Variational Sparse Coding with Learned Thresholding

Kion Fallah [1] **Christopher J. Rozell** [1]

Abstract

Sparse coding strategies have been lauded for their parsimonious representations of data that leverage low dimensional structure. However, inference of these codes typically relies on an optimization procedure with poor computational scaling in high-dimensional problems. For example, sparse inference in the representations learned in the high-dimensional intermediary layers of deep neural networks (DNNs) requires an iterative minimization to be performed at each training step. As such, recent, quick methods in variational inference have been proposed to infer sparse codes by learning a distribution over the codes with a DNN. In this work, we propose a new approach to variational sparse coding that allows us to learn sparse distributions by thresholding samples, avoiding the use of problematic relaxations. We first evaluate and analyze our method by training a linear generator, showing that it has superior performance, statistical efficiency, and gradient estimation compared to other sparse distributions. We then compare to a standard variational autoencoder using a DNN generator on the Fashion MNIST and CelebA datasets.

1. Introduction

Variational inference has become a ubiquitous tool in unsupervised learning of a distribution of latent features. These variational distributions can offer approximations in cases where inference over a true distribution is computationally expensive. Once inference is performed, latent features can be used for a variety of machine learning tasks, such as summarizing a dataset or training a generative model. The structure and statistical properties of latent features depend on the practitioner's choice of a prior distribution. Sparse distributions, in which only a few features are non-zero for each input data sample, have been favored for encouraging statistically efficient representations, especially when input data has low-dimensional structure (Olshausen & Field, 2004; Elad, 2010).

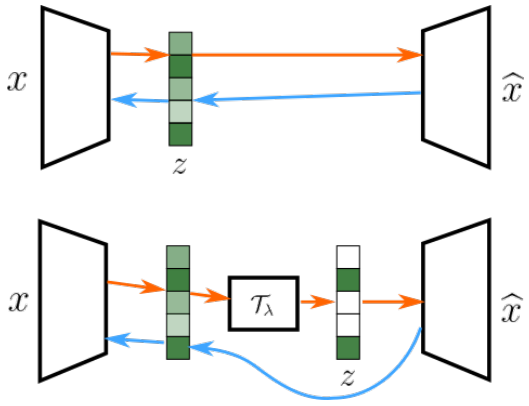


Figure 1. Comparison of standard black-box variational inference and our proposed approach for variational sparse coding. Orange arrows depict a forward pass, blue arrows depict automatic differentiation. *Top:* Sampling from standard variational inference approaches result in all features non-zero. *Bottom:* Our approach incorporates sparsity via a shifted soft-threshold function. We utilize a straight-through estimator, skipping the gradient of the shifted soft-threshold, for exact sparsity without numerical instability during training.

To perform inference in sparse models with low computational cost, recent black-box variational inference (BBVI) (Ranganath et al., 2013) methods have been proposed to learn distribution parameters with DNNs using automatic differentiation (Kingma & Welling, 2014; Rezende et al., 2014). Unfortunately, these approaches either do not explicitly learn sparse features (Barello et al., 2018) or rely on relaxations that can lead to poor gradient estimation during training (Tonolini et al., 2020).

Motivated by the success of soft-thresholding in iterative optimization procedures (Daubechies et al., 2003; Donoho & Johnstone, 1994), we propose a BBVI method to learn sparse distributions by thresholding samples drawn from a Laplacian or Gaussian distribution. We analytically show that thresholded samples from the former are identically distributed to a Spike-and-Slab distribution, giving practitioners control over the extent of sparsity by adjusting a

[1]ML@GT, Georgia Institute of Technology, Atlanta, Georgia. Correspondence to: Kion Fallah <kion@gatech.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

threshold hyper-parameter. In cases where the degree of sparsity is not known beforehand, we also propose a technique to learn a distribution on the threshold parameter. To train our inference network, we apply a straight-through estimator (Bengio et al., 2013; Oord et al., 2018), leading to favorable training stability and gradient estimation. Finally, we propose a new sampling procedure that encourages feature reuse, encouraging the generator to learn more diverse features. We showcase the performance of our method compared to other inference strategies by training and analyzing a linear generator on whitened image patches (Olshausen & Field, 1996) and a DNN generator on the Fashion MNIST (Xiao et al., 2017) and CelebA (Liu et al., 2015) datasets [1].

2. Related Work

2.1. MAP Estimate/Regressive Inference

Sparsity models have a long history in statistical modeling with methods such as LASSO regularization (Tibshirani, 1996). These models often infer sparse latent features via a maximum a posteriori (MAP) estimate, requiring an iterative optimization procedure to be solved. The seminal work of (Olshausen & Field, 1996) proposes sparse codes as a means of unsupervised learning of a linear generator. Once trained, columns of the generator qualitatively resemble the receptive fields of mammalian cortical cells. Although methods exist to solve this optimization in discrete (Beck & Teboulle, 2009; Yang et al., 2012) and continuous (Rozell et al., 2008) time, their computational cost often scales poorly with dimensionality, making them prohibitive to use in modern deep learning settings.

An alternative approach for inference has been to use DNNs to regress sparse codes for given input data. One method "unrolls" iterations of the ISTA algorithm (Gregor & LeCun, 2010). These methods are limited for unsupervised learning since they require ground truth codes as supervision during training.

2.2. Variational Inference

Early variational inference approaches applied exponential families for sparse coding, using iterative procedures to fit a variational posterior distribution (Girolami, 2001; Seeger, 2008). Later works explored variational methods for Spike-and-Slab models (Goodfellow et al., 2012; Sheikh & Lücke, 2016), using approximations to analytic solutions for faster inference. None of these approaches effectively scale inference to high-dimensional DNN representations.

The proposal of BBVI, where DNNs are used to estimate parameters of a variational posterior distribution, has led to a leap in the computational efficiency of variational inference (Kingma & Welling, 2014; Rezende et al., 2014). As such, many recent methods have been proposed to train DNN inference networks with various prior distributions. These include Laplacian (Barello et al., 2018), Spike-and-Slab (Tonolini et al., 2020), and Beta-Bernoulli (Singh et al., 2017) distributions. Other work has incorporated sparsity through hierarchical posterior distributions (Salimans, 2016), evolutionary variational algorithms (Drefs et al., 2022), or group-sparsity in connections to generator networks (Ainsworth et al., 2018; Moran et al., 2021). We refer the reader to (Zhang et al., 2018) for a review of variational inference.

2.3. Estimating the Variational Bound

Various sampling approaches have been proposed to estimate the variational bound. (Cremer et al., 2017) proposes a sampling procedure based on a tighter bound on the data likelihood introduced in (Burda et al., 2016). Counter-intuitively, using this tighter bound in training leads to a reduced signal-to-noise ratio in the inference network gradient (Rainforth et al., 2018). In (Grover et al., 2014), the authors apply rejection sampling using a computed acceptance probability for each sample. A later work (Bauer & Mnih, 2019) learns this acceptance probability with an additional DNN. Other work uses an alternative gradient estimator in BBVI which ignores certain terms during automatic differentiation to reduce variance (Roeder et al., 2017), with a bias-free estimator proposed by (Tucker et al., 2018).

3. Methods

3.1. Black-box Variational Inference

The capability of DNNs as universal function approximators has been recently applied to learn complex distributions. Given a dataset of training samples $[\mathbf{x}^1, \ldots, \mathbf{x}^N]^T \in \mathbb{R}^{N \times D}$ from target density $p(\mathbf{x})$, BBVI can be applied to train an inference network $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ to learn a distribution over latent features $\mathbf{z}^k \in \mathbb{R}^d$. This distribution can be applied in various machine learning tasks, such as training a generator $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$. To do this, one may employ the variational lower bound (ELBO) to maximize the marginal likelihood over the training samples (Jordan et al., 1998):

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \log \mathbb{E}_{p(\mathbf{z})}\left[p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] \\ &= \log \mathbb{E}_{p(\mathbf{z})}\left[\frac{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})} p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] \\ &\geq \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] - D_{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \,||\, p(\mathbf{z})\big) \\ &= \mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}\right). \end{aligned} \tag{1}$$

Training under this bound requires the practitioner to select a

[1] Code available at: https://github.com/kfallah/variational-sparse-coding.

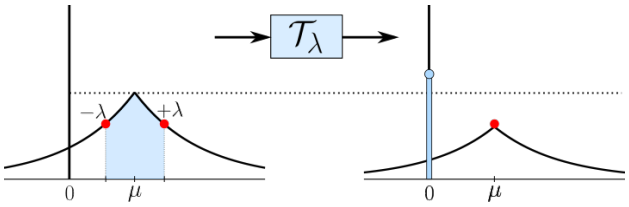


Figure 2. Visual depiction of the shifted soft-threshold applied to a Laplacian distribution. Probability mass $\mu \pm \lambda$ is collapsed to the origin. The red points in the left are shifted to meet at μ on the right.

prior $p(\mathbf{z})$ and a distribution family for the posterior $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$. In this work, we follow the common assumption of independence for both, $p(\mathbf{z}) = \prod_{i=1}^{d} p(z_i), q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) = \prod_{i=1}^{d} q_{\boldsymbol{\phi}}(z_i \mid \mathbf{x})$. For each prior distribution we consider, we will use the same distribution family for our variational posterior.

Parameters $(\boldsymbol{\phi}, \boldsymbol{\theta})$ can be trained via a stochastic version of the variational expectation maximization algorithm (Neal & Hinton, 1998; Barello et al., 2018). Using batched training samples $\mathbf{x}^k$, this algorithm iterates in two steps from initializations $(\boldsymbol{\phi}^{(0)}, \boldsymbol{\theta}^{(0)})$:

1. Expectation: $\boldsymbol{\phi}^{(t)} = \arg\max_{\boldsymbol{\phi}} \mathcal{L}\left(\boldsymbol{\theta}^{(t-1)}, \boldsymbol{\phi}; \mathbf{x}^k\right)$
2. Maximization: $\boldsymbol{\theta}^{(t)} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{\phi}^{(t)}; \mathbf{x}^k\right)$.

In the expectation step a differentiable transform $g_\phi(\mathbf{x}^k, \boldsymbol{\epsilon}^{j,k})$ of auxiliary samples $\boldsymbol{\epsilon}^{j,k} \sim p(\boldsymbol{\epsilon})$ is used to estimate the expectation over the inference network distribution. Coined the "reparameterization trick," this technique has been show to reduce the variance of the gradient estimates through the inference network (Kingma & Welling, 2014; Rezende et al., 2014). In practice, the two steps of the EM algorithm are often approximated concurrently with a gradient step using a single sample $J = 1$.

In the next sections, we will describe 1) a method for reparameterization that results in sparse latent features and 2) a new sampling procedure in the expectation step that improves performance for distributions with sparse priors.

3.2. Reparameterization for Thresholded Samples

A favorable choice of sparse prior distribution is the Spike-and-Slab $p(z_i) = \gamma p(s_i) + (1-\gamma)\delta(z_i)$ (Mitchell & Beauchamp, 1988), which allows sparse random variables to be sampled by first sampling a Bernoulli random variable with probability γ. If the random variable is non-zero, then a sample is drawn from the slab distribution $p(s_i)$. A Spike-and-Slab is favorable to setting $p(z_i)$ equal to Normal, Laplacian, or Cauchy distributions because these latter distributions will never result in samples with features exactly equal to zero. This allows one to benefit from precise sparsity for downstream computations (e.g., decoding, image generation). Unfortunately, current BBVI approaches depend on continuous approximations (controlled by temperature parameter τ) to each Bernoulli random variable τ (Tonolini et al., 2020; Jang et al., 2017; Maddison et al., 2017). Tuning τ requires a trade-off between variance and bias when estimating gradients, potentially leading to poor performance.

Rather than parameterize $p(z_i)$ with discrete random variables, we propose an approach that applies shifted soft-thresholding of samples drawn from a Laplacian distribution. Let $p(s) = \text{Laplace}(\mu, b)$ be a Laplacian pdf with shift μ and scale b. We can draw a sample $s \sim p(s)$ and apply the shifted soft-threshold function:

$$\mathcal{T}(s; \lambda, \mu) = \text{sign}(s-\mu)\max\big(|s-\mu| - \lambda, 0\big) + \mathbb{I}\left[|s-\mu| > \lambda\right]\mu. \quad (2)$$

For ease of notation, we will use the shorthand $\mathcal{T}_\lambda(s)$ to refer to $\mathcal{T}(s; \lambda, \mu)$. Intuitively, this function is a soft-threshold around μ, where points mapped to μ are set exactly to zero. This is visualized in Figure 2. When $\mu = 0$, this function is precisely the soft-threshold function (Donoho & Johnstone, 1994). Next, we will show how this shifted soft-threshold can be used to reparameterize to a Spike-and-Slab without needing discrete random variables.

