Convolutional dictionary learning based auto-encoders for natural exponential-family distributions

Bahareh Tolooshams [*1] **Andrew H. Song** [*2] **Simona Temereanca** [3] **Demba Ba** [1]

Abstract

We introduce a class of auto-encoder neural networks tailored to data from the natural exponential family (e.g., count data). The architectures are inspired by the problem of learning the filters in a convolutional generative model with sparsity constraints, often referred to as convolutional dictionary learning (CDL). Our work is the first to combine ideas from convolutional generative models and deep learning for data that are naturally modeled with a non-Gaussian distribution (e.g., binomial and Poisson). This perspective provides us with a scalable and flexible framework that can be re-purposed for a wide range of tasks and assumptions on the generative model. Specifically, the iterative optimization procedure for solving CDL, an unsupervised task, is mapped to an unfolded and constrained neural network, with iterative adjustments to the inputs to account for the generative distribution. We also show that the framework can easily be extended for discriminative training, appropriate for a supervised task. We 1) demonstrate that fitting the generative model to learn, in an unsupervised fashion, the latent stimulus that underlies neural spiking data leads to better goodness-of-fit compared to other baselines, 2) show competitive performance compared to state-of-the-art algorithms for supervised Poisson image denoising, with significantly fewer parameters, and 3) characterize the gradient dynamics of the shallow binomial auto-encoder.

[*]Equal contribution [1]School of Engineering and Applied Sciences, Harvard University, Cambridge, MA [2]Massachusetts Institute of Technology, Cambridge, MA [3]Brown University, Boston, MA. Correspondence to: Bahareh Tolooshams <btolooshams@seas.harvard.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

1. Introduction

Learning shift-invariant patterns from a dataset has given rise to work in different communities, most notably in signal processing (SP) and deep learning. In the former, this problem is referred to as convolutional dictionary learning (CDL) (Garcia-Cardona & Wohlberg, 2018). CDL imposes a linear *generative model* where the data are generated by a sparse linear combination of shifts of localized patterns. In the latter, convolutional neural networks (NNs) (LeCun et al., 2015) have excelled in identifying shift-invariant patterns.

Recently, the iterative nature of the optimization algorithms for performing CDL has inspired the utilization of NNs as an efficient and scalable alternative, starting with the seminal work of (Gregor & Lecun, 2010), and followed by (Tolooshams et al., 2018; Sreter & Giryes, 2018; Sulam et al., 2019). Specifically, the iterative steps are expressed as a recurrent NN, and thus solving the optimization simply becomes passing the input through an unrolled NN (Hershey et al., 2014; Monga et al., 2019). At one end of the spectrum, this perspective, through weight-tying, leads to architectures with significantly fewer parameters than a generic NN. At the other end, by untying the weights, it motivates new architectures that depart, and could not be arrived at, from the generative perspective (Gregor & Lecun, 2010; Sreter & Giryes, 2018).

The majority of the literature at the intersection of generative models and NNs assumes that the data are real-valued and therefore are not appropriate for binary or count-valued data, such as neural spiking data and photon-based images (Yang et al., 2011). Nevertheless, several works on Poisson image denoising, arguably the most popular application involving non real-valued data, can be found separately in both communities. In the SP community, the negative Poisson data likelihood is either explicitly minimized (Salmon et al., 2014; Giryes & Elad, 2014) or used as a penalty term added to the objective of an image denoising problem with Gaussian noise (Ma et al., 2013). Being rooted in the dictionary learning formalism, these methods operate in an *unsupervised* manner. Although they yield good denoising performance, their main drawbacks are scalability and computational efficiency.

In the deep learning community, NNs tailored to image denoising (Zhang et al., 2017; Remez et al., 2018; Feng et al., 2018), which are reminiscent of residual learning, have shown great performance on Poisson image denoising. However, since these 1) are not designed from the generative model perspective and/or 2) are *supervised* learning frameworks, it is unclear how they can be adapted to the classical CDL, where the task is *unsupervised* and the interpretability of the parameters is important. NNs with a generative flavor, namely variational auto-encoders (VAEs), have been extended to utilize non real-valued data (Nazábal et al., 2020; Liang et al., 2018). However, these architectures cannot be adapted to solve the CDL task.

To address this gap, we make the following contributions[1]:

Auto-encoder inspired by CDL for non real-valued data We introduce a flexible class of auto-encoder (AE) architectures for data from the natural exponential-family that combines the perspectives of generative models and NNs. We term this framework, depicted in Fig. 1, the deep convolutional exponential-family auto-encoder (DCEA).

Unsupervised learning of convolutional patterns We show through simulation that DCEA performs CDL and learns convolutional patterns from binomial observations. We also apply DCEA to real neural spiking data and show that it fits the data better than baselines.

Supervised learning framework DCEA, when trained in a supervised manner, achieves similar performance to state-of-the-art algorithms for Poisson image denoising with orders of magnitude fewer parameters compared to other baselines, owing to its design based on a generative model.

Gradient dynamics of shallow exponential auto-encoder Given some assumptions on the binomial generative model with dense dictionary and "good" initializations, we prove in Theorem 1 that shallow exponential auto-encoder (SEA), when trained by gradient descent, recovers the dictionary.

2. Problem Formulation

Natural exponential-family distribution For a given observation vector $\mathbf{y} \in \mathbb{R}^N$, with mean $\boldsymbol{\mu} \in \mathbb{R}^N$, we define the log-likelihood of the *natural exponential family* (McCullagh & Nelder, 1989) as

$$\log p(\mathbf{y}|\boldsymbol{\mu}) = f\big(\boldsymbol{\mu}\big)^{\mathrm{T}}\mathbf{y} + g(\mathbf{y}) - B\big(\boldsymbol{\mu}\big), \tag{1}$$

where we have assumed that, conditioned on $\boldsymbol{\mu}$, the elements of $\mathbf{y}$ are independent. The natural exponential family includes a broad family of probability distributions such as the Gaussian, binomial, and Poisson. The functions $g(\cdot)$, $B(\cdot)$, as well as the invertible *link function* $f(\cdot)$, all depend on the choice of distribution.

[1]The code can be found at https://github.com/btolooshams/dea

Convolutional generative model We assume that $f(\boldsymbol{\mu})$ is the sum of scaled and time-shifted copies of C finite-length filters (dictionary) $\{\mathbf{h}_c\}_{c=1}^C \in \mathbb{R}^K$, each localized, i.e., $K \ll N$. We can express $f(\boldsymbol{\mu})$ in a convolutional form: $f(\boldsymbol{\mu}) = \sum_{c=1}^C \mathbf{h}_c * \mathbf{x}^c$, where $*$ is the convolution operation, and $\mathbf{x}^c \in \mathbb{R}^{N-K+1}$ is a train of scaled impulses which we refer to as *code vector*. Using linear-algebraic notation, $f(\boldsymbol{\mu}) = \sum_{c=1}^C \mathbf{h}_c * \mathbf{x}^c = \mathbf{H}\mathbf{x}$, where $\mathbf{H} \in \mathbb{R}^{N \times C(N-K+1)}$ is a matrix that is the concatenation of C Toeplitz (i.e., banded circulant) matrices $\mathbf{H}^c \in \mathbb{R}^{N \times (N-K+1)}$, $c = 1, \dots, C$, and $\mathbf{x} = [(\mathbf{x}^1)^{\mathrm{T}}, \dots, (\mathbf{x}^C)^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{C(N-K+1)}$.

