Bidirectional Variational Autoencoders

Bart Kosko  
Department of Electrical and Computer Engineering  
University of Southern California  
Los Angeles, CA, USA  
kosko@usc.edu

Olaoluwa Adigun  
Department of Electrical and Computer Engineering  
University of Southern California  
Los Angeles, CA, USA  
adigun@usc.edu

Abstract—We present the new bidirectional variational autoencoder (BVAE) network architecture. The BVAE uses a single neural network both to encode and decode instead of an encoder-decoder network pair. The network encodes in the forward direction and decodes in the backward direction through the same synaptic web. Simulations compared BVAEs and ordinary VAEs on four image tasks of image reconstruction, classification, interpolation, and generation. The image datasets included MNIST handwritten digits, Fashion-MNIST, CIFAR-10, and CelebA-64 face images. The bidirectional structure of BVAEs cut the parameter count by almost 50% and still slightly outperformed the unidirectional VAEs.

Index Terms—variational autoencoder, bidirectional backpropagation, directional likelihoods, ELBO, evidence lower bound

I. DIRECTIONAL VARIATIONAL AUTOENCODERS

This paper introduces the new bidirectional variational autoencoder (BVAE) network. This architecture uses a single parametrized network for encoding and decoding. It trains with the new bidirectional backpropagation algorithm that jointly optimizes the network’s bidirectional likelihood [1], [2]. The algorithm uses the same synaptic weights both to predict the target y given the input x and to predict the converse x given y. Ordinary or unidirectional VAEs use separate networks to encode and decode.

Unidirectional variational autoencoders (VAEs) are unsupervised machine-learning models that learn data representations [3], [4]. They both learn and infer with directed probability models that often use intractable probability density functions [5]. A VAE seeks the best estimate of the data likelihood \( p(x|\theta) \) from samples \( \{ x^{(n)} \}_{n=1}^{N} \) if \( x \) depends on some observable feature \( z \) and if \( \theta \) represents the system parameters. The intractability involves marginalizing out the random variable \( z \) to give the likelihood \( p(x|\theta) \):

\[
p(x|\theta) = \mathbb{E}_{z|\theta}[p(x|z, \theta)] = \int_{z} p(x|z, \theta) p(z|\theta) dz. \tag{1}
\]

Kingma and Welling introduced VAEs to solve this computational problem [5]. The VAE includes a new recognition (or encoding) model \( q(z|x, \phi) \) that approximates the intractable likelihood \( q(z|x, \theta) \). The probability \( q(z|x, \phi) \) represents a probabilistic encoder while \( p(z|x, \theta) \) represents a probabilistic decoder. These probabilistic models use two neural networks with different synaptic weights. Figure 1a shows the architecture of such a unidirectional VAE. The recognition model doubles the number of parameters and the computational cost of this solution.

The new bidirectional backpropagation (B-BP) algorithm trains a neural network to run forwards and backwards by jointly maximizing the respective directional probabilities. This among other things allows such a network to run backward from output code words to expected input patterns. Running a unidirectionally trained network backwards just produces noise. B-BP jointly maximizes the forward likelihood \( q_f(z|x, \theta) \) and backward likelihood \( p_b(x|z, \theta) \) or the equivalent sum of their respective log-likelihoods:

\[
\theta^* = \arg \max_{\theta} q_f(z|x, \theta) p_b(x|z, \theta) \tag{2}
\]

\[
= \arg \max_{\theta} \ln q_f(z|x, \theta) + \ln p_b(x|z, \theta). \tag{3}
\]

A BVAE approximates the intractable \( q(z|x, \theta) \) with the forward likelihood \( q_f(z|x, \theta) \). Then the probabilistic encoder is \( q_f(z|x, \theta) \) and the probabilistic decoder is \( p_b(x|z, \theta) \). So the two densities share parameter \( \theta \) and there is no need for a separate network. Figure 1b shows the architecture of a BVAE. VAEs vary based on the choice of latent distribution, the method of training, and the use of joint modeling with other generative models, among other factors. The \( \beta \)-VAE introduced the adjustable hyperparameter \( \beta \). It balances the latent channel capacity of the encoder network and the reconstruction error of the decoder network [6]. It trains on a weighted sum of the reconstruction error and the Kullback-Leibler divergence \( D_{KL}(q(z|x, \phi)||p(z|\theta)) \). The \( \beta \)-TCVAE (Total Correlation Variational Autoencoder) extends \( \beta \)-VAE to learning isolating sources of disentanglement [7]. A disentangled \( \beta \)-VAE modifies the \( \beta \)-VAE by progressively increasing the information capacity of the latent code while training [8].

