Bidirectional Backpropagation Autoencoding Networks for Image Compression and Denoising

Olaoluwa Adigun
Signal and Image Processing Institute
Department of Electrical and Computer Engineering
Los Angeles, California 90089-2564.
adigun@usc.edu

Bart Kosko
Signal and Image Processing Institute
Department of Electrical and Computer Engineering
Los Angeles, California 90089-2564.
kosko@usc.edu

***Abstract*—A bidirectional autoencoder learns or approximates an identity mapping as it trains a single network with a version of the new bidirectional backpropagation algorithm. Ordinary unidirectional autoencoders find many uses in image processing and in large language models. But they use separate networks for encoding and decoding. Bidirectional autoencoders use the same synaptic weights for encoding and decoding. The forward pass encodes while the backward pass decodes. Bidirectional autoencoders improved network performance and significantly reduced memory usage and used fewer parameters. Simulations compared unidirectional with bidirectional autoencoders for image compression and denoising. The models trained on the MNIST handwritten-digit and CIFAR-10 image datasets. The performance measures were the peak signal-to-noise ratio and the index of structural similarity. Bidirectional autoencoders outperformed unidirectional autoencoders and still reduced the number of trainable synaptic parameters by about 50%.**

***Index Terms*—Bidirectional backpropagation, autoencoders, bidirectional networks, large language models, bidirectional autoencoders, variational autoencoders.**

I. Bidirectional Autoencoders

We show how the new *bidirectional* backpropagation algorithm can train an autoencoder (AE) network. The resulting bidirectional AE uses a single network and the same synapses for forward and backward passes. Ordinary unidirectional AEs use separate networks for encoding and decoding.

AE networks themselves learn or approximate identity mappings from unlabeled data or patterns. AEs can compress or summarize patterns or text. This lets them generate somewhat new patterns from old patterns. They can combine with large-language models (LLMs) such as chat-AI GPTs to improve the performance of LLMs [1]–[3]. AEs also apply to a wide range of problems in data compression or dimension reduction [4]–[6], image denoising [7]–[11], feature extraction [12], anomaly detection [13]–[16], collaborative filtering [17], [18], and sentiment analysis [19]–[21].

Figure 1 shows the bidirectional architecture of a *bidirectional* AE (BAE). The BAE takes as input the noisy pattern $\mathbf{x}$ and passes it through the network N_θ to produce the encoded vector $\mathbf{z}$. The decoding step passes $\mathbf{z}$ back through the transposed synaptic weight matrices of N_θ.

Figure 2 compares the architecture of unidirectional AEs with the new BAEs. A unidirectional AE needs a separate network for decoding while a BAE uses the same network for encoding and decoding. Figure 3 shows examples of the discretized beta probability densities that model the output layer of autoencoders. Algorithm 1 gives the pseudocode for training BAEs with a simple form of the new bidirectional backpropagation algorithm.

Bidirectional backpropagation [22], [23] maximizes a network's joint likelihood $p_f(\mathbf{y}|\mathbf{x},\theta)p_b(\mathbf{x}|\mathbf{y},\theta)$. The forward probability $p_f(\mathbf{y}|\mathbf{x},\theta)$ describes the forward pass of input pattern $\mathbf{x}$ from the input layer to the output layer. The backward probability $p_b(\mathbf{x}|\mathbf{y},\theta)$ describes the backward pass of target $\mathbf{y}$ from the output layer to the input layer. BAE training differs from bidirectional backpropagation because it maximizes just the network's backward likelihood.

Figure 4 shows that BAEs increase the peak signal-to-noise ratio (PSNR) on tasks of image compression and denoising using the MNIST handwritten digit dataset. Table I shows the BAE gain in PSNR. BAEs reduced the parameter count by about 50% on the image-compression task. Table II shows the BAE gain in PSNR on denoising images corrupted with additive noise and with multiplicative noise.

Table III compares unidirectional AEs and BAEs on image compression with the CIFAR-10 image dataset. The table shows that BAEs increase the PSNR, they increase the structural similarity index measure (SSIM), and they reduce the parameter count. Table IV shows a similar bidirectional benefit on denoising the CIFAR-10 image dataset.