We refer to the input/output domain of $f(\cdot)$ as the data and dictionary domains, respectively. We interpret $\mathbf{y}$ as a time-series and the non-zero elements of $\mathbf{x}$ as the times when each of the C filters are active. When $\mathbf{y}$ is two-dimensional (2D), i.e., an image, $\mathbf{x}$ encodes the spatial locations where the filters contribute to its mean $\boldsymbol{\mu}$.

Exponential convolutional dictionary learning (ECDL) Given J observations $\{\mathbf{y}^j\}_{j=1}^J$, we estimate $\{\mathbf{h}_c\}_{c=1}^C$ and $\{\mathbf{x}^j\}_{j=1}^J$ that minimize the negative log-likelihood $\sum_{j=1}^J l(\mathbf{x}^j) = -\sum_{j=1}^J \log p(\mathbf{y}^j|\{\mathbf{h}_c\}_{c=1}^C, \mathbf{x}^j)$ under the convolutional generative model, subject to sparsity constraints on $\{\mathbf{x}^j\}_{j=1}^J$. We enforce sparsity using the ℓ_1 norm, which leads to the non-convex optimization problem

$$\min_{\substack{\{\mathbf{h}_c\}_{c=1}^C \\ \{\mathbf{x}^j\}_{j=1}^J}} \sum_{j=1}^J \overbrace{-(\mathbf{H}\mathbf{x}^j)^{\mathrm{T}}\mathbf{y}^j + B\big(f^{-1}\big(\mathbf{H}\mathbf{x}^j\big)\big)}^{l(\mathbf{x}^j)} + \lambda\|\mathbf{x}^j\|_1, \tag{2}$$

where the regularizer λ controls the degree of sparsity. A popular approach to deal with the non-convexity is to minimize the objective over one set of variables, while the others are fixed, in an alternating manner, until convergence (Agarwal et al., 2016). When $\{\mathbf{x}^j\}_{j=1}^J$ is being optimized with fixed $\mathbf{H}$, we refer to the problem as convolutional sparse coding (CSC). When $\mathbf{H}$ is being optimized with $\{\mathbf{x}^j\}_{j=1}^J$ fixed, we refer to the problem as convolutional dictionary update (CDU).

3. Deep convolutional exponential-family auto-encoder

We propose a class of auto-encoder architectures to solve the ECDL problem, which we term deep convolutional exponential-family auto-encoder (DCEA). Specifically, we make a one-to-one connection between the CSC/CDU steps and the encoder/decoder of DCEA depicted in Fig. 1. We focus only on CSC for a single $\mathbf{y}^j$, as the CSC step can be parallelized across examples.

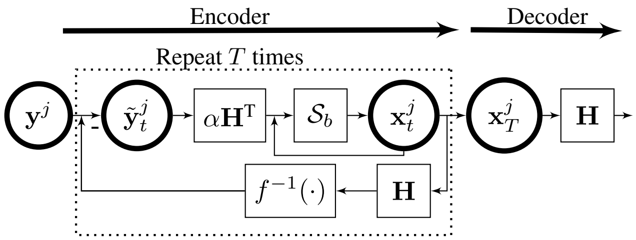

Figure 1. DCEA architecture for ECDL. The encoder/decoder structure mimics the CSC/CDU sequence in CDL. The encoder performs T iterations of CSC. Each iteration uses *working observations* (residuals) $\widetilde{\mathbf{y}}_t^j$ obtained by iteratively modifying $\mathbf{y}^j$ using the filters $\mathbf{H}$ and a nonlinearity $f^{-1}(\cdot)$ that depends on the distribution of $\mathbf{y}^j$. The dictionary $\mathbf{H}$ is updated through the backward pass.

3.1. The architecture

Encoder The forward pass of the encoder maps the input $\mathbf{y}^j$ into the sparse code $\mathbf{x}^j$. Given the filters $\{\mathbf{h}_c\}_{c=1}^C$, the encoder solves the ℓ_1-regularized optimization problem from Eq. (2),

$$\min_{\mathbf{x}^j} l(\mathbf{x}^j) + \lambda\|\mathbf{x}^j\|_1, \tag{3}$$

in an iterative manner by unfolding T iterations of the proximal gradient algorithm (Parikh & Boyd, 2014). For Gaussian observations, Eq. (3) becomes an ℓ_1-regularized least squares problem, for which several works have unfolded the proximal iteration into a recurrent network (Gregor & Lecun, 2010; Wang et al., 2015; Sreter & Giryes, 2018; Tolooshams et al., 2020).

We use $\mathcal{S}_b \in \{\mathrm{ReLU}_b, \mathrm{Shrinkage}_b\}$ to denote a proximal operator with bias $b \geq 0$. We consider three operators,

$$\begin{aligned} \mathrm{ReLU}_b(\mathbf{z}) &= (\mathbf{z} - b) \cdot \mathbb{1}_{\{\mathbf{z} \geq b\}} \\ \mathrm{Shrinkage}_b(\mathbf{z}) &= \mathrm{ReLU}_b(\mathbf{z}) - \mathrm{ReLU}_b(-\mathbf{z}), \end{aligned} \tag{4}$$

where $\mathbb{1}$ is an indicator function. If we constrain the entries of $\mathbf{x}^j$ to be non-negative, we use $\mathcal{S}_b = \mathrm{ReLU}_b$. Otherwise, we use $\mathcal{S}_b = \mathrm{Shrinkage}_b$. A single iteration of the proximal gradient step is given by

$$\begin{aligned} \mathbf{x}_t^j &= \mathcal{S}_b\left(\mathbf{x}_{t-1}^j - \alpha\nabla_{\mathbf{x}_{t-1}^j} \log p\big(\mathbf{y}^j|\{\mathbf{h}_c\}_{c=1}^C, \mathbf{x}_{t-1}^j\big)\right) \\ &= \mathcal{S}_b\big(\mathbf{x}_{t-1}^j + \alpha\mathbf{H}^{\mathrm{T}}(\underbrace{\mathbf{y} - f^{-1}\big(\mathbf{H}\mathbf{x}_{t-1}^j\big)}_{\widetilde{\mathbf{y}}_t^j})\big), \end{aligned} \tag{5}$$

where $\mathbf{x}_t^j$ denotes the sparse code after t iterations of unfolding, and α is the step size of the gradient update. The term $\widetilde{\mathbf{y}}_t^j$ is referred to as *working observation*. The choice of α, which we explore next, depends on the generative distribution. We also note that there is a one-to-one mapping between the regularization parameter λ and the bias b of $\mathcal{S}_b$. We treat λ, and therefore b, as hyperparameters that we tune to the desired sparsity level. The matrix $\mathbf{H}^{\mathrm{T}}$ effectively computes the correlation between $\widetilde{\mathbf{y}}_t^j$ and $\{\mathbf{h}_c\}_{c=1}^C$. Assuming that we unfold T times, the output of the encoder is $\mathbf{x}_T^j$.

The architecture consists of two nonlinear activation functions: $\mathcal{S}_b(\cdot)$ to enforce sparsity, and $f^{-1}(\cdot)$, the inverse of the link function. For Gaussian observations $f^{-1}(\cdot)$ is linear with slope 1. For other distributions in the natural exponential family, the encoder uses $f^{-1}(\cdot)$, a mapping from the dictionary domain to the data domain, to transform the input $\mathbf{y}^j$ at each iteration into a working observation $\widetilde{\mathbf{y}}_t^j$.