Importance weighted autoencoders (IWAEs) use priority weights to derive a strictly tighter lower bound on the log-likelihood [9]. Variants of IWAE include the partially importance weighted auto-encoder (PIWAE), the multiply importance weighted auto-encoder (MIWAE), and the combined importance weighted auto-encoder (CIWAE) [10].

Hyperspherical VAEs use a non-Gaussian latent probability density. They use a von Mises-Fisher (vMF) latent density that
gives in turn a hyperspherical latent space [4]. Other VAEs include the Consistency Regularization for Variational AutoEncoder (CRVAE) [3], the InfoVAE [11], and the Hamiltonian VAE [12] and so on. All these VAEs use separate networks to encode and decode. Vincent et alia [13] suggests the use of tied weights in stacked autoencoder networks. This is a form of constraint that parallels the working of restricted Boltzmann machines RBMs [14] and thus a simple type of bidirectional associative memory or BAM [15]. It forces the weights to be symmetric using $W^T$ on the backward pass. The building block here is a shallow network with no hidden layer [16], [17]. They further suggest that combining this constraint with a nonlinear activation would most likely lead to poor reconstruction error.

Bidirectional autoencoders BAEs [18] extend bidirectional neural representations to image compression and denoising. BAEs differ from autoencoders with tied weights because they relax the constraint by extending the bidirectional assumption over the depth of a deep network. BAEs differ from bidirectional VAEs because they do not require the joint optimization of the directional likelihoods. This limits the generative capability of BAEs.

The next sections review ordinary VAEs and introduce probabilistic BVAEs using the new B-BP algorithm. Section IV compares them on the four standard image test datasets: MNIST handwritten digits, Fashion-MNIST, CIFAR-10, and CelebA-64 datasets. We find that BVAEs cut the number of tunable parameters roughly in half while still performing slightly better than the unidirectional VAEs.

**II. UNIDIRECTIONAL VARIATIONAL AUTOENCODERS**

Let $p(x|\theta)$ denote the data likelihood and $z$ denote the hidden variable. The data likelihood simplifies as

$$p(x|\theta) = \frac{p(x, z|\theta)}{q(z|x, \theta)} = \frac{p(x|z, \theta)p(z|\theta)}{q(z|x, \theta)}. \quad (4)$$

The likelihood $q(z|x, \theta)$ is intractable to solve. So unidirectional VAEs introduce a new likelihood that represents the recognition or encoding model. The term $q_f(z|x, \phi)$ represents the forward likelihood of the encoding network that approximates the intractable likelihood $q(z|x, \theta)$:

$$p(x|\theta) = \frac{p(x, z|\theta)p(z|\theta)}{q(z|x, \theta)} = \frac{p(x, z|\theta)p(z|\theta)q_f(z|x, \phi)}{q(z|x, \theta)q_f(z|x, \phi)}. \quad (5)$$

The corresponding data log-likelihood $\ln p(x|\theta)$ is

$$\ln p(x|\theta) = \ln \left[ \frac{p(x, z, \theta)p(z, \theta)q_f(z|x, \phi)}{q(z|x, \theta)q_f(z|x, \phi)} \right] = \ln p(x|z, \theta) + \ln \frac{p(z|\theta)}{q_f(z|x, \phi)} + \ln \frac{q_f(z|x, \phi)}{q(z|x, \theta)} \quad (6)$$

$$= \ln p(x|z, \theta) - \ln \frac{q_f(z|x, \phi)}{p(z|\theta)} + \ln \frac{q_f(z|x, \phi)}{q(z|x, \theta)}. \quad (7)$$

