Identifiable Deep Generative Models via Sparse Decoding

Gemma E. Moran  
*Columbia University*  
gm2918@columbia.edu

Dhanya Sridhar  
*Mila - Quebec AI Institute and Université de Montréal*

Yixin Wang  
*University of Michigan*

David M. Blei  
*Columbia University*

Reviewed on OpenReview: [https://openreview.net/forum?id=vd0onGWZbE](https://openreview.net/forum?id=vd0onGWZbE)

Abstract

We develop the sparse VAE for unsupervised representation learning on high-dimensional data. The sparse VAE learns a set of latent factors (representations) which summarize the associations in the observed data features. The underlying model is sparse in that each observed feature (i.e. each dimension of the data) depends on a small subset of the latent factors. As examples, in ratings data each movie is only described by a few genres; in text data each word is only applicable to a few topics; in genomics, each gene is active in only a few biological processes. We prove such sparse deep generative models are identifiable: with infinite data, the true model parameters can be learned. (In contrast, most deep generative models are not identifiable.) We empirically study the sparse VAE with both simulated and real data. We find that it recovers meaningful latent factors and has smaller heldout reconstruction error than related methods.

1 Introduction

In many domains, high-dimensional data exhibits variability that can be summarized by low-dimensional latent representations, or factors. These factors can be useful in a variety of tasks including prediction, transfer learning, and domain adaptation (Bengio et al., 2013).

To learn such factors, many researchers fit deep generative models (DGMs) (Kingma and Welling, 2014; Rezende et al., 2014). A DGM models each data point with a latent low-dimensional representation (its factors), which is passed through a neural network to generate the observed features. Given a dataset, a DGM can be fit with a variational autoencoder (VAE), a method that provides both the fitted parameters to the neural network and the approximate posterior factors for each data point.

In this paper, we make two related contributions to the study of deep generative models. First, we develop the sparse DGM. This model is a DGM where each observed feature only depends on a subset of the factors – that is, the mapping from factors to features is sparse. This notion of sparsity often reflects the type of underlying patterns that we anticipate finding in real-world data. In genomics, each gene is associated with a few biological processes; in text, each term is applicable to a few underlying topics; in movie ratings, each movie is associated with a few genres.

In detail, the model implements sparsity through a per-feature masking variable. When generating a data point, this mask is applied to the latent representation before producing each feature. In practice, we learn the per-feature mask with the help of the Spike-and-Slab Lasso prior (Ročková and George, 2018),...
Figure 1: In a DGM, a feature $x_{ij}$ depends on all factors, $z_{ik}$. A sparse DGM is displayed where features $x_{i1}, x_{i2}$ depend only on $z_{i1}$; $x_{i3}$ depends on $z_{i1}$ and $z_{i2}$; and $x_{i4}, x_{i5}$ depend only on $z_{i2}$. The features are passed through the same neural network $f_\theta$.

Second, we study identifiability in sparse DGMs. A model is identifiable if each setting of its parameters produces a unique distribution of the observed data. With an unidentifiable DGM, even with infinite data from the model, we cannot distinguish the true factors. Identifiable factor models are important to tasks such as domain adaptation (Locatello et al., 2020), transfer learning (Dittadi et al., 2021), and fair classification (Locatello et al., 2019a). Most DGMs are not identifiable (Locatello et al., 2019b).

Specifically, we prove that a sparse DGM is identifiable if each latent factor has at least two “anchor features.” An anchor feature is a dimension of the data which depends on only one latent factor. For example, in text data, an anchor feature is a word that is applicable to only one underlying topic (Arora et al., 2013). Note that the anchor features do not need to be known in advance for our results to hold; we only need to assume they exist. Further, we do not need to assume that the true factors are drawn independently; in many applications, such an independence assumption is unrealistic (Träuble et al., 2021).

The sparse VAE and the anchor assumption are related in that the SSL prior will likely yield the kind of sparse mapping that satisfies the anchor assumption. What this means is that if the data comes from an identifiable DGM, i.e., one that satisfies the anchor assumption, then the SSL prior will encourage the sparse VAE to more quickly find a good solution. In this sense, this paper is in the spirit of research in Bayesian statistics, which establishes theoretical conditions on identifiability and then produces priors that favor identifiable distributions.

Empirically, we compare the sparse VAE to existing algorithms for fitting DGMs: the VAE (Kingma and Welling, 2014), $\beta$-VAE (Higgins et al., 2017), Variational Sparse Coding (VSC, Tonolini et al., 2020), and OI-VAE (Ainsworth et al., 2018). We find that (i) on synthetic data, the sparse VAE recovers ground truth factors, including when the true factors are correlated; (ii) on text and movie-ratings data, the sparse VAE achieves better heldout predictive performance; (iii) on semi-synthetic data, the sparse VAE has better heldout predictive performance on test data that is distributed differently to training data; (iv) the sparse VAE finds interpretable structure in text, movie-ratings and genomics data.

In summary, the contributions of our paper are as follows:

- We develop a sparse DGM which has Spike-and-Slab Lasso priors (Ročková and George, 2018) and develop an algorithm for fitting this model, the sparse VAE.
- We prove that sparse DGMs are identified under an anchor feature assumption.
- We study the sparse VAE empirically. It outperforms existing methods on synthetic, semi-synthetic, and real datasets of text, ratings, and genomics.

Related Work. The sparse DGM provides a contribution to sparse methods in linear and nonlinear representation learning. The sparse DGM has a sparsity-inducing prior on the factor-to-feature mapping.
(i.e., the decoder). Similar priors have also been applied in linear factor analysis to induce sparsity in the factor-to-feature mapping (i.e., the loadings matrix); see, for example Bernardo et al. (2003); Bhattacharya and Dunson (2011); Carvalho et al. (2008); Knowles and Ghahramani (2011); Ročková and George (2016).

Beyond linearity, Barello et al. (2018) considers a hybrid linear/nonlinear VAE which uses a sparse linear decoder and a neural network encoder. In nonlinear representation learning, Tonolini et al. (2020) imposes sparsity-inducing priors directly on the latent factors. Instead, the sparse DGM of this paper involves a sparsity-inducing prior on the factor-to-feature mapping. Rhodes and Lee (2021) indirectly induces sparsity in the factor-to-feature mapping of their DGM via $L_1$ regularization of the mapping’s Jacobian.

The sparse VAE closely relates to the OI-VAE (Ainsworth et al., 2018), a method for modeling grouped features where the factor-to-group mapping is sparse. However, there are several differences. First, the OI-VAE uses a Group Lasso prior while the sparse VAE has a Spike-and-Slab Lasso prior (SSL, Ročková and George, 2018). The SSL prior mitigates common issues with the Lasso prior, including under-regularization of small coefficients and over-regularization of large coefficients (Ghosh et al., 2016), and a sub-optimal posterior contraction rate (Castillo et al., 2015). Second, we study the identifiability of sparse DGMs, a property that Ainsworth et al. (2018) does not investigate.

This paper also contributes to the literature that uses the anchor feature assumption for identifiability. In linear models, the anchor feature assumption has led to identifiability results for topic models (Arora et al., 2013), non-negative matrix factorization (Donoho and Stodden, 2003), and linear factor analysis (Bing et al., 2020). The key idea is that the anchor assumption removes the rotational invariance of the latent factors. This idea also arises in identifiable factor analysis (Rohe and Zeng, 2020) and independent component analysis, where rotational invariance is removed by assuming that the components are non-Gaussian (Eriksson and Koivunen, 2004).

Finally, this paper is related to methods in identifiable nonlinear representation learning. To achieve identifiability, most methods require weak supervision: Khemakhem et al. (2020); Mita et al. (2021); Zhou and Wei (2020) require auxiliary data; Locatello et al. (2020) requires paired samples; Kügelgen et al. (2021) relies on data augmentation; and Hälvä et al. (2021); Ahuja et al. (2022); Lachapelle et al. (2022) leverage known temporal or spatial dependencies among the samples. In contrast, the sparse DGM does not need auxiliary information or weak supervision for identifiability; instead, the anchor feature assumption is sufficient. Relatedly, Horan et al. (2021) achieves identifiable representations under a “local isometry” assumption, meaning a small change in a factor corresponds to a small change in the features. In contrast to this paper, however, Horan et al. (2021) requires the factors to be independent; we do not need such an independence assumption.

## 2 The Sparse Variational Autoencoder

The observed data is a vector of $G$ features $x_i \in \mathbb{R}^G$ for $i \in \{1, \ldots, N\}$ data points. The goal is to estimate a low-dimensional set of factors $z_i \in \mathbb{R}^K$ where $K \ll G$.