Decoder & training We apply the decoder $\mathbf{H}$ to $\mathbf{x}_T^j$ to obtain the linear predictor $\mathbf{H}\mathbf{x}_T^j$. This decoder completes the forward pass of DCEA, also summarized in Algorithm 2 of the **Appendix**. We use the negative log-likelihood $\mathcal{L}_{\mathbf{H}}^{\text{unsup.}} = \sum_{j=1}^J l(\mathbf{x}^j)$ as the loss function applied to the decoder output for updating the dictionary. We train the weights of DCEA, fully specified by the filters $\{\mathbf{h}_c\}_{c=1}^C$, by gradient descent through backpropagation. Note that the ℓ_1 penalty is not a function of $\mathbf{H}$ and is not in the loss function.

3.2. Binomial and Poisson generative models

We focus on two representative distributions for the natural exponential family: binomial and Poisson. For the binomial distribution, $\mathbf{y}^j$ assumes integer values from 0 to M_j. For the Poisson distribution, $\mathbf{y}^j$ can, in principle, be any non-negative integer values, although this is rare due to the exponential decay of the likelihood for higher-valued observations. Table 1 summarizes the relevant parameters for these distributions.

The fact that binomial and Poisson observations are integer-valued and have limited range, whereas the underlying $\boldsymbol{\mu}_j = f^{-1}(\mathbf{H}\mathbf{x}^j)$ is real-valued, makes the ECDL challenging. This is compounded by the nonlinearity of $f^{-1}(\cdot)$, which distorts the error in the *data* domain, when mapped to the *dictionary* domain. In comparison, in Gaussian CDL 1) the observations are real-valued and 2) $f^{-1}(\cdot)$ is linear.

This implies that, for successful ECDL, $\mathbf{y}^j$ needs to assume a diverse set of integer values. For the binomial distribution, this suggests that M_j should be large. For Poisson, as well as binomial, the maximum of $\boldsymbol{\mu}_j$ should also be large. This explains why the performance is generally lower in Poisson image denoising for a lower peak, where the peak is defined as the maximum value of $\boldsymbol{\mu}_j$ (Giryes & Elad, 2014).

Practical design considerations for architecture As the encoder of DCEA performs iterative proximal gradient steps, we need to ensure that $\mathbf{x}_T^j$ converges. Convergence analysis of ISTA (Beck & Teboulle, 2009) shows that if $l(\mathbf{x}^j)$ is convex and has L–Lipschitz continuous gradient, which loosely means the Hessian is upper-bounded everywhere by $L > 0$, choosing $\alpha \in (0, 1/L]$ guarantees convergence. For the

Table 1. Generative models for DCEA.

	$\mathbf{y}^j$	$f^{-1}(\cdot)$	$B(\mathbf{z})$	$\widetilde{\mathbf{y}}_t^j$	$\mathbf{x}_t^j$
Gaussian	$\mathbb{R}$	$I(\cdot)$	$\mathbf{z}^{\mathrm{T}}\mathbf{z}$	$\mathbf{y}^j - \mathbf{H}\mathbf{x}_{t-1}^j$	$\mathcal{S}_b\left(\mathbf{x}_{t-1}^j + \alpha\mathbf{H}^{\mathrm{T}}\widetilde{\mathbf{y}}_t^j\right)$
Binomial	$[0..M_j]$	sigmoid$(\cdot)$	$-\mathbf{1}^{\mathrm{T}}\log(\mathbf{1}-\mathbf{z})$	$\mathbf{y}^j - M_j \cdot \text{sigmoid}(\mathbf{H}\mathbf{x}_{t-1}^j)$	$\mathcal{S}_b\left(\mathbf{x}_{t-1}^j + \alpha\mathbf{H}^{\mathrm{T}}(\frac{1}{M_j}\widetilde{\mathbf{y}}_t^j)\right)$
Poisson	$[0..\infty)$	$\exp(\cdot)$	$\mathbf{1}^{\mathrm{T}}\mathbf{z}$	$\mathbf{y}^j - \exp(\mathbf{H}\mathbf{x}_{t-1}^j)$	$\mathcal{S}_b\left(\mathbf{x}_{t-1}^j + \alpha\mathbf{H}^{\mathrm{T}}\left(\text{Elu}(\widetilde{\mathbf{y}}_t^j)\right)\right)$

Gaussian distribution, L is the square of the largest singular value of $\mathbf{H}$ (Daubechies et al., 2004), denoted $\sigma^2_{\max}(\mathbf{H})$, and therefore $\alpha \in (0, 1/\sigma^2_{\max}(\mathbf{H})]$. For the Binomial distribution, $L = \frac{1}{4}\sigma^2_{\max}(\mathbf{H})$, and therefore $\alpha \in (0, 4/\sigma^2_{\max}(\mathbf{H})]$.

The gradient of the Poisson likelihood is not Lipschitz continuous. Therefore, we cannot set α a priori. In practice, the step size at every iteration is determined through a backtracking line search (Boyd & Vandenberghe, 2004), a process that is not trivial to replicate in the forward pass of a NN. We observed that the lack of a principled approach for picking α results, at times, in the residual assuming large negative values, leading to instabilities. Therefore, we imposed a finite lower-bound on the residual through an exponential linear unit (Elu) (Clevert et al., 2016) which, empirically, we found to work the best compared to other nonlinear activation units. The convergence properties of this approach require further theoretical analysis that is outside of the scope of this paper.

4. Connection to unsupervised/supervised paradigm

We now analyze the connection between DCEA and ECDL. We first examine how the convolutional generative model places constraints on DCEA. Then, we provide a theoretical justification for using DCEA for ECDL by proving dictionary recovery guarantees under some assumptions. Finally, we explain how DCEA can be modified for a supervised task.

4.1. Constrained structure of DCEA

We discuss key points that allow DCEA to perform ECDL.

- **Linear 1-layer decoder** In ECDL, the only sensible decoder is a one layer decoder comprising $\mathbf{H}$ and a linear activation. In contrast, the decoder of a typical AE consists of multiple layers along with nonlinear activations.
- **Tied weights across encoder and decoder** Although our encoder is *deep*, the same weights ($\mathbf{H}$ and $\mathbf{H}^{\mathrm{T}}$) are repeated across the layers.
- **Alternating minimization** The forward pass through the encoder performs the CSC step, and the backward pass, via backpropagation, performs the CDU step.

4.2. Theory for unsupervised ECDL task

Here, we prove that under some assumptions, when training the shallow exponential-family auto-encoder by approximate gradient descent, the network recovers the dictionary corresponding to the binomial generative model. Consider a SEA, where the encoder is unfolded only once. Recent work (Nguyen et al., 2019) has shown that, using observations from a sparse *linear* model with a *dense* dictionary, the Gaussian SEA trained by approximate gradient descent recovers the dictionary. We prove a similar result for the binomial SEA trained with binomial observations with mean $f(\boldsymbol{\mu}) = \mathbf{A}\mathbf{x}^*$, i.e., a sparse *nonlinear* generative model with a *dense* dictionary $\mathbf{A} \in \mathbb{R}^{n\times p}$.