Figure 5 compares the encoded or compressed features projected onto a 2D space. The projection used the t-distributed stochastic neighboring embedding (t-SNE) method [24]. The projected bidirectionally encoded features separated more easily into their respective categories.

Preliminary results also showed that the bidirectional backpropagation architecture extends to variational autoencoders. Here the forward error measures the Kullback-Leibler divergence between the encoded vector $\mathbf{z}$ and a target prior probability. The backward error measures the reconstruction error. The bidirectional framework offers a simple alternative to the reparameterization trick in variational AEs [25].

The next section presents the unidirectional AEs used in our simulations.

1946-0759/23/$31.00 ©2023 IEEE
DOI 10.1109/ICMLA58977.2023.00107

II. Unidirectional Autoencoders

An ordinary or unidirectional AE consists of two contiguous networks. These are the encoder network and the decoder network.

Figure 2 shows the architecture of such an autoencoder. The terms θ and ϕ denote the respective weights of the encoder network N_θ and the decoder network N_ϕ. The encoder has output activation $\mathbf{a}^z = N_\theta(\mathbf{x})$ where $\mathbf{x}$ is the input vector or signal. The decoder has output activation $\mathbf{a}^y = N_\phi(\mathbf{a}^z)$. It gives the *reconstructed* input or signal where $\mathbf{a}^z$ is the *encoded* signal.

We can model the output layer of neural networks for image-related tasks as discretized independent *beta* random variables $Y_1, ..., Y_K$. Ma and Leijon [26] have found that the beta probability density gives a reasonable model for such image pixel values. The random variables are $Y_k|_{X=\mathbf{x}} \sim$ Beta($\alpha = 1 + y_k$, $\beta = 2 - y_k$) where we discretize the beta density. This choice of the two beta parameters coincides with a *continuous* Bernoulli [27], [28]. This discretization models the finite cardinality of the set of all pixel values. Y_k denotes the k^{th} neuron or pixel at the output layer of the decoder. It has the target pixel value $y_k = \frac{c}{255}$ for some $c \in \{0, 1, ..., 255\}$.

The pixel values are not continuous and again the support of the beta structure assumes discretized values. This allows multi-level representation with 2 or more levels and assists image representation because it gives a multi-level model for the 256 possible values per pixel.

The decoder's output negative log-likelihood equals the double cross-entropy between the output activation $\mathbf{a}^y$ and the target $\mathbf{y}$. This gives the output likelihood $p(y_k|\mathbf{x}, \theta, \phi)$ as

$$p(y_k|\mathbf{x},\theta,\phi) = \text{Beta}(a_k^y|\alpha = 1 + y_k, \beta = 2 - y_k) \tag{1}$$

$$= \frac{\Gamma(3)}{\Gamma(1+y_k)\Gamma(2-y_k)}(a_k^y)^{(y_k)}(1-a_k^y)^{(1-y_k)}. \tag{2}$$

The corresponding log-likelihood is

$$l(y_k|\mathbf{x},\theta,\phi) = \ln p(y_k|\mathbf{x},\theta,\phi) \tag{3}$$

$$= \ln \frac{2}{\Gamma(1+y_k)\Gamma(2-y_k)}(a_k^y)^{(y_k)}(1-a_k^y)^{(1-y_k)} \tag{4}$$

$$= \ln 2 - \ln\Gamma(1+y_k) - \ln\Gamma(2-y_k) + y_k \ln a_k^y + (1-y_k)\ln(1-a_k^y) \tag{5}$$

$$= \psi(y_k) + y_k \ln a_k^y + (1-y_k)\ln(1-a_k^y) \tag{6}$$

where $\psi(y_k) = \ln 2 - \ln\Gamma(1+y_k) - \ln\Gamma(2-y_k)$. Then the negative log-likelihood simplifies as

$$-l(y_k|\mathbf{x},\theta,\phi) = -\ln p(y_k|\mathbf{x},\theta,\phi) \tag{7}$$

$$= -\psi(y_k) - y_k \ln a_k^y - (1-y_k)\ln(1-a_k^y) \tag{8}$$

$$= -\psi(y_k) + \mathcal{E}(y_k, a_k^y, \theta, \phi) \tag{9}$$

where $\mathcal{E}(y_k, a_k^y, \theta, \phi)$ is the double cross-entropy between y_k and a_k^y.