In a standard DGM, the vector $z_i$ is a latent variable that is fed through a neural network to reconstruct the distribution of $x_i$. The sparse DGM introduces an additional parameter $w_{jk} \in \mathbb{R}^K$, $j \in \{1, \ldots, G\}$, a sparse per-feature vector that selects which of the $K$ latent factors are used to produce the $j$th feature of $x_i$. Further, the sparse DGM models the prior covariance between factors with the positive definite matrix $\Sigma_z \in \mathbb{R}^{K \times K}$. The sparse DGM is:

$$
\begin{align*}
    w_{jk} &\sim \text{Spike-and-Slab Lasso}(\lambda_0, \lambda_1, a, b), \quad k = 1, \ldots, K \\
    z_i &\sim \mathcal{N}_K(0, \Sigma_z), \quad i = 1, \ldots, N \\
    x_{ij} &\sim \mathcal{N}(f_\theta(w_j \odot z_i), \sigma_j^2), \quad j = 1, \ldots, G 
\end{align*}
$$

where $f_\theta : \mathbb{R}^K \rightarrow \mathbb{R}^G$ is a neural network with weights $\theta$, $\sigma_j^2$ is the per-feature noise variance and $\odot$ denotes element-wise multiplication. The Spike-and-Slab Lasso (Ročková and George, 2018) is a sparsity-inducing prior which we will describe below. (Any sparsity-inducing prior on $w_{jk}$ would be appropriate for a sparse DGM; we choose SSL because it enjoys theoretical guarantees and fast computation in more classical settings.)
Algorithm 1: The sparse VAE

\begin{algorithm}
\textbf{input}: data \(X\), hyperparameters \(\lambda_0, \lambda_1, a, b, \Sigma_z\)
\textbf{output}: factor distributions \(q_\phi(z|x)\), selector matrix \(W\), parameters \(\theta\)
\textbf{while} not converged \textbf{do}
\begin{algorithmic}
  \For{\(j = 1, \ldots, G\); \(k = 1, \ldots, K\)}
    \State Update \(E[\gamma_{jk}|w_{jk}, \eta_k] = \left[1 + \frac{1 - \eta_k}{\eta_k} \frac{\psi_0(w_{jk})}{\psi_1(w_{jk})}\right]^{-1}\);
  \EndFor
  \For{\(k = 1, \ldots, K\)}
    \State Update \(\eta_k = \frac{\sum_j^G E[\gamma_{jk}|w_{jk}, \eta_k] + a - 1}{a + b + G - 2}\);
  \EndFor
\end{algorithmic}
\State Update \(\theta, \phi, W\) with SGD according to Eq. 8;
\end{algorithm}

In the data generating distribution in Eq. 1, the parameter \(w_{jk}\) controls whether the distribution of \(x_{ij}\) depends on factor \(k\). If \(w_{jk} \neq 0\), then \(z_{ik}\) contributes to \(x_{ij}\), while if \(w_{jk} = 0\), \(z_{ik}\) cannot contribute to \(x_{ij}\). If \(w_j\) is sparse, then \(x_{ij}\) depends on only a small set of factors. Such a sparse factor-to-feature relationship is shown in Figure 1.

Note that \(x_{ij}\) depends on the same set of factors for every sample \(i\). This dependency only makes sense when each \(x_{ij}\) has a consistent meaning across samples. For example, in genomics data \(x_{ij}\) corresponds to the same gene for all \(i\) samples. (This dependency is not reasonable for image data, where pixel \(j\) has no consistent meaning across samples.)

**Spike-and-Slab Lasso.** The parameter \(w_{jk}\) has a SSL prior (Ročková and George, 2018), which is defined as a hierarchical model,

\begin{align}
\eta_k &\sim \text{Beta}(a, b), \\
\gamma_{jk} &\sim \text{Bernoulli}(\eta_k), \\
w_{jk} &\sim \gamma_{jk}\psi_1(w_{jk}) + (1 - \gamma_{jk})\psi_0(w_{jk}).
\end{align}

In this prior, \(\psi_s(w) = \frac{\lambda_s}{2} \exp(-\lambda_s|w|)\) is the Laplace density and \(\lambda_0 \gg \lambda_1\). The variable \(w_{jk}\) is drawn \textit{a priori} from either a Laplacian “spike” parameterized by \(\lambda_0\), and is consequentially negligible, or a Laplacian “slab” parameterized by \(\lambda_1\), and can be large.

Further, the variable \(\gamma_{jk}\) is a binary indicator variable that determines whether \(w_{jk}\) is negligible. The Beta-Bernoulli prior on \(\gamma_{jk}\) allows for uncertainty in determining which factors contribute to each feature.

Finally, the parameter \(\eta_k \in [0, 1]\) controls the proportion of features that depend on factor \(k\). By allowing \(\eta_k\) to vary, the sparse DGM allows each factor to contribute to different numbers of features. In movie-ratings data, for example, a factor corresponding to the action/adventure genre may be associated with more movies than a more esoteric genre.

Notice the prior on \(\eta_k\) helps to “zero out” extraneous factor dimensions and consequently estimate the number of factors \(K\). If the hyperparameters are set to \(a \propto 1/K\) and \(b = 1\), the Beta-Bernoulli prior corresponds to the finite Indian Buffet Process prior (Griffiths and Ghahramani, 2005).

2.1 Inference

In inference, we are given a dataset \(x_{1:N}\) and want to calculate the posterior \(p(\theta, W, \Gamma, \eta, z_{1:N}|x_{1:N})\). We will use approximate maximum \textit{a posteriori} (MAP) estimation for \(W, \theta\) and \(\eta\), and amortized variational inference for \(z_{1:N}\). The full procedure for approximating this posterior is called a sparse VAE.
The exact MAP objective is
\[
\sum_{i=1}^{N} \log p_{\theta}(x_i|W, \eta) = \sum_{i=1}^{N} \log \left[ \int p_{\theta}(x_i|z_i, W)p(z_i)dz_i \right] + \log \int p(W|\Gamma)p(\Gamma|\eta)p(\eta)d\Gamma,
\]
where \( \Gamma = \{\gamma_{j,k}\}_{j,k=1}^{G,K} \) denotes the binary indicators. We lower bound Eq. 5 with variational approximations of the posterior of \( z_{1:N} \) and the exact posterior of the latent \( \Gamma \), which comes from the SSL prior. We approximate the posterior of the factors \( p_{\theta}(z_i|x_i) \) by the variational family:
\[
q_{\phi}(z_i|x_i) = N_K(\mu_{\phi}(x_i), \sigma^2_{\phi}(x_i)),
\]
where \( \mu_{\phi}() \) and \( \sigma^2_{\phi}() \) are neural networks with weights \( \phi \), as in a standard VAE (Kingma and Welling, 2014; Rezende et al., 2014).

These approximations yield our final objective function:
\[
\mathcal{L}(\theta, \phi, W, \eta) = \sum_{i=1}^{N} \left\{ \mathbb{E}_{q_{\phi}(z_i|x_i)}[\log p_{\theta}(x_i|z_i, W)] - D_{KL}(q_{\phi}(z_i|x_i)||p(z_i)) \right\} \] 
\[+ \mathbb{E}_{\Gamma|W,\eta} \left[ \log[p(W|\Gamma)p(\Gamma|\eta)p(\eta)] \right].
\]

To optimize \( \mathcal{L}(\theta, \phi, W, \eta) \), we alternate between an expectation step and a maximization step. In the E-step, we calculate the complete conditional posterior of \( \Gamma \). In the M-step, we take gradient steps in model parameters and variational parameters with the help of reparameterization gradients (Kingma and Welling, 2014; Rezende et al., 2014). The sparse VAE procedure is in Algorithm 1.

The sparse VAE has a higher computational burden than a VAE. A VAE typically requires one forward pass through the neural network at each gradient step. In contrast, the sparse VAE requires \( G \) forward passes. This additional computation is because for each of the \( G \) features, the sparse VAE uses a different decoder input \( w_{ij} \odot z_i \), for \( j = 1, \ldots, G \). (The VAE and sparse VAE optimize the same number of neural network parameters, however).

### 2.2 Synthetic example

For intuition, consider a synthetic dataset. We generate \( N = 1000 \) samples from a factor model, each with \( G = 7 \) features and \( K = 2 \) latent factors. We take the true factors to be correlated and generate them as \( z_i \sim N(0, C) \) where the true factor covariance \( C \) has diagonal entries \( C_{kk} = 1 \) and off-diagonal entries \( C_{kk'} = 0.6 \) for \( k \neq k' \).
Given the factors, the data is generated as:

\[ x_i \sim N_G(f(z_i), \sigma^2 I_G) \]  \hspace{1cm} \text{with} \sigma^2 = 0.5. \tag{9} 

The factor-to-feature mapping \( f : \mathbb{R}^K \rightarrow \mathbb{R}^G \) is sparse:

\[ f(z_i) = (z_{i1}, 2z_{i1}, 3z_{i1}^2, 4z_{i2}, 5z_{i2}, 6 \sin(z_{i2}), 7z_{i1} \cdot z_{i2}). \]

This mapping corresponds to the \( W \) matrix in Figure 2c.

We compare the sparse VAE with a VAE. For both, we initialize with an overestimate of the true number of factors \((K_0 = 5)\). The sparse VAE finds factors that reflect the true generative model (Figure 2a). Moreover, the sparse VAE correctly finds the true number of factors by setting three columns of \( W \) to zero (Figure 2d). In contrast, the VAE recovers the second factor fairly well, but doesn’t recover the first factor (Figure 2b).

We analyze this synthetic data further in §4, as well as text, movie ratings and genomics data.

3 Identifiability of Sparse DGMs

For an identifiable model, given infinite data, it is possible to learn the true parameters. In both linear and nonlinear factor models, the factors are usually unidentifiable without additional constraints (Yalcin and Amemiya, 2001; Locatello et al., 2019b). This lack of identifiability means that the true factors \( z_i \) can have the same likelihood as another solution \( \tilde{z}_i \). Even with infinite data, we cannot distinguish between these representations based on the likelihood alone. Further inductive biases or model constraints are necessary to narrow down the set of solutions.

In this section, we demonstrate that sparsity is a useful inductive bias for model identifiability. Consider the sparse deep generative model:

\[ z_i \sim N_K(0, C), \quad i = 1, \ldots, N \]
\[ x_{ij} \sim N((f_\theta(w_j \odot z_i))_j, \sigma_j^2), \quad j = 1, \ldots, G, \] \hspace{1cm} \text{(10)}

where \( C \) is the true covariance matrix of the factors. Informally, if the masking variable \( W \) is sparse in a particular way, then we can identify the latent factors, up to coordinate-wise transformation. (Note: we only make an assumption about the sparse support of \( W \), not its distribution; the SSL prior in §2 is a tool to explore the space of sparse \( W \) in the process of fitting the model.)