Theorem 1 *Suppose the generative model satisfies (A1) - (A14) (see* **Appendix***). Given infinitely many examples (i.e., $J \to \infty$), the binomial SEA with $\mathcal{S}_{\mathbf{b}} =$ ReLU$_{\mathbf{b}}$ trained by approximate gradient descent followed by normalization using the learning rate of $\kappa = O(p/s)$ (i.e., $\mathbf{w}_i^{(l+1)} =$ normalize$(\mathbf{w}_i^{(l)} - \kappa g_i)$) recovers* $\mathbf{A}$. *More formally, there exists $\delta\in(0,1)$ such that at every iteration l, $\forall i$ $\|\mathbf{w}_i^{(l+1)} - \mathbf{a}_i\|_2^2 \leq (1-\delta)\|\mathbf{w}_i^{(l)} - \mathbf{a}_i\|_2^2 + \kappa \cdot O(\frac{\max(s^2, s^3/p^{\frac{2}{3}+2\xi})}{p^{1+6\xi}})$.*

Theorem 1 shows recovery of $\mathbf{A}$ for the cases when p grows faster than s and the amplitude of the codes are bounded on the order of $O(\frac{1}{p^{\frac{1}{3}+\xi}})$. We refer the reader to the **Appendix** for the implications of this theorem and its interpretation.

4.3. DCEA as a supervised framework

For the *supervised* paradigm, given the desired output (i.e., clean image, $\mathbf{y}^j_{\text{clean}}$, in the case of image denoising), we relax the DCEA architecture and untie the weights (Sreter & Giryes, 2018; Simon & Elad, 2019; Tolooshams et al., 2020) as follows

$$\mathbf{x}_t^j = \mathcal{S}_{\mathbf{b}}\big(\mathbf{x}_{t-1}^j + \alpha(\mathbf{W}^e)^{\mathrm{T}}(\mathbf{y} - f^{-1}\big(\mathbf{W}^d\mathbf{x}_{t-1}^j)\big)\big), \quad (6)$$

where we still use $\mathbf{H}$ as the decoder. We use $\{\mathbf{w}_c^e\}_{c=1}^C$ and $\{\mathbf{w}_c^d\}_{c=1}^C$ to denote the filters associated with $\mathbf{W}^e$ and $\mathbf{W}^d$, respectively. We train the bias vector $\mathbf{b}$, unlike in the unsupervised setting where we tune it by grid search (Tolooshams et al., 2020; Tasissa et al., 2020). Compared to DCEA for ECDL, the number of parameters to learn has increased three-fold.

Although the introduction of additional parameters implies the framework is no longer exactly optimizing the parameters of the convolutional generative model, DCEA still maintains the core principles of the convolutional generative model. First, DCEA performs CSC, as $\mathbf{W}^e$, $\mathbf{W}^d$, and $\mathbf{H}$ are convolutional matrices and $\mathcal{S}_\mathbf{b}$ ensures sparsity of $\mathbf{x}_T^j$. Second, the encoder uses $f^{-1}(\cdot)$, as specified by natural exponential family distributions. Therefore, we allow only a moderate departure from the generative model to balance the problem formulation and the problem-solving mechanism. Indeed, as we show in the Poisson image denoising of Section 5, the denoising performance for DCEA with untied weights is superior to that of DCEA with tied weights.

Indeed, the constraints can be relaxed further. For instance, 1) the proximal operator $\mathcal{S}_\mathbf{b}$ can be replaced by a NN (Mardani et al., 2018), 2) the inverse link function $f^{-1}(\cdot)$ can be replaced by a NN (Gao et al., 2016), 3) $\mathbf{W}^d$, $\mathbf{W}^e$, and $\mathbf{H}$ can be untied across different iterations (Hershey et al., 2014), and 4) the linear 1-layer decoder can be replaced with a deep nonlinear decoder. These would increase the number of trainable parameters, allowing for more expressivity and improved performance. Nevertheless, as our goal is to maintain the connection to *sparsity* and the *natural exponential family*, while keeping the number of parameters small, we do not explore these possibilities in this work.

5. Experiments

We apply our framework in three different settings.

- **Poisson image denoising** (*supervised*) We evaluate the performance of supervised DCEA in Poisson image denoising and compare it to state-of-the-art algorithms.
- **ECDL for simulation** (*unsupervised*) We use simulations to examine how the unsupervised DCEA performs ECDL for *binomial* data. With access to ground-truth data, we evaluate the accuracy of the learned dictionary. Additionally, we conduct ablation studies in which we relax the constraints on DCEA and assess how accuracy changes.
- **ECDL for neural spiking data** (*unsupervised*) Using neural spiking data collected from mice (Temereanca et al., 2008), we perform unsupervised ECDL using DCEA. As is common in the analysis of neural data (Truccolo et al., 2005), we assume a *binomial* generative model.

5.1. Denoising Poisson images

We evaluated the performance of DCEA on Poisson image denoising for various peaks. We used the peak signal-to-noise-ratio (PSNR) as a metric. DCEA is trained in a *supervised* manner on the PASCAL VOC image set (Everingham et al., 2012) containing $J = 5{,}700$ training images. $\mathcal{S}_\mathbf{b}$ is set to ReLU$_\mathbf{b}$. We used two test datasets: 1) Set12 (12 images) and 2) BSD68 (68 images) (Martin et al., 2001).

Methods We trained two versions of DCEA to assess whether relaxing the generative model, thus increasing the number of parameters, helps improve the performance: 1) DCEA constrained (DCEA-C), which uses $\mathbf{H}$ as the convolutional filters and 2) DCEA unconstrained (DCEA-UC), which uses $\mathbf{H}$, $\mathbf{W}^e$, and $\mathbf{W}^d$, as suggested in Eq. (6). We used $C = 169$ filters of size 11×11, where we used convolutions with strides of 7 and followed a similar approach to (Simon & Elad, 2019) to account for all shifts of the image when reconstructing. In terms of the number of parameters, DCEA-C has $20{,}618$ $(= 169 \times 11 \times 11 + 169)$ and DCEA-UC has $61{,}516$ $(= 3 \times 169 \times 11 \times 11 + 169)$, where the last terms refer to the bias $\mathbf{b}$. We set $\alpha = 1$.

We unfolded the encoder for $T = 15$ iterations. We initialized the filters using draws from a standard Gaussian distribution scaled by $\sqrt{1/L}$, where we approximate L using the iterative power method. We used the ADAM optimizer with an initial learning rate of 10^{-3}, which we decrease by a factor of 0.8 every 25 epochs, and trained the network for 400 epochs. At every iteration, we crop a random 128×128 patch, $\mathbf{y}^j_{\text{clean}}$, from a training image and normalize it to $\boldsymbol{\mu}_{j,\text{clean}} = \mathbf{y}^j_{\text{clean}}/Q^j$, where $Q^j = \max(\mathbf{y}^j_{\text{clean}})/\text{peak}$, such that the maximum value of $\boldsymbol{\mu}_{j,\text{clean}}$ equals the desired peak. Then, we generate a count-valued Poisson image with rate $\boldsymbol{\mu}_{j,\text{clean}}$, i.e., $\mathbf{y}^j \sim \text{Poisson}(\boldsymbol{\mu}_{j,\text{clean}})$. We minimized the mean squared error between the clean image, $\mathbf{y}^j_{\text{clean}}$, and its reconstruction, $Q^j \widehat{\boldsymbol{\mu}}_j = Q^j \exp(\mathbf{H}\mathbf{x}^j_T)$.

We compared DCEA against the following baselines. For a fair comparison, we do not use the binning strategy (Salmon et al., 2014) of these methods, as a pre-processing step.

- **Sparse Poisson dictionary algorithm (SPDA)** This is a patch-based dictionary learning framework (Giryes & Elad, 2014), using the Poisson generative model with the ℓ_0 pseudo-norm to learn the dictionary in an *unsupervised* manner, for a given noisy image. SPDA uses 400 filters of length 400, which results in 160,000 parameters.
- **BM3D + VST** BM3D is an image denoising algorithm based on a sparse representation in a transform domain, originally designed for Gaussian noise. This algorithm applies a variance-stabilizing transform (VST) to the Poisson images to make them closer to Gaussian-perturbed images (Makitalo & Foi, 2013).
- **Class-agnostic denoising network (CA)** This is a denoising residual NN for both Gaussian and Poisson images (Remez et al., 2018), trained in a *supervised* manner.