Proposition 3.1 *Samples from a Laplace distribution $s \sim p(s)$ passed through a shifted soft-threshold $z = \mathcal{T}_\lambda(s)$ are equivalently distributed as a Spike-and-Slab distribution $z \overset{i.i.d.}{\sim} p(z)$.*

Note that under this rule for z, $p(z \mid |s-\mu| \leq \lambda) = \delta(z)$, where the spike probability $(1-\gamma)$ can be found by computing $p(|s-\mu| \leq \lambda)$:

$$p(|s-\mu| \leq \lambda) = 2\int_0^\lambda \frac{1}{2b}\exp\left(-sb^{-1}\right) ds = 1 - \exp\left(-\lambda b^{-1}\right) = (1-\gamma). \quad (3)$$

We show in Appendix A that $p(z \mid |s-\mu| > \lambda)$ is distributed as $p(s)$. Combining these facts and marginalizing over $p(s)$ yields:

$$z \sim \frac{\gamma}{2b}\exp\left(\frac{-|z-\mu|}{b}\right) + (1-\gamma)\delta(z).$$

Using this technique, we can sample sparse random variables $\mathbf{z}^k$ for a corresponding input data $\mathbf{x}^k$ from an inference network trained via the EM algorithm (1). For a given input data sample, we encode the parameters of a base distribution that is either a Gaussian or Laplacian via $q_{\boldsymbol{\phi}}(\mathbf{s} \mid \mathbf{x}^k)$.

These can be used to both compute the KL divergence and to reparameterize a sample to be passed through the shifted soft-threshold. The threshold parameter can be either set as a fixed hyperparameter $\boldsymbol{\lambda} = \boldsymbol{\lambda}_0$ (equating to a fixed prior on the spike probability) or learned via variational inference with a factorial gamma prior $p(\boldsymbol{\lambda}) = \prod_{i=1}^{d} \Gamma(\alpha_0, \frac{\alpha_0}{\lambda_0})$ (Jankowiak & Obermeyer, 2018; Garrigues & Olshausen, 2010). In the case where we perform inference on $\boldsymbol{\lambda}$, we assume independence with the base distribution $p(\mathbf{s})$.

One challenge during training is dealing with the non-differentiable $T_{\boldsymbol{\lambda}}$ in the first term of (1). We have found that the subgradient of this function leads to shrinkage of the base distribution parameters, eventually leading to numerical instability or posterior collapse (Lucas et al., 2019). For further discussion on this issue, refer to Appendix C. As a result, we employ a straight-through estimator (Bengio et al., 2013; Jang et al., 2017; Oord et al., 2018), skipping the shifted soft-threshold when we differentiate the loss by passing the gradient from the generator directly to the base distribution parameters. This is depicted in Figure 1.

This results in our final training objective with a reweighted KL term (Higgins et al., 2017). Following the notation of (Oord et al., 2018), we denote by $\text{sg}[\cdot]$ the stopgradient operator (identity in the forward pass with partial derivatives equal to zero).

$$\widetilde{\mathbf{z}}^k = \mathbf{s}^k + \mathcal{T}_{\boldsymbol{\lambda}^k}\left(\text{sg}\left[\mathbf{s}^k\right]\right) - \text{sg}\left[\mathbf{s}^k\right] \tag{4}$$

$$\begin{aligned}\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^k\right) = \; & \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^k \mid \widetilde{\mathbf{z}}^k)\right] \\ & - \beta_1 D_{KL}\big(q_\phi(\mathbf{s} \mid \mathbf{x}^k) \,||\, p(\mathbf{s})\big) \\ & - \beta_2 D_{KL}\big(q_\phi(\boldsymbol{\lambda} \mid \mathbf{x}^k) \,||\, p(\boldsymbol{\lambda})\big)\end{aligned} \tag{5}$$

Note that in the case where we fix $\boldsymbol{\lambda}^k = \boldsymbol{\lambda}_0$, the second term in (4) has no gradient and the third term in (5) is omitted. More details on reparameterization for the sampling procedure $(\mathbf{s}^k, \boldsymbol{\lambda}^k) \sim q_{\boldsymbol{\phi}}(\mathbf{s}, \boldsymbol{\lambda} \mid \mathbf{x}^k)$ and on computing the loss terms in (5) are included in Appendix B.

3.3. Max ELBO Sampling

A favorable property of MAP estimates is that priority of which latent features are non-zero is given to those that best represent features in input data. These developed features lead to the highest decrease in loss, a property exploited in greedy MAP inference procedures (Tropp & Gilbert, 2007). Unfortunately, BBVI does not inherently have this property, with the factorial prior in the KL divergence term in (5) encouraging all features to be equally likely for each input data sample. To address this, we propose a sampling procedure that biases towards reuse of developed features.

Before introducing our approach, we note the standard approach in BBVI for using multiple samples to approximate the expectation with respect to $q_{\boldsymbol{\phi}}$ in (5). Given a sampling budget J, one may draw J i.i.d. samples $\mathbf{z}^{j,k} \sim q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}^k)$ from the posterior and compute the ELBO for each sample.

$$\mathcal{L}^{j,k} = \log p_{\boldsymbol{\theta}}(\mathbf{x}^{j,k} \mid \mathbf{z}^{j,k}) - \beta D_{KL}\big(q_\phi(\mathbf{z}^{j,k} \mid \mathbf{x}^k) \,||\, p(\mathbf{z})\big) \tag{6}$$

Then, to estimate the training objective, one may average over the loss from each sample:

$$\widehat{\mathcal{L}}^k_{avg} = \frac{1}{J}\sum_{j=1}^{J} \mathcal{L}^{j,k}. \tag{7}$$

In the case where we have a sparse prior, each of the J samples taken here will have different support and thus encourage development of different latent features.

Rather than give each sample equal probability, we introduce a new sampling strategy motivated by the approximation to expectations utilized in (Olshausen & Field, 1996; Connor et al., 2020). In these works, it is observed that the selected prior distribution concentrates most of its probability mass around the maximum value. This leads to an approximation of the expectation over the prior latent variable with a delta at the max value $\widehat{\mathbf{z}}$, motivating a MAP estimate:

$$\begin{aligned}\log \mathbb{E}_{p(\mathbf{z})}\left[p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] &\approx \log \int_z p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) \delta(\widehat{\mathbf{z}}) d\mathbf{z} \\ &= \max_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) + \log p(\mathbf{z}).\end{aligned} \tag{8}$$

Although our training setting differs in that we take our expectation over the variational posterior $\mathbb{E}_{q_{\boldsymbol{\phi}}}$, in many cases

Algorithm 1 Training with Thresholded Samples

Input: Training batch $\mathbf{x}^k$, threshold hyper-parameter $\boldsymbol{\lambda}_0$, whether to use a GammaPrior (along with gamma hyper-prior $\boldsymbol{\alpha}_0$), network initializations $(\boldsymbol{\phi}^0, \boldsymbol{\theta}^0)$, number of samples J, and number of iterations T.
for $t = 1$ **to** T **do**
  for $j = 1$ **to** J **do**
    $\boldsymbol{\epsilon}^{j,k} \sim p(\boldsymbol{\epsilon})$
    if GammaPrior **then**
      $\left(\mathbf{s}^{j,k}, \boldsymbol{\lambda}^{j,k}\right) \leftarrow g_{\boldsymbol{\phi}}\left(\mathbf{x}^k, \boldsymbol{\epsilon}^{j,k}\right)$
    else
      $\mathbf{s}^{j,k} \leftarrow g_{\boldsymbol{\phi}}\left(\mathbf{x}^k, \boldsymbol{\epsilon}^{j,k}\right)$
      $\boldsymbol{\lambda}^{j,k} \leftarrow \boldsymbol{\lambda}_0$
    end if
    $\widetilde{\mathbf{z}}^{j,k} \leftarrow \mathbf{s}^{j,k} + \mathcal{T}_{\boldsymbol{\lambda}^{j,k}}\left(\text{sg}\left[\mathbf{s}^{j,k}\right]\right) - \text{sg}\left[\mathbf{s}^{j,k}\right]$
    $\mathcal{L}^{j,k} \leftarrow \log p_{\boldsymbol{\theta}}(\mathbf{x}^k \mid \widetilde{\mathbf{z}}^{j,k}) - \beta D_{KL}$
  end for
  $\widehat{\mathcal{L}}^k = \arg\max_j \mathcal{L}^{j,k}$
  $(\boldsymbol{\phi}^t, \boldsymbol{\theta}^t) = \arg\max \widehat{\mathcal{L}}^k$
end for

we explicitly regularize this distribution with a KL divergence penalty against a sparse prior. To this same end, rather than approximate our expectation $\mathbb{E}_{q_\phi}$ with the average over several samples, we approximate it with a single sample with the highest likelihood:

$$\widehat{\mathcal{L}}^k_{max} = \max_j \mathcal{L}^{j,k}. \tag{9}$$

Although this provides a biased estimate of the loss, we will show in later sections that this heuristic provides an increase in performance and latent feature reuse during training. We combine this method with our sampling approach to outline our full training procedure in Algorithm 1.

4. Experiments

4.1. Linear Generator

4.1.1. Sparse Coding Performance

In our first experiment, we test different inference methods on sparse coding of the whitened image patches used in (Olshausen & Field, 1996). We train on 80,000 16x16 training patches with a dictionary of 256 elements (i.e., a linear generator $\boldsymbol{\theta} = \mathbf{A} \in \mathbb{R}^{256\times256}$). We start in this simplified setting since it allows us to compare against baseline sparse coding strategies (that use MAP estimates) and so we can analyze the dictionary we learn for each method. As a baseline, we infer coefficients $\widehat{\mathbf{z}}^k$ using the FISTA algorithm (Beck & Teboulle, 2009) to optimize objective (10). This objective includes a data fidelity term, sparsity-encouraging ℓ_1 penalty on latent features, and a Frobenius norm regularizer on the dictionary to prevent unbounded growth:

$$\min_{\mathbf{A}} \|\mathbf{x}^k - \mathbf{A}\widehat{\mathbf{z}}^k\|_2^2 + \lambda\|\widehat{\mathbf{z}}^k\|_1 + \kappa\|\mathbf{A}\|_F^2. \tag{10}$$

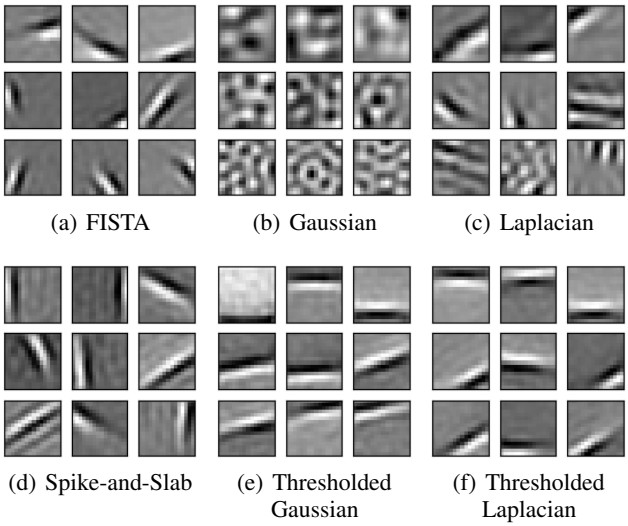
In all of our experiments, we set the sparsity hyperparameter such that roughly 10% features are non-zero for each batch of data. For FISTA, this occurs when $\lambda = 20$. We use this objective to measure validation performance of our variational methods.

To train our inference networks, we use an ELBO objective with a Frobenius norm regularizer on the dictionary:

$$\min_{\mathbf{A},\boldsymbol{\phi}} \; -\mathcal{L}(\mathbf{A}, \boldsymbol{\phi}; \mathbf{x}^k) + \kappa||\mathbf{A}||_F^2 \tag{11}$$

$$\log p_{\mathbf{A}}(\mathbf{x}^k|\mathbf{z}^k) = -\|\mathbf{x}^k - \mathbf{A}\mathbf{z}^k\|_2^2, \tag{12}$$

where $\mathbf{z}^k$ is either found via the inference procedure outlined in section 3.2 or by previous methods that used Gaussian (Kingma & Welling, 2014), Laplacian (Barello et al., 2018), or Spike-and-Slab (Tonolini et al., 2020) prior distributions. For the Spike-and-Slab prior, we approximate each Bernoulli random variable with a straight-through estimator of a Gumbel-Softmax sample (Jang et al., 2017). For each method, we apply the sampling procedure outlined in Section 3.3 with $J = 1$ and $J = 20$. Details on training hyper-parameters are provided in Appendix D.



(a) FISTA (b) Gaussian (c) Laplacian

(d) Spike-and-Slab (e) Thresholded Gaussian (f) Thresholded Laplacian

Figure 3. Entries of the learned linear generator (dictionary) for different approaches for inferring sparse codes. FISTA uses an iterative MAP estimate that optimizes objective (10) directly. Others use a variational approach with different prior distributions using $J = 20$ samples. The Spike-and-Slab, Thresholded Gaussian, and Thresholded Laplacian methods learn dictionaries with most entries visually resembling the dictionary learned by FISTA.

Figure 3 plots the columns of the learned dictionary for each inference method for $J = 20$. As expected, FISTA learns dictionary entries that qualitatively resemble Gabor wavelets (Olshausen & Field, 1996). The Laplacian prior (Barello et al., 2018) learns a few dictionary entries that resemble wavelets, with most entries resembling noise. Although the entries learned by all sparse priors resemble wavelets, only the Thresholded Laplacian is capable of learning this structure with $J = 1$. Full learned dictionaries for all methods under both sampling budgets are visualized in Appendix F.

Table 1 compares average quantitative performance over three training runs for each inference method. The Laplacian prior with a fixed threshold performs the best among variational methods on objective (10). We note that methods that learn the threshold parameters tend to increase the number of active latent features, leading to an increase in L1 penalty in the validation loss. Although the Spike-and-Slab prior minimizes the importance weighted (IWAE) loss from (Burda et al., 2016) with $K = 200$, it performs worse on the sparse coding objective. We note that the IWAE loss measures the tightness of the ELBO bound (reconstruction + KL loss), whereas the thresholding in our method has a trade-off between deviation of the KL loss and posterior expressivity.

