Latent Bernoulli Autoencoder

Jiri Fajtl [1] **Vasileios Argyriou** [1] **Dorothy Monekosso** [2] **Paolo Remagnino** [1]

Abstract

In this work, we pose the question whether it is possible to design and train an autoencoder model in an end-to-end fashion to learn representations in the multivariate Bernoulli latent space, and achieve performance comparable with the state-of-the-art variational methods. Moreover, we investigate how to generate novel samples and perform smooth interpolation and attributes modification in the binary latent space. To meet our objective, we propose a simplified, deterministic model with a straight-through gradient estimator to learn the binary latents and show its competitiveness with the latest VAE methods. Furthermore, we propose a novel method based on a random hyperplane rounding for sampling and smooth interpolation in the latent space. Our method performs on a par or better than the current state-of-the-art methods on common CelebA, CIFAR-10 and MNIST datasets.

1. Introduction

Unsupervised representation learning is a very exciting direction in machine learning, particularly given the plethora of easily available unlabeled data. There are many machine learning algorithms that would greatly benefit from low dimensional, highly expressive features, whether for object detection, classification, reinforcement learning or as generative models for compression, super resolution or novel samples generation. This direction has been successfully pursued with autoencoder models and particularly the variational autoencoder (VAE) (Kingma & Welling, 2014) and its derivatives. Recently, fully deterministic, regularized (Ghosh et al., 2020) and discrete, vector-quantized (VQ-VAE) (van den Oord et al., 2017) autoencoders have been proposed, demonstrating performance comparable to theirs stochastic counterparts.

[1]Kingston University, London, UK [2]Leeds Beckett University, Leeds, UK. Correspondence to: Jiri Fajtl <j.fajtl@kingston.ac.uk>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

In this work we focus on a deterministic class of autoencoders learning a discrete representation, specifically the multivariate Bernoulli distribution without enforcing any prior on the latent space and trained with a gradient based method in an end-to-end fashion.

Binary representations appear to be attractive for a number of applications, for example the realization of an encoder for sparse distributed representations for methods such as the Hierarchical Temporal Memory (HTM) (Hawkins & Ahmad, 2016), modelling neurobiological processes (Bethge & Berens, 2008), data compression, memory addressing (Rae et al., 2016), for gating (hard attention)(Xu et al., 2015) or for general representation learning (Bengio et al., 2012; 2013a). The authors believe that good, as defined by Bengio et al. 2013a, binary features have also great potential in application to energy based memory models such as Hopfield networks operating on a single, capsule (Sabour et al., 2017) like neuron level.

Current neural network learning algorithms are almost exclusively based on very successful gradient based learning methods. However, the need for differentiability of each layer represents a challenge if one desires to train stochastic neurons or other non-differentiable functions such as quantization. Number of techniques have been proposed allowing gradient propagation through such neurons such as re-parametrization, (Kingma & Welling, 2014), surrogate gradient functions (Bengio et al., 2013b) or continuous relaxation of non-differentiable nodes (Jang et al., 2017). In our method we follow the approach behind the straight-through estimator (Hinton, 2012; Bengio et al., 2013b) due its conceptually simple setup.

Sampling from and interpolating in the discrete latent space is equally challenging. Unlike multimodal, Gaussian and many other real-valued distributions, the multivariate Bernoulli distribution concentrates most of the information on the second and higher moments, since the marginals are strictly unimodal and entirely described by the mean $p = \mathbb{E}[\mathrm{b_i}]$, directly giving rise to variance $Var[b_i] = p(1-p)$ for Bernoulli variable b_i at dimension i. This also seems to play a key role in biological neurons, where the binary, pairwise correlations provide strikingly accurate encoding for neuronal firing patterns in primate retina (Nirenberg & Victor, 2007; Schneidman et al., 2006; Shlens et al., 2006).

Given that our model learns a distribution with unknown prior, and based on the aforementioned premise, we propose to parametrize the learned distribution by its first two moments, also motivated by the cross moment model by Mishra et al. 2012. These parameters are learned from latents encoded on the training data. To sample and interpolate in the multivariate Bernoulli latent space we then propose a novel method based on a random hyperplane rounding technique derived from MAX-CUT algorithm (Goemans & Williamson, 1995). Within this work we abbreviate the Latent Bernoulli Autoencoder as LBAE.

We evaluate our method on the datasets CelebA and CIFAR-10 and, for completeness, MNIST and show that our method is competitive with the current state-of-the-art variational and deterministic autoencoders. Our model shows high performance particularly on the interpolation task, which is remarkable, considering we are operating in the discrete latent space. To best of our knowledge, none of the existing discrete distribution autoencoders is able to perform sensible interpolation in the latent space. For example, a state of the art method VQ-VAE (van den Oord et al., 2017) does not suggest how to do so and even explicit methods, as in Berthelot et al. 2019, admit difficulties in accomplishing this task. Finally, we present a simple method for attributes modification in the latent space, also with competitive results. PyTorch code and trained models are publicly available on github[1]. Our work brings the following contributions:

- We show that a tanh function followed by a straight-through estimator with an unity surrogate function in the backward pass can be used to efficiently train an autoencoder with state-of-the-art performance.
- We propose a novel technique to generate correlated Bernoulli samples, perform smooth interpolation and modify sample attributes in the discrete latent space.
- We show that, albeit its simplicity, our method performs equally well or better than the state-of-the-art using the FID, KID and Precision/Recall metrics.

2. Related Work

Unsupervised representation learning has been successfully pursued with autoencoder models, particularly the variational autoencoder (VAE) (Kingma & Welling, 2014) due to its simplicity and well defined probabilistic framework. VAE unfortunately suffers from number of issues, most notably producing blurred images (Dumoulin et al., 2017) and posterior collapse (Razavi et al., 2019a). A number of methods have been proposed to improve the image quality with reconstruction loss based on perceptual similarity in the feature space of an external CNN (Dosovitskiy & Brox, 2016; Hou et al., 2017) or in its own latent space (Zhang et al., 2019). Success of the Generative Adversarial Networks (GAN) to learn image distribution motivated application of the adversarial training to the latent space distribution in the Adversarial Autoencoders (AAE) (Makhzani et al., 2016) and its generalization in Wasserstein Autoencoders (WAE) (Tolstikhin et al., 2018). More recently Dai & Wipf 2019 introduced a 2 stage VAE where the second stage learns the latent space distribution, in principle, performs a density estimation. From the work of Ghosh et al. 2020 it is apparent that deterministic autoencoders are competitive with the VAE and its derivatives, only for the price of ex-post density estimation.