Unidirectional or ordinary backpropagation (BP) trains the AE. This gradient method finds the model weights θ^* and ϕ^* that locally maximize the decoder's output likelihood. This just minimizes the double cross-entropy:

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} p(\mathbf{y}|\mathbf{x},\theta,\phi) \tag{10}$$

$$= \arg\max_{\theta,\phi} \ln p(\mathbf{y}|\mathbf{x},\theta,\phi) \tag{11}$$

$$= \arg\min_{\theta,\phi} -\ln p(\mathbf{y}|\mathbf{x},\theta,\phi) \tag{12}$$

$$= \arg\min_{\theta,\phi} \mathcal{E}(\mathbf{y}, \mathbf{a}^y, \theta, \phi) \tag{13}$$

because the logarithm is a monotonic function and because $-\psi(y_k)$ does not depend on θ or ϕ.

Unidirectional BP trains on only the forward error $\mathcal{E}_M(\mathbf{y}, \mathbf{a}^y, \theta, \phi)$ over M training samples $\{\mathbf{x}^{(m)}\}_{m=1}^M$. This forward error simplifies as

$$\mathcal{E}_M(\mathbf{y}, \mathbf{a}^y, \theta, \phi) = \sum_{m=1}^{M} \mathcal{E}(\mathbf{y}^{(m)}, \mathbf{a}^{y(m)}, \theta, \phi) \tag{14}$$

$$= \sum_{m=1}^{M}\sum_{k=1}^{K} \mathcal{E}(y_k^{(m)}, a_k^{y(m)}, \theta, \phi) \tag{15}$$

$$= -\sum_{m=1}^{M}\sum_{k=1}^{K} \Big(y_k^{(m)} \ln\left(a_k^{y(m)}\right) + \left(1-y_k^{(m)}\right)\ln\left(1-a_k^{y(m)}\right)\Big). \tag{16}$$

The corresponding log-likelihood is

$$\ln p_M(\mathbf{y}|\mathbf{x},\theta,\phi) = \ln \prod_{m=1}^{M} p(\mathbf{y}^{(m)}|\mathbf{x}^{(m)},\theta,\phi) \tag{17}$$

$$= \sum_{m=1}^{M} \ln \prod_{k=1}^{K} p(y_k^{(m)}|\mathbf{x}^{(m)},\theta,\phi) \tag{18}$$

$$= \sum_{m=1}^{M}\sum_{k=1}^{K} \ln p(y_k^{(m)}|\mathbf{x}^{(m)},\theta,\phi) \tag{19}$$

$$= \sum_{m=1}^{M}\sum_{k=1}^{K} \Big(\ln 2 - \ln\Gamma(1+y_k^{(m)}) + y_k^{(m)} \ln a_k^{y(m)} + (1-y_k^{(m)})\ln(1-a_k^{y(m)})\Big) \tag{20}$$

where $y_k^{(m)}$ is the target at the k^{th} output neuron and where $a_k^{y(m)}$ is the activation of the k^{th} neuron at the output layer of the decoder. Note that $y_k^{(m)} = x_k^{(m)}$ because the autoencoder approximates an identity map.

III. Bidirectional Autoencoders

A bidirectional network runs forward and backward through the *same* synaptic weights [23], [29]–[31]. A forward inference passes through a given rectangular weight matrix W while the backward inference passes through the matrix transpose W^T.

A BAE N_θ learns or approximates an identity mapping from an input pattern space to the same or similar output pattern space. The data-encoding from the pattern $\mathbf{x}$ to the