Table 1. Performance of different inference methods with a linear generator on whitened image patches. Validation loss computed on objective (10). Multi-information measures the statistical efficiency of inferred codes. The importance weighted autoencoder (IWAE) loss (Burda et al., 2016) is a tight bound on the ELBO, but does not necessarily depict success in the sparse coding task. A table with standard deviations for $J = 20$ is included in Table 6 in Appendix F

Method/Prior Distribution	Validation Loss		Multi-Information		IWAE Loss	
	J=1	J=20	J=1	J=20	J=1	J=20
FISTA (baseline)	1.01E+02	–	8.75E+01	–	–	–
Gaussian	1.35E+03	1.35E+03	7.34E+02	7.36E+02	2.57E-01	2.16E-01
Laplacian	5.96E+02	5.79E+02	5.36E+02	5.34E+02	**2.20E-01**	2.22E-01
Spike-and-slab	2.52E+02	2.39E+02	1.94E+02	1.96E+02	2.41E-01	**2.12E-01**
Thresholded Gaussian	2.32E+02	2.30E+02	**1.67E+02**	**1.76E+02**	1.52E+00	1.33E+00
Thresholded Gaussian+Gamma	2.85E+02	2.70E+02	2.35E+02	2.29E+02	1.17E+00	1.13E+00
Thresholded Laplacian	**1.98E+02**	**1.94E+02**	1.80E+02	1.91E+02	9.30E-01	1.13E+00
Thresholded Laplacian+Gamma	2.23E+02	2.11E+02	2.38E+02	2.33E+02	9.81E-01	1.12E+00

To measure the statistical efficiency in the inferred codes, we follow previous work in estimating the multi-information, $\sum_{i=1}^{d} h(z_i) - h(\mathbf{z})$, of the inferred codes (Fallah et al., 2020; Eichhorn et al., 2009; Bethge, 2006). This quantity is minimized when the features have high joint entropy while being independent of one another, denoting a reduction in redundancy. Among variational methods, the Thresholded Gaussian minimizes this quantity.

To further understand the benefit of our proposed method, we investigate the quality of the gradient estimates through both the dictionary and inference network at the end of training. To measure this, we use the signal-to-noise ratio (SNR) metric proposed in (Rainforth et al., 2018):

$$\text{SNR}(\theta) = \left| \mathbb{E}\left[\nabla_\theta \mathcal{L}\right] / \sigma\left[\nabla_\theta \mathcal{L}\right] \right| .$$

Where θ is a single parameter from our dictionary or inference network and $\mathcal{L}$ is our training loss. In our experiments, we average the SNR over all parameters in our inference network or dictionary.

The benefit of measuring the gradient quality through a relative metric like SNR is that it weights the variance by the importance of each parameter. For example, dictionary entries that are rarely activated for a given input data will have a low expected gradient, but may have high variance. We evaluate the SNR over 1000 sparse code samples for each data point in our validation set. Figure 4 compares the SNR for both the dictionary and inference network for various methods. It can be seen that the Spike-and-Slab suffers from low SNR in the case of $J = 1$, with a significant increase for $J = 20$. We speculate that this metric explains the lack of clear wavelets in the dictionary learned by the Spike-and-Slab for $J = 1$.

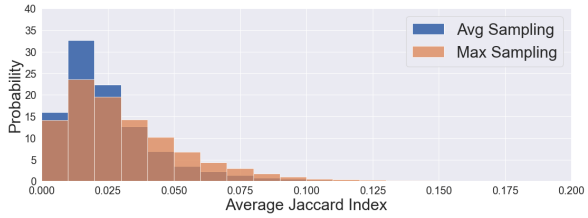
4.1.2. Sampling Performance

To test whether our sampling procedure leads to more feature reuse, we measure the consistency in the features used over several forward passes for fixed input data. Each forward pass leads to a sparse code with a different support set, indicating use of different features. Hence, we can compute the Jaccard index between the support set of several samples $\mathbf{z}^{1,k}, \ldots, \mathbf{z}^{J,k} \sim q_\phi(\mathbf{z} \mid \mathbf{x}^k)$ for fixed input to measure consistency (Jaccard, 1912). Intuitively, the Jaccard index measures similarity between discrete sets, providing a value in $[0, 1]$ based on how similar the support is over different forward passes. Let $S(\mathbf{z}^{j,k})$ denote the support of $\mathbf{z}^{j,k}$ (i.e.

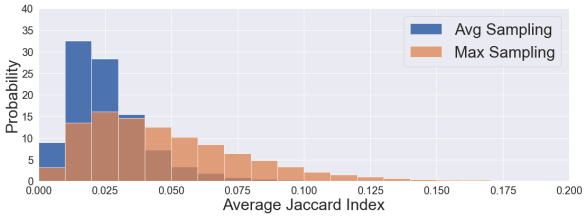
(a) Inference Network Gradient SNR

(b) Generator Gradient SNR

Figure 4. Comparison of the SNR of the (a) inference network and (b) linear generator gradients at the end of training. Thresholded methods perform the best, with all sparse priors experiencing higher SNR with $J = 20$ samples under max ELBO sampling.



(a) J=20



(b) J=100

Figure 5. Histogram of the average pairwise Jaccard index over 20 forward passes on two trained inference networks. The pairwise Jaccard index measures the consistency of the support set of inferred codes. Max ELBO sampling leads to a higher average Jaccard index, indicating reuse of developed latent features.

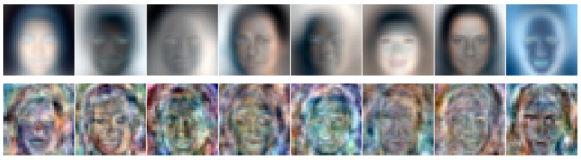
$i \in S(\mathbf{z}^{j,k})$ if $z_i^{j,k} \neq 0$), the Jaccard index is computed as:

$$C(\mathbf{z}^{m,k}, \mathbf{z}^{n,k}) = \frac{\left|S(\mathbf{z}^{m,k}) \cap S(\mathbf{z}^{n,k})\right|}{\left|S(\mathbf{z}^{m,k}) \cup S(\mathbf{z}^{n,k})\right|}. \tag{13}$$

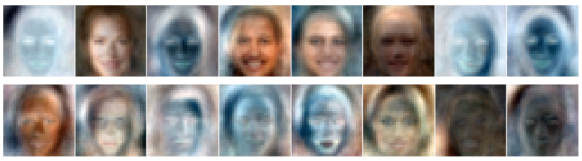
For any given input data $\mathbf{x}^k$ we can draw J samples from our inference network and compute the average pairwise Jaccard index as $\frac{1}{J(J-1)} \sum_m^J \sum_{n \neq m}^J C(\mathbf{z}^{m,k}, \mathbf{z}^{n,k})$. To normalize for the fact that each training methods leads to a different number of non-zero dictionary entries (e.g., an inference network that only uses a subset of the available support is more likely to be consistent), we measure consistency using the support of features with $\|\mathbf{a}_i\|_2^2 > 1\text{E}{-}01$.

We use this metric to compare the consistency of inference networks trained under the average ELBO scheme from (7) and the proposed max ELBO scheme in (9) using $J = [20, 100]$ samples. We observe that both methods result in 10% of features non-zero on average.

We plot the consistency on the validation set averaged over three training runs as a histogram in Figure 5. It can be seen that the max ELBO sampling procedure leads to higher consistency (indicating reuse of developed features) compared to average sampling. Qualitatively, the dictionary entries learned in the average sampling scheme has several repeated entries, indicating poor reuse in comparison to the qualitatively diverse entries learned by max sampling. We visualize the full dictionary for both in Figure F.2 in Appendix F.



(a) Gaussian



(b) Thresholded Gaussian+Gamma

Figure 6. Selected entries of an estimated dictionary for variational autoencoders trained with a (a) Gaussian prior and (b) Thresholded Gaussian+Gamma prior with a 512 dimensional latent space[3]. For each method, the first row denotes dictionary entries with the highest magnitude and the bottom row denotes random entries from the middle of the dictionary.

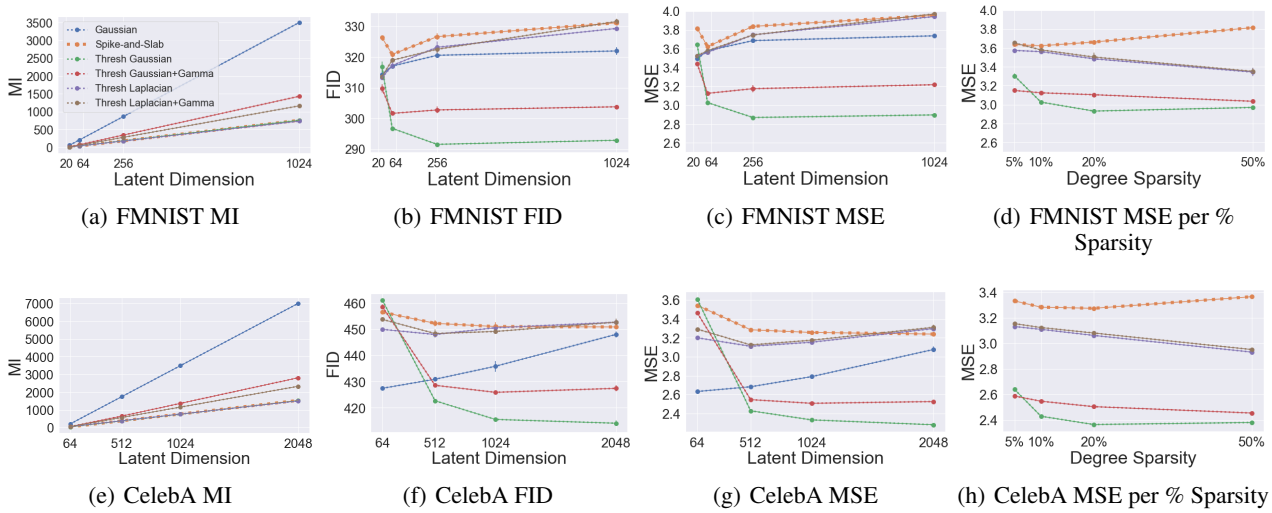
4.2. DNN Generator

We next investigate the efficacy of variational sparse coding in unsupervised learning on the Fashion MNIST (FMNIST) (Xiao et al., 2017) and CelebA datasets (Liu et al., 2015). We follow a similar methodology as the previous section, except that we apply a DNN generator in place of the linear dictionary (more details in Appendix E). Following the principle of overcomplete representations in sparse coding, where the robustness of latent features increases with the dimensionality (Olshausen, 2013), we compare performance as we sweep dimensionality.

Figure 7 shows the quantitative performance of different prior methods trained with either increasing latent dimensionality or percentage of latent features non-zero. For both datasets, as the dimensionality increases with a fixed degree of sparsity, it can be seen that the Gaussian prior (corresponding to a variational autoencoder (Kingma & Welling, 2014; Higgins et al., 2017)) has a significant increase in redundancy, measured via the multi-information, with a slight increase in the validation Fréchet inception distance (FID) (Heusel et al., 2017) and MSE. Alternatively, all the sparse priors show superior scaling with dimensionality, with the Thresholded Gaussian and Thresholded Gaussian+Gamma depicting superior performance to the Spike-and-Slab model from Tonolini et al. (2020). Additionally, all thresholded methods scale better with MSE as the degree of sparsity (corresponding to the % of features non-zero) increases.

We observe the same qualitative generative interpretability

[3]This visualization is from a training run where the sparsity prior is set to encourage 5% features non-zero.



(a) FMNIST MI (b) FMNIST FID (c) FMNIST MSE (d) FMNIST MSE per % Sparsity

(e) CelebA MI (f) CelebA FID (g) CelebA MSE (h) CelebA MSE per % Sparsity

Figure 7. Comparison of variational autoencoder with different prior distributions as the latent feature dimensionality increases on the Fashion MNIST (a-d) and CelebA (e-h) datasets. (a, e) Multi-information scales at a higher rate with the Gaussian prior, indicating inferior statistical efficiency at higher latent dimensions. (b, f) The Fréchet Inception Distance of images reconstructed by learned DNN generator. (c, g) Mean squared error on validation data for a fixed degree of sparsity. (d, h) Mean squared error for a fixed dimensionality with an increase in the degree of sparsity (indicating the % of non-zero latent features set by the prior).

when sweeping each individual latent feature as in previous work that investigates Gaussian priors (Higgins et al., 2017) and sparse priors (Tonolini et al., 2020). In these works, it is observed that each latent feature corresponds to a specific semantic change in the generated image (such as changes in skin color, background color, hairstyle, etc.). Unfortunately, in the case where one trains a DNN generator, they are unable to observe the generator features the same way one could with the columns of the linear dictionary.