Most of the methods learn real-valued latent space owning to the established gradient-based optimizations. VQ-VAE (van den Oord et al., 2017) is perhaps the first competitive deterministic, autoencoder that learns discrete representations. As in Ghosh et al. 2020, this method does not impose any prior on the learned latent distribution, thus it requires some form of external post density estimation. Authors propose the PixelCNN (Van den Oord et al., 2016), an autoregressive density estimator which learns a categorical prior over the stored latents encoded from the training dataset.

Learning discrete representations with a gradient-based optimization is not straightforward. Bengio et al. 2013b proposed four methods, addressing the learning through stochastic neurons, most notably the straight-through gradient estimator, originally described by Hinton 2012. The straight-through estimator is also used in the VQ-VAE model to allow gradient flow over non differentiable, nearest neighbour operation in the forward pass. Chung et al. 2017 then introduces a straight-through estimator with the slope annealing trick. Over the training period, this method gradually reduces the difference between the non-differentiable function in the forward pass and the surrogate in the backward pass to converge to the discrete distribution in the limit. This method is somewhat similar to the ST Gumbel-Softmax (Jang et al., 2017). The Gumbel-Softmax was also applied to the autoencoder model in JointVAE (Dupont, 2018).

3. Bernoulli Latent Space

3.1. Learning the Bernoulli Latent Space

The base of our method is a deterministic autoencoder with encoder $\mathbf{z} = g_\phi(\mathbf{X})$, parametrized by ϕ, that produces typically real-valued latent representation $\mathbf{z}$ for input $\mathbf{X}$. Decoder $\mathbf{X}' = f_\theta(\mathbf{z})$, parametrized by θ, attempts to reconstructs $\mathbf{X}$ from $\mathbf{z}$. Our model is trained with a single, common objective function $\mathcal{L}(\theta, \phi) = \mathbb{E}[L(\mathbf{X}, \mathbf{X}')]$, where L is the reconstruction loss function. To discretize an N dimensional latent $\mathbf{z} \in \mathbb{R}^N$ into the binary range $\mathbf{b} \in \{-1, 1\}^N$

[1] https://github.com/ok1zjf/lbae

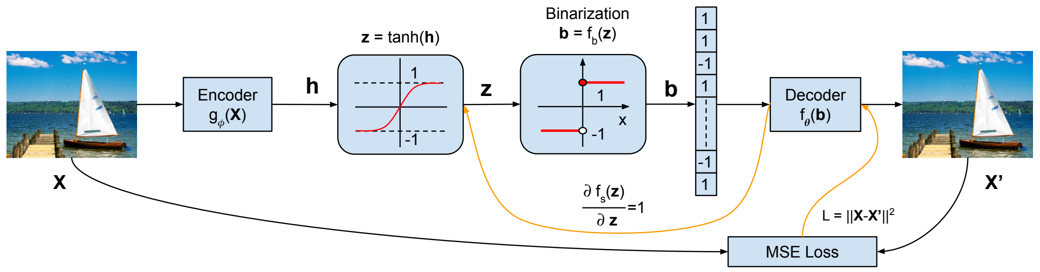


Figure 1. For N dimensional latent space the information bottleneck of a typical autoencoder is in LBAE replaced with tanh() followed by binarization $f_b() \in \{-1,1\}^N$ with unit gradient surrogate function $f_s()$ for backward pass.

we threshold $\mathbf{z}$ at zero as follows:

$$b_i = f_b(z_i) = \begin{cases} 1, & \text{if} \quad z_i \geq 0 \\ -1, & \text{otherwise.} \end{cases} \tag{1}$$

We chose to represent the binary values by $\{-1,1\}$ rather than $\{0,1\}$ due to its computational benefits such as zero being the threshold level and $b_i^2 = 1\ \forall i$. The latents can be easily converted between these two ranges without loss of information.

Since $f_b()$ is not differentiable, we define a surrogate differentiable function $f_s(\mathbf{z}) = \mathbf{z}$ with unit gradient $\nabla_{\mathbf{z}} f_s = 1$ operating in the same domain as $f_b()$. $f_s()$ is then used in the backward pass. During the backpropagation this allows the gradient to flow through the binarization operation and lets the encoder correct its output in the direction of the binarized quantities read by the decoder. The rounding during the binarization brings an additional error that is not corrected during the backpropagation and manifests as a noise. This noise can be reduced by lowering the learning rate but it slows down the training or hinders the convergence altogether. To alleviate this weakness we add $\tanh()$ before the binarization, which limits the gradient flow from the decoder and minimizes the optimization overshoot during the gradient descent.

3.2. Sampling Correlated Multivariate Bernoulli Latents

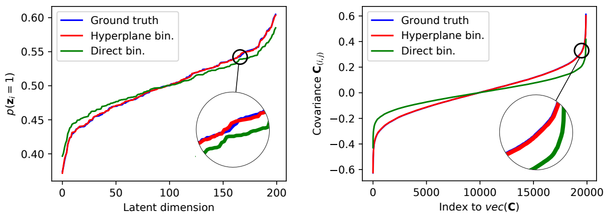
Our goal is to realize a generative model of the form $\mathbf{x} \sim p(\mathbf{x} \mid \mathbf{b}; \theta)$ for $\mathbf{b} \sim p(\mathbf{b}), \mathbf{b} \in \{-1,1\}^N$. Unlike VAE, we do not enforce any prior on the latent space during the training, thus the learned distribution $p(\mathbf{b})$ is unknown. Therefore, to efficiently sample novel latents we first learn $p(\mathbf{b})$ from the distribution of the training dataset in the latent space and parametrize it by its first two moments.

The most straightforward way to learn and sample from the correlated Bernoulli distribution would appear to treat it as a Gaussian distribution with the binarization step. Let us consider a matrix $\mathbf{Y} \in \{-1,1\}^{(N\times K)}$ of K N-dimensional latent vectors encoded on training data. Given expected value $\mathbb{E}[\mathbf{Y}] \in R^N$ and covariance $\Sigma = \mathbb{E}[\mathbf{Y}\mathbf{Y}^{\mathrm{T}}] - \mathbb{E}[\mathbf{Y}]\,\mathbb{E}[\mathbf{Y}]^T$ we can sample a latent $\mathbf{b}$ from the distribution as:

$$\mathbf{z} \sim \mathcal{N}_N(0, \mathbf{I}_N) \tag{2}$$

$$\mathbf{b} = f_b(\mathbf{L}\mathbf{z} + \mathbb{E}[\mathbf{Y}]), \quad \mathbf{b} \in \{-1,1\}^N, \tag{3}$$

where $\Sigma = \mathbf{L}\mathbf{L}^T$ is a lower triangular Cholesky decomposition. This approach, however, does not produce Bernoulli samples with the correct distribution. To mitigate this issue, we propose a method inspired by the cross moment model method (Mishra et al., 2012) and random hyperplane rounding technique for MAX-CUT (Goemans & Williamson, 1995). In Figure 2 we can see that a distribution generated by the direct binarization (Eq. 3) (green) exhibits noticeable error compared to the ground truth (blue). The red plot shows distribution generated with the proposed random hyperplane method.