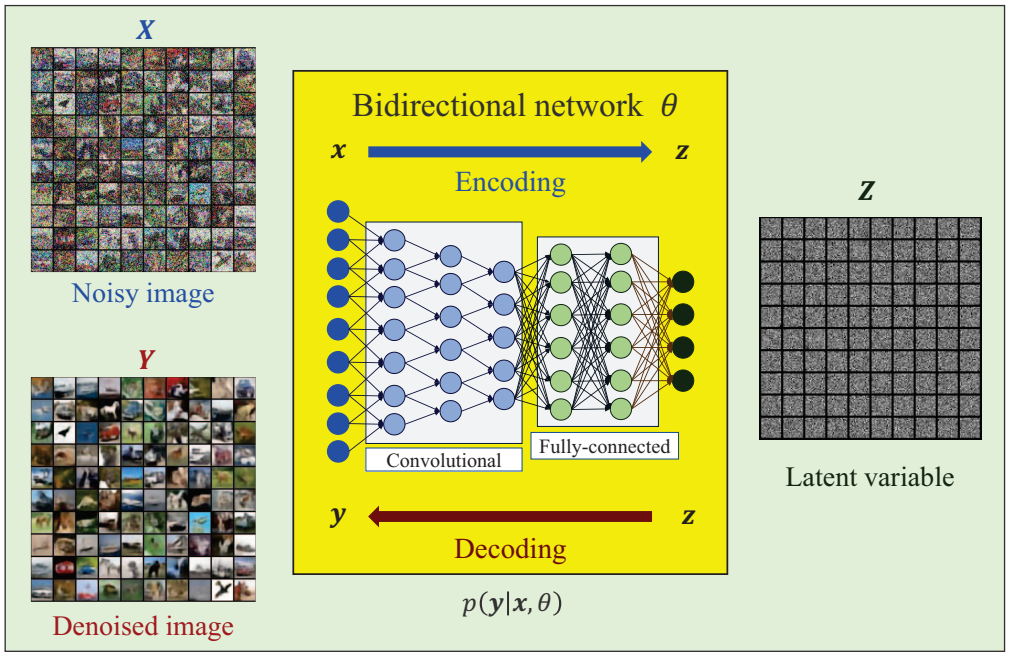


Fig. 1: Image denoising with a bidirectional autoencoding network: The architecture of the new bidirectional autoencoder consists of a single bidirectional network N_θ for encoding and decoding. The encoding and decoding use the same synaptic weights. This bidirectional network runs as an encoder in the forward direction and as a decoder in the backward direction.

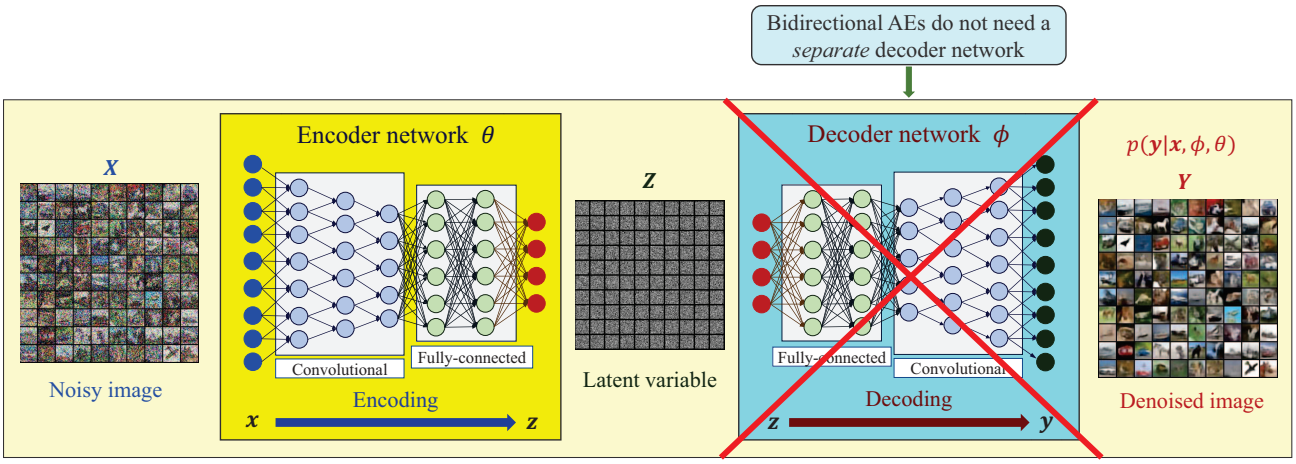
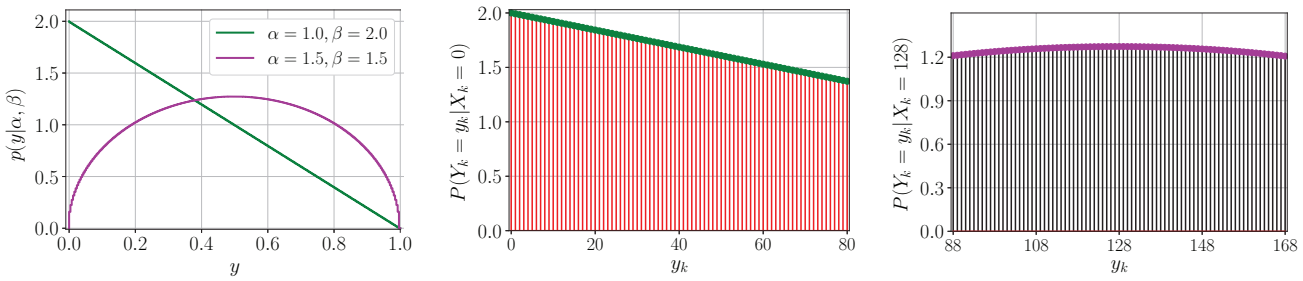