To this end, we estimate a linear dictionary for each fully trained CelebA inference network to visualize what is encoded by each latent feature. Given a collection of data samples $\mathbf{X} = \left[\mathbf{x}^1, \ldots, \mathbf{x}^n\right]^T$ and corresponding inferred codes $\mathbf{Z} = \left[\mathbf{z}^1, \ldots, \mathbf{z}^n\right]^T$ from an inference network with fixed weights fully trained with a DNN generator $\mathbf{z}^k \sim q_\phi(\mathbf{z} \mid \mathbf{x}^k)$, one can estimate a linear dictionary as (Isely et al., 2010; Fallah et al., 2020):

$$\mathbf{X} = \mathbf{A}\mathbf{Z}$$

$$\widehat{\mathbf{A}} = \left(\mathbf{X}\mathbf{Z}^T\right)\left(\mathbf{Z}\mathbf{Z}^T\right)^{-1}.$$

We use this technique with each trained inference network to analyze which features in the training dataset are represented by each latent feature. We plot selected entries from dictionaries estimated for the Gaussian and Thresholded Gaussian+Gamma in Figure 6. Interestingly, the first eight entries from the Gaussian prior resemble the qualitative features observed in the β-VAE (Higgins et al., 2017). Poor statistical efficiency in the Gaussian prior indicates

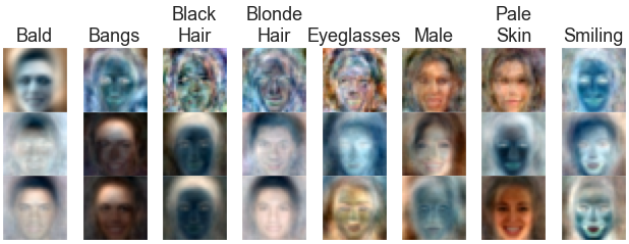


Figure 8. Estimated dictionary entries for latent features with the highest absolute point-biserial correlation with the CelebA binary attribute labels. From top to bottom, the rows denote the Gaussian prior, Thresholded Gaussian prior, and Thresholded Gaussian+Gamma prior.

that certain latent features are not encoding useful information from training data, leading to corresponding estimated dictionary entries that resemble noise (visualized in the second row). Meanwhile, all the entries for the sparse prior more clearly resemble features in the training dataset. While the qualitative results here are similar to previous research comparing learned features from principal component analysis (PCA) and independent component analysis (ICA) on face datasets (Bartlett et al., 2002), a detailed quantitative comparison with those results is beyond the scope of the present paper. We visualize more estimated dictionary plots in Appendix G.

Finally, we use the binary attribute labels included with the CelebA dataset to measure correlation between latent fea-

tures and attributes in the input data. Given vectors of 40 binary attributes for each data sample $\mathbf{y}^1, \ldots, \mathbf{y}^N$ and corresponding latent features $\mathbf{z}^1, \ldots, \mathbf{z}^N$ from a trained inference network, we can compute the point-biserial correlation coefficient between each attribute m and each dimension of latent feature n as:

$$\rho_{pb}^{m,n} = \frac{M_1 - M_0}{s_N} \frac{\sqrt{n_1 n_0}}{\sqrt{N}}.$$

Where M_1 and M_0 are the sample mean of latent features z_n^k corresponding to when attribute label y_m^k equals one or zero, respectively. s_N is the sample standard deviation over all features, and n_1 and n_0 respectively denote the total count of attribute labels that equal 1 and 0.

We plot the estimated dictionary entry corresponding to the feature with the highest absolute correlation for selected attributes in Figure 8. Noting that features can be positive or negative for a given input data, it can be seen that the sparse priors lead to dictionary entries with features that visually resemble the semantic meaning of each attribute label.

5. Discussion

In this work, we have presented a new method for variational sparse coding by thresholding samples from an inference network. This simple approach avoids discrete random variables in its parameterization, and we show that it leads to superior performance and gradient estimation. Compared to a standard Gaussian prior, it provides more statistical efficiency in its use of latent features. One area of future work may consider alternative discrete random variable estimators that employ control variates (Tucker et al., 2017). Another interesting direction is the substitution of the KL divergence with distance metrics that perform better when the prior lies on a manifold (as is the case with sparse signals) (Patrini et al., 2020).

6. Acknowledgements

The authors would like to thank Adam Charles, Francesco Tonolini, and Linxing Preston Jiang for their correspondence regarding their experience with variational sparse coding. This work was partially supported by NSF CAREER award CCF-1350954, ONR grant number N00014-15-1-2619 and AFRL grant number FA8750-19-C-0200.

References

Ainsworth, S. K., Foti, N. J., Lee, A. K. C., and Fox, E. B. oi-VAE: Output Interpretable VAEs for Nonlinear Group Factor Analysis. pp. 10, 2018.

Barello, G., Charles, A. S., and Pillow, J. W. Sparse-Coding Variational Auto-Encoders. preprint, Neuroscience, August 2018. URL http://biorxiv.org/lookup/doi/10.1101/399246.

Bartlett, M., Movellan, J., and Sejnowski, T. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13(6):1450–1464, November 2002. ISSN 1941-0093. doi: 10.1109/TNN.2002.804287. Conference Name: IEEE Transactions on Neural Networks.

Bauer, M. and Mnih, A. Resampled Priors for Variational Autoencoders. *arXiv:1810.11428 [cs, stat]*, April 2019. URL http://arxiv.org/abs/1810.11428. arXiv: 1810.11428.

Beck, A. and Teboulle, M. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, January 2009. ISSN 1936-4954. doi: 10.1137/080716542. URL http://epubs.siam.org/doi/10.1137/080716542.

Bengio, Y., Léonard, N., and Courville, A. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432 [cs]*, August 2013. URL http://arxiv.org/abs/1308.3432. arXiv: 1308.3432.

Bethge, M. Factorial coding of natural images: how effective are linear models in removing higher-order dependencies? *Journal of the Optical Society of America A*, 23(6):1253, June 2006. ISSN 1084-7529, 1520-8532. doi: 10.1364/JOSAA.23.001253. URL https://www.osapublishing.org/abstract.cfm?URI=josaa-23-6-1253.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance Weighted Autoencoders. *arXiv:1509.00519 [cs, stat]*, November 2016. URL http://arxiv.org/abs/1509.00519. arXiv: 1509.00519.

Connor, M. C., Canal, G. H., and Rozell, C. J. Variational Autoencoder with Learned Latent Structure. *arXiv:2006.10597 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2006.10597. arXiv: 2006.10597.

Cremer, C., Morris, Q., and Duvenaud, D. Reinterpreting Importance-Weighted Autoencoders. *arXiv:1704.02916 [stat]*, August 2017. URL http://arxiv.org/abs/1704.02916. arXiv: 1704.02916.

Daubechies, I., Defrise, M., and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *arXiv:math/0307152*, November 2003. URL http://arxiv.org/abs/math/0307152. arXiv: math/0307152.

Donoho, D. L. and Johnstone, I. M. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, September 1994. ISSN 0006-3444. doi: 10.1093/biomet/81.3.425. URL https://doi.org/10.1093/biomet/81.3.425.

Drefs, J., Guiraud, E., and Lücke, J. Evolutionary variational optimization of generative models. *Journal of Machine Learning Research*, 23(21):1–51, 2022. URL http://jmlr.org/papers/v23/20-233.html.

Eichhorn, J., Sinz, F., and Bethge, M. Natural Image Coding in V1: How Much Use Is Orientation Selectivity? *PLoS Computational Biology*, 5(4):e1000336, April 2009. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000336. URL https://dx.plos.org/10.1371/journal.pcbi.1000336.

Elad, M. *Sparse and Redundant Representations*. Springer New York, New York, NY, 2010. ISBN 978-1-4419-7010-7 978-1-4419-7011-4. doi: 10.1007/978-1-4419-7011-4. URL http://link.springer.com/10.1007/978-1-4419-7011-4.

Fallah, K., Willats, A. A., Liu, N., and Rozell, C. J. Learning sparse codes from compressed representations with biologically plausible local wiring constraints. preprint, Neuroscience, October 2020. URL http://biorxiv.org/lookup/doi/10.1101/2020.10.23.352443.

Garrigues, P. and Olshausen, B. Group Sparse Coding with a Laplacian Scale Mixture Prior. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

Girolami, M. A Variational Method for Learning Sparse and Overcomplete Representations. *Neural Computation*, 13(11):2517–2532, November 2001. ISSN 0899-7667, 1530-888X. doi: 10.1162/089976601753196003. URL https://direct.mit.edu/neco/article/13/11/2517-2532/6475.

Goodfellow, I. J., Courville, A., and Bengio, Y. Large-Scale Feature Learning With Spike-and-Slab Sparse Coding. pp. 8, 2012.

Gregor, K. and LeCun, Y. Learning Fast Approximations of Sparse Coding. pp. 8, 2010.

Grover, A., Gummadi, R., Lazaro-Gredilla, M., Schuurmans, D., and Ermon, S. Variational Rejection Sampling. pp. 10, 2014.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. Beta-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK. pp. 22, 2017.

Isely, G., Hillar, C., and Sommer, F. Deciphering subsampled data: adaptive compressive sampling as a principle of brain communication. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://papers.nips.cc/paper/2010/hash/a0160709701140704575d499c997b6ca-Abstract.html.

Jaccard, P. The Distribution of the Flora in the Alpine Zone.1. *New Phytologist*, 11(2):37–50, 1912. ISSN 1469-8137. doi: 10.1111/j.1469-8137.1912.tb05611.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8137.1912.tb05611.x.

Jang, E., Gu, S., and Poole, B. Categorical Reparameterization with Gumbel-Softmax. *arXiv:1611.01144 [cs, stat]*, August 2017. URL http://arxiv.org/abs/1611.01144. arXiv: 1611.01144.

Jankowiak, M. and Obermeyer, F. Pathwise Derivatives Beyond the Reparameterization Trick. *arXiv:1806.01851 [cs, stat]*, July 2018. URL http://arxiv.org/abs/1806.01851. arXiv: 1806.01851.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An Introduction to Variational Methods for Graphical Models. In Jordan, M. I. (ed.), *Learning in Graphical Models*, pp. 105–161. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-010-6104-9 978-94-011-5014-9. doi: 10.1007/978-94-011-5014-9_5. URL http://link.springer.com/10.1007/978-94-011-5014-9_5.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. URL http://arxiv.org/abs/1312.6114. arXiv: 1312.6114.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Lucas, J., Tucker, G., Grosse, R., and Norouzi, M. Don't Blame the ELBO! A Linear VAE Perspective on Posterior Collapse. *arXiv:1911.02469 [cs, stat]*, November 2019. URL http://arxiv.org/abs/1911.02469. arXiv: 1911.02469.

Maddison, C. J., Mnih, A., and Teh, Y. W. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv:1611.00712 [cs, stat]*, March 2017. URL http://arxiv.org/abs/1611.00712. arXiv: 1611.00712.

Meyer, G. P. An Alternative Probabilistic Interpretation of the Huber Loss. pp. 5261–5269, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Meyer_An_Alternative_Probabilistic_Interpretation_of_the_Huber_Loss_CVPR_2021_paper.html.

Mitchell, T. J. and Beauchamp, J. J. Bayesian Variable Selection in Linear Regression. *Journal of the American Statistical Association*, 83(404): 1023–1032, December 1988. ISSN 0162-1459. doi: 10.1080/01621459.1988.10478694. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1988.10478694. Publisher: Taylor & Francis.

Moran, G. E., Sridhar, D., Wang, Y., and Blei, D. M. Identifiable Variational Autoencoders via Sparse Decoding. *arXiv:2110.10804 [cs, stat]*, October 2021. URL http://arxiv.org/abs/2110.10804. arXiv: 2110.10804.

Neal, R. M. and Hinton, G. E. A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants. In Jordan, M. I. (ed.), *Learning in Graphical Models*, NATO ASI Series, pp. 355–368. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-011-5014-9. doi: 10.1007/978-94-011-5014-9_12. URL https://doi.org/10.1007/978-94-011-5014-9_12.

Olshausen, B. A. Highly overcomplete sparse coding. In *Human Vision and Electronic Imaging XVIII*, volume 8651, pp. 168–176. SPIE, March 2013. doi: 10.1117/12.2013504.

Olshausen, B. A. and Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583): 607–609, June 1996. ISSN 1476-4687. doi: 10.1038/381607a0. URL https://www.nature.com/articles/381607a0. Number: 6583 Publisher: Nature Publishing Group.

Olshausen, B. A. and Field, D. J. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487, August 2004. ISSN 0959-4388. doi: 10.1016/j.conb.2004.07.007. URL https://www.sciencedirect.com/science/article/pii/S0959438804001035.

Oord, A. v. d., Vinyals, O., and Kavukcuoglu, K. Neural Discrete Representation Learning. *arXiv:1711.00937 [cs]*, May 2018. URL http://arxiv.org/abs/1711.00937. arXiv: 1711.00937.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Patrini, G., Berg, R. v. d., Forré, P., Carioni, M., Bhargav, S., Welling, M., Genewein, T., and Nielsen, F. Sinkhorn AutoEncoders. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, pp. 733–743. PMLR, August 2020. URL https://proceedings.mlr.press/v115/patrini20a.html. ISSN: 2640-3498.

Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., and Teh, Y. W. Tighter Variational Bounds are Not Necessarily Better. *arXiv:1802.04537 [cs, stat]*, March 2018. URL http://arxiv.org/abs/1802.04537. arXiv: 1802.04537.

Ranganath, R., Gerrish, S., and Blei, D. M. Black Box Variational Inference. *arXiv:1401.0118 [cs, stat]*, December 2013. URL http://arxiv.org/abs/1401.0118. arXiv: 1401.0118.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv:1401.4082 [cs, stat]*, May 2014. URL http://arxiv.org/abs/1401.4082. arXiv: 1401.4082.