(a) Sorted marginal probabilities

(b) Vectorized, sorted covariances

Figure 2. Ground truth (200bits latents, MNIST train data) and the distribution sampled with the random hyperplane method appear identical while the direct rounding method exhibits a clear error. Note the ground truth (blue) is mostly hidden behind the red.

Our method can be summarized in the following three steps: (**1**) parametrize distribution of the training dataset in latent space by first two moments, (**2**) relax each latent dimension to an unit vector on a hypersphere with a position corresponding to its correlation with other dimensions, (**3**) sample latent $\mathbf{b}$ by randomly splitting the sphere through the centre with a hyperplane normal $\mathbf{r}$ and assigning binary state 1 to dimensions corresponding to vectors on one side of the plane and -1 to the rest.

The distribution of $\mathbf{Y}$ is parametrized by first moments and second non-central moments, similar to Mishra et al. 2012, in matrix $\mathbf{M}$ as:

$$\mathbf{M} = \begin{bmatrix} \mathbb{E}[\mathbf{Y}\mathbf{Y}^{\mathrm{T}}] & \mathbb{E}[\mathbf{Y}] \\ \mathbb{E}[\mathbf{Y}]^{T} & 1 \end{bmatrix}, \mathbf{M} \in [-1,1]^{(N+1)\times(N+1)}. \tag{4}$$

For N dimensional latent space we generate $N+1$ unit length vectors on sphere $S^{(N+1)}$. These vectors are organized as rows in matrix $\mathbf{V} \in \mathbb{R}^{(N+1)\times(N+1)}, \forall i \in [1,..,N+1], \|\mathbf{V}_i\| = 1$, where $\mathbf{V}_i$ is an i^{th} row of $\mathbf{V}$. Each vector $\mathbf{V}_i$ represents one dimension in the latent space. We express the covariances $\mathbf{M}$ as probabilities of vectors $\mathbf{V}_i, \mathbf{V}_j$ pointing in the same or opposite direction. For positive, high covariance between dimensions i and j the angle $\alpha_{i,j}$ between corresponding vectors $\mathbf{V}_i$ and $\mathbf{V}_j$ will be small and $P(\mathbf{V}_i, \mathbf{V}_j) \to 1$, while for negative covariance $\alpha_{i,j} \to \pi$ with $P(\mathbf{V}_i, \mathbf{V}_j) \to 0$. For non correlated dimensions $\mathbf{V}_i \perp \mathbf{V}_j$ with $P(\mathbf{V}_i, \mathbf{V}_j) \approx \frac{1}{2}$. Bits of positively correlated dimensions share the same state (-1 or 1) while negatively correlated take opposite states. We set the probabilities as:

$$P(\mathbf{V}_i, \mathbf{V}_j) = \frac{M_{i,j}+1}{2}, \forall(i,j), P(\mathbf{V}_i, \mathbf{V}_j) \in [0,1] \tag{5}$$

and express them as a function of the angle $\alpha_{i,j}$ or dot product $\langle \mathbf{V}_i, \mathbf{V}_j \rangle$.

$$\begin{aligned} P(\mathbf{V}_i, \mathbf{V}_j) &= 1 - \frac{\alpha_{i,j}}{\pi}, \forall(i,j),\ \alpha_{i,j} \in [0,\pi], && (6) \\ &= 1 - \frac{\cos^{-1}(\langle \mathbf{V}_i, \mathbf{V}_j \rangle)}{\pi}. && (7) \end{aligned}$$

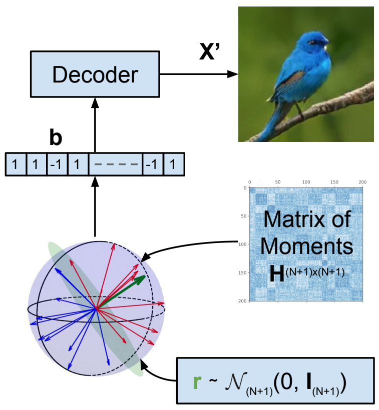
We define the dot products as Gram matrix $H_{i,j} = \langle \mathbf{V}_i, \mathbf{V}_j \rangle$, $\mathbf{H} \in \mathbb{R}^{(N+1)\times(N+1)}$ as a function of $\mathbf{M}$,

$$\begin{aligned} H_{i,j} &= \cos\left(\left(1 - \frac{1}{2}(M_{i,j}+1)\right)\pi\right) && (8) \\ &= \cos\left(\frac{\pi}{2}(1 - M_{i,j})\right). && (9) \end{aligned}$$

To obtain $\mathbf{V}$ we perform a square root of $\mathbf{H}$ by lower triangular Cholesky decomposition

$$\mathbf{H} = \mathbf{V}\mathbf{V}^{T} \quad s.t. \quad \mathbf{H} \succcurlyeq 0, \tag{10}$$

where $\mathbf{V}$ is a row-normal lower triangular matrix with rows being the desired unit vectors on $S^{(N+1)}$. The $\mathbf{V}_{(N+1)}$ represents the boundary conditions for the first moments $\mathbb{E}[\mathbf{Y}]$. Concretely, it defines the positive hemisphere in S where all vectors receive positive binary state. In other words, this boundary vector orients the hypersphere space according to the marginals $\mathbb{E}[\mathbf{Y}]$. Finally, to generate a novel latent $\mathbf{b}$ we split the sphere S with a random plane through the center and then assign positive binary states to latent dimensions represented by vectors $\mathbf{V}_i$ in one hemisphere and negative to the rest. Vectors sharing hemisphere with $\mathbf{V}_{(N+1)}$ (yellow in Figure 4) will receive positive values. For a random hyperplane given by normal $\mathbf{r} \sim \mathcal{N}_{(N+1)}(0, \mathbf{I}_{(N+1)})$ (green in Figure 4) we generate the latent $\mathbf{b}$ with bits at each dimension as:



Decoder
X'
b
1 1 -1 1 - - - - -1 1
Matrix of Moments $\mathbf{H}^{(N+1)x(N+1)}$
0 50 100 150 200
$\mathbf{r} \sim \mathcal{N}_{(N+1)}(0, \mathbf{I}_{(N+1)})$

Figure 3. Distribution of the training dataset in the latent space is parametrized by matrix $\mathbf{H}$ of first two moments. Each dimension in the latent space is represented by an unit vector on a hypersphere.

$$b_i = \begin{cases} 1, \text{ if } f_b(\langle \mathbf{V}_i, \mathbf{r} \rangle) = f_b(\langle \mathbf{V}_{(N+1)}, \mathbf{r} \rangle) \\ -1, \text{ otherwise} \end{cases}, \tag{11}$$
$$\forall i \in [1,..,N], \mathbf{r} \in \mathbb{R}^{(N+1)\times 1}.$$

In vector form the Eq. 11 is then:

$$\begin{aligned} &\mathbf{b} = f_r(\mathbf{r}) = f_b(\mathbf{V}\mathbf{r})_{-(N+1)} f_b(\mathbf{V}_{(N+1)}\,\mathbf{r}), \\ &\mathbf{b} \in \{-1,1\}^N, \end{aligned} \tag{12}$$

where subscript $_{-(N+1)}$ denotes all but $(N+1)$ dimensions. The expression $f_b(\langle \mathbf{V}_{(N+1)}, \mathbf{r} \rangle)$, and its vectorized form $f_b(\mathbf{V}_{(N+1)}\,\mathbf{r})$, returns the boundary decision bit. If positive, the hyperplane normal $\mathbf{r}$ is located in the same hemisphere as the boundary vector $\mathbf{V}_{(N+1)}$. Finally, an image $\mathbf{X}'$ is decoded from the binary latent $\mathbf{b}$ as $\mathbf{X}' = f_\theta(\mathbf{b})$.

3.3. Interpolation in the Bernoulli Latent Space

For each latent of the images we are interpolating, we first lookup up a hyperplane normal responsible for generating this latent vector according to Section 3.2. Then, intermediate latents are interpolated on the sphere $S^{(N+1)}$ between the endpoints with spherical linear interpolation (SLERP) (Shoemake, 1985).

Let us consider $\mathrm{s} \in \{-1,1\}^N$ to be our latent vector for which we desire to find a hyperplane normal $\mathbf{r} \in \mathbb{R}^{(N+1)}$ that would generate back the latent s as per Eq. 11. Intuitively, one could attempt to find the solution as $\mathbf{r} = \mathbf{V}^{-1}[\mathrm{s}, 1]$ which, indeed, recovers s back as: $\mathrm{s} = f_r(\mathbf{L}\mathbf{r})_{-(N+1)}$, where $[\mathrm{s}, 1]$ denotes s concatenated

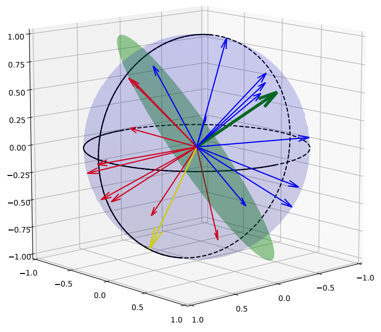


Figure 4. Each dimension in the latent space is represented by an unit vector on a hypersphere. Pairwise correlations are given by angle between vectors; the smaller angle the higher correlation between corresponding dimensions. New samples are generated by splitting the sphere with a random plane (green) and assigning positive states to dimensions (red) on the side of the plane shared by the boundary vector (yellow) and negative to the rest (blue).

with 1. By setting the boundary decision bit to positive state $[\mathbf{s}, 1]$ we will produce a hyperplane normal in the same hemisphere as the boundary vector $\mathbf{V}_{(N+1)}$, consequently this hemisphere represents positive binary states.

The hyperplane found this way is, however, not suitable for the interpolation. Interpolating between such hyperplanes produces exact copies of the source latent till the midpoint where it instantly flips to the target and stays there till the end of interpolation. The source/target flip happens over less than $1/10^6$ degrees step. It appears that the hyperplanes found this way are degenerate in some sense. They produce latents very far from the main distribution manifold. To find the nature of this behaviour is a subject of ongoing research.

Instead, we found that the most suitable latent-hyperplane inversion can be carried out by placing the hyperplane normal close to the centroids of the positive and negative vectors in $\mathbf{V}$. First, we get the centroid for all positive vectors in $\mathbf{s}$.

$$\mathbf{r}_p = \sum\nolimits_i^N \left(u(s_i)\,\mathbf{V}_i\right)^T,\ \mathbf{r}_p \in \mathbb{R}^{(N+1)\times 1}, \tag{13}$$

where $u(s_i) = \frac{1}{2}(1 + s_i)$ changes the range of its argument from $\{-1, 1\}$ to $\{0, 1\}$. $\mathbf{r}_p$ is a prototype of the hyperplane normal but it typically does not reproduce $\mathbf{s}$ accurately, causing reconstruction error in the pixel space when decoded. To mitigate this, we propose an iterative process that tilts the normal $\mathbf{r}_p$ towards the vectors incorrectly placed behind the hyperplane. The process stops when the Hamming distance between $\mathbf{s}$ and $f_r(\mathbf{r}_p)$ does not decrease, which typically takes < 4 steps. Similarly, we create a normal for the negative vectors $\mathbf{r}_n = \sum_i^N \left(u(-s_i)\,\mathbf{V}_i\right)^T$. The final normal is then:

$$\mathbf{r} = \frac{\mathbf{r}_p}{\parallel \mathbf{r}_p \parallel} - \frac{\mathbf{r}_n}{\parallel \mathbf{r}_n \parallel}. \tag{14}$$

A vector of error bits between $\mathbf{s}$ and its reconstruction with hyperplane normal $\mathbf{r}$ is calculated as:

$$E_b(\mathbf{s}, \mathbf{r}) = u(-f_r(\mathbf{r}) \odot \mathbf{s}), \tag{15}$$

where $\odot$ is Hadamard product. Hamming distance d is then:

$$d = \sum_{i=0}^{N} E_b(\mathbf{s}, \mathbf{r})_i,\ d \in \{0, ..., N\}. \tag{16}$$

Algorithm 1 summarizes the process of looking up a hyperplane normal for a given Bernoulli latent vector and $\mathbf{V}$.