Now take the expectation of (8) with respect to $q_f(z|x, \phi)$:

$$E_{z|x,\phi} \left[ \ln p(x|\theta) \right] = \int_z q_f(z|x, \phi) \ln p(x|\theta) \, dz = \ln p(x|\theta) \int_z q_f(z|x, \phi) \, dz \quad (9)$$

$$= \ln p(x|\theta) \quad (10)$$

because $q_f(z|x, \phi)$ is a probability density function and its integral over the domain of $z$ equals 1. The expectation of the term on the right-hand side of (8) with respect $q_f(z|x, \phi)$ is

$$E_{z|x,\phi} \left[ \ln p(x|z, \theta) - \ln \frac{q_f(z|x, \phi)}{p(z|\theta)} + \ln \frac{q_f(z|x, \phi)}{q(z|x, \theta)} \right] = E_{z|x,\phi} \left[ \ln p(x|z, \theta) \right] - D_{KL}(q_f(z|x, \phi)||p(z|\theta)) + D_{KL}(q_f(z|x, \phi)||q(z|x, \theta)) \quad (12)$$

where

$$D_{KL}(q_f(z|x, \phi)||p(z|\theta)) = E_{z|x,\phi} \left[ \ln \frac{q_f(z|x, \phi)}{p(z|\theta)} \right] \quad (13)$$

![Diagram of Unidirectional VAE architecture](image1)

**Fig. 1:** Bidirectional vs. unidirectional variational autoencoders: Unidirectional VAEs use the forward passes of two separate networks for encoding and decoding. Bidirectional VAEs run their encoding on the forward pass and decoding on the backward pass with the same synaptic webs–weight matrices in both directions. This cuts the number of tunable parameters roughly in half. (a) The decoder network with parameter $\theta$ approximates $p(x|z, \theta)$ and the encoder network with parameter $\phi$ approximates $q(z|x, \theta)$. (b) Bidirectional VAEs use the forward pass of a network with parameter $\theta$ to approximate $q(z|x, \theta)$ and the backward pass of the network to approximate $p(x|z, \theta)$. 

(a) Unidirectional VAE architecture

(b) Bidirectional VAE architecture
because the negative of the natural logarithm is convex. So
\[
\ln p(x|\theta) \geq \mathbb{E}_{z|x,\phi} \left[ \ln p(x|z,\theta) \right] - D_{KL} \left( q_f(z|x,\phi) \| p(z|\theta) \right) \]
\[
\mathcal{L}(x,\theta,\phi)
\]
(22)

and
\[
\mathcal{L}(x,\theta,\phi) = \mathbb{E}_{z|x,\phi} \left[ \ln p(x|z,\theta) \right] - D_{KL} \left( q_f(z|x,\phi) \| p(z|\theta) \right)
\]
(23)

where \(\mathcal{L}(x,\theta,\phi)\) is the evidence lower bound (ELBO) on the data log-likelihood \(p(x|\theta)\).

Unidirectional VAEs train on the estimate \(\hat{\mathcal{L}}_{\text{ELBO}}(x,\theta,\phi)\) of the ELBO using ordinary or unidirectional backpropagation (BP). This estimate involves using the forward pass \(q_f(z|x,\phi)\) to approximate the intractable encoding model \(q(z|x,\theta)\) and the forward pass \(p_f(x|z,\theta)\) to approximate the encoding model. The gradient update rules for the encoder and decoder networks at the \((n+1)^{th}\) iteration or training epoch are

\[
\theta^{(n+1)} = \theta^{(n)} + \eta \nabla_\theta \hat{\mathcal{L}}_{\text{ELBO}}(x,\theta,\phi) \bigg|_{\theta=\theta^{(n)},\phi=\phi^{(n)}}
\]
(24)