Results Table 2 shows that DCEA outperforms SPDA and BM3D + VST, and shows competitive performance against CA, with an order of magnitude fewer parameters. Fig. 2 shows the denoising performance of DCEA-C and DCEA-UC on two test images from Set12 (see **Appendix** for more examples). We summarize a few additional points from this

Table 2. PSNR performance (in dB) of Poisson image denoising for five different models on test images for peak 1, 2, and 4: 1) SPDA, 2) BM3D+VST, 3) Class-agnostic, 4) DCEA constrained (DCEA-C), and 5) DCEA unconstrained (DCEA-UC).

		Man	Couple	Boat	Bridge	Camera	House	Peppers	Set12	BSD68	# of Params
Peak 1	SPDA	.	.	21.42	19.20	20.23	22.73	19.99	20.39	·	160,000
	BM3D+VST	21.62	21.14	21.47	19.22	20.37	22.35	19.89	·	21.01	N/A
	Class-agnostic	**22.49**	22.11	**22.38**	19.83	**21.59**	22.87	**21.43**	**21.51**	21.78	655,544
	DCEA-C (ours)	22.03	21.76	21.80	19.72	20.68	21.70	20.22	20.72	21.27	20,618
	DCEA-UC (ours)	22.44	**22.16**	22.34	**19.87**	21.47	**23.00**	20.91	21.37	**21.84**	61,516
Peak 2	SPDA	.	.	21.73	20.15	21.54	**25.09**	21.23	21.70	·	160,000
	BM3D+VST	23.11	22.65	22.90	20.31	22.13	24.18	21.97	·	22.21	N/A
	Class-agnostic	**23.64**	**23.30**	**23.66**	20.80	**23.25**	24.77	**23.19**	**22.97**	22.90	655,544
	DCEA-C (ours)	23.10	2.79	22.90	20.60	22.01	23.22	21.70	22.02	22.31	20,618
	DCEA-UC (ours)	23.57	**23.30**	23.51	**20.82**	22.94	24.52	22.94	22.79	**22.92**	61,516
Peak 4	SPDA	.	.	22.46	20.55	21.90	26.09	22.09	22.56	·	160,000
	BM3D+VST	24.32	24.10	24.16	21.50	23.94	26.04	24.07	·	23.54	N/A
	Class-agnostic	24.77	24.60	24.86	21.81	**24.87**	**26.59**	**24.83**	**24.40**	23.98	655,544
	DCEA-C (ours)	24.26	24.08	24.25	21.59	23.60	25.11	23.68	23.51	23.54	20,618
	DCEA-UC (ours)	**24.82**	**24.69**	**24.89**	**21.83**	24.66	26.47	24.71	24.37	**24.10**	61,516

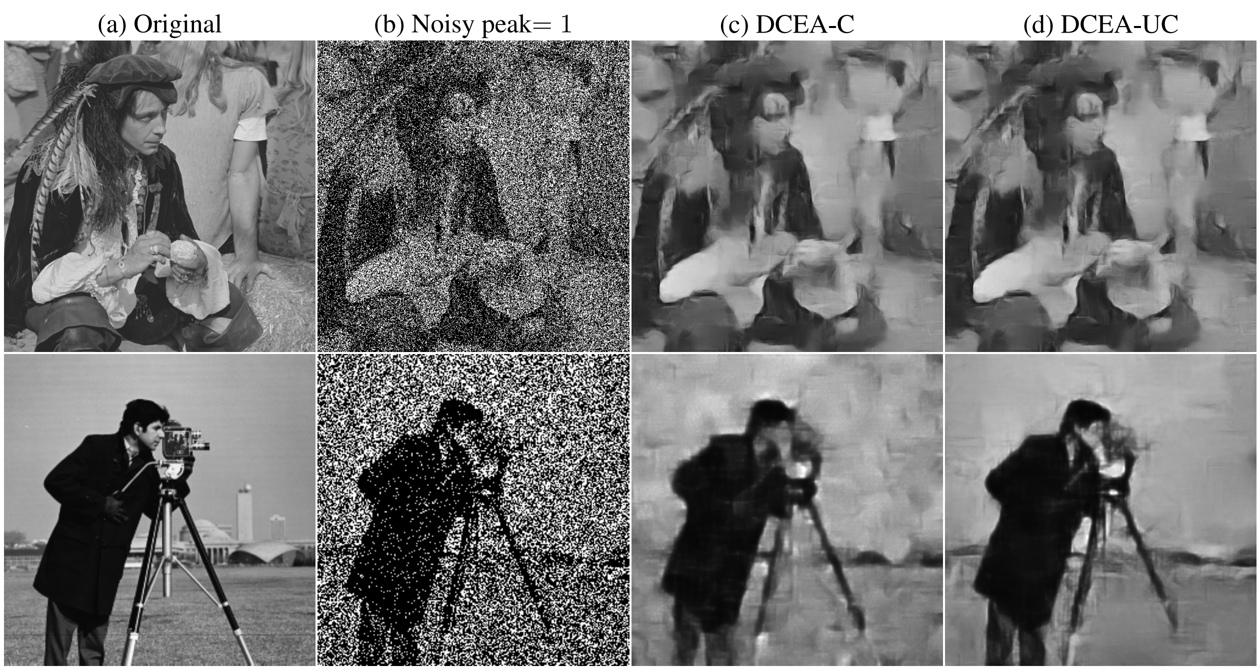

Figure 2. Denoising performance on test images with peak= 1. (a) Original, (b) noisy, (c) DCEA-C, and (d) DCEA-UC.

experiment.

- SPDA vs. DCEA-UC DCEA-UC is significantly more computationally efficient compared to SPDA. SPDA takes several minutes, or hours in some cases, to denoise a single Poisson noisy image whereas, upon training, DCEA performs denoising in less than a second.

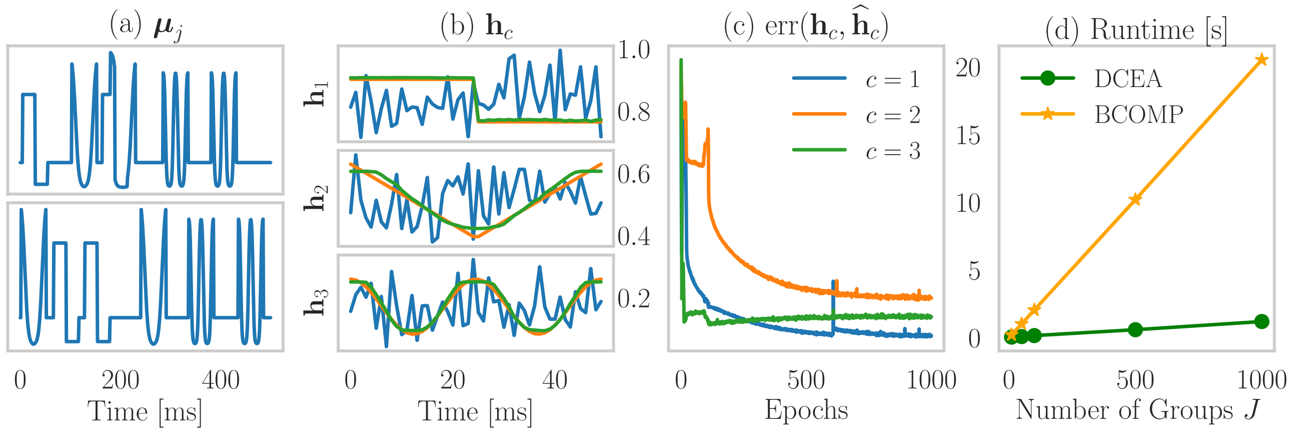

Figure 3. Simulated results with DCEA. (a) Example rate functions, $\boldsymbol{\mu}_j$, for two different groups. (b) Initial (blue), true (orange), and learned (green) filters for binomial data. (c) err$(\mathbf{h}_c, \widehat{\mathbf{h}}_c)$ over $1,000$ epochs. (d) Total runtime for inference for DCEA and BCOMP.