Roeder, G., Wu, Y., and Duvenaud, D. Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference. *arXiv:1703.09194 [cs, stat]*, May 2017. URL http://arxiv.org/abs/1703.09194. arXiv: 1703.09194.

Rozell, C. J., Johnson, D. H., Baraniuk, R. G., and Olshausen, B. A. Sparse Coding via Thresholding and Local Competition in Neural Circuits. *Neural Computation*, 20(10):2526–2563, October 2008. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.2008.03-07-486. URL https://direct.mit.edu/neco/article/20/10/2526-2563/7343.

Salimans, T. A Structured Variational Auto-encoder for Learning Deep Hierarchies of Sparse Features. *arXiv:1602.08734 [cs, stat]*, February 2016. URL http://arxiv.org/abs/1602.08734. arXiv: 1602.08734.

Seeger, M. W. Bayesian Inference and Optimal Design for the Sparse Linear Model. pp. 55, April 2008.

Sheikh, A.-S. and Lücke, J. Select-and-Sample for Spike-and-Slab Sparse Coding. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Singh, R., Ling, J., and Doshi-Velez, F. Structured Variational Autoencoders for the Beta-Bernoulli Process. pp. 9, 2017.

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

Tonolini, F., Jensen, B. S., and Murray-Smith, R. Variational Sparse Coding. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, pp. 690–700. PMLR, August 2020. URL https://proceedings.mlr.press/v115/tonolini20a.html. ISSN: 2640-3498.

Tropp, J. A. and Gilbert, A. C. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, December 2007. ISSN 1557-9654. doi: 10.1109/TIT.2007.909108. Conference Name: IEEE Transactions on Information Theory.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, D., and Sohl-Dickstein, J. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. *arXiv:1703.07370 [cs, stat]*, November 2017. URL http://arxiv.org/abs/1703.07370. arXiv: 1703.07370.

Tucker, G., Lawson, D., Gu, S., and Maddison, C. J. Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives. *arXiv:1810.04152 [cs, stat]*, November 2018. URL http://arxiv.org/abs/1810.04152. arXiv: 1810.04152.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747.

Yang, A. Y., Zhou, Z., Ganesh, A., Sastry, S. S., and Ma, Y. Fast L1-Minimization Algorithms For Robust Face Recognition. *arXiv:1007.3753 [cs]*, August 2012. URL http://arxiv.org/abs/1007.3753. arXiv: 1007.3753.

Zhang, C., Butepage, J., Kjellstrom, H., and Mandt, S. Advances in Variational Inference. *arXiv:1711.05597 [cs, stat]*, October 2018. URL http://arxiv.org/abs/1711.05597. arXiv: 1711.05597.

A. Thresholded Laplacian Derivation

To determine the distribution of our thresholded samples $p(z)$, we marginalize the conditional distribution with respect to $p(s)$. The shifted soft-threshold can be viewed as a shift of the pdf of $p(s)$ towards the mean value μ, as depicted in Figure 2.

We thus break up the marginalization over $p(s)$ into two disjoint regions: the region that gets collapsed into the origin $p(|s-\mu| \leq \lambda)$ and the regions that get shifted towards the mean $p(|s-\mu| \geq \lambda)$. We can then write the expectation:

$$\begin{aligned} p(z) &= \mathbb{E}_{p(s)}\left[p(z \mid s)\right] \\ &= p(z \mid |s-\mu| \leq \lambda) p(|s-\mu| \leq \lambda) \\ &\quad + p(z \mid |s-\mu_0| \geq \lambda) p(|s-\mu| \geq \lambda) \\ &= (1-\gamma)\delta(z) + \gamma p(z \mid |s-\mu| \geq \lambda) \end{aligned}$$

Where $(1-\gamma) = 1 - \exp(-\lambda b^{-1})$ and $\gamma = \exp(-\lambda b^{-1})$, as shown in the main text.

To derive $p(z \mid |s-\mu| \geq \lambda)$, we write the pdf of $p(s)$ with the shift applied from soft-thresholding:

$$\begin{aligned} &\begin{cases} \frac{1}{2b}\exp\left(-\frac{|s-\mu+\lambda|}{b}\right) & s-\mu \leq 0 \\ \frac{1}{2b}\exp\left(-\frac{|s-\mu-\lambda|}{b}\right) & s-\mu > 0 \end{cases} \\ =&\frac{1}{2b}\exp\left(-\frac{|s-\mu+\text{sign}(s-\mu)\lambda|}{b}\right) \\ =&\frac{1}{2b}\exp\left(-\frac{|s-\mu|}{b}\right)\exp\left(-\lambda b^{-1}\right) \end{aligned}$$

Where the third line follows from the equality condition of the triangle inequality (both terms have the same sign) and from the fact that λ is always positive.

This is precisely equal to $p(z \mid |s-\mu| \geq \lambda)$ multiplied by a normalizing constant $\frac{1}{Z}$ to become a valid distribution (accounting for the probability mass that was moved to zero). This normalizing constant is exactly one minus the mass moved to zero $Z = \gamma = \exp\left(-\lambda b^{-1}\right)$. This leads to our final conditional probability:

$$p(z \mid |s-\mu_0| \geq \lambda) = \frac{1}{2b}\exp\left(-\frac{|s-\mu|}{b}\right) \tag{14}$$

B. Reparameterization Details for Thresholded Samples

To reparameterize thresholded samples, we first draw samples s_i^k for each feature space dimension from a base distribution that is either Gaussian or Laplacian. For an input data $\mathbf{x}^k$, our encoder network outputs a shift and log-scale for each dimension of our latent feature for both distributions. The KL divergence loss is always computed on the base distribution, summing over each feature dimension.

For a Gaussian distribution, we apply the standard reparameterization technique from (Kingma & Welling, 2014) using $(\mu_i^k, 2\log\sigma_i^k)$ output from a DNN:

$$\epsilon_i^{j,k} \sim \mathcal{N}(0,1)$$

$$z_i^{j,k} = g_\phi(\mathbf{x}^k, \epsilon_i^{j,k})_i = \sigma_i^k \epsilon_i^{j,k} + \mu_i^k$$

with a KL divergence depending on scale prior σ_0^2 (Kingma & Welling, 2014):

$$D_{KL}^{\mathcal{N}} = \left(\left(\mu_i^k\right)^2 + \left(\sigma_i^k\right)^2\right)\left(2\sigma_0\right)^{-1} + \log\frac{\sigma_0}{\sigma_i^k} - \frac{1}{2}$$

For a Laplacian distribution, we apply the reparameterization rule from (Connor et al., 2020) using $(\mu_i^k, \log b_i^k)$ output from a DNN:

$$\epsilon_i^{j,k} \sim \text{Unif}\left(-\frac{1}{2}, \frac{1}{2}\right)$$

$$z_i^{j,k} = g_\phi(\mathbf{x}^k, \epsilon_i^{j,k})_i = -b\ \text{sign}(\epsilon_i^{j,k})\log\left(1 - 2\epsilon_i^{j,k}\right)$$

Applying the KL divergence loss derived in (Meyer, 2021) with a scale prior b_0:

$$D_{KL}^{L} = \frac{|\mu_i^k|}{b_0} + \frac{b_i^k \exp\left(-|\mu_i^k|\left(b_i^k\right)^{-1}\right)}{b_0} + \log\frac{b_0}{b_i^k} - 1$$

To prevent numerical instability during training, we apply clamping to the log term in $g_\phi(\mathbf{x}^k, \epsilon_i^{j,k})_i$ to have a minimum value of $1\text{E}{-}06$. Additionally, we multiply a warm-up variable ω to the inferred scale variable b_i^k. At the beginning of training we set this variable $\omega^{(0)} = 0.1$ and increment it at each training iteration by $2\text{E}{-}04$ until it reaches a maximum value of 1.0.

When applying thresholding to either distribution, we either apply a fixed threshold set as a hyper-parameter $\lambda_i = \lambda_0$ or infer a threshold for each feature dimension. To perform reparameterization, we follow the pathwise derivative methods proposed in (Jankowiak & Obermeyer, 2018), parameterized by a neural network that outputs $(\log\alpha_i^k, \log\beta_i^k)$. For numerical stability, we clamp these values to be between $[1\text{E}{-}06, 1\text{E}{+}06]$. We use both the reparameterization and KL divergence implementations included in PyTorch 1.10 (Paszke et al., 2019). In all cases, we set the prior $\alpha_0 = 3, \beta_0 = \frac{3}{\lambda_0}$. We find that on average this encourages the inferred threshold values to equal λ_0.

Our reparameterization procedure differs from the Spike-and-Slab from (Tonolini et al., 2020) in a few ways. First and foremost, we avoid the need of applying continuous approximations to Bernoulli random variables by applying shifted soft-thresholding. This also results in our model not needing to estimate a spike probability for each latent

variable, but rather just parameters for the base distribution. Finally, rather than use the KL divergence for Spike-and-Slabs derived in (Tonolini et al., 2020), we only penalize the base distribution with a KL divergence.

C. Straight-Through Estimator Details

The shifted soft-threshold is non-differentiable, with a subgradient of 0 when a latent component $z_i = 0$. This is problematic since it serves as a block of gradient flowing from the generator to the inference network whenever a latent feature is not used (a common occurrence when sparsity is imposed on latent features). In these circumstances, although the gradient is blocked from the generator, a gradient is still computed for the KL divergence term. This empirically causes numerical instability (gradients go to NaN) or the inference network to primarily minimize the KL divergence term, leading to posterior collapse (Lucas et al., 2019). Note that this same problem would occur with a reparameterization procedure that simply sets $z_i = 0$ without applying $\mathcal{T}_{\boldsymbol{\lambda}}$ in the cases where $\mathcal{T}_{\boldsymbol{\lambda}}$ would result in a zero-valued latent component.

Posterior collapse measures the extent to which the inference network is indistinguishable from the prior (Lucas et al., 2019):

$$\mathbb{P}_{\mathbf{x}\sim p(\mathbf{x})}\left[D_{KL}\left(q_\phi(z_i,\mathbf{x})\|p(z_i)\right)\leq\epsilon\right]\geq 1-\delta. \tag{15}$$

To measure another tangible problem of all latent features values going to zero, we also introduce the metric of feature collapse to measure whether inferred latent features $\widehat{\mathbf{z}}$ are zero for most images $\mathbf{x}$:

$$\mathbb{P}_{\mathbf{x}\sim p(\mathbf{x})}\left[|\widehat{z}_i|\leq\epsilon\right]\geq 1-\delta. \tag{16}$$

We measure these two quantities in Table 2 using subgradient and straight-through estimators. It can be observed that using a single sample, the subgradient leads to numerical instability in all three runs. Increasing the number of samples avoids this instability, at the cost of complete posterior and feature collapse in all three runs. Meanwhile, the straight-through estimator avoids this issue.

D. Linear Generator Training Details

For each method, we train using $80,000$ image patches and validate using $16,000$ image patches of dimension 16×16 with a dictionary of 256 entries. We train for 300 epochs using a batch size of 100. Our initial learning rate for the dictionary is $5\text{E}{-01}$ and we apply an exponential decay by a factor of 0.99 each epoch. Our inference network is trained with an initial learning rate of $1\text{E}{-02}$, using an SGD+Nesterov optimizer with a CycleScheduler. For FISTA, we set $\kappa = 1\text{E}{-03}$. For the variational methods, we set $\kappa = 1\text{E}{-04}$, and the scale parameter of our prior $p(\mathbf{z})$ equal to 0.1 when applicable. Unless specified, we set the KL weight as $\beta = 1\text{E}{-02}$ for each method. We run each method for three trials, using the random seeds $\{0, 1337, 747\}$.

For FISTA, we multiply λ by a warmup parameter ω that starts at $\omega^{(0)} = 0.1$ and is incremented by $1\text{E}{-04}$ each training iteration, capping at a maximum value of $\omega^{(t)} = 1.0$. We find that this prevents all dictionary entries from going to zero when training with a Frobenius norm regularizer.

For the Spike-and-Slab, we use the same warmup strategy proposed in (Tonolini et al., 2020). We set $\omega^{(0)} = 0.0$, incrementing by $2\text{E}{-04}$ each iteration after iteration 1500. This parameter is also capped out at $\omega^{(t)} = 1.0$. Additionally, we apply a warmup to the temperature parameter τ for the Gumbel-Softmax used in the Spike-and-Slab. This starts at $\tau^{(0)} = 1.0$ and is decreased by a multiplicative factor of 0.9995 until it reaches a minimum value of $\tau^{(t)} = 5\text{E}{-01}$. We set the prior for the spike probability as 0.1.

For the Thresholded Gaussian and Laplacian methods, we use the analytic results to set $\boldsymbol{\lambda}_0$ respectively as 0.52 and 0.25. We find that these values lead to 10% of the latent features non-zero for each input data on average. When using a Gamma prior on the threshold parameter, we reduce the corresponding KL weight by a multiplicative factor of $1\text{E}{-01}$ (resulting in $\beta_1 = 1\text{E}{-02}$ and $\beta_2 = 1\text{E}{-03}$). We set $\boldsymbol{\alpha}_0 = 3$ for each method that uses a Gamma prior.

Each inference network uses an MLP backbone with the same architecture, with a separate projection head for each distribution parameter. The architecture for the backbone is outlined in Table 3. For each distribution parameter, an additional 256×256 linear layer is added on top of the output of this network.