\[
\phi^{(n+1)} = \phi^{(n)} + \eta \nabla_\phi \hat{\mathcal{L}}_{\text{ELBO}}(x,\theta,\phi) \bigg|_{\theta=\theta^{(n)},\phi=\phi^{(n)}}
\]
(25)

where \(\eta\) is the learning rate, \(\phi^{(n)}\) is the encoder parameter, and \(\theta^{(n)}\) is the decoder parameter after \(n\) training iterations.
The expectation of the right-hand term is
\[
\E_{z|x, \theta} \left[ \ln p(x|z, \theta) \right] - D_{KL}(q_f(z|x, \theta) || p(z|\theta)) + D_{KL}(q_f(z|x, \theta) || q(z|x, \theta)) \]

The log-likelihood of the BVAE is such that
\[
\ln p(x|\theta) \geq \E_{z|x, \theta} \left[ \ln p(x|z, \theta) \right] - D_{KL}(q_f(z|x, \theta) || p(z|\theta)) + D_{KL}(q_f(z|x, \theta) || q(z|x, \theta)) \]

because the KL-divergence \( D_{KL}(q_f(z|x, \theta) || q(z|x, \theta)) \geq 0 \). So
\[
\mathcal{L}(x, \theta) = \E_{z|x, \theta} \left[ \ln p(x|z, \theta) \right] - D_{KL}(q_f(z|x, \theta) || p(z|\theta)) + D_{KL}(q_f(z|x, \theta) || q(z|x, \theta)) \]

where \( \mathcal{L}(x, \theta) \) is the ELBO on \( \ln p(x|\theta) \) and we take the expectation \( \E_{z|x, \theta} \) with respect to \( q_f(z|x, \theta) \).

Bidirectional VAEs train on the estimate \( \tilde{\mathcal{L}}_{\text{ELBO}}(x, \theta) \) of the ELBO that uses bidirectional neural representation [2]. This estimate involves using the forward pass \( f(z|x, \theta) \) to approximate the intractable encoding model \( q(z|x, \theta) \) and the reverse pass \( p_b(z|x, \theta) \) to approximate the decoding model.

The update rule at the \( (n+1)^{th} \) iteration or training epoch is
\[
\theta^{(n+1)} = \theta^{(n)} + \eta \nabla_{\theta} \tilde{\mathcal{L}}_{\text{ELBO}}(x, \theta) \bigg|_{\theta=\theta^{(n)}}
\]

where \( \eta \) is the learning rate and \( \theta^{(n)} \) is the autoencoder network parameter just after the \( n^{th} \) training iteration. Figure 2 shows the probabilistic approximation of a BVAE with the directional likelihoods of a bidirectional network.

## IV. Simulations

We compared the performance of unidirectional VAEs and bidirectional VAEs using different tasks, datasets, network architectures, and loss functions. We first describe the image test sets for our experiments.

### A. Datasets

The simulations compared results on four standard image datasets: MNIST handwritten digits [19], Fashion-MNIST [20], CIFAR-10 [21], and CelebA [22] datasets.

The MNIST handwritten digit dataset contains 10 classes of handwritten digits \( \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \). This dataset consists of 60,000 training samples with 6,000 samples per class, and 10,000 test samples with 1,000 samples per class. Each image is a single-channel image with dimension \( 28 \times 28 \).

The Fashion-MNIST dataset is a database of fashion images. It is made of 10 classes namely ankle boot, bag, coat, dress, pullover, sandal, shirt, sneaker, trouser, and t-shirt / top. Each class has 6,000 training samples and 1,000 testing samples. Each image is also a single-channel image with dimension \( 28 \times 28 \).

The CIFAR-10 dataset consists of 60,000 color images from 10 categories. Each image has size \( 32 \times 32 \times 3 \). The 10 pattern categories are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class consists of 5,000 training samples and 1,000 testing samples.

The CelebA dataset is a large-scale face dataset of 10,177 celebrities [22]. This dataset is made up of 202,599 color
Fig. 3: Bidirectional VAE with residual network architecture: This cuts the tunable parameters roughly in half compared with unidirectional VAEs. (a) is the bidirectional convolutional layer. Convolution runs in the forward pass and convolution transpose runs in reverse with the same set of convolution masks. (b) is the architecture of a bidirectional residual block with bidirectional skip connections.