Algorithm 1 Latent $\mathbf{s}$ to hyperplane normal $\mathbf{r}$ inversion

```
Function latent_to_hyperplane(s, V)
$\mathbf{r} = \sum_i^N (u(s_i)\,\mathbf{V}_i)^T$ # r is a mean vector of rows in V at $s_i$ = 1 (Eq. 13)
$d_{best} = N$ # Start with the maximum Hamming distance (all bits are different at all $N$ dimensions)
repeat
    $\mathbf{e} = E_b(\mathbf{s}, \mathbf{r})$ # Error bits vector between s and its reconstruction with hyperplane normal r (Eq. 15)
    $d = \sum_i^N e_i$ # Hamming distance
    if $d \geq d_{best}$ then
        return r # If the distance does not improve return the hyperplane normal $r$
    end if
    $\mathbf{r} = \mathbf{r} + \sum_i^N (e_i\,\mathbf{V}_i)^T$ # Add vectors at the error bits positions
    $\mathbf{r} = \mathbf{r} / \parallel \mathbf{r} \parallel$
    $d_{best} = d$
until $True$
```

We then interpolate T normals between source $\mathbf{r}_s$ and target $\mathbf{r}_t$ on the hypersphere, and for each generate a latent vector according to Eq. 12 and decode it as an image $\mathbf{X'}_i = f_\theta(\mathbf{r}_i)$

4. Evaluation

In this section we evaluate how well our method reconstructs images from latents, generates new images, interpolates between existing images and modifies image attributes in latent space. In Supplementary Material, Appendix B we briefly look at the compression capabilities. We trained and tested our model on the CelebA (Liu et al., 2015), CIFAR-10 (Krizhevsky & Hinton, 2009) and MNIST (LeCun et al., 2010) datasets with the default train/test splits and image resolutions in Table 3. To evaluate LBAE against VAE with the LBAE identical architecture, we modified the LBAE encoder to output (μ, σ) and trained it in the VAE setup. We call this model VAE(ours).

For all datasets we use almost identical models, varying in the latent dimensions (Table 3). Encoder and decoder are CNN networks with residual connections, where the decoder mirrors the encoder with transposed convolutions. The model was trained with ADAM(Kingma & Ba, 2015) with learning rate 10^{-3}, no weight decay and 512 batch size.

Table 1. FID scores. Results are taken from the corresponding publications for VPGA,LPGA (Zhang et al., 2019), VAE, WAE-MMD, RAE-L2, RAE-SN (Ghosh et al., 2020) and Best GAN, 2 Stage VAE (Dai & Wipf, 2019). For fair comparison, results for VAE, WAE-MMD, RAE-L2 and RAE-SN are split into $\mathcal{N}(0,1)$ and $\mathcal{N}(\mu,\Sigma)$ columns. The VAE(ours) architecture is identical to LBAE.

	MNIST				CIFAR-10				CelebA			
	Reco.	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$	Interp.	Reco.	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$	Interp.	Reco.	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$	Interp.
Best GAN		10				70				49		
VAE	18.26	19.21		18.21	57.94	106.37		88.62	39.12	48.12		44.49
VAE (ours)	8.77	18.52			37.94	68.43			34.96	56.08		
2 Stage VAE		12.6				72.9				44.4		
WAE-MMD	10.03	20.42		14.34	35.97	117.44		76.89	34.81	53.67		40.93
RAE-L2	10.53		22.22	14.54	32.24		80.8	62.54	43.52		51.13	45.98
RAE-SN	15.65		19.67	15.15	27.61		84.25	63.62	36.01		44.74	39.53
LPGA		12.06				55.87				**14.53**		
VPGA		**11.67**				**51.51**				24.73		
LBAE (ours)	**8.11**	88.13	**11.36**	**9.80**	**19.37**	71.48	**53.55**	**34.41**	**7.71**	64.65	**34.95**	**14.87**

Table 2. KID Scores scaled by 10^3 as in Dai & Wipf 2019.

	MNIST			CIFAR-10			CelebA		
	Reco.	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$	Reco.	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$	Reco.	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$
VAE (ours)	6.43	12.41		30.87	74.1		30.49	58.83	
2 Stage VAE		**6.7**			59.3			**40.9**	
WAE-MMD		137.8			**58.7**			59.7	
LBAE (ours)	**5.39**	84.48	**6.34**	**13.01**	74.4	**51.9**	**6.15**	75.29	**30.33**

Table 3. Image resolutions, latent sizes and training epochs.

	Image resolution	Latent size		Epochs
		LBAE (bits)	VAE(ours) (float32)	
MNIST	32x32x1, zero padded from 28x28	200	16	2000
CIFAR-10	32x32x3	600	128	2000
CelebA	64x64x3, cropped to 1:1 and scaled	1500	64	500

Mean squared error is used as the reconstruction loss except for MNIST where we use the binary cross entropy. The model architecture is shown in more details in Supplementary Material, Appendix A. The training is slower compared to the VAE due to the gradient propagation through the tanh() and binarization, nevertheless comparable to other methods such as the 2 stage VAE(Dai & Wipf, 2019) which requires 420 epochs on CelebA, 3000 on CIFAR-10 and 1200 on MNIST.

As the evaluation metrics we use the Fréchet Inception Distance (FID) (Lucic et al., 2018), Kernel Inception Distance (KID) (Bińkowski et al., 2018) and Precision/Recall (Sajjadi et al., 2018). For consistency purposes we use reference implementations for all metrics [2,3,4]. To compute FID and KID we use 10k reference and evaluation images.

[2]https://github.com/bioinf-jku/TTUR

[3]https://github.com/mbinkowski/MMD-GAN

[4]https://github.com/msmsajjadi/precision-recall-distributions

4.1. Reconstruction and Random Samples Generation

In Tables 1 and 2 we show that our model achieves the lowest reconstruction FID and KID scores. This can be attributed to the prior-free training, where the model is not constrained to approximate any prior, which is believed to produce blurry images in the case of VAE. From Figure 7 it is apparent that LBAE reconstructions are sharper than typical VAE outputs. We can also see that, on the generative task, LBAE outperforms all except the VPGA method, when sampled with the proposed hyperplane rounding method. When sampled from the binarized normal distribution $f_b(\sim \mathcal{N}_N(0,\mathbf{I}_N))$, our scores are worse. This can be also seen perceptually in Figure 5 where the generated images are sharp but composed of features with wrong consistency. This suggest that the correlation between the dimensions in the latent space is, indeed the major source of information.