E. DNN Generator Training Details

For CelebA, we use $150,000$ training samples and $19,000$ validation samples. All samples are center cropped to 140×140 pixels and then resized to 64×64. We also apply a random horizontal flip to training samples only. We train for 300 epochs using a batch size of 512 across two Nvidia RTX 3080s. We use an initial learning rate of $3\text{E}{-04}$ using the Adam optimizer with $\beta = (0.5, 0.999)$, weight decay equal to $1\text{E}{-05}$, and a sample budget of $J = 10$. Additionally, we use the automatic mixed precision (AMP) and DistributedDataParallel implementations included in PyTorch 1.10 (Paszke et al., 2019). We set $\beta = 1\text{E}{-03}$ for the KL divergence term in all methods. In the case where we have a Gamma distribution on the threshold parameter, we multiply the KL term by a factor of $1\text{E}{-01}$ (leading to $\beta_1 = 1\text{E}{-03}$ and $\beta_2 = 1\text{E}{-04}$). The network architecture for the encoder and the decoder are included in Table 4. All

Table 2. Posterior collapse and feature collapse for different sparsity estimators. Thresholding methods use a straight-through (ST) estimator or a subgradient (SG) estimator. NaN denotes a run that failed due to gradients equal to NaN.

Method/Prior Distribution	% Posterior Collapse ($\epsilon = 1\text{E}{-}02, \delta = 5\text{E}{-}02$)		% Feature Collapse ($\epsilon = 1\text{E}{-}02, \delta = 5\text{E}{-}02$)	
	J=1	J=20	J=1	J=20
Thresholded Laplacian (SG)	NaN	100.00%	NaN	100.00%
Thresholded Laplacian (ST)	4.17%	4.56%	0.00%	0.00%

Table 3. Inference Network Backbone for Linear Generator

Input $\in \mathbb{R}^{16\times 16}$
Linear: 256×512
ReLU
Linear: 512×1024
ReLU
Linear: 1024×512
ReLU
Linear: 512×256
ReLU
Output $\in \mathbb{R}^{256}$

other methodology is kept consistent with the linear generator. We run each method for three trials, using the random seeds $\{0, 1337, 747\}$.

For FMNIST, we train with all $60,000$ training samples and evaluate on all $10,000$ validation samples. Hyperparameters are kept the same as CelebA experiments, except for a KL divergence weight $\beta = 1\text{E}{-}02$ and sampling budget $J = 20$. A smaller network architecture is used for FMNIST, depicted in Table 5.

F. Additional Results for Linear Generator

F.1. Full Dictionary for Different Priors

F.2. Full Dictionary for Different Sampling Methods

G. Additional Results for DNN Generator

G.1. Recovered Dictionary at Different Dimensionality

Table 4. Network Architecture for CelebA Experiments

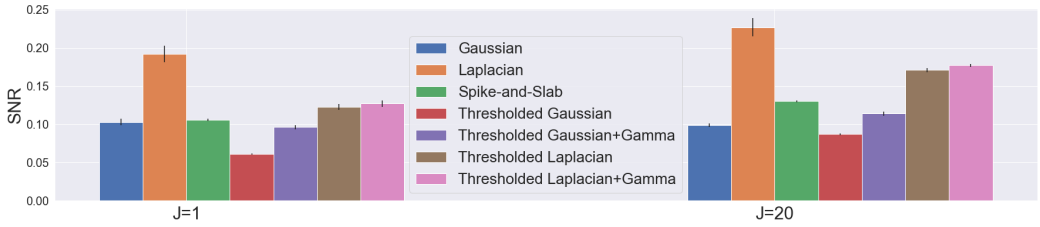
Encoder Network	Decoder Network
Input $\in \mathbb{R}^{64\times64\times3}$	Input $\in \mathbb{R}^{d}$
conv: chan: 64, kern: 4, stride: 2, pad: 1	Linear: $d \times 1024$ Units
BatchNorm: feat: 64	ReLU
ReLU	convTranpose: chan: 128, kern: 4, stride: 2, pad: 1
conv: chan: 64, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 128
BatchNorm: feat: 64	ReLU
ReLU	convTranpose: chann: 128, kern: 4, stride: 2, pad: 1
conv: chan: 128, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 128
BatchNorm: feat: 128	ReLU
ReLU	convTranpose: chan: 64, kernel: 4, stride: 2, pad: 1
conv: chan: 256, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 64
BatchNorm: feat: 256	ReLU
ReLU	convTranpose: chan: 64, kernel: 4, stride: 2, pad: 1
conv: chan: 256, kern: 4, stride: 2, pad: 0	BatchNorm: feat: 64
BatchNorm: feat: 256	ReLU
ReLU	convTranpose: chan: 3, kernel: 4, stride: 2, pad: 1
	Sigmoid
Output $\in \mathbb{R}^{256}$	Output $\in \mathbb{R}^{64\times64\times3}$

Table 5. Network Architecture for FMNIST Experiments

Encoder Network	Decoder Network
Input $\in \mathbb{R}^{28\times28\times1}$	Input $\in \mathbb{R}^{d}$
conv: chan: 64, kern: 4, stride: 2, pad: 1	Linear: $d \times 128$ Units
BatchNorm: feat: 64	ReLU
ReLU	convTranpose: chan: 128, kern: 4, stride: 2, pad: 0
conv: chan: 64, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 128
BatchNorm: feat: 64	ReLU
ReLU	convTranpose: chann: 64, kern: 4, stride: 1, pad: 0
conv: chan: 128, kern: 4, stride: 1, pad: 0	BatchNorm: feat: 64
BatchNorm: feat: 128	ReLU
ReLU	convTranpose: chan: 64, kernel: 4, stride: 2, pad: 1
conv: chan: 64, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 64
BatchNorm: feat: 128	ReLU
ReLU	convTranpose: chan: 1, kernel: 4, stride: 2, pad: 1
	Sigmoid
Output $\in \mathbb{R}^{128}$	Output $\in \mathbb{R}^{28\times28\times1}$

Table 6. A copy of Table 1 from the main text with $J = 20$ samples with standard deviations included

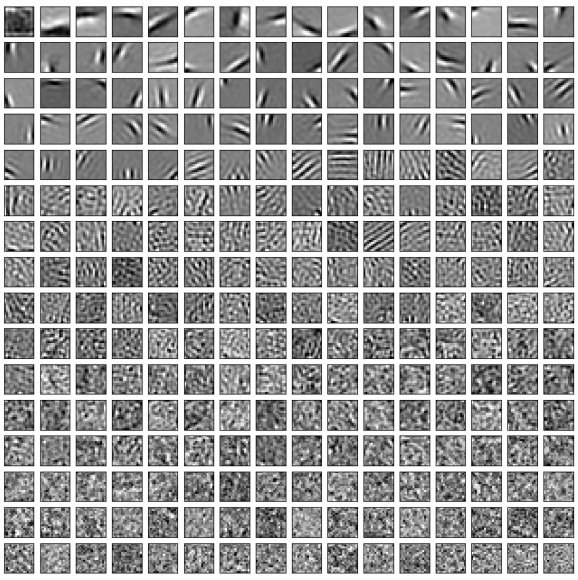
Method/Prior Distribution	Validation Loss	Multi-Information	IWAE Loss
FISTA (baseline)	1.01E+02 $\pm$ 7.00E-01	8.75E+01 $\pm$ 5.43E-01	–
Gaussian	1.35E+03 $\pm$ 2.08E+00	7.36E+02 $\pm$ 3.06E-01	2.16E-01 $\pm$ 2.64E-04
Laplacian	5.79E+02 $\pm$ 1.51E+00	5.34E+02 $\pm$ 5.29E-01	2.22E-01 $\pm$ 7.97E-04
Spike-and-slab	2.39E+02 $\pm$ 4.27E+00	1.96E+02 $\pm$ 2.57E+00	2.12E-01 $\pm$ 2.64E-02
Thresholded Gaussian	2.30E+02 $\pm$ 1.87E+00	1.76E+02 $\pm$ 2.08E-01	1.33E+00 $\pm$ 3.80E-03
Thresholded Gaussian+Gamma	2.70E+02 $\pm$ 1.87E+00	2.29E+02 $\pm$ 3.79E-01	1.13E+00 $\pm$ 2.47E-03
Thresholded Laplacian	1.94E+02 $\pm$ 1.66E+00	1.91E+02 $\pm$ 1.53E-01	1.13E+00 $\pm$ 1.29E-02
Thresholded Laplacian+Gamma	2.11E+02 $\pm$ 1.47E+00	2.33E+02 $\pm$ 5.77E-01	1.12E+00 $\pm$ 8.09E-03



(a) Inference Network SNR

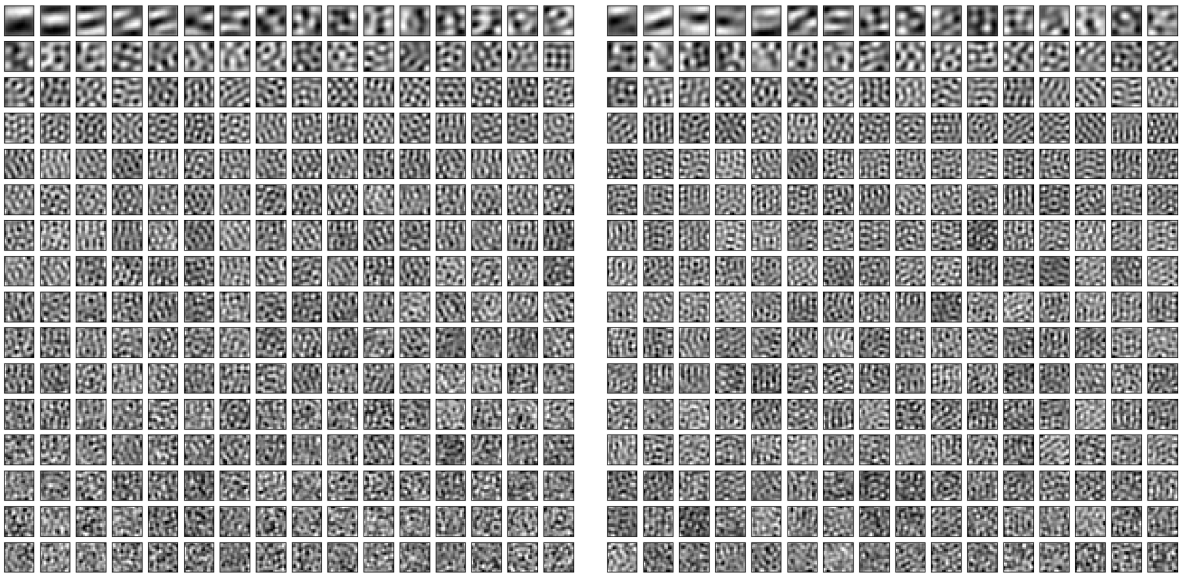
(b) Generator SNR

Figure 9. Comparison of SNR for (a) inference network and (b) linear generator for all methods using $J = 1$ and $J = 20$ with max ELBO sampling.



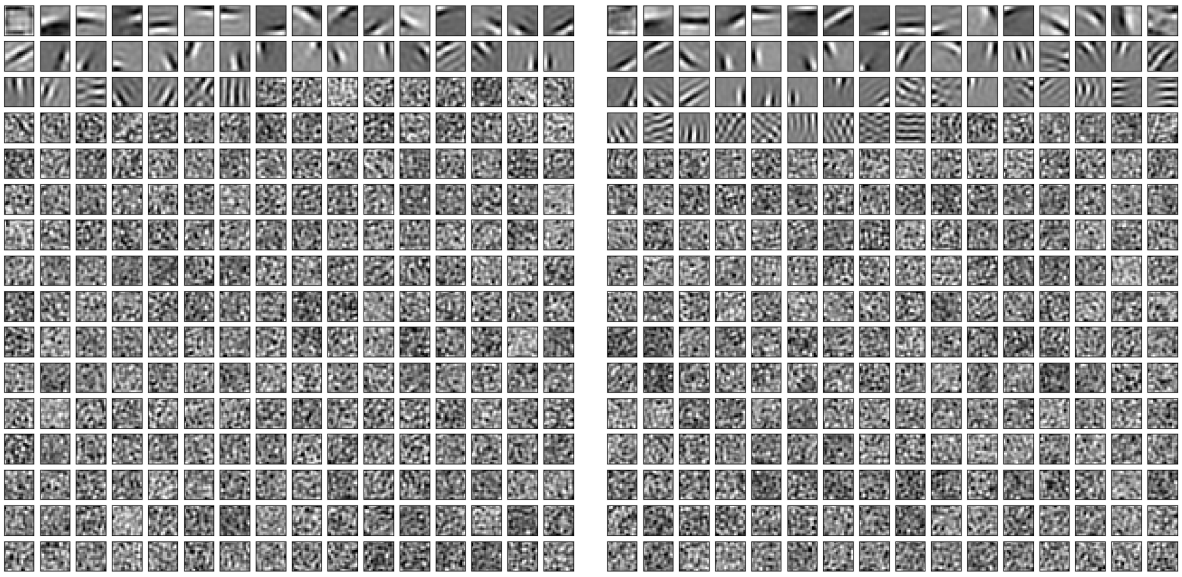
(a)

Figure 10. Full dictionary sorted by magnitude in descending order for FISTA with $\lambda = 20$ and a Frobenius norm penalty of $1\mathrm{E}{-}03$.