B. Tasks

We compared the performance of bidirectional VAEs and unidirectional VAEs on the following four tasks.

1) Image compression and reconstruction: We tested the self-mapping of the image datasets with unidirectional and bidirectional VAEs. This involved the encoding of images with latent variable $z$ and the subsequent decoding to reconstruct the image after the latent sampling. We evaluated the performance of VAEs on this task using the Peak Signal-to-Noise Ratio (PSNR), Fréchet Inception Distance (FID) [23], [24], and Structural Similarity Index Measure (SSIM) [25].

2) Downstream image classification: We trained simple classifiers on the latent space features of VAEs. These VAEs compress the input images and a simple classifier maps the latent features to their corresponding classes. We evaluated the classification accuracy of this downstream classification.

The VAE-extracted features for the MNIST handwritten dataset trained on simple linear classifiers. We used a neural classifier with one hidden layer of 256 logistic hidden neurons to classify the VAE-extracted features from the Fashion-MNIST and CIFAR-10 datasets.

We used the $t$-distributed stochastic neighbor embedding ($t$-SNE) method to visualize the reduced features. This method uses a statistical approach to map the high-dimensional representation of data $\{x_i\}_{i=1}^N$ to their respective low-dimensional
TABLE I: MNIST handwritten digits dataset with VAEs. We used the residual network architecture. The dimension of the latent variable is 64. The BVAEs each used 42.2 MB storage memory and the VAEs each used 84.4 MB storage memory.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Generative Task</th>
<th>Reconstruction and Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAE</td>
<td>22.1M</td>
<td>NLL ↓ 86.72, AU 18, FID ↓ 4.340</td>
<td>PSNR ↑ 20.69, SSIM ↑ 0.8959, rFID ↓ 3.610, Accuracy ↑ 94.97%</td>
</tr>
<tr>
<td>Bidirectional VAE</td>
<td>11.1M</td>
<td>NLL ↓ 86.09, AU 15, FID ↓ 3.370</td>
<td>PSNR ↑ 20.64, SSIM ↑ 0.8988, rFID ↓ 3.050, Accuracy ↑ 95.26%</td>
</tr>
<tr>
<td>β-VAE (β = 0.5)</td>
<td>22.1M</td>
<td>NLL ↓ 88.12, AU 23, FID ↓ 2.021</td>
<td>PSNR ↑ 22.19, SSIM ↑ 0.9506, rFID ↓ 1.711, Accuracy ↑ 93.15%</td>
</tr>
<tr>
<td>Bidirectional β-VAE (β = 0.5)</td>
<td>11.1M</td>
<td>NLL ↓ 88.09, AU 26, FID ↓ 2.034</td>
<td>PSNR ↑ 22.39, SSIM ↑ 0.9332, rFID ↓ 1.702, Accuracy ↑ 93.32%</td>
</tr>
<tr>
<td>β-VAE (β = 1.5)</td>
<td>22.1M</td>
<td>NLL ↓ 86.99, AU 15, FID ↓ 4.855</td>
<td>PSNR ↑ 19.75, SSIM ↑ 0.8726, rFID ↓ 4.370, Accuracy ↑ 95.20%</td>
</tr>
<tr>
<td>Bidirectional β-VAE (β = 1.5)</td>
<td>11.1M</td>
<td>NLL ↓ 87.21, AU 15, FID ↓ 5.589</td>
<td>PSNR ↑ 19.80, SSIM ↑ 0.8736, rFID ↓ 5.011, Accuracy ↑ 96.19%</td>
</tr>
<tr>
<td>β-TCVAE (β = 0.5)</td>
<td>22.1M</td>
<td>NLL ↓ 88.18, AU 24, FID ↓ 1.975</td>
<td>PSNR ↑ 22.21, SSIM ↑ 0.9299, rFID ↓ 1.662, Accuracy ↑ 93.37%</td>
</tr>
<tr>
<td>Bidirectional β-TCVAE (β = 0.5)</td>
<td>11.1M</td>
<td>NLL ↓ 87.99, AU 26, FID ↓ 2.000</td>
<td>PSNR ↑ 22.30, SSIM ↑ 0.9325, rFID ↓ 1.760, Accuracy ↑ 94.17%</td>
</tr>
<tr>
<td>β-TCVAE (β = 1.0)</td>
<td>22.1M</td>
<td>NLL ↓ 86.87, AU 18, FID ↓ 3.544</td>
<td>PSNR ↑ 20.68, SSIM ↑ 0.8988, rFID ↓ 3.143, Accuracy ↑ 95.08%</td>
</tr>
<tr>
<td>Bidirectional β-TCVAE (β = 1.0)</td>
<td>11.1M</td>
<td>NLL ↓ 86.68, AU 18, FID ↓ 3.419</td>
<td>PSNR ↑ 20.75, SSIM ↑ 0.9007, rFID ↓ 3.085, Accuracy ↑ 95.66%</td>
</tr>
<tr>
<td>β-TCVAE (β = 1.5)</td>
<td>22.1M</td>
<td>NLL ↓ 87.20, AU 14, FID ↓ 5.625</td>
<td>PSNR ↑ 19.64, SSIM ↑ 0.8600, rFID ↓ 5.052, Accuracy ↑ 95.56%</td>
</tr>
<tr>
<td>Bidirectional β-TCVAE (β = 1.5)</td>
<td>11.1M</td>
<td>NLL ↓ 86.90, AU 15, FID ↓ 4.968</td>
<td>PSNR ↑ 19.86, SSIM ↑ 0.8756, rFID ↓ 4.548, Accuracy ↑ 96.54%</td>
</tr>
<tr>
<td>IWAE</td>
<td>22.1M</td>
<td>NLL ↓ 86.02, AU 20, FID ↓ 2.958</td>
<td>PSNR ↑ 19.30, SSIM ↑ 0.8627, rFID ↓ 2.774, Accuracy ↑ 96.40%</td>
</tr>
<tr>
<td>Bidirectional IWAE</td>
<td>11.1M</td>
<td>NLL ↓ 86.00, AU 19, FID ↓ 3.330</td>
<td>PSNR ↑ 19.21, SSIM ↑ 0.8587, rFID ↓ 3.030, Accuracy ↑ 96.29%</td>
</tr>
</tbody>
</table>

