) using the energy in equation (36), we obtain the following coupled stochastic differential equations for inference and learning in  $L_0$ -LSC:

$$\tau_u \dot{\mathbf{u}} = -A^T(A\mathbf{s} - \mathbf{x})\Theta(|\mathbf{u}| - \mathbf{u}_0) - \lambda \text{sign}(\mathbf{u}) + \sqrt{2}\xi(t) \quad (40)$$

$$\mathbf{s} = f(|\mathbf{u}|) \quad (41)$$

$$\tau_A \dot{A} = -(A\mathbf{s} - \mathbf{x})\mathbf{s}^T. \quad (42)$$

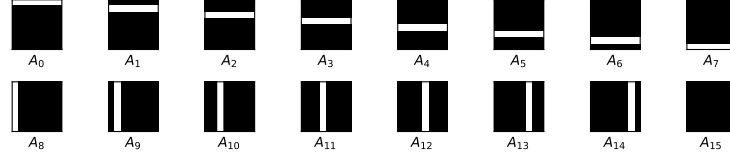
where  $\Theta(u)$  is the Heaviside function and  $\xi(t)$  is independent Gaussian white noise. Importantly, we can also learn  $u_0$ , and therefore the activation probability,  $\pi$ , via the dynamics

$$\dot{u}_0 \propto \left\langle \left\langle -\frac{\partial E}{\partial u_0} \right\rangle_{\mathbf{s}|\mathbf{x}} \right\rangle_{\mathbf{x} \sim \mathcal{D}} \quad (43)$$

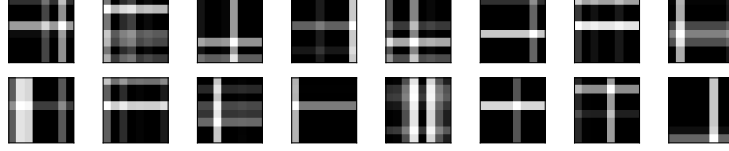
$$= \left\langle \left\langle A^T(A\mathbf{s} - \mathbf{x}) \cdot \mathbf{1}(\mathbf{s} > 0) \right\rangle_{\mathbf{s}|\mathbf{x}} \right\rangle_{\mathbf{x} \sim \mathcal{D}} \quad (44)$$

## 4 Results

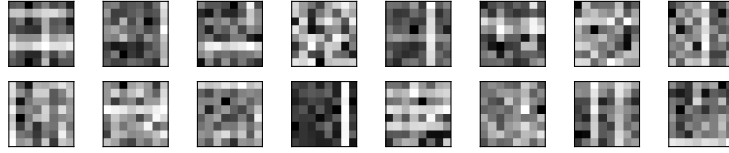
To study the efficacy of  $L_0$ -LSC, we first apply it to an artificial dataset consisting of images of bars in different orientations. This provides a useful test case for evaluation since the causes that generate the data are known. We then turn to a dataset of natural scenes where the ground truth is unknown.



(a) Bars Dictionary



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



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

Figure 3: The synthetic Bars dataset used as a toy problem. a) The dictionary is the collection of vertical and horizontal lines. b) An example of a sample drawn from the dataset. c) Another sample with noise introduced.

## 4.1 Inference on Bars Dataset

For the bars dataset, samples are generated from a dictionary  $A$  consisting of vertical and horizontal lines (Fig. 3a). We compare results obtained on this dataset against DSC as well as another method for training sparse coding, the locally competitive algorithm (LCA) (Rozell et al., 2008).

We synthetically generate data as a linear combination of the dictionary with additive Gaussian noise (Eq. 1) where,  $n_i \sim N(0, \sigma^2)$  and the coefficients are distributed according to  $L_0$  zero-inflated exponential prior (Beckett et al., 2014) (Eq. 32). A sample drawn from this model without noise and with noise is shown in Fig. 3b and 3c.

When trained on this dataset, all three algorithms were successful at learning the correct dictionary. However,  $L_0$ -LSC can better capture the posterior distribution than either DSC or LCA due to the fact that it directly enforces  $L_0$  sparsity. In both DSC and LCA, the sparsity is controlled by adjusting the parameter  $\lambda$ . However, the relationship between  $\lambda$  and  $L_0$  sparsity (Fig. 4a) is rather indirect and no analytic expression is known. On the other hand, in  $L_0$ -LSC a specific level of  $L_0$ -sparsity can be directly enforced by setting  $u_0 = -\lambda^{-1} \log(\pi)$ .

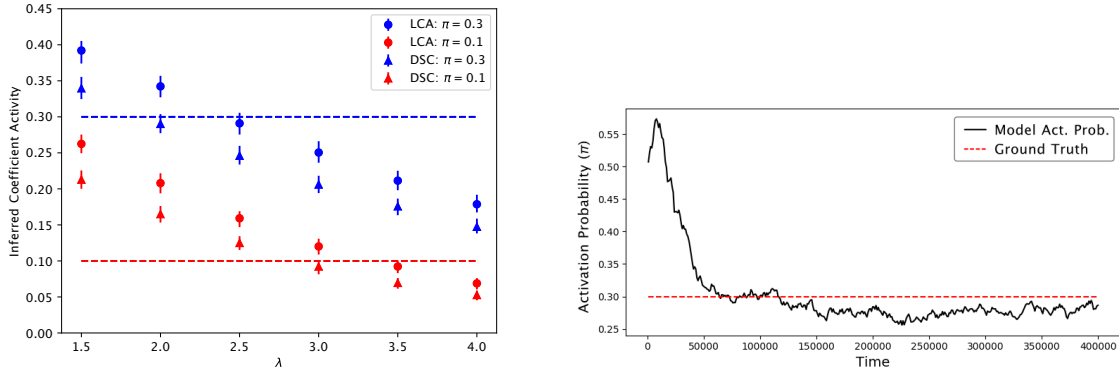


Figure 4: a) LCA and DSC are trained on data generated with activation probability  $\pi = 0.3$  (blue) and  $\pi = 0.1$  (red). For both, a sweep in sparsity parameter  $\lambda$  is made. While a correspondence between  $\lambda$  and  $\pi$  exists, there is no analytic expression to automatically adapt these parameters to the data. Even with data of known sparsity, it is impossible to select the correct parameter  $\lambda$  to use. b) With  $L_0$ -sparse LSC, the activation probability  $\pi$  is directly related to the parameter  $u_0 = -\lambda^{-1} \log \pi$  and can be learned directly without a parameter search.

Moreover, the activation probability  $\pi$  can be learned by LSC without any guesswork or parameter search (Eq. 43). Specifically, simultaneous to the evolution of  $A, \mathbf{u}$ , the threshold parameter  $u_0$  is treated as a variable evolves through gradient descent,  $\dot{u}_0 \propto \nabla_{u_0} E$ .

Figure 4b shows the convergence of model parameter  $\pi$  to match (approximately) the actual

level of sparsity in the data. To further characterize the coefficients, the distributions of the non-negative coefficients of the three algorithms were also plotted in Fig. 5. Using a fixed dictionary, the algorithm was run to infer either the MAP estimate (DSC and LCA) or to sample from the posterior ( $L_0$ -LSC). This was done with a correctly learned dictionary (Fig. 3a) as well as a random dictionary (i.e. uncorrelated gaussian noise). In addition to having the correct  $L_0$ -sparsity,  $L_0$ -LSC correctly samples the posterior, which when averaged over the data matches the desired prior (Fig. 5c), as expected from theory. This is in contrast to non-stochastic algorithms where the inferred latent variable distribution often exhibits a more pronounced peak at zero compared to the prior. A more quantitative analysis is provided in Appendix B.