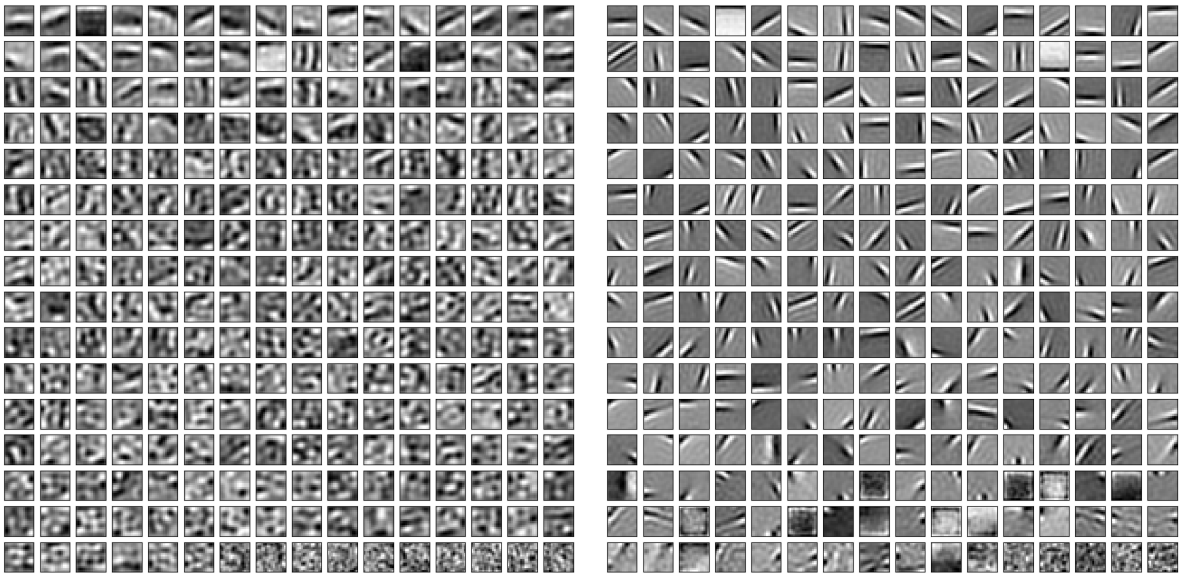
(a) $J = 1$ (b) $J = 20$

Figure 11. Full dictionary sorted by magnitude in descending order for **Gaussian** prior with $J = 1$ and $J = 20$ under max ELBO sampling.



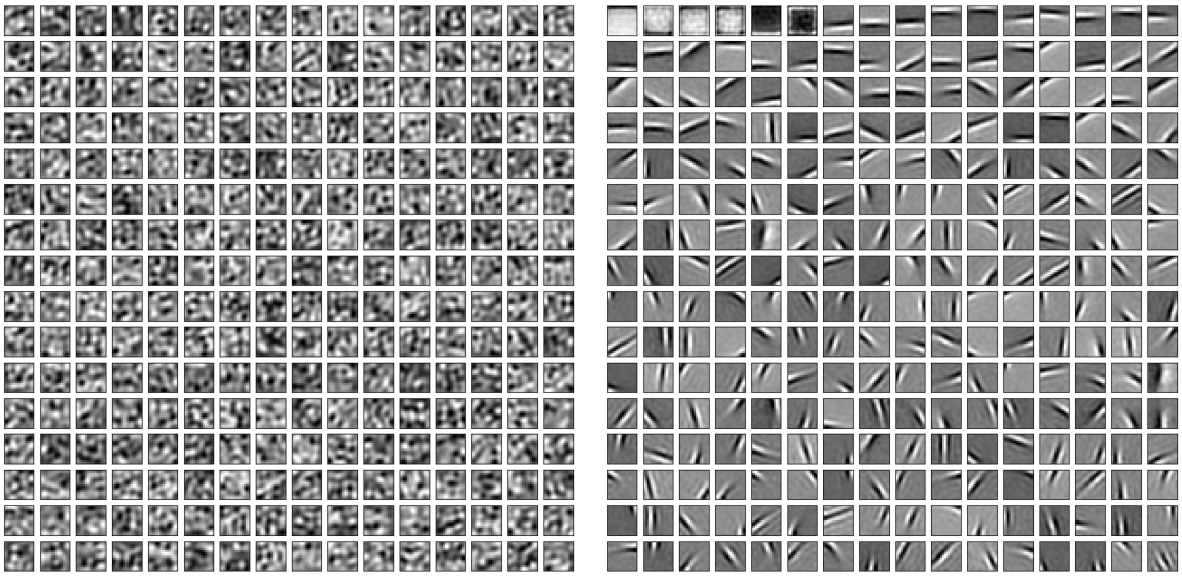
(a) $J = 1$ (b) $J = 20$

Figure 12. Full dictionary sorted by magnitude in descending order for **Laplacian** prior with $J = 1$ and $J = 20$ under max ELBO sampling.



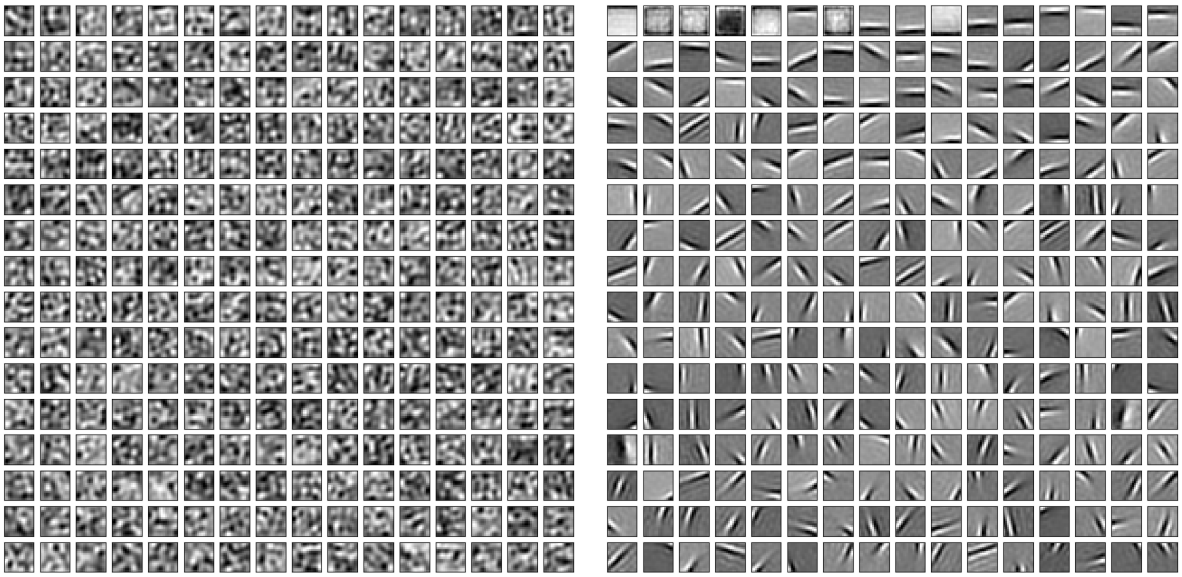
(a) $J = 1$ (b) $J = 20$

Figure 13. Full dictionary sorted by magnitude in descending order for **Spike-and-Slab** prior with $J = 1$ and $J = 20$ under max ELBO sampling.



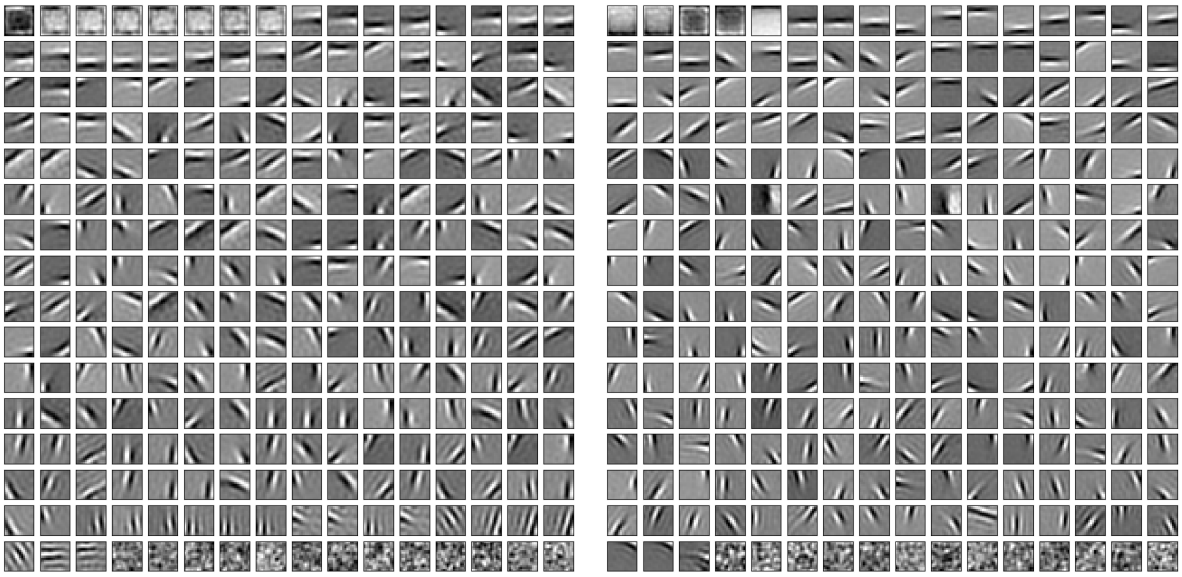
(a) $J = 1$ (b) $J = 20$

Figure 14. Full dictionary sorted by magnitude in descending order for **Thresholded Gaussian** prior with $J = 1$ and $J = 20$ under max ELBO sampling.



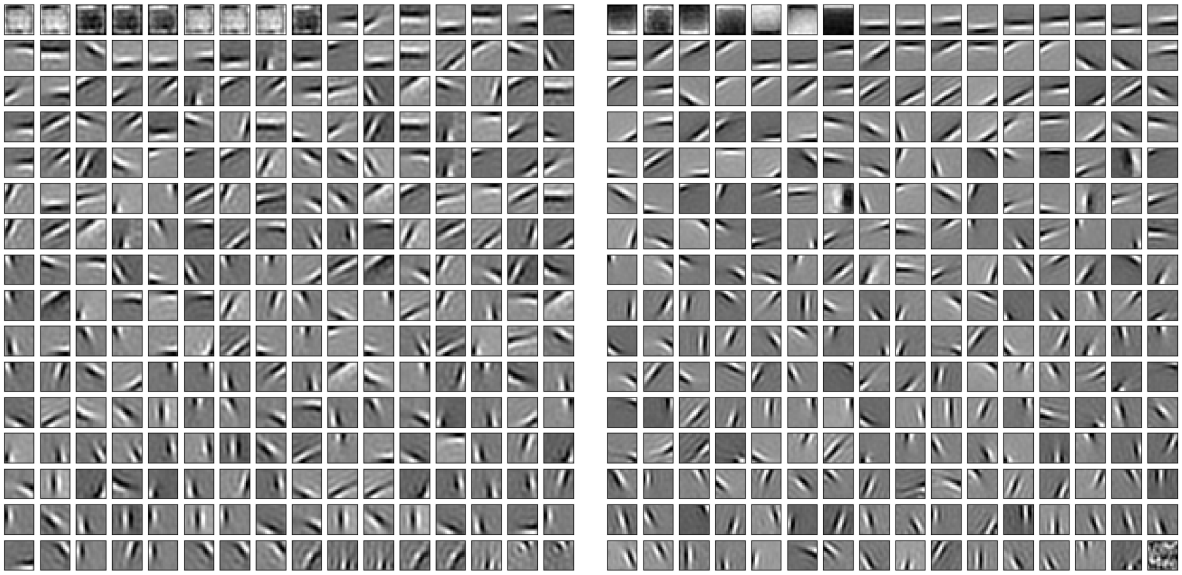
(a) $J = 1$ (b) $J = 20$

Figure 15. Full dictionary sorted by magnitude in descending order for **Thresholded Gaussian + Gamma threshold prior** prior with $J = 1$ and $J = 20$ under max ELBO sampling.



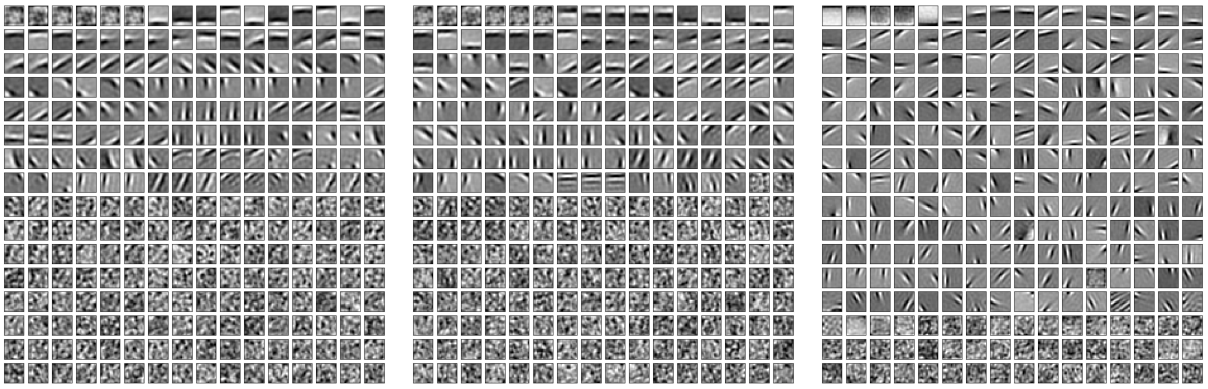
(a) $J = 1$ (b) $J = 20$

Figure 16. Full dictionary sorted by magnitude in descending order for **Thresholded Laplacian** prior with $J = 1$ and $J = 20$ under max ELBO sampling.