- **CA vs. DCEA-UC** DCEA-UC achieves competitive performance against CA, despite an order of magnitude difference in the number of parameters (650K for CA vs. 61K for DCEA). We conjecture that given the same number of parameters, DCEA-UC would outperform CA. For example, we found that replacing the linear decoder with a nonlinear two-layer decoder ($\approx$ 120K number of parameters) in DCEA-UC resulted in an increase in PSNR of $\sim$0.2 dB.

- **DCEA-C vs. DCEA-UC** We also observe that DCEA-UC achieves better performance than DCEA-C. As discussed in Section 4.3, the relaxation of the generative model, which allows for a three-fold increase in the number of parameters, helps improve the performance.

5.2. Application to simulated neural spiking data

5.2.1. Accuracy of ECDL for DCEA

We simulated time-series of neural spiking activity from $J = 1{,}000$ neurons according to the binomial generative model. We used $C = 3$ templates of length $K = 50$ and, for each example j, generated $f(\boldsymbol{\mu}_j) = \mathbf{H}\mathbf{x}^j \in \mathbb{R}^{500}$, where each filter $\{\mathbf{h}_c\}_{c=1}^3$ appears five times uniformly random in time. Fig. 3(a) shows an example of two different means, $\boldsymbol{\mu}_{j_1}$, $\boldsymbol{\mu}_{j_2}$ for $j_1 \neq j_2$. Given $\boldsymbol{\mu}_j$, we simulated two sets of binary time-series, each with $M_j = 25$, $\mathbf{y}^j \in \{0, 1, \ldots, 25\}^{500}$, one of which is used for training and the other for validation.

Methods For DCEA, we initialized the filters using draws from a standard Gaussian, tuned the regularization parameter λ (equivalently b for $\mathcal{S}_b$) manually, and trained using the unsupervised loss. We place non-negativity constraints on $\mathbf{x}^j$ and thus use $\mathcal{S}_b = \text{ReLU}_b$. For baseline, we developed and implemented a method which we refer to as binomial convolutional orthogonal matching pursuit (BCOMP). At present, there does not exist an optimization-based framework for ECDL. Existing dictionary learning methods for non-Gaussian data are patch-based (Lee et al., 2009; Giryes & Elad, 2014). BCOMP combines efficient convolutional greedy pursuit (Mailhé et al., 2011) and binomial greedy pursuit (Lozano et al., 2011). BCOMP solves Eq. (2), but uses $\|\mathbf{x}^j\|_0$ instead of $\|\mathbf{x}^j\|_1$. For more details, we refer the reader to the **Appendix**.

Results Fig. 3(b) demonstrates that DCEA (green) is able to learn $\{\mathbf{h}_c\}_{c=1}^3$ accurately. Letting $\{\widehat{\mathbf{h}}_c\}_{c=1}^3$ denote the estimates, we quantify the error between a filter and its estimate using the standard measure (Agarwal et al., 2016), err$(\mathbf{h}_c, \widehat{\mathbf{h}}_c) = \sqrt{1 - \langle \mathbf{h}_c, \widehat{\mathbf{h}}_c \rangle^2}$, for $\|\mathbf{h}_c\| = \|\widehat{\mathbf{h}}_c\| = 1$. Fig. 3(c) shows the error between the true and learned filters by DCEA, as a function of epochs (we consider all possible permutations and show the one with the lowest error). The fact that the learned and the true filters match demonstrates that DCEA is indeed performing ECDL. Finally, Fig. 3(d) shows the runtime for both DCEA (on GPU) and BCOMP (on CPU) on CSC task, as a function of number of groups J, where DCEA is much faster. This shows that DCEA, due to 1) its simple implementation as an unrolled NN and 2) the ease with which the framework can be deployed to GPU, is an efficient/scalable alternative to optimization-based BCOMP.

5.2.2. Generative model relaxation for ECDL

Here, we examine whether DCEA with untied weights, which implies a departure from the original convolutional generative model, can still perform ECDL accurately. To this end, we repeat the experiment from Section 5.2.1 with DCEA-UC, whose parameters are $\mathbf{H}$, $\mathbf{W}^e$, and $\mathbf{W}^d$. Fig. 4

shows the learned filters, $\widehat{\mathbf{w}}_c^e$, $\widehat{\mathbf{w}}_c^d$, and $\widehat{\mathbf{h}}_c$ for $c = 1$ and 2, along with the true filters. For visual clarity, we only show the learned filters for which the distance to the true filters are the closest, among $\widehat{\mathbf{w}}_c^e$, $\widehat{\mathbf{w}}_c^d$, and $\widehat{\mathbf{h}}_c$. We observe that none of them match the true filters. In fact, the error between the learned and the true filters are bigger than the initial error.

This is in sharp contrast to the results of DCEA-C (Fig. 3(b)), where $\mathbf{H} = \mathbf{W}^e = \mathbf{W}^d$. This shows that, to accurately perform ECDL, the NN architecture needs to be strictly constrained such that it optimizes the objective formulated from the convolutional generative model.

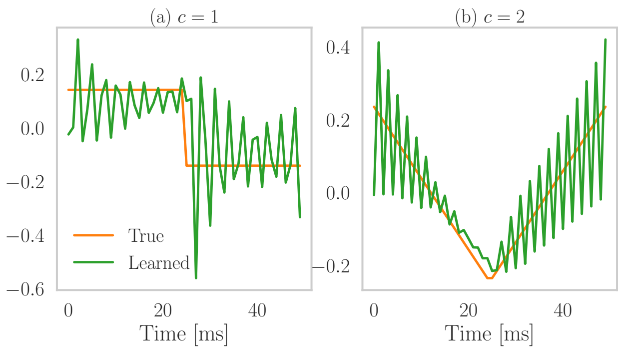

Figure 4. The learned (green) and true (orange) filters for DCEA-UC, when the weights are untied.

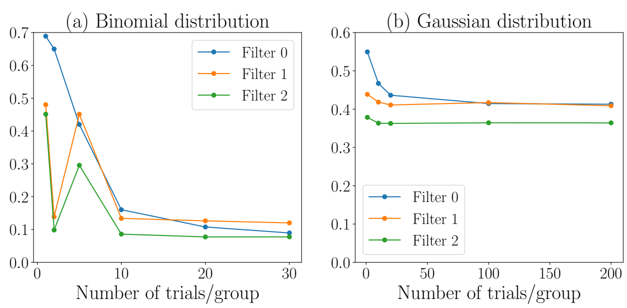

Figure 5. Dictionary error, err($\mathbf{h}_c, \widehat{\mathbf{h}}_c$), as a function of number of trials M_j per group for the (a) Binomial and (b) Gaussian models. Each point represents the median of 20 independent trials.