Specifically, we prove that for any solutions \( z \) and \( \tilde{z} \) with equal likelihood, we have:

\[ z_{ik} = g_k(\tilde{z}_{ik}), \quad i = 1, \ldots, N, \] \hspace{1cm} \text{(11)}

for all factors \( k = 1, \ldots, K \) (up to permutation of \( k \)), where \( g_k : \mathbb{R} \rightarrow \mathbb{R} \) are invertible and differentiable functions.

This definition of identifiability is weaker than the canonical definition, for which we would need the two solutions \( z_i, \tilde{z}_i \) to be exactly equal. We use this weaker notion of identifiability because our goal is to isolate the dimensions of \( z_i \) which drive the observed response, and not necessarily find their exact value—i.e. we want to avoid mixing the dimensions of \( z_i \). For example, we want to be able to distinguish between solutions \( z \) and \( \tilde{z} = Pz \), for arbitrary rotation matrices \( P \).

Starting point. This paper builds on a body of work in identifiable linear unsupervised models (Arora et al., 2013; Bing et al., 2020; Donoho and Stodden, 2003). These works assume the existence of anchor features, features which depend on only one factor. The existence of anchor features helps with identifiability because they leave a detectable imprint in the covariance of the observed data.

We extend these results to the nonlinear setting. The key insight is that if two anchor features have the same nonlinear dependence on a factor, then we can apply results from the linear setting.

We first formally state the anchor feature assumption.
A1. (Anchor feature assumption) For every factor, \( z_k \), there are at least two features \( x_{ij}, x_{ij'} \) which depend only on that factor. Moreover, the two features have the same mapping \( f_j \) from the factors \( z_k \); that is, for all \( i = 1, \ldots, N \):

\[
\mathbb{E}[x_{ij}|z_i] = f_j(z_{ik}), \quad \mathbb{E}[x_{ij'}|z_i] = f_j(z_{ik}).
\]

(12)

We refer to such features as “anchor features.”

The connection between the anchor feature assumption and the masking variable \( W \) is the following: If feature \( j \) is an anchor for factor \( k \), then \( w_{jk} \neq 0 \) and \( w_{jk'} = 0 \) for all \( k \neq k' \).

Roadmap. We prove the identifiability of the sparse DGM in two parts. First, we prove that the latent factors are identifiable up to coordinate-wise transformation when the anchor features are known (Theorem 1). Second, we prove that the anchor features can be detected from the observed covariance matrix (Theorem 2). Together, Theorem 1 and Theorem 2 give the identifiability of the sparse DGM.

Known anchor features. The first result, Theorem 1, proves that the sparse DGM factors are identifiable if we are given the anchor features. If feature \( j \) is an anchor feature for factor \( k \), we set \( w_{jk} = 1 \) and \( w_{jk'} = 0 \) for all \( k' \neq k \).

Theorem 1. Suppose we have infinite data drawn from the model in Eq. 10 and A1 holds. Assume we are given the rows of \( W \) corresponding to the anchor features. Suppose we have two solutions with equal likelihood: \( \{\theta, \tilde{z}\} \) and \( \{\theta, \hat{z}\} \), with

\[
p_{\tilde{z}}(x|\tilde{z}, W) = p_{\hat{z}}(x|\hat{z}, W).
\]

Then, the factors \( \tilde{z} \) and \( \hat{z} \) are equal up to coordinate wise transformations. For \( g_k : \mathbb{R} \rightarrow \mathbb{R} \) and \( i = 1, \ldots, N \),

\[
(\tilde{z}_1, \ldots, \tilde{z}_i, \ldots) = (g_1(\hat{z}_1), \ldots, g_K(\hat{z}_K)).
\]

(14)

The proof of Theorem 1 is in Appendix A.1. The proof idea is that if the anchor features are known, then we are given noisy transforms of the coordinates of \( z \). Knowing these noisy transforms allows us to identify \( z \) up to coordinate-wise transform.

Unknown anchor features. The second result proves that the anchor features can be determined from an approximation of the covariance matrix of the data \( X \).

This result requires additional assumptions. The next assumption concerns the weights of the neural network \( f_\theta \). This assumption needs some additional notation. After applying the chain rule to the first layer, we rewrite the derivative of mapping \( f_\theta \) as follows:

\[
\frac{\partial (f_\theta(w_j \odot z_i))_j}{\partial z_{ik}} = \sum_{d=1}^{D_1} \frac{\partial m_\theta(u_{ijd})_j}{\partial u_{ijd}} H_{dk}^{(1)} w_{jk},
\]

(15)

where \( \{H_{dk}^{(1)}\}_{d,k=1}^{D_1,K} \) are the weights of the first neural network layer and \( D_1 \) is the dimension of the first layer, \( u_{ijd} = \sum_{k=1}^{K} H_{dk}^{(1)} w_{jk} z_{ik} \) is the first layer before the activation function, and \( m_\theta : \mathbb{R}^{D_1} \rightarrow \mathbb{R}^G \) is the rest of the neural network which takes as input the first layer \( u_{ij} = (u_{ijd})_{d=1}^{D_1} \). Let \( B_{ijk} = \sum_{d=1}^{D_1} \frac{\partial m_\theta(u_{ij})_j}{\partial u_{ijd}} H_{dk}^{(1)} \).

A2. Suppose \( j \) is an anchor feature for factor \( k \). For another feature \( l \), we assume that

\[
|w_{jk}| \sum_{i=1}^{N} |B_{ijk}|^2 \geq \sum_{i=1}^{N} \sum_{k'=1}^{K} |w_{lk'}| |B_{iljk'}| |B_{iljk'}|,
\]

with equality when \( l \) is also an anchor feature for \( k \) and inequality otherwise.

This assumption ensures that the covariance between two anchor features is larger than the covariance between an anchor feature and a non-anchor feature. This consequence then allows us to pinpoint the
anchor features in the covariance matrix of the data. In Appendix A.3, we show this assumption holds for a neural network with ReLU activations and independent weights.

Finally, we also require an assumption regarding the covariance of the factors: this is the same as assumption (iii) of Bing et al. (2020), which studies linear models.

A3. Denote the true covariance matrix of the factors as Cov($z_i$) = $C$. We assume min{$C_{kk}, C_{k'k'}$} > |$C_{kk'}$|, for all $k, k' = 1, \ldots, K$.

Assumption A3 requires that the variance of each factor be greater than its covariance with any other factor. To gain some intuition, consider a dataset of news articles where the latent factors are topics, sports and politics. Assumption A3 would be violated if every article about sports also discussed politics (and vice versa). In this case, the anchor word for sports will be equally as correlated with an anchor word for politics as it is with another anchor word for sports. That is, there is no discernible difference in the data that helps us distinguish the sports and politics factors.

Theorem 2. Assume the model Eq. 10 with A1–3 holds. Then, the set of anchor features can be determined uniquely from $\frac{1}{N} \sum_{i=1}^{N} \text{Cov}(x_i)$ as $N \to \infty$ (given additional regularity conditions detailed in Appendix A.2).

The proof of Theorem 2 is in Appendix A.2.

Proof idea: We adapt the proof technique of Bing et al. (2020). The proof idea of that paper is that for any two features $x_j$ and $x_{j'}$ that are anchors for the same factor, their covariance Cov($x_j, x_{j'}$) will be greater than their covariance with any other non-anchor feature (under some conditions on the loadings matrix, analogous to A2). Then, the anchor features can be pinpointed from the observed covariance matrix. In the nonlinear case, we can apply a similar strategy: even if the mapping is nonlinear, if the mapping is the same for the anchor features, we can pinpoint the two anchors in the covariance matrix.

Connection to the sparse VAE algorithm. Theorem 2 proves that the existence of anchor features implies a particular structure in the covariance of the data. Consequently, an appropriate estimation procedure can then pinpoint the anchor features. In the sparse VAE, we do not directly consider the covariance matrix as it would involve laborious hyperparameter tuning to determine which covariance entries are “close enough” to be anchor features. Instead, we estimate $W$ with an SSL prior, motivated by the sparsity in the decoder exhibited by anchor features. That is, it is the sparse decoding for the anchor features that is needed for Theorem 2, not the SSL prior; the SSL prior is a modeling choice that will likely yield the kind of sparse mapping that satisfies the anchor assumption. The SSL prior is not the only sparsity-inducing prior that we could have used; however, we found it to work well empirically. We compare the SSL prior to the horseshoe prior (Carvalho et al., 2009) in Appendix C.4.1.

Consistency. An important implication of identifiability is consistency: if we learn the optimal parameters of the sparse VAE, then we will recover the true factors in the limit of infinite data. Specifically, if (i) the variational family $q_\phi(z|x)$ contains the true posterior $p_\theta(z|x)$ and (ii) the ELBO is maximized with respect to the parameters $\theta, \phi$ and $W$, then in the limit of infinite data, the sparse VAE will learn the true factors up to coordinate-wise transform and permutation.

4 Experiments

We empirically study the sparse VAE using a mix of synthetic, semi-synthetic and real data\(^1\). We consider the following questions: 1) How well does the sparse VAE recover ground truth factors when (i) the factors are uncorrelated and (ii) the factors are correlated? 2) How does the heldout predictive performance of the sparse VAE compare to related methods? 3) How does the sparse VAE perform compared to related methods when the correlation between factors is different in the training and test data? 4) Does the sparse VAE find meaningful structure in data?