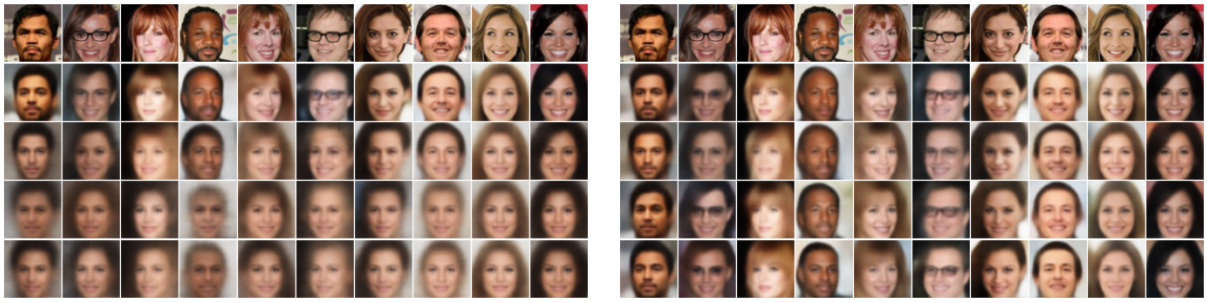
(a) $J = 1$ (b) $J = 20$

Figure 17. Full dictionary sorted by magnitude in descending order for **Thresholded Laplacian + Gamma threshold prior** prior with $J = 1$ and $J = 20$ under max ELBO sampling.



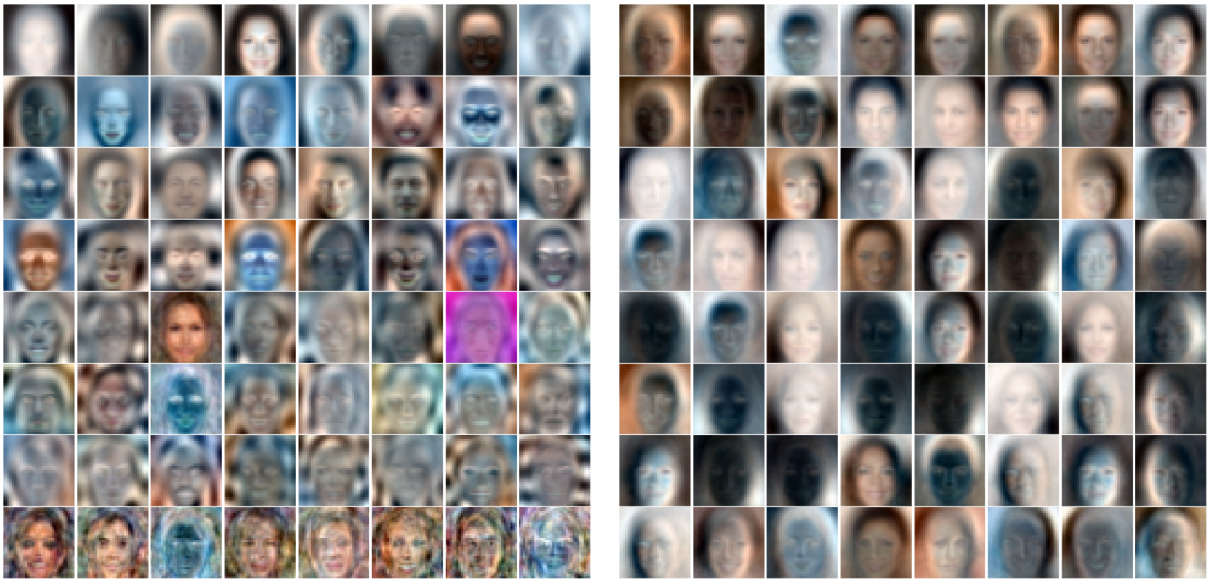
(a) Avg (b) IWAE (c) Max

Figure 18. Full dictionary sorted by magnitude in descending order for different sampling methods using $J = 100$ samples using the Thresholded Laplacian prior. Average sampling and IWAE sampling learn repetetive features in their dictionary since they do not encourage feature reuse during training. On the other hand, max ELBO sampling learns a larger dictionary of diverse features.



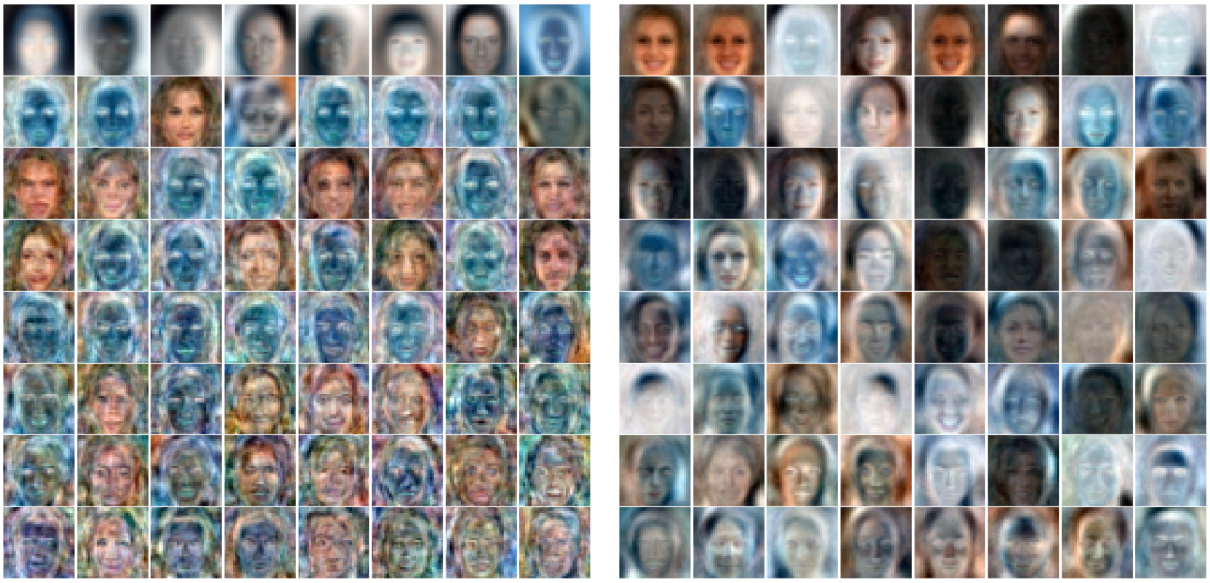
(a) 64 Dimensions (b) 4096 Dimensions

Figure 19. Image reconstruction on CelebA using DNN generator for (a) 64 dimensions and (b) 4096 dimensions. From top to bottom, the rows denote the ground truth images, a Gaussian prior, a Spike-and-Slab prior, a Thresholded Gaussian prior, and a Thresholded Gaussian+Gamma prior.



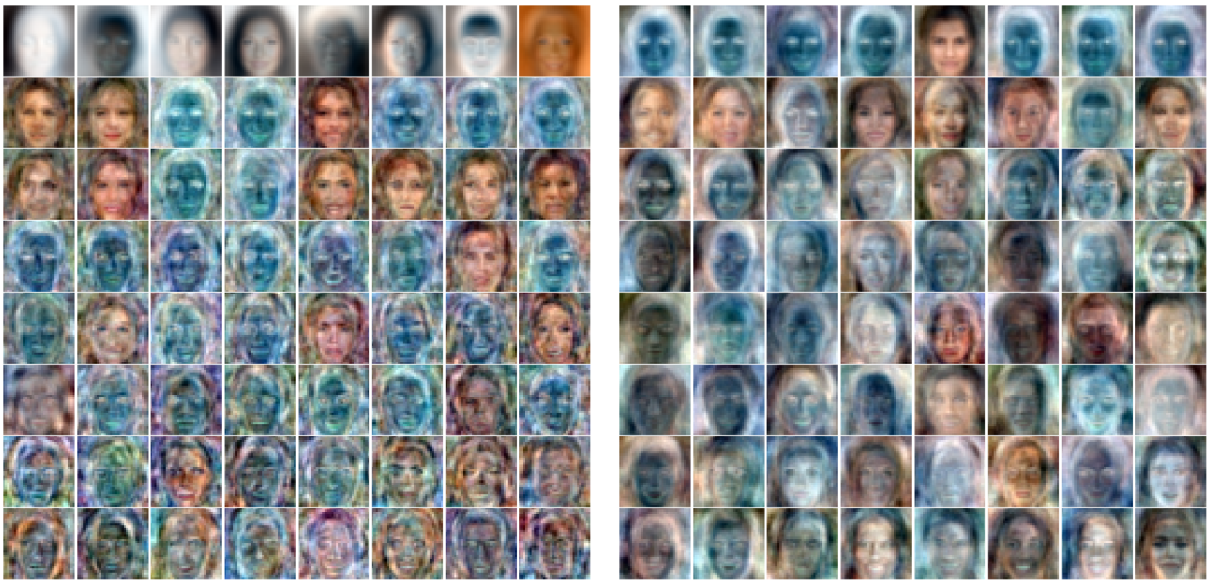
(a) Gaussian (b) Thresholded Gaussian+Gamma

Figure 20. **64 dimensional latent space**. Sampled entries from an estimated dictionary using an inference network trained with a DNN generator on CelebA. Entries are shown in descending order of Frobenius norm magnitude, with each row taking linearly spaced entries from the full dictionary. The sparse prior is set so to encourage 10% of latent features to be non-zero.



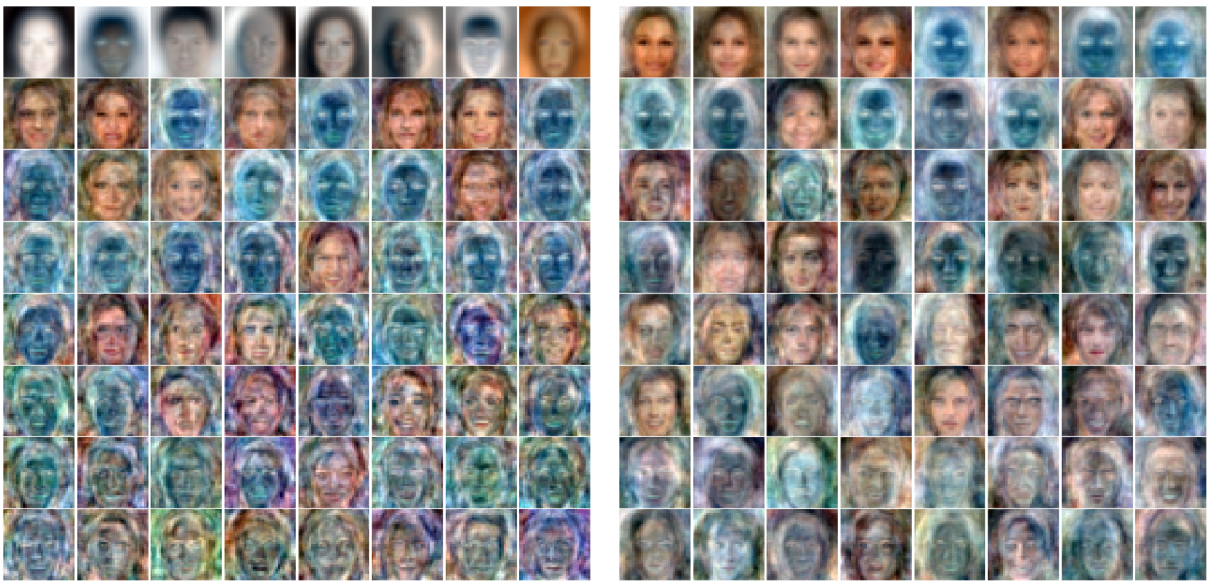
(a) Gaussian (b) Thresholded Gaussian+Gamma

Figure 21. **512 dimensional latent space**. Sampled entries from an estimated dictionary using an inference network trained with a DNN generator on CelebA. Entries are shown in descending order of Frobenius norm magnitude, with each row taking linearly spaced entries from the full dictionary. The sparse prior is set so to encourage 10% of latent features to be non-zero.



(a) Gaussian (b) Thresholded Gaussian+Gamma

Figure 22. **2048 dimensional latent space**. Sampled entries from an estimated dictionary using an inference network trained with a DNN generator on CelebA. Entries are shown in descending order of Frobenius norm magnitude, with each row taking linearly spaced entries from the full dictionary. The sparse prior is set so to encourage 10% of latent features to be non-zero.



(a) Gaussian (b) Thresholded Gaussian+Gamma

Figure 23. **4096 dimensional latent space**. Sampled entries from an estimated dictionary using an inference network trained with a DNN generator on CelebA. Entries are shown in descending order of Frobenius norm magnitude, with each row taking linearly spaced entries from the full dictionary. The sparse prior is set so to encourage 10% of latent features to be non-zero.