Algorithm 1: BVAE Training with bidirectional backpropagation

**Input:** Data \(\{x_n\}_{n=1}^N\) and latent space dimension \(J\).

**Initialize:** Synaptic weights \(\theta \in \{\mathbf{W}_\theta, \mathbf{V}_\theta, \mathbf{W}_\theta\}\), learning rate \(\alpha\), and other hyper-parameters.

1. for iteration \(t = 1, 2, \ldots\) do
2. Pick a mini-batch \(\{x_m\}_{m=1}^B\) of \(B\) samples
3. for \(m = 1, 2, \ldots, B\) do
4. **Forward Pass (Encoding):** Predict the variational mean and log-covariance:
   \[
   \mu_m = \mathbf{W}_\theta \mathbf{N}_\theta (x_m)
   \]
   and
   \[
   \ln \sigma_m^2 = \mathbf{V}_\theta \mathbf{N}_\theta (x_m)
   \]
5. Sample the latent features \(z_m\) from the variational Gaussian distribution with condition \(x_m\):
   \[
   z_m \sim \mathcal{N}(\mu_m, \sigma_m)
   \]
6. **Backward Pass (Decoding):** Map the latent variable back to the input space:
   \[
   \hat{\theta}(m) = \mathbf{N}_\theta^T \left( \mathbf{W}_\theta^T (z_m) \right)
   \]
7. end for
8. Estimate the negative log-likelihood NLL \((x, \theta)\):
   \[
   \text{KLD}(x, \theta) = -\frac{1}{B} \sum_{m=1}^B \mathcal{D}_{KL}(\mathcal{N} (\mu_m, \operatorname{Diag}(\sigma_m^2)) || \mathcal{N}(0, I))
   \]
   \[
   \text{BCE}(x, \theta) = -\frac{1}{B} \sum_{m=1}^B \ln p_\theta(x_m | z_m, \theta)
   \]
   \[
   \hat{\mathcal{L}}_{\text{ELBO}}(x, \theta) = \text{BCE}(x, \theta) + \text{KLD}(x, \theta)
   \]
9. Update \(\theta\) by backpropagating \(\hat{\mathcal{L}}_{\text{ELBO}}(x, \theta)\) through the weights.
10. end for
11. return \(\theta\)