Fig. 2: Image denoising with a unidirectional autoencoding network: A unidirectional autoencoding network consists of two separate sub-networks with respective parameters θ and ϕ. They are the encoder network N_θ and the decoder network N_ϕ. The encoder maps the input space X to the latent space Z. The decoder network maps Z to Y. Bidirectional autoencoding networks use just one network.



(a) Beta distribution (b) Discretized Beta ($X_k = 0$) (c) Discretized Beta ($X_k = 128$)

Fig. 3: Beta probability density function Beta(α, β): A discretized form of Beta($\alpha = 1 + y_k$, $\beta = 2 - y_k$) models the output layer of an autoencoder. The discretized beta describes the probability of the output-image pixel value y given the input image $\mathbf{x}$. (a) The beta densities for $y_k = 0.0$ and $y_k = 0.5$. (b) Discretized beta density for the pixel value $X_k = 0$ for pixel values $y_k \in \{0, 1, ..., 255\}$. The function is nonzero but the figure shows only the values $y_k \in \{0, 1, ..., 80\}$. (c) Discretized beta probability function for the pixel value $X_k = 128$ for pixel values $y_k \in \{0, 1, ..., 255\}$. The figure shows only the values $y_k \in \{88, 89, ..., 168\}$.

latent variable $\mathbf{z}$ passes forward through the network N_θ. The encoding of the input image gives

$$\mathbf{a}^z = N_\theta(\mathbf{x}). \tag{21}$$

The decoding from $\mathbf{z}$ back to $\mathbf{x}$ passes backwards through the same network. So the encoded message decodes as

$$\mathbf{a}^{xb} = N_\theta^T(\mathbf{a}^z). \tag{22}$$

BAE networks train with a form of the bidirectional backpropagation algorithm [23], [30]. This training maximizes the *backward* likelihood $p(\mathbf{x}|\mathbf{y}, \theta)$ of the bidirectional networks. So this bidirectional structure differs in kind from the encoder-only structure of the Bidirectional Encoder Representations from Transformers (BERT) model [32].

The training error function $\mathcal{E}_M(\mathbf{x}, \theta)$ equals the negative log-likelihood of M training samples with the assumption of independent and identical distribution. The negative log-likelihood of Beta($\alpha = 1 + x_k, \beta = 2 - x_k$) gives the cross-entropy:

$$\begin{aligned}\mathcal{E}_M(\mathbf{x}, \theta) &= -\frac{1}{M}\sum_{m=1}^{M}\Big(\mathbf{x}^{(m)^T}\ln\left(\mathbf{a}^{xb(m)}\right) \\ &\quad + \left(\mathbf{1} - \mathbf{x}^{(m)}\right)^T\ln\left(\mathbf{1} - \mathbf{a}^{xb(m)}\right)\Big) \qquad (23) \\ &= -\frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{I}\Big(x_i^{(m)}\ln\left(a_i^{xb(m)}\right) \\ &\quad + \left(1 - x_i^{(m)}\right)\ln\left(1 - a_i^{xb(m)}\right)\Big) \qquad (24)\end{aligned}$$

where $x_i^{(m)}$ is the i^{th} pixel value of the m^{th} sample. The term $a_i^{xb(m)}$ is the activation of the i^{th} input neuron on the backward pass of the m^{th} sample.

Overall: BAE networks significantly reduced memory usage because they reduced the number of synaptic parameters by about 50%. This favors both large-scale neural models and dedicated hardware implementations.