5.2.3. Effect of model mis-specification on ECDL

Here, we examine how model mis-specification in DCEA, equivalent to mis-specifying 1) the loss function (negative log-likelihood) and 2) the nonlinearity $f^{-1}(\cdot)$, affects the accuracy of ECDL. We trained two models: 1) DCEA with sigmoid link and binomial likelihood (DCEA-b), the correct model for this experiment, and 2) DCEA with linear link and Gaussian likelihood (DCEA-g). Fig. 5 shows how the error err($\mathbf{h}_c, \widehat{\mathbf{h}}_c$), at convergence, changes as a function of the number of observations M_j.

We found that DCEA-b successfully recovers dictionaries for large M_j (>15). Not surprisingly, as M_j, i.e. SNR, decreases, the error increases. DCEA-g with 200 observations achieves an error close to 0.4, which is significantly worse than the 0.09 error of DCEA-b with $M_j = 30$. These results highlight the importance, for successful dictionary learning, of specifying an appropriate model. The framework we propose, DCEA, provides a flexible inference engine that can accommodate a variety of data-generating models in a seamless manner.

5.3. Neural spiking data from somatosensory thalamus

We now apply DCEA to neural spiking data from somatosensory thalamus of rats recorded in response to periodic whisker deflections (Temereanca et al., 2008). The objective is to learn the features of whisker motion that modulate neural spiking strongly. In the experiment, a piezoelectric simulator controls whisker position using an *ideal* position waveform. As the interpretability of the learned filters is important, we constrain the weights of encoder and decoder to be $\mathbf{H}$. DECA lets us learn, in an *unsupervised* fashion, the features that best explains the data.

The dataset consists of neural spiking activity from $J = 10$ neurons in response to periodic whisker deflections. Each example j consists of $M_j = 50$ trials lasting 3,000 ms, i.e., $\mathbf{y}^{j,m} \in \mathbb{R}^{3000}$. Fig. 6(a) depicts a segment of data from a neuron. Each trial begins/ends with a baseline period of 500 ms. During the middle 2,000 ms, a periodic deflection with period 125 ms is applied to a whisker by the piezoelectric stimulator. There are 16 total deflections, five of which are shown in Fig. 6(b). The stimulus represents ideal whisker position. The blue curve in Fig. 6(c) depicts the whisker velocity obtained as the first derivative of the stimulus.

Methods We compare DCEA to ℓ_0-based ECDL using BCOMP (introduced in the previous section), and a generalized linear model (GLM) (McCullagh & Nelder, 1989) with whisker-velocity covariate (Ba et al., 2014). For all three methods, we let $C = 1$ and $\mathbf{h}_1 \in \mathbb{R}^{125}$, initialized using the whisker velocity (Fig. 6(c), blue). We set $\lambda = 0.119$ for DCEA and set the sparsity level of BCOMP to 16. As in the simulation, we used $\mathcal{S}_b = \text{ReLU}_b$ to ensure non-negativity of the codes. We used 30 trials from each neuron to learn $\mathbf{h}_1$ and the remaining 20 trials as a test set to assess goodness-of-fit. We describe additional parameters used for DCEA and the post-processing steps in the **Appendix**.

Results The orange and green curves from Fig. 6(c) depict the estimates of whisker velocity computed from the neural spiking data using BCOMP and DCEA, respectively. The figure indicates that the spiking activity of this population of 10 neurons encodes well the whisker velocity, and is most strongly modulated by the maximum velocity of whisker

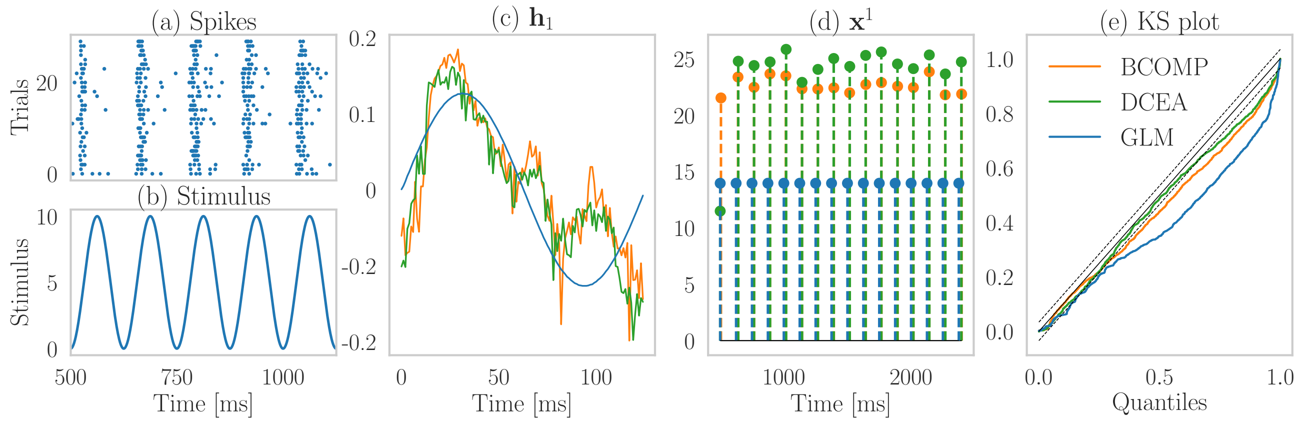

Figure 6. A segment of data from a neuron and result of applying DCEA and BCOMP. (a) A dot indicates a spike from the neuron. (b) Stimulus used to move the whisker. (c) Whisker velocity covariate (blue) used in GLM analysis, along with whisker velocities estimated with BCOMP (orange) and DCEA (green) using all 10 neurons in the dataset. The units are $\frac{\text{mm}}{10}$ per ms. (d) The estimated sparse codes (onset of whisker deflection). (e) Analysis of Goodness-of-fit using KS plots. The dotted lines represent 95% confidence intervals.

movement.

Fig. 6(d) depicts the 16 sparse codes that accurately capture the onset of stimulus in each of the 16 deflection periods. The heterogeneity of amplitudes estimated by DCEA and BCOMP is indicative of the variability of the neural response to whisker deflections repeated 16 times, possibly capturing other characteristics of cellular and circuit response dynamics (e.g., adaptation). This is in sharp contrast to the GLM–detailed in the **Appendix**–which uses the ideal whisker velocity (Fig. 6(c), blue) as a covariate, and assumes that neural response to whisker deflections is constant across deflections.

In Fig. 6(e), we use the Kolmogorov-Smirnov (KS) test to compare how well DCEA, BCOMP, and the GLM fit the data for a representative neuron in the dataset (Brown et al., 2002). KS plots are a visualization of the KS test for assessing the Goodness-of-fit of models to point-process data, such as neural spiking data (see **Appendix** for details). The figure shows that DCEA and BCOMP are a much better fit to the data than the GLM.

We emphasize that 1) the similarity of the learned $\mathbf{h}_1$ and 2) the similar goodness-of-fit of DCEA and BCOMP to the data shows that DCEA performs ECDL. In addition, this analysis shows the power of the ECDL as an *unsupervised* and *data-driven* approach for data analysis, and a superior alternative to GLMs, where the features are hand-crafted.

6. Conclusion

We introduced a class of neural networks based on a generative model for convolutional dictionary learning (CDL) using data from the natural exponential-family, such as count-valued and binary data. The proposed class of networks, which we termed deep convolutional exponential auto-encoder (DCEA), is competitive against state-of-the-art supervised Poisson image denoising algorithms, with an order of magnitude fewer trainable parameters.