representation \(\{y_i\}_{i=1}^N\) based on the similarity of the data-points [26]. This low-dimensional representation provides insight into the degree of separability among the classes.

3) **Image generation:** We compared the generative performance of bidirectional VAEs with their corresponding unidirectional ones. These VAEs trained with the Gaussian latent distribution \(\mathcal{N}(0, I)\). We tested the VAEs using both Gaussian sampler and Gaussian mixture model samplers post-training [27]. We used the estimates of the negative of the data log-likelihood (NLL) and the number of active latent units (AU) [28] as the quantitative metrics for performance evaluation.

4) **Image interpolation:** We conducted linear interpolation of samples. The interpolations involve a convex combination of two images over 10 steps. The encoding step transforms the mixture of two images in the latent space. The decoding step reconstructs the interpolated samples.

C. Model Architecture

We used different neural network architectures for various datasets and tasks.

Variational Autoencoders: We used deep convolutional and residual neural network architectures. VAEs that trained on the MNIST handwritten and Fashion-MNIST used the residual architecture. Figure 3c shows the architecture of the bidirectional residual networks that trained on the MNIST datasets. The unidirectional VAEs with this architecture used two such networks each: one for encoding and the other for decoding. The BVAEs used just one of such network each. Encoding runs in the forward pass and decoding runs in the backward pass.

Each of the encoder and decoder networks that trained on the CIFAR-10 dataset used six convolutional layers and two fully connected layers. The corresponding BVAEs used only one network for encoding and decoding each. The dimension of the hidden convolutional layers is \{64 ↔ 128 ↔ 256 ↔ 512 ↔ 1024 ↔ 2048\}. The dimension of the fully connected layers is \{2048 ↔ 1024 ↔ 64\}.

The configuration of the VAEs that trained on the CelebA dataset differs slightly. The sub-networks each used nine convolutional layers and two fully connected layers. The dimension of the hidden convolutional layers is \{128 ↔
128 ↔ 192 ↔ 256 ↔ 384 ↔ 512 ↔ 768 ↔ 1024 ↔ 1024}. The dimension of the fully connected layers is \{4096 ↔ 2048 ↔ 256\}.

The VAEs used generalized nonvanishing (G-NoVa) hidden neurons [29], [30]. The G-NoVa activation \(a(x)\) of input \(x\) is

\[
a(x) = \alpha x + x \sigma(\beta x) = \alpha x + \frac{x}{1 + e^{-\beta x}} \quad (40)
\]

where \(\alpha > 0\) and \(\beta > 0\). Each layer of a BVAE performs probabilistic inference in both the forward and backward passes. The convolutional layers use bidirectional kernels. The kernels run convolution in the forward pass and transposed convolution in the backward pass. Transposed convolution projects feature maps to a higher-dimensional space [31].

**Downstream Classification:** We trained simple linear classifiers on VAE-extracted features from the MNIST digit dataset. We trained shallow neural classifiers with one hidden layer and 100 hidden neurons each on the extracted features from the Fashion-MNIST images. Similar neural classifiers with one hidden layer and 256 hidden neurons each trained on VAE-extracted features from the CIFAR-10 dataset.

**D. Training**

We considered four implementations of VAEs and compare them with their respective bidirectional versions. The four VAEs are vanilla VAE [5], \(\beta\)-VAE [6], \(\beta\)-TCVAE [7], and IWAE [9]. We trained these over the four datasets across the four tasks. We used the AdamW optimizer [32] with the OneCycleLR [33] learning rate scheduler. The optimizer trained on their respective ELBO estimates.