## 4.2 Learning the Dictionary Norm

For traditional sparse coding models such as DSC and LCA which update the dictionary based on a single MAP estimate for each data item, it is necessary to normalize the dictionary elements after each update. However if the update is based on samples from the posterior, as specified in equation (8), then this is no longer necessary. As a result, when using LSC, there is no need for normalization. Instead, the dictionary element norms  $\|A_i\|$  will automatically grow or shrink as needed to optimize the model log-likelihood.

The adaptive norm property can also be used to automatically select for the size of the dictionary. For data of dimension  $D$ , we consider a dictionary of size  $K = \Omega \times D$ , to have an (over)completeness of  $\Omega$ . A  $2\times$  overcomplete model was trained using the LSC algorithm using a fixed activation probability  $\pi$ , without normalizing the dictionary  $A$ . The resulting learned dictionary is shown in Fig. 6b. In previous work by (Berkes et al., 2008), Annealed Importance Sampling (AIS) (Neal,

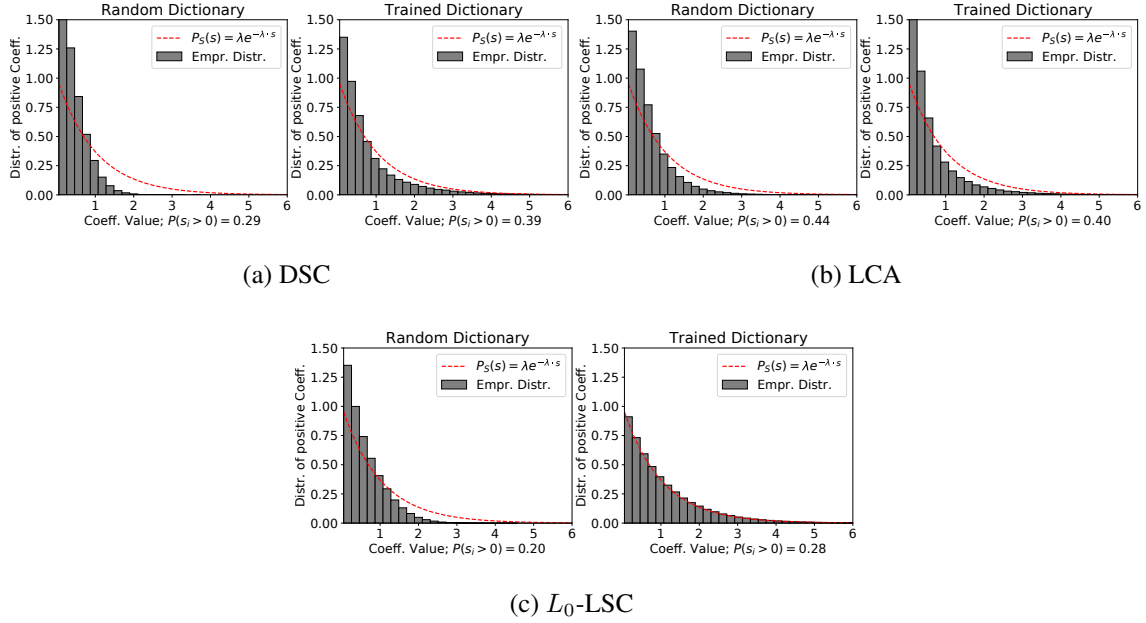


Figure 5: The distribution of non-zero coefficients of each of the three algorithms. The dotted red line shows the prior of coefficients used in generating the dataset. The left panel of each subfigure shows the empirical distribution when each algorithm is run with random dictionaries. The right panel shows the the distribution with learned dictionaries. Only  $L_0$ -LSC, with the correctly trained dictionary achieves the distribution matching the prior.

2001) was used to approximate the marginal likelihood in order to find the optimal dictionary elements. However,  $L_0$ -LSC, without additional procedures, can be used to effectively do the same through attenuation of unnecessary dictionary elements. The learned dictionary contains exactly the bars dictionary and the extra elements decay to nearly zero, as shown in Figure 6a.

When both  $\|\mathbf{A}_i\|$  and  $\pi$  are being learned, a more stable solution is to have duplicated dictionary elements with a reduced activity. This is shown in Figure 7a with a duplicated dictionary but halved activity (Fig. 7b).



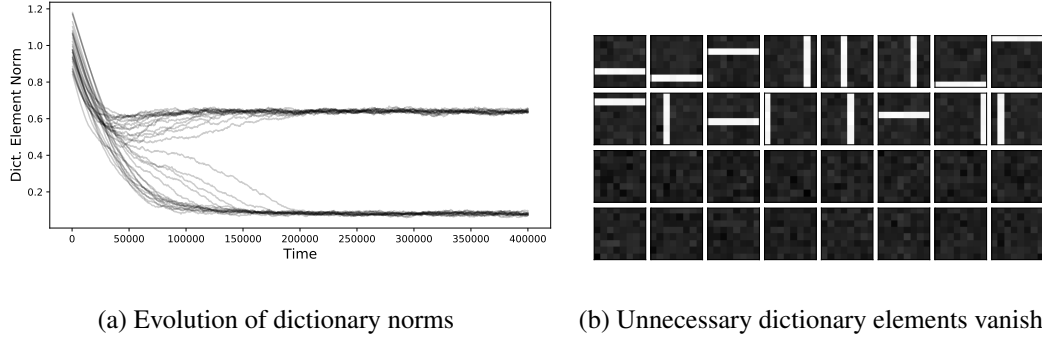


Figure 6: Learning the dictionary size. a) Dictionary norms bifurcate, with half decaying to nearly zero. b) The remaining elements contain exactly one copy of the dictionary elements used to generate the data.

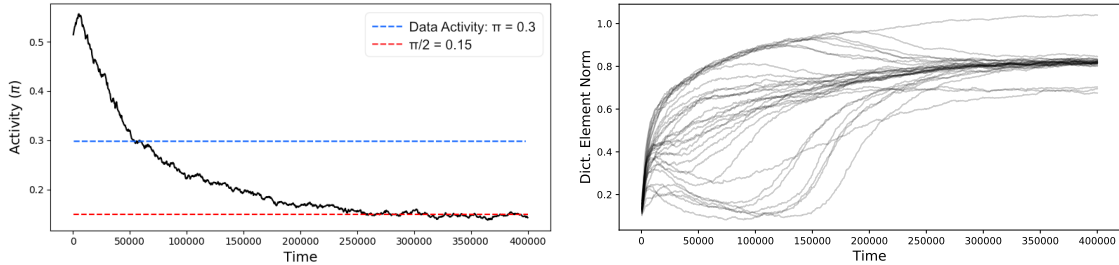
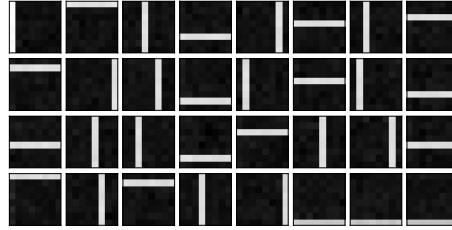


Figure 7: LSC is used to learn both the dictionary size and activation probability of the same  $2\times$  overcomplete model a) The learned dictionary now contains duplicated elements. b) But the activation probability  $\pi$  is half of the actual value used in generating the data.