We compare the sparse VAE to non-negative matrix factorization (NMF) and algorithms for DGMs: the VAE (Kingma and Welling, 2014); $\beta$-VAE (Higgins et al., 2017); VSC (Tonolini et al., 2020); and OI-VAE

---

\(^1\)The sparse VAE implementation may be found at https://github.com/gemoran/sparse-vae-code.
Datasets. We consider the following datasets; further details about the data are given in Appendix C.1.

* Synthetic data. We consider again the synthetic dataset generated as Eq. 9. We vary correlation between the true factors from $\rho = 0, 0.2, 0.4, 0.6, 0.8$.

* PeerRead (Kang et al., 2018). Dataset of word counts for paper abstracts ($N \approx 10,000, G = 500$).

* MovieLens (Harper and Konstan, 2015). Dataset of binary user-movie ratings ($N = 100,000, G = 300$).

* Semi-synthetic PeerRead. A semi-synthetic version of the PeerRead dataset in which the correlations between data-generating factors differ across the test and training data, inducing different training and test distributions.

* Zeisel (Zeisel et al., 2015). Dataset of RNA molecule counts in mouse cortex cells ($N = 3005, G = 558$).

Implementation details. Empirically, we found that setting the prior covariance of the factors to $\Sigma_z = I_K$ worked well, even when the true generative factors were correlated (Figure 3a). Further implementation details for all methods are in Appendix C.

Recovering the ground truth factors. How well does the sparse VAE recover the factors when the ground truth factors are known? We assess this question with simulated data. We use the DCI disentanglement score to measure the fidelity between estimated factors and true factors (Eastwood and Williams, 2018). The DCI score is an average measure of how relevant each estimated factor is for the true factors. This score also penalizes estimated factors that are equally informative for multiple true factors.

We create synthetic datasets with the model given in Eq. 9 and evaluate the DCI metric between estimated and true factors as the true factors are increasingly correlated. As the correlation increases, we expect a standard VAE to conflate the factors while the sparse VAE recovers the two true underlying factors.

We see this phenomenon in Figure 3b; the sparse VAE has higher DCI scores than the VAE in all settings. The sparse VAE is robust to factor correlations up to $\rho = 0.4$, with decreasing performance as $\rho$ is further increased. Here, VSC performs worse than both the sparse VAE and VAE (the MSE scores for VSC in were too large for visualization in Figure 3a). The poor performance of VSC is likely due to the true generative factors in Eq. 9 not being sparse; only the mapping between the factors and features is sparse. Note that we do not compare with NMF in this example; as NMF is a linear method, it cannot reconstruct the nonlinear data generation process using only two latent dimensions.

Heldout reconstruction. We compare the negative log-likelihood on heldout data achieved by the sparse VAE and related methods. All results are averaged over five splits of the data, with standard deviation in
Table 1: On movie-ratings and text data, the sparse VAE achieves the lowest heldout negative log-likelihood (NLL). For the $\beta$-VAE, we show the result with best performing $\beta$.

(a) MovieLens

<table>
<thead>
<tr>
<th>Method</th>
<th>NLL</th>
<th>Recall@5</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse VAE</td>
<td>170.9 (2.1)</td>
<td>0.98 (0.002)</td>
<td>0.98 (0.003)</td>
</tr>
<tr>
<td>VAE</td>
<td>175.9 (2.4)</td>
<td>0.97 (0.001)</td>
<td>0.96 (0.001)</td>
</tr>
<tr>
<td>$\beta$-VAE ($\beta = 2$)</td>
<td>178.2 (2.4)</td>
<td>0.95 (0.002)</td>
<td>0.93 (0.002)</td>
</tr>
<tr>
<td>OI-VAE</td>
<td>212.1 (2.6)</td>
<td>0.51 (0.004)</td>
<td>0.50 (0.004)</td>
</tr>
<tr>
<td>VSC</td>
<td>192.2 (2.3)</td>
<td>0.79 (0.008)</td>
<td>0.77 (0.009)</td>
</tr>
<tr>
<td>NMF</td>
<td>198.6 (2.5)</td>
<td>0.90 (0.004)</td>
<td>0.85 (0.004)</td>
</tr>
</tbody>
</table>

(b) PeerRead

<table>
<thead>
<tr>
<th>Method</th>
<th>NLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse VAE</td>
<td>245.0 (2.0)</td>
</tr>
<tr>
<td>VAE</td>
<td>252.6 (1.4)</td>
</tr>
<tr>
<td>$\beta$-VAE ($\beta = 2$)</td>
<td>254.5 (3.0)</td>
</tr>
<tr>
<td>OI-VAE</td>
<td>260.6 (1.2)</td>
</tr>
<tr>
<td>VSC</td>
<td>252.9 (2.0)</td>
</tr>
<tr>
<td>NMF</td>
<td>267.2 (1.0)</td>
</tr>
</tbody>
</table>

Table 2: On the semi-synthetic PeerRead data, the sparse VAE achieves the lowest heldout negative log-likelihood. We create three settings where the difference between training and test data distributions range from high (hardest) to low (easiest).

<table>
<thead>
<tr>
<th>Difference between train and test</th>
<th>Negative log-likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>High</td>
</tr>
<tr>
<td>Sparse VAE</td>
<td>52.4 (0.4)</td>
</tr>
<tr>
<td>VAE</td>
<td>54.6 (0.5)</td>
</tr>
<tr>
<td>$\beta$-VAE ($\beta = 2$)</td>
<td>54.7 (0.3)</td>
</tr>
<tr>
<td>OI-VAE</td>
<td>65.7 (0.3)</td>
</tr>
<tr>
<td>VSC</td>
<td>58.7 (0.6)</td>
</tr>
<tr>
<td>NMF</td>
<td>60.1 (0.02)</td>
</tr>
</tbody>
</table>

Parentheses. Figure 3a shows that the sparse VAE performs better than the compared methods on synthetic data. On MovieLens and PeerRead datasets, Tables 1a and 1b show that the SparseVAE achieves the lowest heldout negative log-likelihood among the compared methods. For the MovieLens dataset, Table 1a additionally shows that the sparse VAE has the highest heldout Recall@5 and normalized discounted cumulative gain (NDCG@10), which compare the predicted rank of heldout items to their true rank (Liang et al., 2018).

**Different training and test distributions.** How does the sparse VAE perform when the factors that generate data are correlated differently across training and test distributions? This particular type of shift in distribution affects many real world settings. For example, we may estimate document representations from scientific papers where articles about machine learning also often discuss genomics, and want to use the representations to analyze new collections of papers where articles about machine learning rarely involve genomics. We hypothesize that because the sparse VAE associates each latent factor with only a few features (e.g., words) even when the factors are correlated, it will reconstruct differently distributed test data better than the related methods.

We assess this question using the semi-synthetic PeerRead dataset, where the train and test data were generated by factors with different correlations. Table 2 summarizes the results from three settings where the difference between training and test data distributions range from high (hardest) to low (easiest). We report the average negative log-likelihood across five semi-simulated datasets. The sparse VAE performs the best, highlighting its ability to estimate models which generalize better to test data where the factors have a different distribution.

**Interpretability.** We now examine whether the sparse VAE finds interpretable structure in the data. For each factor in the MovieLens dataset, we consider the four movies with the largest selector variable $\hat{w}_{jk}$ values (Table 3a). The sparse VAE finds clear patterns: for example, the top movies in first factor are all science fiction movies; the second factor contains animated children’s movies; the third factor contains three Star Wars movies and an Indiana Jones movie, all blockbuster science fiction movies from the 1980s.
### Table 3: On movie-ratings and text data, the sparse VAE finds meaningful topics via the matrix $W$.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Movies</th>
<th>(a) MovieLens</th>
<th></th>
<th>Topic</th>
<th>Words</th>
<th>(b) PeerRead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>The Fifth Element; Alien; Gattaca; Aliens</td>
<td></td>
<td></td>
<td>A</td>
<td>task; policy; planning; heuristic; decision</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A Bug’s Life; Monsters, Inc.; Toy Story 3 &amp; 2</td>
<td></td>
<td></td>
<td>B</td>
<td>information; feature; complex; sparse; probability</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Star Wars V, IV &amp; IV; Indiana Jones 1</td>
<td></td>
<td></td>
<td>C</td>
<td>given; network; method; bayesian; neural</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4: On the Zeisel data, the sparse VAE has the best heldout cell type prediction accuracy, followed closely by a VAE.

<table>
<thead>
<tr>
<th>Method</th>
<th>Sparse VAE</th>
<th>VAE ($\beta = 1$)</th>
<th>NMF</th>
<th>VSC</th>
<th>OI-VAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.95 (0.003)</td>
<td>0.94 (0.016)</td>
<td>0.91 (0.007)</td>
<td>0.89 (0.062)</td>
<td>0.80 (0.019)</td>
</tr>
</tbody>
</table>

We perform the same analysis for the PeerRead dataset (Table 3b). The sparse VAE finds some themes of computer science papers: the first factor contains words related to reinforcement learning; the second factor contains words about information theory; the third factor involves words about neural networks. We note that NMF also finds interpretable topics in both MovieLens and PeerRead (Appendix C.5.1); however, NMF has worse heldout predictive performance than the sparse VAE (Tables 1a and 1b). The sparse VAE retains the interpretability of NMF while incorporating flexible function estimation.

Finally, we consider the single-cell genomics dataset of Zeisel et al. (2015). We examine how well the factors learned by each method predict the cell type in a multinomial regression (note the cell type is not used when learning the factors). The sparse VAE is the best performing method (Table 4). Although the VAE is competitive in this setting, we see that the sparse VAE does better than methods such as OI-VAE and VSC that were designed to produce interpretable results.