We analyzed gradient dynamics of shallow exponential-family auto-encoder (i.e., unfold the encoder once) for binomial distribution and proved that when trained with approximate gradient descent, the network recovers the dictionary corresponding to the binomial generative model.

We also showed using binomial data simulated according to the convolutional exponential-family generative model that DCEA performs dictionary learning, in an unsupervised fashion, when the parameters of the encoder/decoder are constrained. The application of DCEA to neural spike data suggests that DCEA is superior to GLM analysis, which relies on hand-crafted covariates.

Acknowledgements

Bahareh Tolooshams acknowledges the support of the NSF-Simons Center for Mathematical and Statistical Analysis of Biology at Harvard University (supported by NSF grant no. DMS-1764269), the Harvard FAS Quantitative Biology Initiative, and AWS Machine Learning Research Awards. Demba Ba and Bahareh Tolooshams acknowledge the partial support of the ARO under grant number W911NF-16-1-0368. This is part of a collaboration between US DOD, UK MOD and UK Engineering and Physical Research Council (EPSRC) under the Multidisciplinary University Research Initiative. Andrew H. Song acknowledges support from a Samsung scholarship.

References

Agarwal, A., Anandkumar, A., Jain, P., Netrapalli, P., and Tandon, R. Learning sparsely used overcomplete dictionaries via alternating minimization. *SIAM Journal on Optimization*, 26:2775–2799, 2016.

Ba, D., Temereanca, S., and Brown, E. Algorithms for the analysis of ensemble neural spiking activity using simultaneous-event multivariate point-process models. *Frontiers in Computational Neuroscience*, 8:6, 2014.

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

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Brown, E. N., Barbieri, R., Ventura, V., Kass, R. E., and Frank, L. M. The time-rescaling theorem and its application to neural spike train data analysis. *Neural Computation*, 14(2):325–346, 2002.

Clevert, D., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations*, 2016.

Daubechies, I., Defrise, M., and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2012.

Feng, W., Qiao, P., and Chen, Y. Fast and accurate poisson denoising with trainable nonlinear diffusion. *IEEE Transactions on Cybernetics*, 48(6):1708–1719, 2018.

Gao, Y., Archer, E. W., Paninski, L., and Cunningham, J. P. Linear dynamical neural population models through nonlinear embeddings. In *Advances in Neural Information Processing Systems 29*, pp. 163–171. Curran Associates, Inc., 2016.

Garcia-Cardona, C. and Wohlberg, B. Convolutional dictionary learning: A comparative review and new algorithms. *IEEE Transactions on Computational Imaging*, 4(3):366–381, September 2018.

Giryes, R. and Elad, M. Sparsity-based poisson denoising with dictionary learning. *IEEE Transactions on Image Processing*, 23(12):5057–5069, 2014.

Gregor, K. and Lecun, Y. Learning fast approximations of sparse coding. In *International Conference on Machine Learning*, pp. 399–406, 2010.

Hershey, J. R., Roux, J. L., and Weninger, F. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv:1409.2574*, pp. 1–27, 2014.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521:436–444, 2015.

Lee, H., Raina, R., Teichman, A., and Ng, A. Y. Exponential family sparse coding with applications to self-taught learning. In *Proc. the 21st International Jont Conference on Artificial Intelligence*, IJCAI, pp. 1113–1119, 2009.

Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pp. 689–698, 2018.

Lozano, A., Swirszcz, G., and Abe, N. Group orthogonal matching pursuit for logistic regression. *Journal of Machine Learning Research*, 15:452–460, 2011.

Ma, L., Moisan, L., Yu, J., and Zeng, T. A dictionary learning approach for poisson image deblurring. *IEEE Transactions on medical imaging*, 32(7):1277–1289, 2013.

Mailhé, B., Gribonval, R., Vandergheynst, P., and Bimbot, F. Fast orthogonal sparse approximation algorithms over local dictionaries. *Signal Processing*, 91:2822–2835, 2011.

Makitalo, M. and Foi, A. Optimal inversion of the generalized anscombe transformation for poisson-gaussian noise. *IEEE Transactions on Image Processing*, 22(1):91–103, Jan 2013.

Mardani, M., Sun, Q., Vasawanala, S., Papyan, V., Monajemi, H., Pauly, J., and Donoho, D. Neural proximal gradient descent for compressive imaging. In *Proc. Advances in Neural Information Processing Systems 31*, pp. 9573–9683, 2018.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pp. 416–423, July 2001.

McCullagh, P. and Nelder, J. *Generalized Linear Models*. Chapman & Hall/CRC, 1989.

Monga, V., Li, Y., and Eldar, Y. C. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *arXiv:1912.10557*, 2019.

Nazábal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. Handling incomplete heterogeneous data using vaes. *Pattern recognition*, 107:107501, 2020. ISSN 0031-3203.

Nguyen, T. V., Wong, R. K. W., and Hegde, C. On the dynamics of gradient descent for autoencoders. In *Proc. Machine Learning Research*, volume 89, pp. 2858–2867. PMLR, 16–18 Apr 2019.

Parikh, N. and Boyd, S. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.

Remez, T., Litany, O., Giryes, R., and Bronstein, A. M. Class-aware fully-convolutional gaussian and poisson denoising. *CoRR*, abs/1808.06562, 2018.

Salmon, J., Harmany, Z., Deledalle, C.-A., and Willett, R. Poisson noise reduction with non-local pca. *Journal of Mathematical Imaging and Vision*, 48(2):279–294, Feb 2014.

Simon, D. and Elad, M. Rethinking the csc model for natural images. In *Proc. Advances in Neural Information Processing Systems 33 (NeurIPS)*, pp. 2271–2281, 2019.

Sreter, H. and Giryes, R. Learned convolutional sparse coding. In *Proc. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2191–2195, 2018.

Sulam, J., Aberdam, A., Beck, A., and Elad, M. On multi-layer basis pursuit, efficient algorithms and convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

Tasissa, A., Theodosis, E., Tolooshams, B., and Ba, D. Dense and sparse coding: Theory and architectures. *arXiv:2006.09534*, 2020.

Temereanca, S., Brown, E. N., and Simons, D. J. Rapid changes in thalamic firing synchrony during repetitive whisker stimulation. *Journal of Neuroscience*, 28(44): 11153–11164, 2008.

Tolooshams, B., Dey, S., and Ba, D. Scalable convolutional dictionary learning with constrained recurrent sparse auto-encoders. In *Proc. 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2018.

Tolooshams, B., Dey, S., and Ba, D. Deep residual autoencoders for expectation maximization-inspired dictionary learning. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2020. doi: 10.1109/TNNLS.2020.3005348.

Truccolo, W., Eden, U. T., Fellows, M., Donoghue, J., and Brown, E. N. A Point Process Framework for Relating Neural Spiking Activity to Spiking History, Neural Ensemble, and Extrinsic Covariate Effects. *Journal of Neurophysiology*, 93(2):1074–1089, 2005.

Wang, Z., Liu, D., Yang, J., Han, W., and Huang, T. Deep networks for image super-resolution with sparse prior. In *Proc. the IEEE International Conference on Computer Vision*, pp. 370–378, 2015.

Yang, F., Lu, Y. M., Sbaiz, L., and Vetterli, M. Bits from photons: Oversampled image acquisition using binary poisson statistics. *IEEE Transactions on image processing*, 21(4):1421–1436, 2011.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, July 2017.