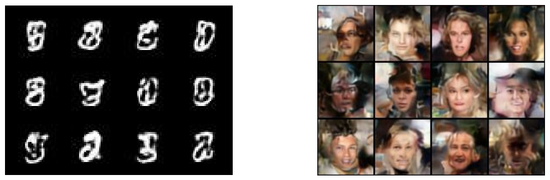
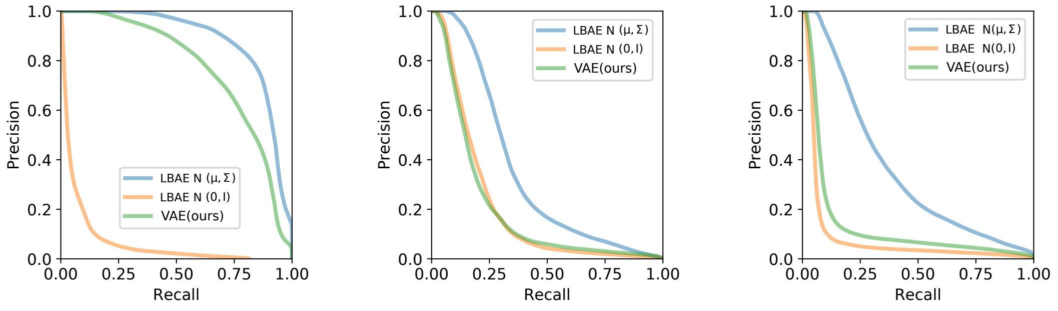


Figure 5. MNIST and CelebA images generated by LBAE from latents $\mathbf{b} = f_b(\sim \mathcal{N}_N(0,\mathbf{I}_N))$

Table 4. Precision / Recall evaluation between LBAE and methods VAE, WAE-MMD, RAE-L2, RAE-SN from Ghosh et al. 2020.

	MNIST		CIFAR-10		CelebA	
	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$	$\mathcal{N}(0,1)$	$\mathcal{N}(\mu,\Sigma)$
VAE	**0.96 / 0.92**		0.25 / 0.55		0.54 / 0.66	
VAE (ours)	0.88 / 0.93		**0.55 / 0.74**		**0.62 / 0.64**	
WAE-MMD	0.93 / 0.88		0.38 / 0.68		0.59 / 0.68	
RAE-L2		0.92 / 0.87		0.41 / 0.77		0.36 / 0.64
RAE-SN		0.89 / 0.95		0.36 / 0.73		0.54 / 0.68
LBAE (ours)	0.37 / 0.44	**0.92 / 0.97**	0.48 / 0.76	**0.66 / 0.87**	0.50 / 0.57	**0.73 / 0.82**



(a) MNIST (b) CIFAR-10 (c) CelebA

Figure 6. Precision / Recall curves.

Table 5. Precision/Recall and FID scores for sampling from GMM, except our method LBAE where we sample from the matrix of moments with the random hyperplane method.

	MNIST		CIFAR-10		CelebA	
	FID ↓	Precision /Recall ↑	FID ↓	Precision /Recall ↑	FID ↓	Precision /Recall ↑
VAE	17.66	0.95 / 0.96	103.78	0.37 / 0.56	45.52	0.50 / 0.66
WAE-MMD	9.39	0.98 / 0.95	93.53	0.51 / 0.81	42.73	0.69 / 0.77
RAE-L2	**8.69**	**0.98 / 0.98**	74.16	0.57 / 0.81	47.97	0.44 / 0.65
RAE-SN	11.74	0.98 / 0.97	75.3	0.52 / 0.81	40.95	0.55 / 0.74
LBAE (ours)	11.36	0.92 / 0.97	**53.55**	**0.66 / 0.87**	**34.95**	**0.73 / 0.82**

Note that the very high performance of the 2 Stage VAE(Dai & Wipf, 2019) and the VPGA, LPGA (Zhang et al., 2019) on the CelebA can be, in large part, attributed to the image preprocessing. For example the Dai & Wipf 2019 authors center-crop 108×108 patch and resize it to 64×64. This augmentation removes most of the background which simplifies the generative task.

While FID and KID metrics indicate a similarity between the quality of generated and reference images, they do not explain other important attributes such as the coverage of the generated distribution. To disentangle the FID/KID 1D quality measure we evaluate our method by the Precision/Recall metric (Sajjadi et al., 2018). Precision measures qualitative distance between the generated and reference images, and recall how well the entire reference distribution (e.g. all classes) is represented by the randomly generated images. We set the entire test datasets of respective benchmarks as the reference distributions. In Figure 6 we show the Precision/Recall curves for random images generated by sampling from normal, binarized distribution $\mathcal{N}(0, \mathbf{I})$, the LBAE method, noted as $\mathcal{N}(\mu, \Sigma)$, and VAE(ours) - VAE model with the LBAE architecture. We can see that our method shows consistently higher, balanced precision and recall with the exception of sampling from $\mathcal{N}(0, \mathbf{I})$. In Table 4 we then compare our method with Ghosh et al. 2020. Here, again, the LBAE achieves relatively high precision as well as recall. This signifies that the generated images represent the entire distribution equally well and that the image quality is close to the reference distribution.

In Table 5 we compare our model with Ghosh et al. 2020 results obtained by sampling from a GMM (10 Gaussians) trained on latents encoded on the training data. With the exception of MNIST, our model outperforms the GMM sampling on both FID and Precision/Recall scales. Note

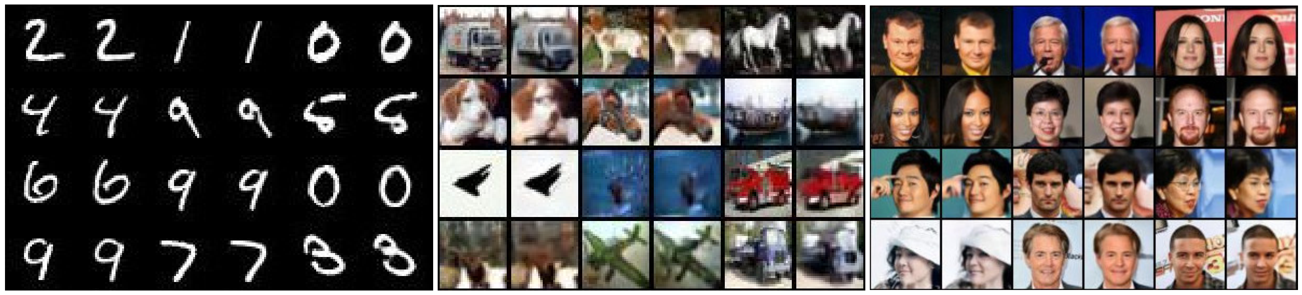


Figure 7. Reconstruction on the MNIST, CIFAR-10 and CelebA test datasets with the LBEA method. The ground truth image on the left is followed by the reconstruction on right.