### 5 Discussion

We developed the sparse DGM, a model with SSL priors that induce sparsity in the mapping between factors and features. Under an anchor feature assumption, we proved that the sparse DGM model has identifiable factors. To fit the sparse DGM, we develop the sparse VAE algorithm. On real and synthetic data, we show the sparse VAE performs well. (i) It has good heldout predictive performance. (ii) It generalizes to out of distribution data. (iii) It is interpretable.

There are a few limitations of this work. First, the sparse DGM is designed for tabular data, where each feature $j$ has a consistent meaning across samples. Image data does not have this property as a specific pixel has no consistent meaning across samples. Second, the sparse VAE algorithm is more computationally intensive than a standard VAE. This expense is because at each gradient step, it requires a forward pass through the network for each of the $G$ features. Finally, we did not provide theoretical estimation guarantees for the sparse VAE algorithm. Similarly to other algorithms for fitting DGMs, estimation guarantees for the sparse VAE are difficult to obtain due to the nonconvexity of the objective function. The majority of works which consider identifiability in DGMs also do not provide estimation guarantees for their algorithms, including Locatello et al. (2020); Khemakhem et al. (2020). We leave such theoretical study of the sparse VAE algorithm to future work.

### Acknowledgments

Funding to support this research was provided for by the Eric and Wendy Schmidt Center, the Canada-CIFAR AI Chairs program, National Science Foundation Grant NSF-CHE-2231174, NSF IIS 2127869, ONR N00014-17-1-2131, ONR N00014-15-1-2209, the Simons Foundation, the Sloan Foundation, and Open Philanthropy.
References


A Proofs

A.1 Proof of Theorem 1

We consider the case of known anchor features, where we have at least two anchor features per factor. We consider the case where the likelihood is Gaussian. We have two solutions, $(\hat{\theta}, \hat{z}, \hat{\sigma}^2)$ and $(\hat{\theta}, \tilde{z}, \tilde{\sigma}^2)$, with equal likelihood. We show that $\tilde{z}$ must be a coordinate-wise transform of $\hat{z}$.

We have

$$\sum_{i=1}^{N} \sum_{j=1}^{G} \frac{1}{\sigma_j^2} [x_{ij} - f_{\theta_j}(\hat{w}_{j} \odot \hat{z}_i)]^2 = \sum_{i=1}^{N} \sum_{j=1}^{G} \frac{1}{\sigma_j^2} [x_{ij} - f_{\theta_j}(\tilde{w}_{j} \odot \tilde{z}_i)]^2,$$

(17)

where $f_{\theta_j}(\cdot)$ denotes $f_{\theta}(\cdot)_j$, the $j$th output of $f_{\theta}$.

We prove that, given the anchor feature for factor $k$, $z_{ik}$ is identifiable. As all factors are assumed to have anchor features, this holds for all $k \in \{1, \ldots, K\}$. Suppose $j$ is an anchor feature for $k$. That is, $w_{j1} = 0$ for all $l \neq k$. Then, $\hat{w}_{j} \odot \hat{z}_i = (0, \ldots, 0, \tilde{w}_{jk}\tilde{z}_{ik}, 0, \ldots, 0)$. With slight abuse of notation, we then write $f_{\theta_j}(\hat{w}_{j} \odot \hat{z}_i) = f_{\theta_j}(\tilde{w}_{jk}\tilde{z}_{ik})$.

Now, for Eq. 17 to hold for any value of $\{x_{ij}\}_{i,j=1}^{N,G}$ we must have, for all $j = 1, \ldots, G$:

$$\sum_{i=1}^{N} \frac{1}{\sigma_j^2} [x_{ij} - f_{\theta_j}(\hat{w}_{jk}\tilde{z}_{ik})]^2 = \sum_{i=1}^{N} \frac{1}{\sigma_j^2} [x_{ij} - f_{\theta_j}(\tilde{w}_{jk}\tilde{z}_{ik})]^2$$

(18)

$$\left(\frac{1}{\sigma_j^2} - \frac{1}{\sigma_j^2}\right) \sum_{i=1}^{N} x_{ij}^2 - 2 \sum_{i=1}^{N} x_{ij} \left(\frac{f_{\theta_j}(\hat{w}_{jk}\tilde{z}_{ik})}{\sigma_j^2} - \frac{f_{\theta_j}(\tilde{w}_{jk}\tilde{z}_{ik})}{\sigma_j^2}\right) + \sum_{i=1}^{N} \left(\frac{f_{\theta_j}(\hat{w}_{jk}\tilde{z}_{ik})}{\sigma_j^2} - \frac{f_{\theta_j}(\tilde{w}_{jk}\tilde{z}_{ik})}{\sigma_j^2}\right) = 0.$$  

(19)

For Eq. 19 to hold for all $x_{ij}$, we must have the coefficients equal to zero:

$$\frac{1}{\sigma_j^2} - \frac{1}{\sigma_j^2} = 0 \implies \sigma_j^2 = \tilde{\sigma}_j^2$$  

(20)

and

$$\frac{f_{\theta_j}(\hat{w}_{jk}\tilde{z}_{ik})}{\sigma_j^2} - \frac{f_{\theta_j}(\tilde{w}_{jk}\tilde{z}_{ik})}{\sigma_j^2} = 0 \implies f_{\theta_j}(\hat{w}_{jk}\tilde{z}_{ik}) = f_{\theta_j}(\tilde{w}_{jk}\tilde{z}_{ik}).$$  

(21)

If $f_{\theta_j} : \mathbb{R} \to \mathbb{R}$ is invertible, then we have

$$\tilde{z}_{ik} = \hat{w}_{jk}^{-1} f_{\theta_j}^{-1}(\hat{w}_{jk}\tilde{z}_{ik}).$$  

(22)

That is, $z_{ik}$ is identifiable up to coordinate-wise transformation.

If $f_{\theta_j}$ is not invertible, then there exists bijective functions $g : \mathbb{R} \to \mathbb{R}$, $h : \mathbb{R} \to \mathbb{R}$ such that $\hat{w}_{jk}\tilde{z}_{ik} = g(\tilde{w}_{jk}\tilde{z}_{ik})$ and the equality is satisfied

$$f_{\theta_j}(\hat{w}_{jk}\tilde{z}_{ik}) = f_{\theta_j}(h(g(\tilde{w}_{jk}\tilde{z}_{ik}))) \quad \text{(as $f_{\theta_j}$ is not invertible, $\hat{w}_{jk}\tilde{z}_{ik} \neq h(g(\tilde{w}_{jk}\tilde{z}_{ik}))$)}$$  

(23)

$$= f_{\theta_j}(\tilde{w}_{jk}\tilde{z}_{ik}).$$  

(24)

As we have $\tilde{z}_{ik} = \hat{w}_{jk}^{-1} g(\tilde{w}_{jk}\tilde{z}_{ik})$, $z_{ik}$ is identifiable up to coordinate-wise transformation.
A.2 Proof of Theorem 2

To prove the result, we:

1. Approximate the covariance \( \frac{1}{N} \sum_{i=1}^{N} \text{Cov}(x_i) \) using a first order Taylor approximation.

2. Show that an anchor feature for factor \( k \) always has larger covariance with another anchor feature for \( k \) than other non-anchor features. Specifically: for any \( k \in \{1, \ldots, K\} \) and \( j = \text{anchor for } k \), we have
   
   \[ \sum_{i=1}^{N} \text{Cov}(x_{ij}, x_{il}) = C_{kk} w_{jk}^2 \sum_{i=1}^{N} B_{jk}^2 \text{ for all } l = \text{anchor for } k, \]
   
   \[ \sum_{i=1}^{N} \text{Cov}(x_{ij}, x_{il}) < C_{kk} w_{jk}^2 \sum_{i=1}^{N} B_{jk}^2 \text{ for all } l \neq \text{anchor for } k. \]

3. Apply Theorem 1 of Bing et al. (2020) to prove that the anchor features can be identified from the covariance of the data.

**Regularity condition.** Let \( \mu(z_i) = \mathbb{E}[x_i|z_i] \). We assume: for all \( z_i \) in a neighborhood of \( \hat{z}_i = \mathbb{E}[z_i] \),

\[
\frac{1}{N} \sum_{i=1}^{N} \mu(z_i) = \frac{1}{N} \sum_{i=1}^{N} \hat{z}_i + \frac{\partial \mu(z_i)}{\partial z_i} \bigg|_{z_i = \hat{z}_i} (z_i - \hat{z}_i) + o_P(1)
\]

(25)

where \( o_P(1) \) denotes a random variable converging to 0 in probability.

**Step 1.** Here is the marginal covariance of the Sparse VAE:

\[
\text{Var}(x_i) = \mathbb{E} \left[ \text{Var}(x_i|z_i) \right] + \text{Var} \left( \mathbb{E} \left[ x_i|z_i \right] \right) = \Sigma + \text{Var}(\mu(z_i))
\]

(26)

(27)

Take the first order Taylor expansion \( \mu(z_i) \approx \hat{z}_i + \frac{\partial \mu(z_i)}{\partial z_i} \bigg|_{z_i = \hat{z}_i} (z_i - \hat{z}_i) \).