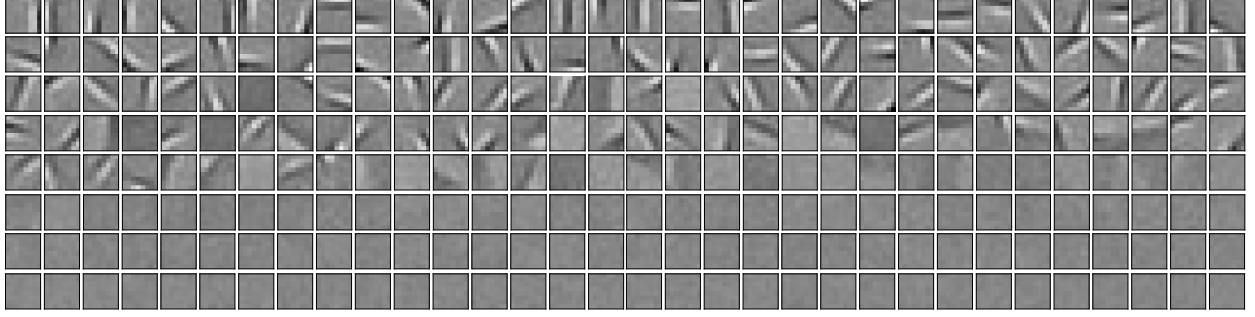


Figure 8: With activity fixed ( $\pi = 0.5$ ), only a fraction of the total dictionary elements have significant norm, the rest vanish. The dictionary elements are sorted by their respective norms.

### 4.3 Natural Image Patches

We ran the  $L_0$ -LSC algorithm on a dataset of  $8 \times 8$  image patches of whitened natural scenes from the Van Hateren dataset (Hateren & Schaaf, 1998; Olshausen, 2013). First, the model activity was fixed at  $\pi = 0.5$  and we used  $L_0$ -LSC to learn a  $4 \times$  overcomplete dictionary ( $K = 4 \times 64 = 256$ ). We can see in Figure 8 that a little more than half of the dictionary was utilized. The unused dictionary elements had a comparatively insignificant norm. In contrast to prior efforts to determine the optimal number of dictionary elements based on approximating the log-likelihood (Berkes et al., 2008), this result emerges directly from dictionary learning in Langevin sparse coding.

Then, unfixing  $\pi$ , we allow the activity to be learned. Repeating the experiment at different levels of overcompleteness  $\Omega$ , a correspondence between the activity and overcompleteness is plotted in Figure 9a. This relationship happens to be very well modeled by  $\pi \propto \Omega^{-1}$ . As a consequence, the expected number of active dictionary elements,  $\pi \times K = \pi \times \Omega \times D$  stays nearly constant irrespective of the overcompleteness  $\Omega$ .

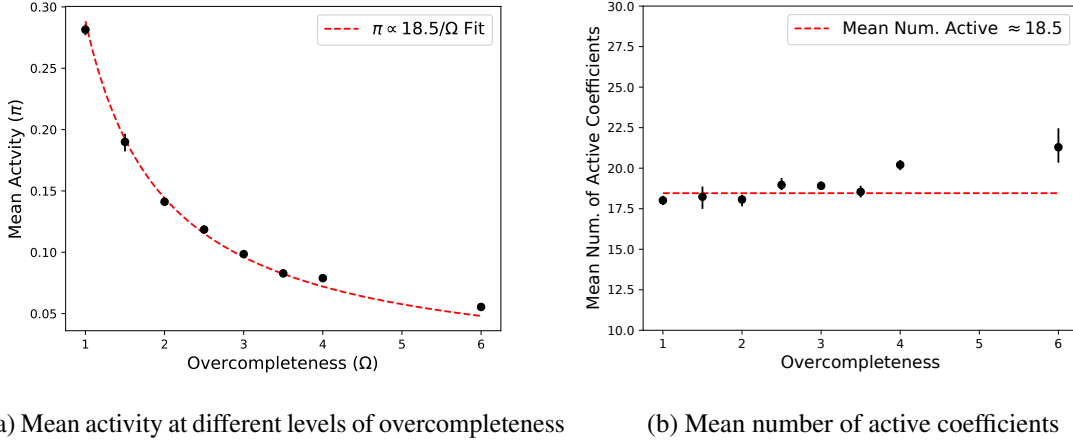


Figure 9: a) Using LSC to learn dictionaries for natural scenes at different levels of completeness  $\Omega$ , the relationship  $\pi \propto 1/\Omega$  is obtained. b) This implies that the mean number of dictionary elements used to code each image is constant irrespective of the total number of dictionary elements learned. Error bars on both plots denote the 10% - 90% range

## 5 Discussion

Our main contribution in this paper is to show that by using Langevin dynamics to sample from posterior distributions, we obtain a set of continuous-time equations over analog state variables that enable probabilistically correct inference and learning in a latent variable model. While the use of Langevin dynamics for sampling in probabilistic models *per se* is not new (Cheng et al., 2018), our emphasis here is to show how these dynamics play out in the case of the sparse coding model, and to point the way toward their efficient implementation in analog, electronic circuits that harness natural sources of stochasticity, for which we provide an example in Appendix C. The basic operations involve computing inner products, thresholding, lateral inhibition, and thresholding, in addition to injection of a Gaussian noise source. The first four of these are shared with LCA, for

which there already exist examples of both efficient analog implementations (Shapero et al., 2012; Sheridan et al., 2017), and digital implementation using spiking neurons (Davies et al., 2021). In the latter case LCA was shown to achieve the highest efficiency gains. The only additional component required for implementing LSC or  $L_0$ -LSC beyond these existing implementations is the injection of a Gaussian noise source. This would seem quite natural since noise is intrinsic to any physical system, however shaping the noise to be Gaussian and i.i.d., and whether this is strictly required, remain important issues to resolve.

Finding efficient implementations is key to making probabilistic models tractable and scalable to practical problems of interest such as image analysis. Indeed, latent variable models such as Boltzmann machines are often considered intractable due to the inner loop required to sample over hidden unit states conditioned on input data. For this reason, practitioners often turn to approximations such as restricted Boltzmann machines (RBM’s) (Hinton & Salakhutdinov, 2006) or variational inference (VAE’s) (Kingma & Welling, 2013) so as to make the problem tractable by eliminating “explaining away” – i.e., dependencies among hidden units conditioned on the data. But for most problems of interest in perception, explaining away is key (Olshausen, 2014). So doing away with explaining away in the interest of making the problem tractable simply dodges the very problem that needs to be solved. Here we show that there is alternative approach that tackles sampling from posteriors head on and makes it tractable via dynamics that could be naturally realized in a physical system.

An important next step will be to improve the efficiency of sampling by developing richer dynamical models. It is well known that the first-order Langevin dynamics we have utilized here can be slow to mix and reach equilibrium (Hennequin et al., 2014). Adding higher-order terms to the dynamics such as momentum or even third-order terms has been shown to dramatically

improve mixing time (Mou et al., 2021), and it has even been proposed that the balanced excitatory and inhibitory recurrent networks in cortex could serve such a function (Hennequin et al., 2014; Echeveste et al., 2020). The model we have proposed here could be modified along similar lines, and indeed this is a topic of ongoing work. Yet another route is to harness recent improvements in Hamiltonian Monte Carlo (Sohl-Dickstein et al., 2014).