Algorithm 1 Training a bidirectional autoencoder with a form of bidirectional backpropagation for image denoising.

Require: Dataset $\mathcal{D} = \left\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right\}_{i=1}^{N_{tr}}$, batch size L, learning rate η, number M of epochs, and number B of iterations *per* epoch.

Require: Initialize the network parameter $\theta^{(0)}$.

1: **for** $m = 0$ to M **do**

2: Select a batch of L samples from $\mathcal{D}$

3: Encode the input signal or vector on the forward pass:

$$\mathbf{a}^{z(l)} = N_\theta\left(\mathbf{x}^{(l)}\right)$$

4: Decode the latent variable on the backward pass:

$$\mathbf{a}^{x(l)} = N_\theta^T\left(\mathbf{a}^{z(l)}\right)$$

5: Compute the backward error E_b:

$$\begin{aligned}E_b(\theta) = -\frac{1}{L}\sum_{l=1}^{L}\sum_{k=1}^{K}\Big(&y_k^{(l)}\ln a_k^{xb(l)} \\ &+ \left(1 - y_k^{(l)}\right)\ln\left(1 - a_k^{xb(l)}\right)\Big)\end{aligned}$$

6: Update the weights:

$$\theta^{(m+1)} = \theta^{(m)} - \eta\nabla_\theta\ E_b(\theta)\Big|_{\theta=\theta^{(m)}}$$

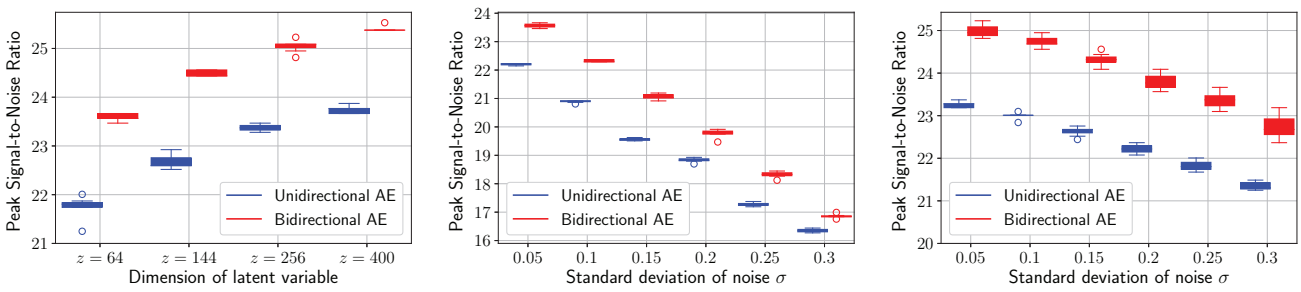
7: **end for**

IV. Simulation Results

The supercomputer simulations compared the performance of unidirectional with bidirectional AEs. They tested these autoencoders on image compression and reconstruction and on image denoising.

A. Model Architecture

We tested two types of autoencoders. The AEs were either fully connected or convolutional. The models used the new logistic *nonvanishing* (NoVa) hidden neurons [33], [34] because NoVa neurons outperformed rectified linear unit (ReLU)



(a) (b) (c)

Fig. 4: Image compression and image denoising with fully connected autoencoders on the MNIST handwritten digit dataset: BAEs always outperformed the corresponding unidirectional AEs on these tasks. The AEs each used a latent variable of dimension 256. The plots show the peak signal-to-noise ration (PSNR) after training. The AEs each trained over 200 epochs. (a) Data compression and reconstruction. (b) Denoising additive Gaussian noise. (c) Denoising multiplicative Gaussian (speckle) noise.

TABLE I: Image compression and reconstruction of the MNIST handwritten digit dataset with fully connected autoencoders: Bidirectional AEs always reduced the number of parameters by about 50% and outperformed the corresponding unidirectional AEs. The AEs trained over 200 epochs.