Then

\[
\text{Var}(\mu(z_i)) \approx \frac{\partial \mu(z_i)}{\partial z_i} \bigg|_{z_i = \hat{z}_i} \text{Var}(z_i) \left( \frac{\partial \mu(z_i)}{\partial z_i} \bigg|_{z_i = \hat{z}_i} \right)^T.
\]

(28)

Now, \( \mu(z_i) = \{f_{\theta_j}(w_j \odot z_i)\}_{j=1}^{G} \). Then, for any two features, \( j \) and \( l \), we have

\[
\text{Cov}(x_{ij}, x_{il}) \approx \sum_{k=1}^{K} \sum_{k'=1}^{K} \frac{\partial f_{\theta_j}(w_j \odot z_i)}{\partial z_{ik}} \bigg|_{z_i = \hat{z}_i} \frac{\partial f_{\theta_l}(w_l \odot z_i)}{\partial z_{ik'}} \bigg|_{z_i = \hat{z}_i} C_{kk'}
\]

(29)

where \( \text{Var}(z_i) = C \).

**Step 2.** We re-write \( f_{\theta_j}(\cdot) \) to expose the first layer of the neural network (before the activation function is applied):

\[
f_{\theta_j}(w_j \odot z_i) = m_{\theta_j}(u_{ij})
\]

(30)

where \( u_{ij} = \{u_{ijd}\}_{d=1}^{D_1} \) with

\[
u_{ijd} = \sum_{k=1}^{K} \tilde{H}^{(1)}_{dk} w_{jk} z_{ik},
\]

(31)

where \( \{\tilde{H}^{(1)}_{dk}\}_{d,k=1}^{D_1,K} \) are the weights of the first neural network layer with dimension \( D_1 \).
Then,
\[
\text{Cov}(x_{ij}, x_{il}) = \sum_{k=1}^{K} \sum_{k'=1}^{K} w_{jk} w_{lk'} D_{ik} H_{dk}^{(1)} \frac{\partial m_{\theta}(u_{ij})}{\partial u_{ijd}} \left( \sum_{d=1}^{D_1} H_{dk'}^{(1)} \frac{\partial m_{\theta}(u_{il})}{\partial u_{ild}} \right) C_{kk'}
\]
\[
= \sum_{k=1}^{K} \sum_{k'=1}^{K} w_{jk} w_{lk'} B_{ijk} B_{ilk'} C_{kk'},
\]
where \( B_{ijk} = \sum_{d=1}^{D_1} H_{dk}^{(1)} \frac{\partial m_{\theta}(u_{ij})}{\partial u_{ijd}} \).

Suppose \( j \) is an anchor feature for \( k \). Then the absolute covariance of feature \( j \) and any other feature \( l \) is:
\[
|\text{Cov}(x_{ij}, x_{il})| = |w_{jk} B_{ijk} \sum_{k'=1}^{K} w_{lk'} B_{ilk'} C_{kk'}| 
\leq |C_{kk} w_{jk} B_{ijk} \sum_{k'=1}^{K} w_{lk'} B_{ilk'}| \quad \text{by A3.}
\]
\[
\leq C_{kk} w_{jk}^2 B_{ijk}^2 \quad \text{by A2.}
\]

**Step 3.** In Step 2, we proved the equivalent of Lemma 2 of Bing et al. (2020), adapted for the Sparse VAE. This allows us to apply Theorem 1 of Bing et al. (2020), which proves that the anchor features can be determined uniquely from the approximate covariance matrix, \( \frac{1}{N} \sum_{i=1}^{N} \text{Cov}(x_{i}), \) as \( N \to \infty. \)

### A.3 Discussion of identifiability assumptions

In this section, we examine the suitability of assumption A2. We do so by showing A2 holds for a three-layer neural network with ReLU activation functions, independently distributed Gaussian weights and no bias terms. Specifically:
\[
E[x_{ij}|w_j, z_i] = H_{j}^{(3)} \text{ReLU}(H_{j}^{(2)} \text{ReLU}(H_{j}^{(1)}(w_j \odot z_i)))
\]
where \( H^{(1)} \in \mathbb{R}^{D \times K} \) are the weights for layer 1 with \( D \) hidden units, \( H^{(2)} \in \mathbb{R}^{D \times D} \) are the weights for layer 2, and \( H_{j}^{(3)} \) denotes the \( j \)th row of the weights for layer 3, \( H^{(3)} \in \mathbb{R}^{G \times D}. \)

We have
\[
\frac{\partial \mu(z_i)}{\partial z_{ik}} = \sum_{d_1=1}^{D} \sum_{d_2=1}^{D} H_{j,d_2}^{(3)} H_{d_2,d_1}^{(2)} H_{d_1,k}^{(1)} \left[ H_{d_2}^{(2)} \text{ReLU}(H_{d_2}^{(1)}(w_j \odot z_i)) > 0 \right] \left[ \sum_{k=1}^{K} H_{d_1,k}^{(1)} w_{jk} z_{ik} > 0 \right].
\]
where \( \mathbb{I}(\cdot) \) is the indicator function.

Assume \( \{w_{jk}\}_{j,k} \sim G_{K} \) are independent and symmetric around zero. Assume all weights are independent and distributed as: \( H_{d_1,d_2}^{(m \ 0 \ i.i.d. N(0, \tau)} \) for all layers \( m = 1, 2, 3. \)

Taking the first order Taylor approximation, for any two features, \( j \) and \( l \), we have
\[
\text{Cov}(x_{ij}, x_{il}) \approx \sum_{k=1}^{K} \sum_{k'=1}^{K} C_{kk'} \sum_{d_1=1}^{D} \sum_{d_2=1}^{D} \sum_{d_1'=1}^{D} \sum_{d_2'=1}^{D} (H_{j,d_2}^{(3)} H_{d_2,d_1}^{(2)} H_{d_1,k}^{(1)} w_{jk})(H_{l,d_2'}^{(3)} H_{d_2',d_1'}^{(2)} H_{d_1',k}^{(1)} w_{lk'})
\times \mathbb{I}\left[ H_{d_2}^{(2)} \text{ReLU}(H_{d_2}^{(1)}(w_j \odot z_i)) > 0 \right] \mathbb{I}\left[ \sum_{k=1}^{K} H_{d_1,k}^{(1)} w_{jk} z_{ik} > 0 \right]
\times \mathbb{I}\left[ H_{d_2}^{(2)} \text{ReLU}(H_{d_2}^{(1)}(w_l \odot z_l)) > 0 \right] \mathbb{I}\left[ \sum_{k=1}^{K} H_{d_1',k}^{(1)} w_{lk} z_{ik} > 0 \right].
\]
As the neural network weights are independent Gaussians, we keep only the terms where $d_1 = d_1'$ and $d_2 = d_2'$:

\[
\text{Cov}(x_{ij}, x_{il}) \approx \sum_{d_2 = 1}^{D} H_{j,d_2}^{(3)} H_{l,d_2}^{(3)} \sum_{d_1 = 1}^{D} \left( H_{d_2,d_1}^{(2)} \right)^2 \sum_{k=1}^{K} \sum_{k'=1}^{K} C_{kk'} H_{d_1,k}^{(1)} H_{d_1,k'}^{(1)} w_{jk} w_{jk'}
\]

\[
\times \mathbb{I} \left[ H_{d_2}^{(2)} \text{ReLU}(H^{(1)}(w_j \odot \hat{z}_i)) > 0 \right] \mathbb{I} \left[ \sum_{k=1}^{K} H_{d_1,k}^{(1)} w_{jk} \hat{z}_{ik} > 0 \right]
\]

\[
\times \mathbb{I} \left[ H_{d_2}^{(2)} \text{ReLU}(H^{(1)}(w_l \odot \hat{z}_i)) > 0 \right] \mathbb{I} \left[ \sum_{k=1}^{K} H_{d_1,k}^{(1)} w_{lk} \hat{z}_{ik} > 0 \right].
\]

(42)

If $j$ and $l$ are both anchor features for factor $k$, we have:

\[
\text{Cov}(x_{ij}, x_{il}) \approx \sum_{d_2 = 1}^{D} \left( H_{j,d_2}^{(3)} \right)^2 \sum_{d_1 = 1}^{D} \left( H_{d_2,d_1}^{(2)} \right)^2 C_{kk} H_{d_1,k}^{(1)} w_{jk}^2
\]

\[
\times \mathbb{I} \left[ H_{d_2}^{(2)} \text{ReLU}(H^{(1)}(w_j \odot \hat{z}_i)) > 0 \right] \mathbb{I} \left[ \sum_{k=1}^{K} H_{d_1,k}^{(1)} w_{jk} \hat{z}_{ik} > 0 \right],
\]

(43)

as $H_{j,d_2}^{(3)} = H_{l,d_2}^{(3)}$ and $w_{j,k} = w_{l,k}$ by definition of an anchor feature.

Meanwhile, if $j$ and $l$ are not anchor features for the same factor, we have

\[
\text{Cov}(x_{ij}, x_{il}) \approx \sum_{d_2 = 1}^{D} H_{j,d_2}^{(3)} H_{l,d_2}^{(3)} A_{d_2}
\]

(44)

where

\[
A_{d_2} = \sum_{d_1 = 1}^{D} \left( H_{d_2,d_1}^{(2)} \right)^2 \sum_{k=1}^{K} \sum_{k'=1}^{K} C_{kk'} H_{d_1,k}^{(1)} H_{d_1,k'}^{(1)} w_{jk} w_{jk'} \mathbb{I} \left[ H_{d_2}^{(2)} \text{ReLU}(H^{(1)}(w_j \odot \hat{z}_i)) > 0 \right]
\]

\[
\times \mathbb{I} \left[ \sum_{k=1}^{K} H_{d_1,k}^{(1)} w_{jk} \hat{z}_{ik} > 0 \right] \mathbb{I} \left[ H_{d_2}^{(2)} \text{ReLU}(H^{(1)}(w_l \odot \hat{z}_i)) > 0 \right] \mathbb{I} \left[ \sum_{k=1}^{K} H_{d_1,k}^{(1)} w_{lk} \hat{z}_{ik} > 0 \right].
\]

(45)

As the weights are independent, we have that $H_{j,d_2}^{(3)}$, $H_{l,d_2}^{(3)}$, $A_{d_2}$ are independent. For large $D$, we then have

\[
\sum_{d_2 = 1}^{D} H_{j,d_2}^{(3)} H_{l,d_2}^{(3)} A_{d_2} \to 0.
\]

(50)

Hence, for two anchor features $j$ and $l$ and non-anchor feature $m$, we have $\text{Cov}(x_{ij}, x_{il}) > \text{Cov}(x_{ij}, x_{im})$. 