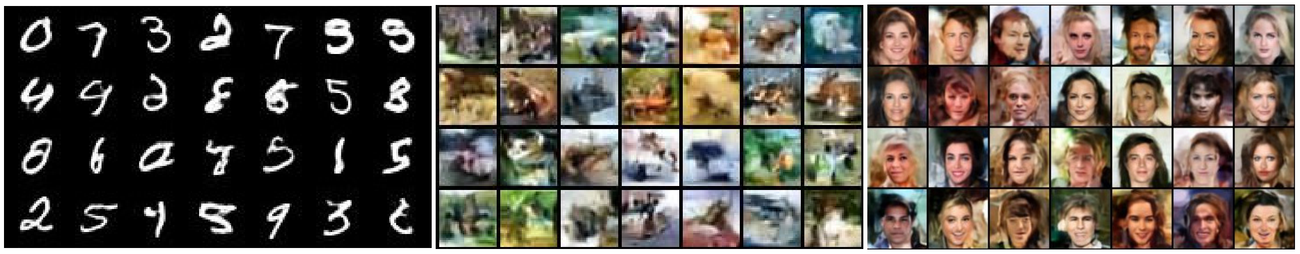


Figure 8. Novel samples generated with the LBAE method.

that the RAE-L2 method with GMM sampling shows lower FID score than on the reconstruction on the test dataset. It is conceivable that latents sampled from GMM, fitted to the training data, are decoded by the RAE-L2 model that overfits on the training data.

4.2. Interpolation in Latent Space

In the Figure 10 we show interpolation between two images over $T = 10$ steps. We can see the interpolation is smooth between the endpoints; there are no abrupt changes in the context nor the image intensities. The composition of the intermediate samples also seems to lie on the path between the endpoints as we intuitively expect. The FID and KID scores for interpolation in Tables 1 and 2 support this observation.

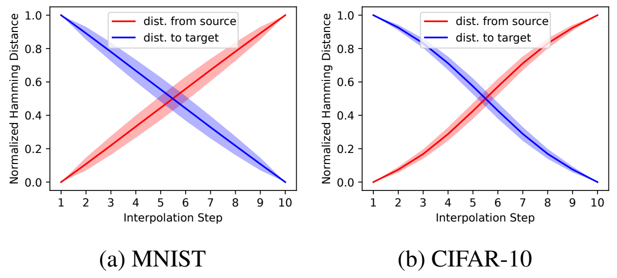


Figure 9. μ and σ of Hamming distance between interpolated latent at step k and source and target latents.

SLERP interpolation given by two endpoints follows the shortest path on the sphere. To understand what path the latents follow in the binary space, we measure Hamming distance between the interpolated latent $\mathbf{b}_k, k \in [1, .., T]$ at step k and the source and target latents. Plot of these distances over 1k interpolations is shown in Figure 9. The Hamming distance is normalized between the source and target latents. We can see that the interpolation in the binary space is almost linear which indicates that the feature manifolds in this space are continuous between the endpoints and that our interpolation method provides a suitable mapping between the binary latent space and the continual space on the sphere.

4.3. Attribute Manipulation in Latent Space

Attributes of the generated samples can be directly modified in the latent space. We demonstrate this on two examples where we add *eyeglasses* or *goatee* CelebA attributes to random test images. This operation does not require the model to be conditionally trained with the attributes, only to collect K latents $\mathbf{Y}^a \in \{-1, 1\}^{(N \times K)}$ with the attribute a and then get the expected value $\mathbf{p} = \mathbb{E}[\mathbf{Y}^a], \mathbf{p} \in \mathbb{R}^N$. Here, the $\mathbf{Y}^a$ are latents encoded on all images with attribute a from the training dataset. To change the attribute a in an image represented by latent $\mathbf{b}$ we set its bits b_i whose expected value p_i is outside threshold D as:

$$b_i = \begin{cases} 1, & \text{if} \quad p_i > D \\ -1, & \text{if} \quad p_i < -D \\ b_i, & \text{otherwise.} \end{cases} \tag{17}$$

The threshold D determines how many bits will be modified, consequently how strongly the source image will be altered. Experimentally we found that $D = 0.1$ provides sat-

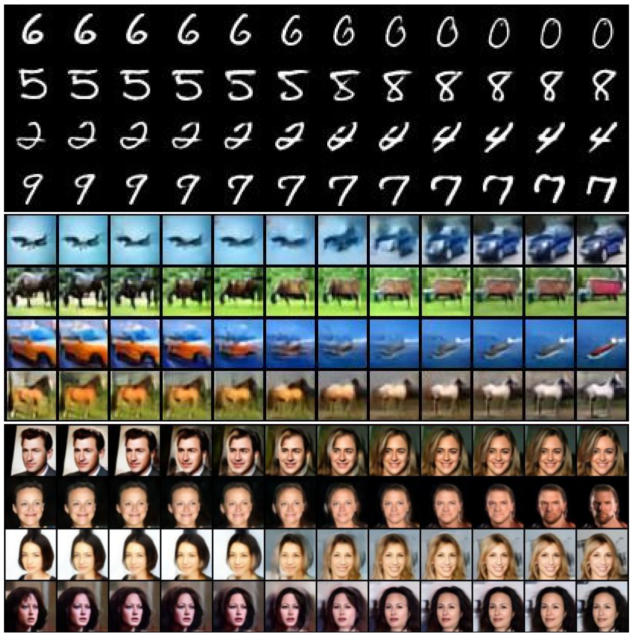


Figure 10. Interpolations between test images from MNIST, CIFAR-10 and CelebA.

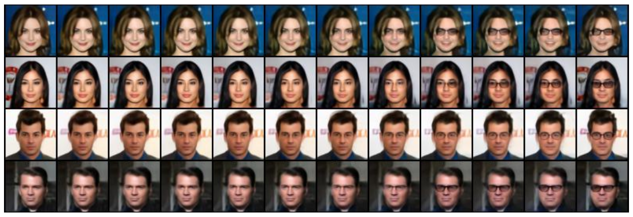
isfactory results and used this value for all our experiments. Interpolation is then performed by the method described in the Section 3.3. Examples of two attributes alterations are shown in Figure 11. More qualitative results are in Supplementary Material, Appendix C.