With an efficient sampler in place, it becomes possible to adapt parameters of a sparse coding model beyond the dictionary, such as the level of sparsity or overcompleteness, which has not been possible in previous MAP-estimate based approaches. Furthermore, through application of a threshold function to the stochastic dynamics, *we demonstrate that inference with an  $L_0$ -sparse prior – which has been avoided in most approaches by using  $L_1$  as a proxy – can be readily computed and implemented* (Sec. 3). As shown in Section 4.1,  $L_0$ -LSC is better at sampling from the posterior distribution as well as capable of learning the activation probability  $\pi$  of the latent variables  $\mathbf{s}$ . In applying the model to natural images (Sec. 4.3), we found that *the mean number of dictionary elements used to encode an image is mostly invariant to the total dictionary size*. This runs counter to previous results (Olshausen, 2013) showing that, on average, the number of elements required for reconstructing a given image decreases with larger dictionaries in which the elements take on more specific and diverse shapes. This discrepancy could possibly be reconciled by the fact that the previous work utilized MAP-estimates rather than sampling, and so the learning was biased accordingly. Nonetheless, it is still intriguing that the mean number of dictionary elements in our case was near constant, suggesting that overcompleteness is an under-utilized degree of freedom. However, another likely culprit is the assumption of a factorial prior, and it may be that an overcomplete dictionary loses its explanatory power under such a prior. Thus, it will be important to consider group sparse coding or other approaches for modeling statistical dependencies

among latent variables (Garrigues & Olshausen, 2010, 2007) in order to fully realize the gains from overcompleteness.

Finally, another contribution of this work is to show how both learning and inference can be mapped to *simultaneous dynamics* at two different time scales. An underlying assumption in all implementations of probabilistic models on digital systems is the notion of a *global clock*. But the global clock is an impossibility for neural systems of any significant complexity. Our work presents an alternative approach to computing sparse coding which allows for simultaneous updates of both latent variables and model parameters such as the dictionary elements. This type of concurrent dynamics removes the need of any such global clock.

More generally, the mixed time-scale analog sampling framework on which LSC is based opens the way to learning richer generative models that capture dependencies among latent variables via horizontal connections (Garrigues & Olshausen, 2007) or via top-down priors (Boutin et al., 2020). And this goes beyond just sparse coding. In the future we hope to develop analogous procedures for learning other latent variable models such as Boltzmann machines and hierarchical Bayesian models (Lee & Mumford, 2003).

## A Time-scaling property of Langevin Dynamics

Consider the results of scaling the time variable  $t$  by a constant  $\tau$

$$\tilde{t} = \tau \cdot t. \quad (45)$$

Recall that the Gaussian white noise  $\xi(t)$  was normalized such that

$$\langle \xi(t) \xi(t') \rangle = I \delta(t - t'). \quad (46)$$

Using the new, scaled time, we have

$$\langle \xi(\tilde{t}) \xi(\tilde{t}') \rangle = I \delta(\tau(t - t')) \quad (47)$$

$$= \frac{1}{\tau} I \delta(t - t'). \quad (48)$$

If we define  $\tilde{\xi}(\tilde{t}) = \sqrt{\tau} \xi(\tau t)$ , we will recover

$$\langle \tilde{\xi}(\tilde{t}) \tilde{\xi}(\tilde{t}') \rangle = I \delta(t - t'). \quad (49)$$

## B Quantifying convergence to prior

To better quantify the convergence to the desired prior, we estimate the KL-divergence from  $p(s_i|\lambda)$ , the target prior to  $p(s_i|A)$  the learned prior based on dictionary  $A$ . Because the learned prior cannot be easily calculated, we rely on samples taken at regular time intervals. The samples are then binned in the same way that generated the histograms in (Fig. 5) .

$$D_{KL}(p(s|\lambda)||p(s|A)) = \left\langle \log \left( \frac{p(s|\lambda)}{p(s|A)} \right) \right\rangle_{s|\lambda} \quad (50)$$

$$\approx \sum_n p_n(\lambda) \log \left( \frac{p_n(\lambda)}{q_n(A)} \right) \quad (51)$$

where

$$p_n(\lambda) = P(n\delta s < s < (n+1)\delta s) \quad (52)$$

with  $\delta s$  being the bin width. Figure 10 shows the evolution of the estimated  $D_{KL}$  over time. As expected, only with LSC does the KL-divergence approach 0.

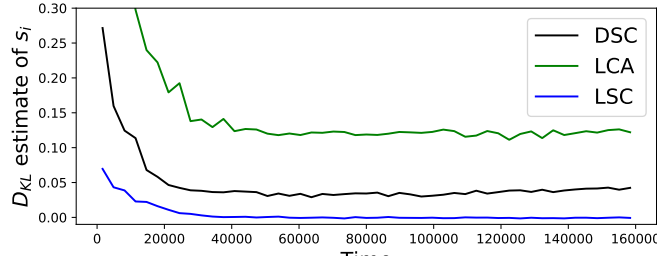


Figure 10: The KL-Divergence for coefficients  $s_i$  is compared for each of the three sparse coding methods. Only with LSC, does the  $D_{KL}$  approach zero.

## C Hardware Implementation

While the results presented above were obtained from simulation on a digital computer using the Euler-Maruyama algorithm, LSC was designed with stochastic analog implementation in mind. In this section, we present one candidate hardware implementation making use of Gilbert cells, a type of fast analog voltage multiplier (Gilbert, 1968).

The first goal is to design an analog circuit capable of simulating the coupled differential equations of continuous-time sparse coding (Eq. 25), explicitly

$$\tau_S \dot{\mathbf{s}} = -A^T(A\mathbf{s} - \mathbf{x}) - \lambda_1 \text{sgn}(\mathbf{s}) \quad (53)$$

$$\tau_A \dot{A} = -(A\mathbf{s} - \mathbf{x})\mathbf{s}^T \quad (54)$$



The core design challenge is to dynamically update  $\mathbf{s}$  through matrix multiplication with  $A^T A$ . Simultaneously, we require the dictionary elements  $A$  to also change in accordance with the value of  $\mathbf{s}$ . One promising approach is using grids of memristors, or programmable resistors (Di Ventra et al., 2009), which have been proposed as an analog implementation of generative adversarial networks (Krestinskaya et al., 2020). However, the limited endurance of memristors prevents extensive rewrites and ultimately a fully analog implementation (Krestinskaya et al., 2020). As an alternative, we propose using arrays of Gilbert cells for matrix multiplication. Because both inputs and the output are voltages, continuous dynamic updates are easily possible.

## C.A Gilbert Cell Matrix Multiplier

To focus on the matrix multiplication, we simplify Eqs. 53-54, at least initially, by ignoring the sparsity term, taking  $\lambda_1 = 0$ . We can also better organize the equations by introducing the reconstruction error variable,  $\Delta = A\mathbf{s} - \mathbf{x}$ . Finally, integrating the differential equations, we obtain

$$\Delta = A\mathbf{s} - \mathbf{x} \quad (55)$$

$$\mathbf{s} = -\tau_s^{-1} \int dt A^T \Delta \quad (56)$$

$$A = -\tau_A^{-1} \int dt \Delta \mathbf{s}^T \quad (57)$$

We represent each of the variables  $A, \mathbf{s}, \mathbf{x}, \Delta$  by proportional electric potentials.

$$V^{(A)} \propto A \quad (58)$$

$$V^{(S)} \propto \mathbf{s} \quad (59)$$

$$V^{(X)} \propto \mathbf{x} \quad (60)$$

Multiplication of matrices consists of element-wise multiplication which is facilitated by Gilbert cells and summation, which is facilitated through operational amplifiers.

A schematic of a Gilbert cell is shown in Figure 11. With inputs given as the voltage differences  $V_A \equiv V_A^+ - V_A^-$  and  $V_B \equiv V_B^+ - V_B^-$ , the multiplier produces an output proportional to the product of the inputs.

$$V_{out} = V_{out}^+ - V_{out}^- = \frac{1}{V_T} (V_A^+ - V_A^-) \cdot (V_B^+ - V_B^-) = \frac{V_A \cdot V_B}{V_T}. \quad (61)$$

The constant  $V_T$  depends on the design of the cell, choice of transistors, and other factors.