(51)
B Inference

B.1 Derivation of ELBO for identity prior covariance

In the main body of the paper, we set the prior covariance to $\Sigma_z = I$ as it performed well empirically and was computationally efficient. For this setting, the objective is derived as

$$
\sum_{i=1}^{N} \log p_{\theta}(\mathbf{x}_i|\mathbf{W},\eta) = \sum_{i=1}^{N} \log \left( \int p_{\theta}(\mathbf{x}_i|z_i,\mathbf{W})p(z_i)dz_i \right) + \log \left( \int p(\mathbf{W}|\Gamma)p(\Gamma|\eta)p(\eta)d\Gamma \right) 
$$

$$
= \sum_{i=1}^{N} \log \left( \mathbb{E}_{q_\phi(z_i|x_i)} \left[ p_{\theta}(\mathbf{x}_i|z_i,\mathbf{W}) \frac{p(z_i)}{q_\phi(z_i|x_i)} \right] \right) + \log \left( \mathbb{E}_{\Gamma|\mathbf{W},\eta} \left[ \frac{p(\mathbf{W}|\Gamma)p(\Gamma|\eta)p(\eta)}{p(\mathbf{W}|\eta)} \right] \right) 
$$

$$
\geq \sum_{i=1}^{N} \{ \mathbb{E}_{q_\phi(z_i|x_i)}[\log p_{\theta}(\mathbf{x}_i|z_i,\mathbf{W})] - D_{KL}(q_\phi(z_i|x_i)||p(z_i)) \} + \mathbb{E}_{\Gamma|\mathbf{W},\eta} [\log p(\mathbf{W}|\Gamma)p(\Gamma|\eta)p(\eta)] 
$$

$$
\equiv \mathcal{L}(\theta,\phi,\mathbf{W},\eta). 
$$

The KL divergence between the variational posterior $q_\phi(z_i|x_i)$ and the prior $z_i \sim \mathcal{N}_K(0, I)$ is:

$$
D_{KL}(q_\phi(z_i|x_i)||p(z_i)) = -\frac{1}{2} \sum_{k=1}^{K} \left[ 1 + \log(\sigma_\phi^2(x_i)) - (\mu_\phi(x_i))^2 - \sigma_\phi^2(x_i) \right] 
$$

The final term of the ELBO in Eq. 53 is:

$$
\mathbb{E}_{\Gamma|\mathbf{W}(t),\eta(t)} [\log p(\mathbf{W}|\Gamma)p(\Gamma|\eta)p(\eta)] = \sum_{k=1}^{K} \sum_{j=1}^{G} \lambda^*(w_{jk}^{(t)}, \eta_k^{(t)})|w_{jk}| + \sum_{k=1}^{K} \left[ \sum_{j=1}^{G} \mathbb{E}[\gamma_{jk}|w_{jk}^{(t)}, \eta_k^{(t)}] + a - 1 \right] \log \eta_k 
$$

$$
+ \left[ G - \sum_{j=1}^{G} \mathbb{E}[\gamma_{jk}|w_{jk}^{(t)}, \eta_k^{(t)}] + b - 1 \right] \log(1 - \eta_k), 
$$

where

$$
\mathbb{E}[\gamma_{jk}|w_{jk}, \eta_k] = \frac{\eta_k \psi_1(w_{jk})}{\eta_k \psi_1(w_{jk}) + (1 - \eta_k) \psi_0(w_{jk})} 
$$

$$
\lambda^*(w_{jk}, \eta_k) = \lambda_1 \mathbb{E}[\gamma_{jk}|w_{jk}, \eta_k] + \lambda_0 (1 - \mathbb{E}[\gamma_{jk}|w_{jk}, \eta_k]). 
$$

As described in Algorithm 1, we alternate between updating $\mathbb{E}[\gamma_{jk}|w_{jk}^{(t)}, \eta_k^{(t)}]$ and $\eta_k$, and taking gradient steps for $\theta, \phi$ and $\mathbf{W}$.

B.2 Derivation of ELBO for general prior covariance

In this section, we detail the sparse VAE ELBO for general $\Sigma_z$. We do not study this algorithm in this paper but we include the derivation here for completeness.

The change in the sparse VAE ELBO for general $\Sigma_z$ is the KL divergence term:

$$
D_{KL}(q_\phi(z_i|x_i)||p(z_i)) = -\frac{1}{2} \left\{ \sum_{k=1}^{K} \left[ 1 + \log(\sigma_\phi^2(x_i)) - \text{tr}(\Sigma_z^{-1} \text{diag}(\sigma_\phi^2(z_i))) - \mu_\phi(z_i)^T \Sigma_z^{-1} \mu_\phi(z_i) - \log |\Sigma_z| \right] \right\}. 
$$
C Details of empirical studies

C.1 Dataset details and preprocessing

PeerRead (Kang et al., 2018). We discard any word tokens that appear in fewer than about 0.1% of the abstracts and in more than 90% of the abstracts. From the remaining word tokens, we consider the $G = 500$ most used tokens as features. The observations are counts of each feature across $N \approx 11,000$ abstracts.

Semi-synthetic PeerRead The training and testing data are distributed differently because we vary the correlations among the latent factors that generate the features across the two datasets. This data was generated as follows: (i) we applied Latent Dirichlet Allocation (LDA, Blei et al., 2003) to the PeerRead dataset using $K = 20$ components to obtain factors $\theta \in \mathbb{R}^{N \times K}$ and topics (loadings) $\beta \in \mathbb{R}^{K \times G}$. (ii) We created train set factors, $\theta_{tr}$, by dropping the last $K_2$ columns of $\theta$ and replacing them with columns calculated as: $\logit \tilde{\theta}_{k} = \logit \theta_{k} + N(0, \sigma^2)$ for each of the first $K_2$ latent dimension $k$. We fix the test set factors as $\theta_{te} = \theta$.

MovieLens (Harper and Konstan, 2015). We consider the MovieLens 25M dataset. Following (Liang et al., 2018), we code ratings four or higher as one and the remaining ratings as zero. We retain users who have rated more than 20 movies. We keep the top $G = 300$ most rated movies. Finally, we randomly subsample $N = 100,000$ users.

Zeisel (Zeisel et al., 2015). We first processed the data following (Zeisel et al., 2015). Next, we normalized the gene counts using quantile normalization (Bolstad et al., 2003; Bolstad, 2018). Finally, following (Lopez et al., 2018), we retained the top $G = 558$ genes with the greatest variability over cells.

C.2 Sparse VAE settings

- For all experiments, the Sparse VAE takes the $\eta_k$ prior hyperparameters to be $a = 1, b = G$, where $G$ is the number of observed features.
- For experiments with Gaussian loss, the prior on the error variance is:

$$\sigma_j^2 \sim \text{Inverse-Gamma}(\nu/2, \nu \xi/2).$$  \hfill (60)

We set $\nu = 3$. The hyperparameter $\xi$ is set to a data-dependent value. Specifically, we first calculate the sample variance of each feature, $x_j$. Then, we set $\xi$ such that the 5% quantile of the sample variances is the 90% quantile of the Inverse-Gamma prior. The idea is: the sample variance is an overestimate of the true noise variance. The smaller sample variances are assumed to correspond to mostly noise and not signal – these variances are used to calibrate the prior on the noise.

C.3 Experimental settings

- For the neural networks, we use multilayer perceptrons with the same number of nodes in each layer. These settings are detailed in Table 6.
- For stochastic optimization, we use automatic differentiation in PyTorch, with optimization using Adam (Kingma and Ba, 2015) with default settings (beta1=0.9, beta2=0.999)
- For LDA, we used Python’s sklearn package with default settings. For NMF, we used sklearn with alpha=1.0.
- The dataset-specific experimental settings are in Table 6.

C.4 Additional experiments

C.4.1 Synthetic data

We consider an additional experiments to answer the questions:
1. How does the sparse VAE perform when the regularization parameter in the sparsity-inducing prior is decreased?

2. Does the Spike-and-Slab Lasso prior perform similarly to an alternate sparsity-inducing prior, the Horseshoe prior (Carvalho et al., 2009)?

**Experiment 1.** We find that the sparse VAE with less regularization (λ₀ = λ₁ = 0.001) performs slightly worse than the sparse VAE with more regularization on W (SparseVAE) in terms of MSE (Figure 4a) and DCI (Figure 4b). The sparse VAE with less regularization still performs well, however – this is because the W it learns also tends to be somewhat sparse (Figure 4c). Note that when λ₀ = λ₁, the Spike-and-Slab Lasso prior is equivalent to a Lasso prior (Tibshirani, 1996).

**Experiment 2.** We find that the Spike-and-Slab Lasso and the horseshoe perform similarly for lower values of the factor correlation ρ, but that the Spike-and-Slab has lower MSE and higher DCI than the horseshoe for higher values of ρ.