5. Relation to VQ-VAE

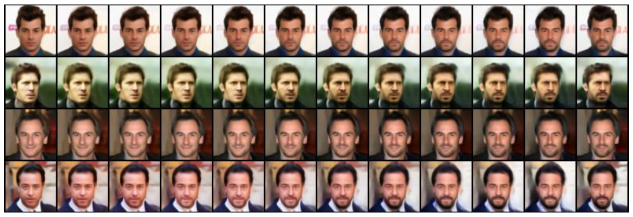
VQ-VAE was introduced by van den Oord et al. 2017 as a discrete, deterministic autoencoder. In a multi-scale, hierarchical organization (Razavi et al., 2019b) it achieves generative performance comparable to GANs.

VQ-VAE learns the discrete representations indirectly, as indexes to a codebook with continual-value embeddings that are then passed to the decoder for reconstruction. The indexes are looked up as nearest codebook neighbours of the encoder output. During training the indexed embeddings in the codebook are moved closer to the encoder output. The non differentiable nearest neighbour operation is replaced with the straight-through gradient estimator in the backward pass. To sample new images authors propose to learn a categorical prior over latents encoded on the training data with PixelCNN (Oord et al., 2016).

LBAE learns the latent codes directly, though we could think of the first fully connected layer in the decoder as a dictionary of embeddings that is implicitly learned. The binary latents, in fact, work as row selectors of the weight matrix, where each row can be considered an embedding vector. Row vectors corresponding to ones in the input are summed together and sent down to the following layers in



(a) Setting the CelebA *eyeglasses* attribute.



(b) Setting the CelebA *goatee* attribute.

Figure 11. Interpolation between test images (left) and the same images (right) with modified attributes.

the decoder. The possibility of training an autoregressive model on this dictionary, in the VQ-VAE fashion, is left for future research.

Unlike LBAE, the VQ-VAE cannot be easily used for interpolation and other operations in the latent space as shown by Berthelot et al. 2019. While VQ-VAE needs to train an external autoregressive model, LBAE can perform the generative tasks in the discrete latent space with its decoder.

6. Conclusion

In this paper, we show that a simple deterministic, discrete latent autoencoder, trained with the straight-through gradient estimator performs on a par with the VAE model, its derivatives and the latest regularized, deterministic autoencoders, on all common tasks such as reconstruction, novel samples generation, interpolation and attribute modification on benchmarks CelebA, CIFAR-10 and MNIST.

We propose a simple, closed form method for sampling from the Bernoulli latent space as well as to perform a smooth interpolation and attribute modification in this space. To our knowledge this is the first successful method that directly learns binary representation of images and allows for smooth interpolation in the discrete latent space.

Our model achieves higher reconstruction as well as generative image quality compared to VAE. Furthermore, our method for random sampling from the latent space covers the entire distribution without over or under representation of any classes, indicating resilience to mode collapse. Equally, on a simple experiment of modifying image attributes we show a potential of the representation power of the Bernoulli latent space.

Acknowledgement

This work has been supported in part by the NATO SPS programme under the funding NATO SPS G5381 MIDAS: *Control of team of mini-UAVs to support counter-terrorism missions*.

References

Bengio, Y., Courville, A. C., and Vincent, P. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1:2012, 2012.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013a.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013b.

Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019.

Bethge, M. and Berens, P. Near-maximum entropy models for binary neural representations of natural images. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 97–104. Curran Associates, Inc., 2008.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018.

Chung, J., Ahn, S., and Bengio, Y. Hierarchical multiscale recurrent neural networks. In *International Conference on Learning Representations*, 2017.

Dai, B. and Wipf, D. Diagnosing and enhancing VAE models. In *International Conference on Learning Representations*, 2019.

Dosovitskiy, A. and Brox, T. Generating images with perceptual similarity metrics based on deep networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 658–666. Curran Associates, Inc., 2016.

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. C. Adversarially learned inference. In *International Conference on Learning Representations*, 2017.

Dupont, E. Learning disentangled joint continuous and discrete representations. In *Advances in Neural Information Processing Systems*, pp. 710–720, 2018.

Ghosh, P., Sajjadi, M. S. M., Vergari, A., Black, M., and Scholkopf, B. From variational to deterministic autoencoders. In *International Conference on Learning Representations*, 2020.

Goemans, M. X. and Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

Hawkins, J. and Ahmad, S. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10:23, 2016. ISSN 1662-5110. doi: 10.3389/fncir.2016.00023.

Hinton, G. Neural networks for machine learning, coursera video lectures. *Coursera*, 2012.

Hou, X., Shen, L., Sun, K., and Qiu, G. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision*, pp. 1133–1141. IEEE, 2017.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *International Conference on Learning Representations*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical Report,University of Toronto, 2009.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

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

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. Are gans created equal? a large-scale study. In *Advances in Neural Information Processing Systems*, pp. 700–709, 2018.

Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016.

Mishra, V. K., Natarajan, K., Tao, H., and Teo, C.-P. Choice prediction with semidefinite optimization when utilities are correlated. *IEEE Transactions on Automatic Control*, 57(10):2450–2463, 2012.

Nirenberg, S. H. and Victor, J. D. Analyzing the activity of large populations of neurons: how tractable is the problem? *Current opinion in neurobiology*, 17(4):397–400, 2007.

Oord, A. V., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR.

Rae, J. W., Hunt, J. J., Harley, T., Danihelka, I., Senior, A., Wayne, G., Graves, A., and Lillicrap, T. P. Scaling memory-augmented neural networks with sparse reads and writes. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 3628–3636, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Razavi, A., van den Oord, A., Poole, B., and Vinyals, O. Preventing posterior collapse with delta-VAEs. In *International Conference on Learning Representations*, 2019a.

Razavi, A., van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pp. 14866–14876, 2019b.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pp. 3859–3869, 2017.

Sajjadi, M. S., Bachem, O., Lucic, M., Bousquet, O., and Gelly, S. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems*, pp. 5228–5237, 2018.

Schneidman, E., Berry, M. J., Segev, R., and Bialek, W. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087): 1007–1012, 2006.

Shlens, J., Field, G. D., Gauthier, J. L., Grivich, M. I., Petrusca, D., Sher, A., Litke, A. M., and Chichilnisky, E. The structure of multi-neuron firing patterns in primate retina. *Journal of Neuroscience*, 26(32):8254–8266, 2006.

Shoemake, K. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pp. 245–254, 1985.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pp. 4790–4798, 2016.

van den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, pp. 2048–2057, 2015.

Zhang, Z., Zhang, R., Li, Z., Bengio, Y., and Paull, L. Perceptual generative autoencoders. In *International Conference on Learning Representations, Workshop DeepGenStruct*, 2019.