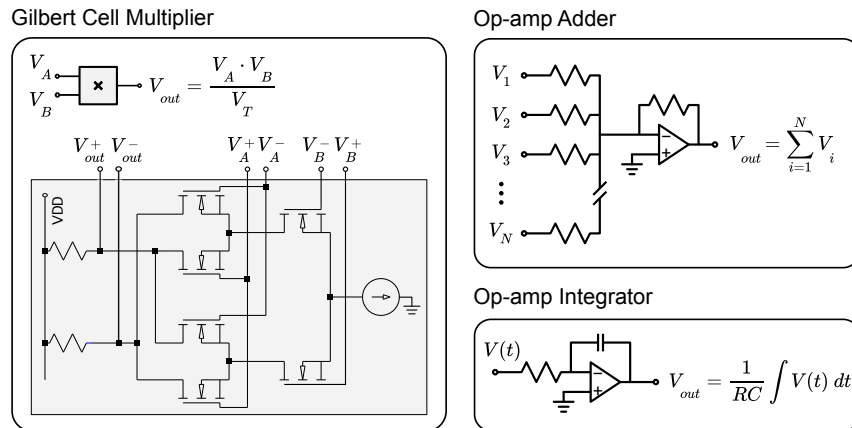


Figure 11: Circuit elements used for multiplication, summation and integration of voltages. The op-amp adder is used to add elements. The op-amp integrator is typically used to integrate over time.

Using operational amplifiers (op-amps) for summing electric potentials (Mancini, 2003) (Fig. 11), matrix multiplication of analog signals can be implemented with a grid of Gilbert cells. This is demonstrated in (Fig. 12a) specifically for  $\Delta = As - x$ . An array of nodes with potentials represent elements  $A$  are shown in red. Wires running horizontally carrying potentials representing

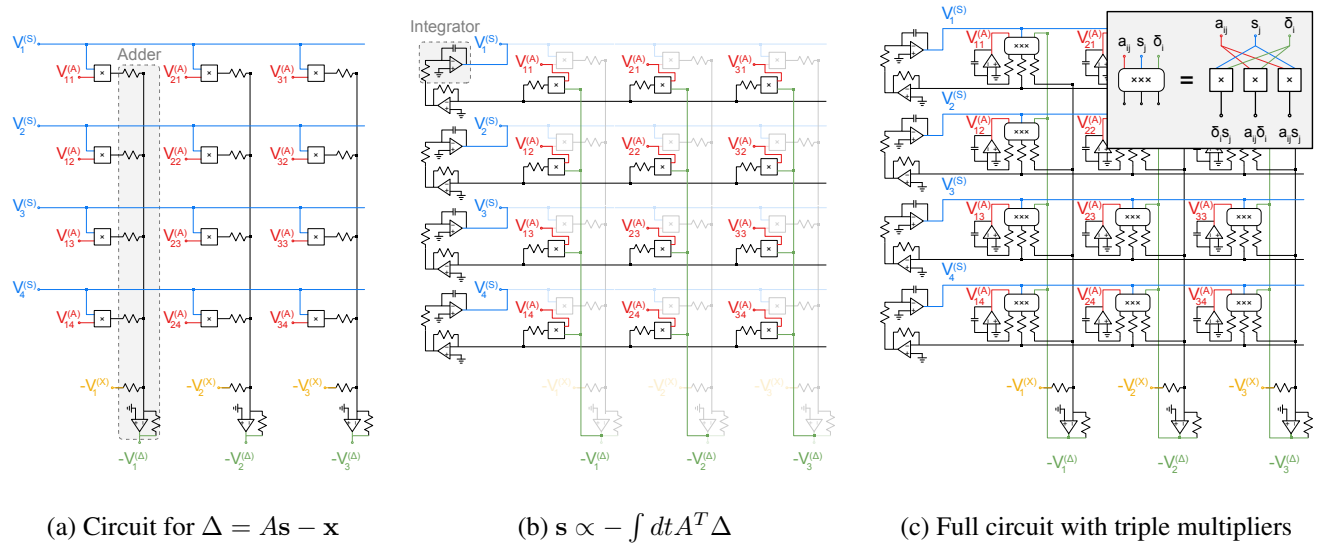


Figure 12: a) An analog matrix multiplier built from a grid of Gilbert cells and op-amp adders. Note the extra row of input  $V^{(X)}$  makes this technically an affine transformation. b) The output of the previous circuit,  $V^{(\Delta)}$ , is fed back to another set of multipliers and integrated. The new multiplier array are weaved into the existing circuit (which are faded to emphasize the newly added multipliers). c) With an array of triple multipliers, this circuit allows for the dictionary elements  $V_{ij}^{(A)}$  to be updated concurrent with the coefficients  $V_i^{(s)}$ . In practice, the Gilbert cells input and output potential differences. However, for brevity each pair of potentials are represented by a single node – see App. D for detailed circuit diagrams.

elements of  $s$  are shown in blue. Each Gilbert cell multiplies the two sets of input potentials to produce the output  $V_T^{-1} V_{ij}^{(A)} V_j^{(S)}$ . This output is then subsequently summed together by a series of op-amp adders shown running vertically in the figure. An extra row of inputs,  $-V_n^{(X)}$  accounts for the needed bias, making this an affine transformation (rather than a matrix multiplication). Finally

the output at the bottom of the figure is

$$V_i^{(\Delta)} = V_T^{-1} \sum_j V_{ij}^{(A)} V_j^{(S)} - V_i^{(X)}. \quad (62)$$

Recall that  $V_T$  is a constant which depends on specific Gilbert cell. This first circuit implements Eq. 55 as desired.

In a similar manner, an implementation of  $\tau_s \dot{s} = -A^T \Delta$  can be obtained. Because the voltages associated with matrix elements  $V_{ij}^{(A)}$  are already present, a second set of Gilbert cells can be woven into the previous circuit (Fig. 12b). Here, the potentials  $V_i^{(\Delta)}$  are propagated through vertical wires and with op-amp adders running horizontally, the resulting product is

$$-V_T^{-1} \sum_i V_{ij}^{(A)} V_i^{(\Delta)} = -V_T^{-1} \sum_i V_{ji}^{(A^T)} V_i^{(\Delta)} = -V_T^{-1} (V^{(A^T)} V^{(\Delta)})_j. \quad (63)$$

To integrate the above output, we make use of op-amp integrators (see. Fig. 11). Passing through a set of integrators and looping back to  $V_j^{(S)}$ , we obtain

$$V^{(S)} = -\tau_S \int dx V_T^{-1} V^{(A^T)} V^{(\Delta)} \quad (64)$$

The time constant is dependent on op-amp integrator (i.e.  $\tau_S = RC$ ) and can be adjusted accordingly. The newly introduced circuitry further enforces Eq. 56.

Lastly, for Eq. 57, another set of Gilbert cells to multiply  $\Delta_i$  and  $s_j$  is added. Its output is then integrated and fed back back into  $V_{ij}^{(A)}$ . Figure 12c presents the complete circuit for implementing the coupled equations 55 - 57. For conciseness, we introduce the triple multiplier comprising of three Gilbert cells (see Fig. 12c inset). With inputs of  $a_{ij}$ ,  $s_j$ ,  $\Delta_i$ , it outputs all pairwise products  $\delta_i s_j$ ,  $a_{ij} \Delta_i$  and  $a_{ij} s_j$ .