To implement the horseshoe prior, we follow Ghosh et al. (2019) and parameterize the half-Cauchy distribution $C^+(0, b)$ as:

$$ a \sim C^+(0, b) \Leftrightarrow a^2 | \lambda \sim \text{Inv-Gamma}(1/2, 1/\lambda), \quad \lambda \sim \text{Inv-Gamma}(1/2, 1/b^2). \quad (61) $$

Then, we assign W a horseshoe prior:

$$ \lambda_{\text{global}} \sim \text{Inv-Gamma}(1/2, 1/b^2_{\text{global}}), \quad (62) $$

$$ \lambda_{\text{local}} \sim \text{Inv-Gamma}(1/2, 1/b^2_{\text{local}}), \quad (63) $$

$$ \tau_{jk} | \lambda_{\text{local}} \sim \text{Inv-Gamma}(1/2, 1/\lambda_{\text{local}}), \quad (64) $$

$$ v_k | \lambda_{\text{global}} \sim \text{Inv-Gamma}(1/2, 1/\lambda_{\text{global}}), \quad (65) $$

$$ w_{jk} | \tau_{jk}, v_k \sim N(0, (t_{jk}^2 v_k^2 I), \quad (66) $$

where $s \sim \text{Inv-Gamma}(a, b)$ is the inverse-Gamma distribution with density $p(s) \propto s^{a-1} \exp\{-b/s\}$ for $s > 0$. We set the hyperparameters to $b_{\text{global}} = b_{\text{local}} = 1.0$.
(a) Heldout mean squared error (lower is better)  
(b) Ground truth factor recovery (higher is better)

Figure 5: Synthetic data. The Sparse VAE with the Spike-and-Slab-Lasso prior performs slightly better than Sparse VAE with a horseshoe prior $W$ in terms of (a) MSE; (b) DCI Disentanglement Score.

With this horseshoe prior on $W$, we again optimize $W$ using coordinate ascent and then use closed-form updates for the inverse-Gamma parameters:

$$
\tau_{jk}^{(t)} = 0.5\|w_{jk}^{(t)}\|^2/(2v_k^{(t-1)} + 1/\lambda_{local}^{(t-1)}), 
$$

$$
v_k^{(t)} = 2/(G+3) \left( \sum_{j=1}^G w_{jk}^{(t)^2}/2\tau_{jk}^{(t)} + 1/\lambda_{global}^{(t-1)} \right), 
$$

$$
\lambda_{local}^{(t)} = 2/(GK+3) \left( \sum_{k,j=1}^{KG} 1/\tau_{jk}^{(t)^2} + 1/b_{local} \right), 
$$

$$
\lambda_{global}^{(t)} = 2/(K+3) \left( \sum_{k=1}^K 1/v_k^{(t)^2} + 1/b_{global} \right).
$$

(67)  
(68)  
(69)  
(70)
Figure 6: MovieLens. The Sparse VAE generally performs as well as or better than a VAE on (a) negative heldout log-likelihood and (b) heldout recall when varying the sample size and number of features (movies). Results are averaged over 5 splits of the data.

C.5 MovieLens: varying sample size and number of features

In this section, we consider the performance of the Sparse VAE when we vary the sample size and number of features (movies) in the MovieLens dataset. The settings are the same as noted in Table 6, except for the latent space dimension which is set to $K = 50$.

Generally, the Sparse VAE performs as well as or better than a VAE on negative heldout log-likelihood and heldout recall (Figure 6). For the largest setting ($N = 100,000, G = 1,000$), the Sparse VAE is about 100 times slower than a VAE.

C.5.1 MovieLens and PeerRead: comparisons with NMF

We include additional comparisons of the Sparse VAE with non-negative matrix factorization (NMF) on the MovieLens and PeerRead datasets.

On the MovieLens data, examples of topics found by NMF include those related to science fiction; animated childrens’ movies; and Star Wars (although Toy Story is included in the Star Wars topic). This performance is similar to the Sparse VAE.

On the PeerRead data, examples of topics found by NMF include those related to reinforcement learning; language modeling; and neural network models.

In summary, NMF finds interpretable topics, similarly to the Sparse VAE. However, the Sparse VAE has better heldout predictive performance than NMF. This provides evidence for the Sparse VAE retaining the interpretability of linear methods while improving model flexibility.

C.6 Compute

- GPU: NVIDIA TITAN Xp graphics card (24GB).
- CPU: Intel E4-2620 v4 processor (64GB).
Table 5: On MovieLens (left) and PeerReads (right), NMF finds meaningful topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Movies</th>
<th>Topic</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alien; Aliens; Terminator 2; Terminator; Blade Runner</td>
<td>A</td>
<td>agent; action; policy; state; game</td>
</tr>
<tr>
<td>B</td>
<td>Shrek 1 &amp; 2; Monsters, Inc.; Finding Nemo; The Incredibles</td>
<td>B</td>
<td>word; representation; vector; embeddings; semantic</td>
</tr>
<tr>
<td>C</td>
<td>Star Wars IV, V &amp; VI; Toy Story</td>
<td>C</td>
<td>model; inference; neural; latent; structure</td>
</tr>
</tbody>
</table>

Table 6: Settings for each experiment.

<table>
<thead>
<tr>
<th></th>
<th>Synthetic data</th>
<th>MovieLens</th>
<th>Semi-synthetic PeerRead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Settings</td>
<td>Value</td>
<td>Settings</td>
</tr>
<tr>
<td># hidden layers</td>
<td>3</td>
<td></td>
<td># hidden layers</td>
</tr>
<tr>
<td># layer dimension</td>
<td>50</td>
<td></td>
<td># layer dimension</td>
</tr>
<tr>
<td>Latent space dimension</td>
<td>5</td>
<td></td>
<td>Latent space dimension</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.01</td>
<td></td>
<td>Learning rate</td>
</tr>
<tr>
<td>Epochs</td>
<td>200</td>
<td></td>
<td>Epochs</td>
</tr>
<tr>
<td>Batch size</td>
<td>100</td>
<td></td>
<td>Batch size</td>
</tr>
<tr>
<td>Loss function</td>
<td>Gaussian</td>
<td></td>
<td>Loss function</td>
</tr>
<tr>
<td>Sparse VAE</td>
<td>$\lambda_1 = 1, \lambda_0 = 10$</td>
<td>$\lambda_1 = 1, \lambda_0 = 10$</td>
<td>$\beta$-VAE</td>
</tr>
<tr>
<td>$\beta$-VAE</td>
<td>$[2, 4, 6, 8, 16]$</td>
<td>$\beta$-VAE</td>
<td>$\gamma = 0.01$</td>
</tr>
<tr>
<td>VSC</td>
<td>$\alpha = 0.01$</td>
<td>VSC</td>
<td>$\alpha = 0.01$</td>
</tr>
<tr>
<td>OI-VAE</td>
<td>$\lambda = 1, p = 5$</td>
<td>OI-VAE</td>
<td>$\lambda = 1, p = 5$</td>
</tr>
<tr>
<td>Runtime per split</td>
<td>CPU, 2 mins</td>
<td>Runtime per split</td>
<td>GPU, 1 hour</td>
</tr>
</tbody>
</table>

Zeisel

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Settings</th>
<th>Value</th>
<th>Settings</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># hidden layers</td>
<td>3</td>
<td># hidden layers</td>
<td>3</td>
<td># hidden layers</td>
<td>3</td>
</tr>
<tr>
<td># layer dimension</td>
<td>100</td>
<td># layer dimension</td>
<td>50</td>
<td># layer dimension</td>
<td>50</td>
</tr>
<tr>
<td>Latent space dimension</td>
<td>15</td>
<td>Latent space dimension</td>
<td>20</td>
<td>Latent space dimension</td>
<td>20</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.01</td>
<td>Learning rate</td>
<td>0.01</td>
<td>Learning rate</td>
<td>0.01</td>
</tr>
<tr>
<td>Epochs</td>
<td>100</td>
<td>Epochs</td>
<td>100</td>
<td>Epochs</td>
<td>100</td>
</tr>
<tr>
<td>Batch size</td>
<td>512</td>
<td>Batch size</td>
<td>128</td>
<td>Batch size</td>
<td>128</td>
</tr>
<tr>
<td>Loss function</td>
<td>Gaussian</td>
<td>Loss function</td>
<td>Softmax</td>
<td>Loss function</td>
<td>Softmax</td>
</tr>
<tr>
<td>Sparse VAE</td>
<td>$\lambda_1 = 1, \lambda_0 = 10$</td>
<td>$\lambda_1 = 0.001, \lambda_0 = 5$</td>
<td>$\beta$-VAE</td>
<td>$[2, 4, 6, 8, 16]$</td>
<td></td>
</tr>
<tr>
<td>$\beta$-VAE</td>
<td>$[2, 4, 6, 8, 16]$</td>
<td>$\beta$-VAE</td>
<td>$\gamma = 0.01$</td>
<td>$\alpha = 0.01$</td>
<td></td>
</tr>
<tr>
<td>VSC</td>
<td>$\alpha = 0.01$</td>
<td>VSC</td>
<td>$\alpha = 0.01$</td>
<td>$\gamma = 0.01$</td>
<td></td>
</tr>
<tr>
<td>OI-VAE</td>
<td>$\lambda = 1, p = 5$</td>
<td>OI-VAE</td>
<td>$\lambda = 1, p = 5$</td>
<td>$\gamma = 0.01$</td>
<td></td>
</tr>
<tr>
<td>Runtime per split</td>
<td>CPU, 15 mins</td>
<td>Runtime per split</td>
<td>GPU, 30 mins</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>