We designed a new framework for bidirectional VAEs and implemented unidirectional VAEs with the Pythae framework [34]. All the models trained on a single A100 GPU. Tables I - IV and Figures 4 - 6 present the results.

**E. Evaluation Metrics**

We considered the performance of the VAE models on generative and compression tasks. We used the following quantitative metrics:

- **Negative Log-Likelihood (NLL):** This is an estimate of the negative of \(\ln p(x|\theta)\). This is computational intractable so we used the Monte Carlo method to estimate this. Lower value means the model generalizes well to unseen data.
- **Number of Active Latent Units (AU) [28]:** This reflects the number of latent variables with variance above a given threshold \(\epsilon\). We have

\[
AU = \sum_{d=1}^{D} \mathbb{I}\left[\text{Cov}\left[\mu_x|x,\phi|,\sigma_z\right] \geq \epsilon\right] \quad (41)
\]

where \(\mathbb{I}\) is an indicator function, \(z_d\) represents the \(d^{th}\) component of the latent variable, and \(\epsilon = 0.01\). Higher AU means the model uses more features for the latent space representation. But having too many active units can lead to overfitting.

- **Peak Signal-to-Noise Ratio (PSNR):** This compares re-constructed images with their target images. Higher value implies better reconstruction from data compression.
- **Structural Similarity Index (SSIM):** This is a perceptual metric. It quantifies the degradation from data compression. Higher SSIM value implies better reconstruction from image compression.
- **Downstream Classification Accuracy:** This is the classification accuracy of simple classifiers that trained on the latent or VAE-extracted features. Higher accuracy means the compression extracts easy-to-classify features.
- **Fréchet Inception Distance (FID) [23]:** This metric evaluates the quality of generated images. It measures the similarity between the distribution of real images and the distribution of generated images. Lower value implies that the generated images are closer to the real images.
Bidirectional VAEs encode and decode through the same synaptic web of a deep neural network. This bidirectional flow captures the joint probabilistic structure of both directions during learning and recall. BVAEs cut the synaptic parameter
TABLE IV: CelebA-64 dataset with VAEs. The dimension of the latent variable is 256. The BVAEs each used 133.8MB memory parameters and the unidirectional VAEs each used 267.6MB memory parameters.

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Generative Task NLL ↓ AU</th>
<th>Reconstruction PSNR ↑ SSIM↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAE</td>
<td>69.1M</td>
<td>6221.9 80</td>
<td>20.64 0.652</td>
</tr>
<tr>
<td>Bidirectional VAE</td>
<td>34.6M</td>
<td>6217.7 542</td>
<td>20.66 0.653</td>
</tr>
<tr>
<td>β-VAE (β = 0.1)</td>
<td>69.1M</td>
<td>6243.0 240</td>
<td>22.35 0.7225</td>
</tr>
<tr>
<td>Bidirectional β-VAE</td>
<td>34.6M</td>
<td>6261.9 90</td>
<td>22.60 0.7243</td>
</tr>
<tr>
<td>β-TCVAE (β = 0.1)</td>
<td>69.1M</td>
<td>6277.2 93</td>
<td>22.15 0.7189</td>
</tr>
<tr>
<td>Bidirectional β-TCVAE</td>
<td>34.6M</td>
<td>6275.2 145</td>
<td>23.55 0.7681</td>
</tr>
<tr>
<td>β-IWA</td>
<td>69.1M</td>
<td>6221.1 77</td>
<td>20.44 0.6521</td>
</tr>
<tr>
<td>Bidirectional β-IWA</td>
<td>34.6M</td>
<td>6217.5 204</td>
<td>20.81 0.6551</td>
</tr>
</tbody>
</table>

count roughly in half compared with unidirectional VAEs. The simulations on the four image test sets showed that the BVAEs still performed slightly better than the unidirectional VAEs.

REFERENCES