Note that in the complete circuit, the voltages  $V_j^{(S)}$ ,  $V_{ij}^{(A)}$ ,  $V_i^{(\Delta)}$  no longer are inputs to the system but rather represent “internal variables.” Only  $V_i^{(X)}$  is set externally and the potential at the remaining nodes evolve according to the coupled equations. The fact that the evolution of  $A$  and  $s$  requires neither external measurement nor global clocking is exactly the desired result sought out of the fully analog system. We further demonstrate in App. E that the triple multiplier array can be easily modified to accept asynchronous batched inputs.

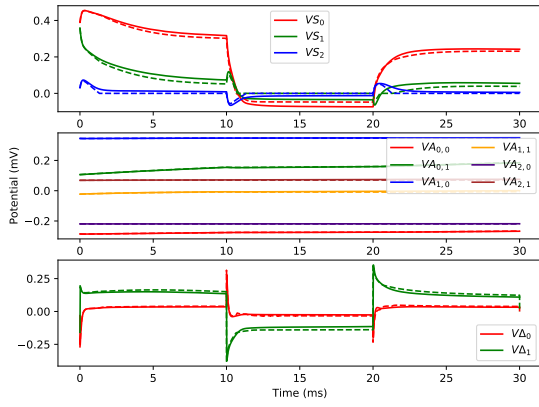
## C.B $L_1$ -SSC Circuit

With matrix multiplication accounted for, we return to the sparse penalty. Specifically, to implement the sign function in (Eq. 53), a high gain open loop op-amp is used as an comparator (App. D.A). The entire analog circuitry was drafted and simulated in LTSpice. The results are compared against solutions to (Eq. 25 - 26) and shown in Fig. 13a, 13b.

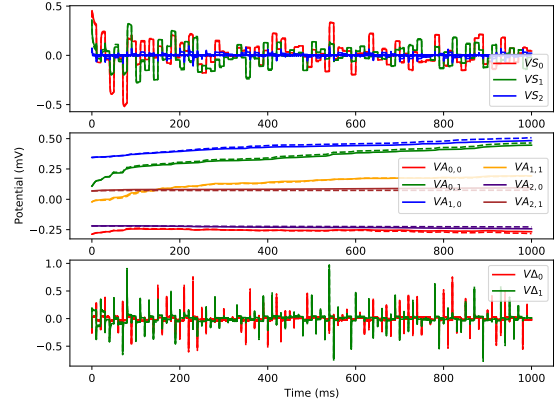
Over the course of one second, 100 inputs  $V^{(X)}$  were presented in 10ms intervals. We highlight the different response from different nodes in the circuit. The faster evolving coefficients  $V^{(S)}$  converge for each input within the short 10ms window. The slower evolving dictionary elements  $V^{(A)}$  exhibits slower and smoother dynamics. Finally, the reconstruction error  $V^{(\Delta)}$  spikes with each presentation of new input and tends towards zero. We see that the simulated circuit dynamics closely follows the theoretical solutions both on short time scales and long time scales.

## C.C $L_0$ -LSC Circuit

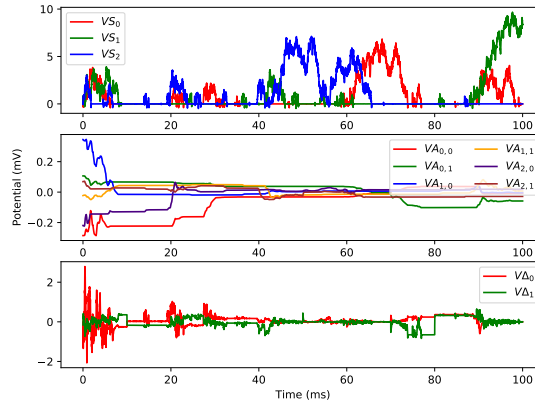
Lastly, we present the design of an analog circuit to implement  $L_0$ -LSC. Its dynamics are modeled by Eqs. 40, 42. Two major changes are required from the above  $L_1$ -SSC design. First is the



(a)  $L_1$ -SSC circuit operating for 30ms



(b)  $L_1$ -SSC circuit for 1s over 100 input batches



(c)  $L_0$ -LSC circuit over 100ms

Figure 13: Dynamics of potentials representing  $s$ ,  $A$  and  $\Delta$  in two analog circuits implementing sparse dictionary learning. The dotted lines depict simulated values and the solid lines are theoretical solutions. (Top) dynamics of latent variable coefficients. (Middle) dynamics of dictionary elements. (Bottom) dynamics of reconstruction error. a) The dynamics shown for a  $L_1$ -sparse SSC circuit. b) The same potentials in the circuit plotted for a longer interval of time where the evolution of  $A$  is more apparent. c) The dynamics shown for a  $L_0$ -LSC circuit

inclusion of a soft-threshold function and second, the injection of white noise. Details of both can be found in App. D.B.

Results from the Spice simulations are shown in Fig. 13c. Similar to  $L1$ -SSC, the circuit is characterized by two populations of fast evolving nodes  $V_i^{(s)}$  and slow evolving nodes  $V_{ij}^{(A)}$ .

While in the Spice simulation, white noise was directly added to the circuit, the ultimate aim is to leverage unavoidable, inherent noise of the circuit. An important direction for further research is the detailed characterization of noise from various electronic components. It has been demonstrated, in the context of photonic networks (Roques-Carmes et al., 2019), systems leveraging non-Gaussian sources of noise can converge to the same distribution as those with white noise.

## D Analog Circuit

The circuitry used in the LTSpice simulation is shown in Fig. 14. One of the triple multipliers is highlighted by the blue box. A integrator is represented by the circuit block in the red box.

Note that unlike the simplified circuit diagram shown in Fig. 12c, the multipliers act on differential voltages and also outputs two potentials  $V_-$ ,  $V_+$ . The triple multiplier, therefore takes in six inputs and produces six outputs. Because of this, the integrator operates on the difference in voltages and contains an subtracting op-amp before the actual integrator. In practices, this is achieved through a pair of cascading opamp circuits. The first, a differential amplifier, performs a subtraction and the second, an integrator amplifier continues with integration. This differential integrator, in purple, is modified in implementing  $L0$ - and  $L1$ -sparse penalties.

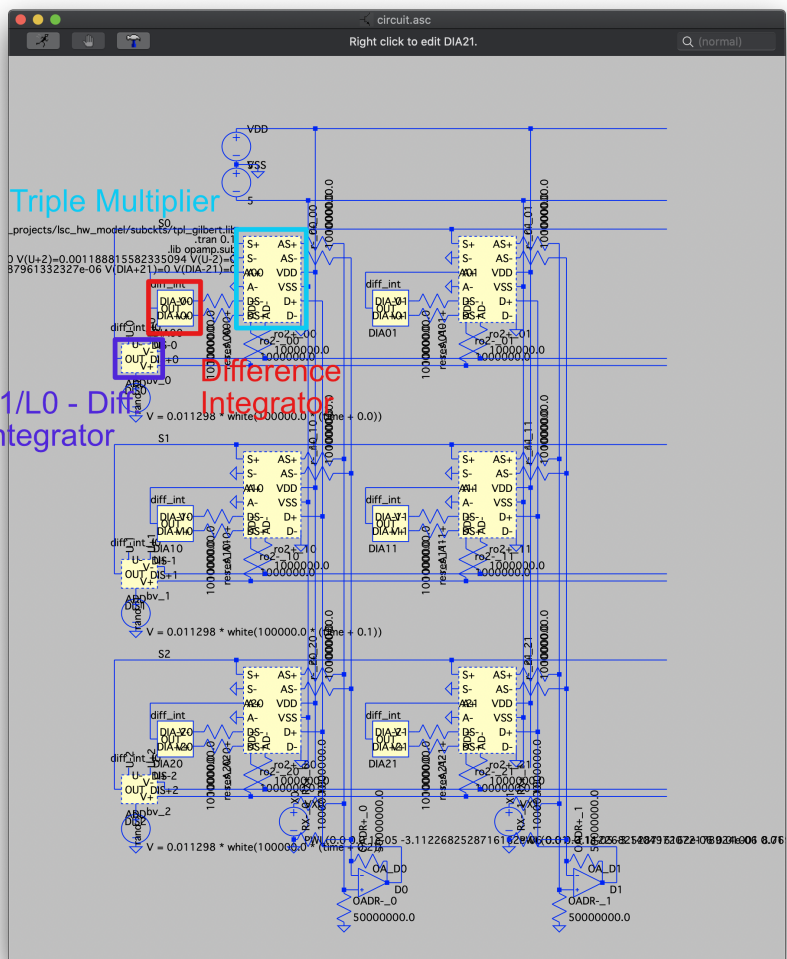
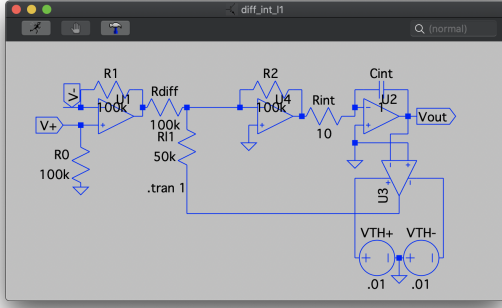
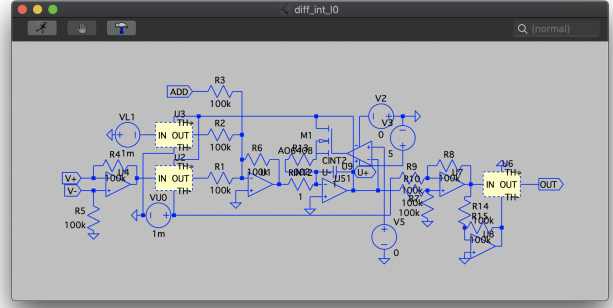


Figure 14: Circuit block diagram used in the LTSpice simulation.





(a) Difference integrator with L1 penalty



(b) Difference integrator with L0 penalty

Figure 15: The circuit diagram for both L1 and L0-sparse differential integrators. a) L1 sparse penalty is achieved by taking the sign function of the output and subtracting the initial input. Specifically, the sign function is implemented using an op-amp. b) The L0-sparse penalty is achieved mainly by passing the output through a threshold function ( $s = f(u)$ ).

## D.A $L_1$ -Sparse

With the L1-penalty, Eq. 55 - 57 becomes

$$\Delta = As - \mathbf{x} \quad (65)$$

$$\mathbf{s} = -\tau_s^{-1} \int dt A^T \Delta \quad (66)$$

$$A = -\tau_A^{-1} \int dt (\Delta \mathbf{s}^T + \lambda_1 \text{sgn}(\mathbf{s})) \quad (67)$$

with the addition of a  $\text{sgn}(\mathbf{s})$  term. The differential integrator circuit is modified accordingly. An open loop op-amp behaves as the sign function. The output is then summed with  $\Delta \mathbf{s}^T$  and subsequently integrated. The strength of the L1 penalty  $\lambda_1$  can be easily adjusted with the summation being a weighted sum.

## D.B $L_0$ -Sparse Circuit

The  $L_0$ -sparse set of equations is

$$\Delta = A\mathbf{s} - \mathbf{x} \quad (68)$$

$$\mathbf{u} = -\tau_s^{-1} \int dt A^T \Delta \quad (69)$$

$$A = -\tau_A^{-1} \int dt (\Delta \mathbf{s}^T + \lambda_1 \text{sgn}(\mathbf{u})) . \quad (70)$$

Recall that the new internal variable  $\mathbf{u}$  and  $\mathbf{s}$  are related via the threshold function  $s_i = f(u_i)$  (Eq. 34). In keeping with the conventions of the simulations, we restrict  $u_i$  to be positive, making the sign function unnecessary. The circuit shown above details the implementation of the threshold function. The potential  $u_0$  is subtracted from  $u_i$  and then a ReLU circuit (Agarap, 2018) is applied making use of the equivalent expression

$$f(u_i) = \max(0, u_i - u_0) \quad (71)$$

in cases where  $u_i \geq 0$ . Finally, to maintain  $u_i \geq 0$  despite the white noise, a transistor switch is added to short the capacitor of the integrator amplifier should  $u_i < 0$ .

## E Batched Circuit

In this section, we describe a simple modification to generalize the current circuit to incorporate batched inputs. The dynamics of SSC with input of batch size  $N$  can be written as

$$\tau_u \dot{\mathbf{u}}_{\mathbf{n}} = \nabla_{\mathbf{u}} E(A, \mathbf{u}_n, \mathbf{x}_n) \quad (72)$$

$$\tau_A \dot{A} = \frac{1}{N} \sum_{n=1}^N \nabla_A E(A, \mathbf{u}_n, \mathbf{x}_n). \quad (73)$$

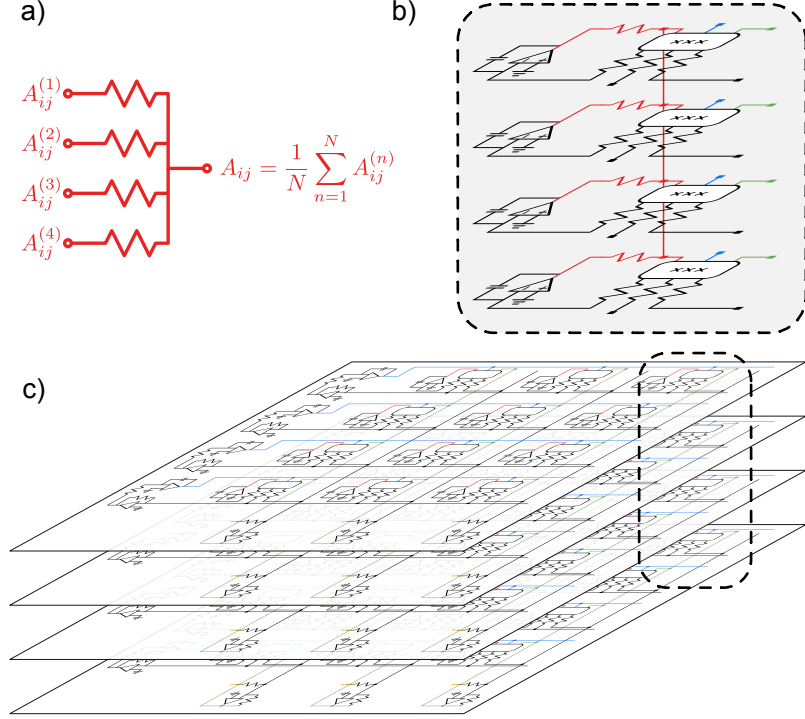


Figure 16: Circuit for batched update. a) A very simple passive averager is shown, where the central node has a potential equaling the mean of all the “inputs”. b) Incorporating this passive averager into a column of multipliers. c) A 3D circuit comprised of layers of the LSC circuit. Each layer corresponds to an individual input of a batch with its unique data  $\mathbf{x}_n$  and coefficients  $\mathbf{u}_n$  but with shared dictionary elements  $A$ .

We can implement the above dynamics by stacking  $N$  identical LSC circuits (see Fig. 16). Each layer would have its own data  $\mathbf{x}_n$  and latent variables  $\mathbf{u}_n$  and be updated independently. However, the dictionary elements  $A$  is shared among all layers and its update is averaged from each layer. A simple passive averaging circuit, shown in the figure below, can be used to link each of the layers.

## References

- Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive science*, 9(1), 147–169.
- Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.
- Beckett, S., Jee, J., Ncube, T., Pompilus, S., Washington, Q., Singh, A., & Pal, N. (2014). Zero-inflated poisson (zip) distribution: Parameter estimation and applications to model data from natural calamities. *Involve, a Journal of Mathematics*, 7(6), 751–767.
- Berkes, P., Orbán, G., Lengyel, M., & Fiser, J. (2011). Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331(6013), 83–87.
- Berkes, P., Turner, R., & Sahani, M. (2008). On sparsity and overcompleteness in image models. In *Advances in neural information processing systems* (pp. 89–96).
- Borders, W. A., Pervaiz, A. Z., Fukami, S., Camsari, K. Y., Ohno, H., & Datta, S. (2019). Integer factorization using stochastic magnetic tunnel junctions. *Nature*, 573(7774), 390–393.
- Boutin, V., Franciosini, A., Ruffier, F., & Perrinet, L. (2020). Effect of top-down connections in hierarchical sparse coding. *Neural Computation*, 32(11), 2279–2309.
- Bussi, G., & Parrinello, M. (2007). Accurate sampling using langevin dynamics. *Physical Review E*, 75(5), 056707.
- Cheng, X., Chatterji, N. S., Bartlett, P. L., & Jordan, M. I. (2018). Underdamped langevin mcmc: A non-asymptotic analysis. In *Conference on learning theory* (pp. 300–323).

- Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., ... Risbud, S. R. (2021). Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5), 911–934.
- Di Ventra, M., Pershin, Y. V., & Chua, L. O. (2009). Circuit elements with memory: memristors, memcapacitors, and meminductors. *Proceedings of the IEEE*, 97(10), 1717–1724.
- Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on information theory*, 52(4), 1289–1306.
- Echeveste, R., Aitchison, L., Hennequin, G., & Lengyel, M. (2020). Cortical-like dynamics in recurrent circuits optimized for sampling-based probabilistic inference. *Nature neuroscience*, 23(9), 1138–1149.
- Garrigues, P., & Olshausen, B. A. (2007). Learning horizontal connections in a sparse coding model of natural images. In *Nips* (pp. 505–512).
- Garrigues, P., & Olshausen, B. A. (2010). Group sparse coding with a laplacian scale mixture prior. In *Advances in neural information processing systems* (pp. 676–684).
- Gilbert, B. (1968). A precise four-quadrant multiplier with subnanosecond response. *IEEE journal of solid-state circuits*, 3(4), 365–373.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

- Hateren, J. H. v., & Schaaf, A. v. d. (1998, Mar). Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings: Biological Sciences*, 265(1394), 359-366.
- Hennequin, G., Aitchison, L., & Lengyel, M. (2014). Fast sampling-based inference in balanced neuronal networks. *Advances in neural information processing systems*, 27.
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786), 504–507.
- Hinton, G. E., & Sejnowski, T. J. (1983). Optimal perceptual inference. In *Proceedings of the ieee conference on computer vision and pattern recognition* (Vol. 448).
- Hoyer, P. O., & Hyvärinen, A. (2003). Interpreting neural response variability as monte carlo sampling of the posterior. In *Advances in neural information processing systems* (pp. 293–300).
- Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Krestinskaya, O., Choubey, B., & James, A. (2020). Memristive gan in analog. *Scientific reports*, 10(1), 1–14.
- Lee, T. S., & Mumford, D. (2003). Hierarchical bayesian inference in the visual cortex. *JOSA A*, 20(7), 1434–1448.
- Lewicki, M. S., & Olshausen, B. A. (1999). Probabilistic framework for the adaptation and comparison of image codes. *JOSA A*, 16(7), 1587–1601.
- Mancini, R. (2003). *Op amps for everyone: design reference*. Newnes.

- Mansinghka, V., & Jonas, E. (2014). Building fast bayesian computing machines out of intentionally stochastic, digital parts. *arXiv preprint arXiv:1402.4914*.
- Mansinghka, V. K., Jonas, E. M., & Tenenbaum, J. B. (2008). Stochastic digital circuits for probabilistic inference. *Massachusetts Institute of Technology, Technical Report MITCSAIL-TR, 2069*.
- Mead, C. A., & Mahowald, M. A. (1988). A silicon model of early visual processing. *Neural networks, 1*(1), 91–97.
- Mitchell, T. J., & Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the american statistical association, 83*(404), 1023–1032.
- Mou, W., Ma, Y.-A., Wainwright, M. J., Bartlett, P. L., & Jordan, M. I. (2021). High-order langevin diffusion yields an accelerated mcmc algorithm. *J. Mach. Learn. Res.*, 22, 42–1.
- Neal, R. M. (2001). Annealed importance sampling. *Statistics and computing, 11*(2), 125–139.
- Olshausen, B. A. (2013). Highly overcomplete sparse coding. In *Human vision and electronic imaging xviii* (Vol. 8651, p. 86510S).
- Olshausen, B. A. (2014). Perception as an inference problem. In G. Mangun & M. Gazzaniga (Eds.), *The cognitive neurosciences* (chap. 27). MIT Press, Cambridge, MA.
- Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research, 37*(23), 3311–3325.
- Olshausen, B. A., & Millman, K. J. (2000). Learning sparse codes with a mixture-of-gaussians prior. In *Advances in neural information processing systems* (pp. 841–847).

- Orbán, G., Berkes, P., Fiser, J., & Lengyel, M. (2016). Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron*, 92(2), 530–543.
- Roques-Carnes, C., Shen, Y., Zanoci, C., Prabhu, M., Atieh, F., Jing, L., . . . others (2019). Photonic recurrent ising sampler. In *Cleo: Qels fundamental science* (pp. FTu4C–2).
- Rozell, C. J., Johnson, D. H., Baraniuk, R. G., & Olshausen, B. A. (2008). Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10), 2526–2563.
- Shapero, S., Charles, A. S., Rozell, C. J., & Hasler, P. (2012). Low power sparse approximation on reconfigurable analog hardware. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(3), 530–541.
- Shelton, J. A., Sheikh, A. S., Berkes, P., Bornschein, J., & Lücke, J. (2011). Select and sample-a model of efficient neural inference and learning. In *Advances in neural information processing systems* (pp. 2618–2626).
- Sheridan, P. M., Cai, F., Du, C., Ma, W., Zhang, Z., & Lu, W. D. (2017). Sparse coding with memristor networks. *Nature nanotechnology*, 12(8), 784.
- Sohl-Dickstein, J., Mudigonda, M., & DeWeese, M. (2014). Hamiltonian monte carlo without detailed balance. In *International conference on machine learning* (pp. 719–726).
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267–288.
- Tropp, J. A. (2006). Algorithms for simultaneous sparse approximation. part ii: Convex relaxation. *Signal Processing*, 86(3), 589–602.



- Wang, Z., Yang, J., Zhang, H., Wang, Z., Huang, T. S., Liu, D., & Yang, Y. (2015). *Sparse coding and its applications in computer vision*. World Scientific.
- Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 681–688).
- Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T. S., & Yan, S. (2010). Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6), 1031–1044.