Dimension of Latent Variable	**Peak Signal-to-Noise Ratio** (PSNR) ↑		**# of Parameters** ↓	
	Unidirectional AE	**Bidirectional AE**	**Unidirectional AE**	**Bidirectional AE**
64	21.75 ± 0.22	**23.59 ± 0.07**	5,145,696	**2,572,848**
144	22.68 ± 0.14	**24.49 ± 0.06**	5,225,856	**2,612,928**
256	23.37 ± 0.07	**25.05 ± 0.12**	5,338,080	**2,669,040**
400	23.74 ± 0.07	**25.40 ± 0.05**	5,482,368	**2,741,184**

neurons and many others. The NoVa activation perturbs a logistic where the activation $a(x)$ from input x is

$$a(x) = bx + x\sigma(cx) = bx + \frac{x}{1 + \exp(-cx)}. \quad (25)$$

We used $b = 0.3$ and $c = 2.0$.

1) Fully Connected Autoencoder: The unidirectional AEs with fully connected layers each used one encoder network and one decoder network. Each encoder used four fully connected hidden layers. The first two hidden layers used 1000 neurons per layer and the other two used 500 neurons per layer. The encoder used identity input activations and sigmoid outputs.

Each decoder network mirrored the encoder and used four fully connected hidden layers. The first two hidden layers used 500 neurons per layer and the other two used 1000 neurons per layer.

BAEs with fully connected layers each used one bidirectional network. Each bidirectional network used four fully connected hidden layers. The first two hidden layers used 1000 neurons per layer and the other two used 500 neurons per layer.

2) Convolutional Autoencoders: The unidirectional convolutional AEs each used a convolutional encoder network and a convolutional decoder network. Each convolutional encoder network used five convolutional layers and two fully connected layers for encoding. The dimensions of the respective input channels and output channels of the convolutional layers were (3, 32, 64, 128, 256) and (32, 64, 128, 256, 512). The two fully connected layers used 2048 neurons and 1024 neurons.

The convolutional decoder network used five convolutional layers and two fully connected layers for decoding. The dimensions of the respective input channels and output channels of the convolutional layers were (512, 256, 128, 64, 32) and (256, 128, 64, 32, 3). The two fully connected layers used 2048 neurons and 1024 neurons.

The bidirectional convolutional AEs used a bidirectional convolutional network for encoding and decoding. Each BAE used five convolutional layers and two fully connected layers. The dimensions of the respective input channels and output channels of the convolutional layers were (3, 32, 64, 128, 256) and (32, 64, 128, 256, 512). The two fully connected layers used 2048 neurons and 1024 neurons. The convolutional autoencoders trained over 300 epochs.

B. Datasets

We compared AEs and BAEs on the MNIST handwritten digit and the CIFAR-10 image dataset [35]. The MNIST handwritten digit dataset contained the 10 classes of the handwritten digits $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. This dataset set consisted of 60,000 training samples with 6,000 samples per class and 10,000 test samples with 1,000 samples per class.

The CIFAR-10 test set consists of 60,000 color images from 10 categories ($K = 10$). Each image had size $32 \times 32 \times 3$. The 10 pattern categories were airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class consisted of 5,000 training samples and 1,000 testing samples.

TABLE II: Image denoising on the MNIST handwritten digit dataset with fully connected autoencoders: BAEs always outperformed their corresponding unidirectional AEs. The AEs trained on *additive* noise and separately on *multiplicative* (speckle) noise. The AEs used a latent variable of dimension 256 and trained over 200 epochs. The unidirectional AEs each used 5.3M trainable parameters while the BAEs each used 2.7M trainable parameters.

Standard deviation of noise σ	**Peak Signal-to-Noise Ratio** (PSNR) ↑			
	Additive Noise		**Multiplicative (Speckle) Noise**	
	Unidirectional AE	**Bidirectional AE**	**Unidirectional AE**	**Bidirectional AE**
0.05	22.20 ± 0.03	**23.56 ± 0.08**	23.25 ± 0.06	**24.99 ± 0.14**
0.1	20.89 ± 0.04	**22.32 ± 0.04**	23.00 ± 0.07	**24.74 ± 0.12**
0.2	18.83 ± 0.07	**19.77 ± 0.14**	22.22 ± 0.10	**23.80 ± 0.17**
0.3	16.35 ± 0.06	**16.86 ± 0.06**	21.35 ± 0.09	**22.74 ± 0.26**

TABLE III: Image compression and reconstruction with convolutional autoencoders on the CIFAR-10 image dataset: Bidirectional AEs with convolutional layers always significantly reduced the number of trainable synaptic parameters by about 50% and slightly outperformed their corresponding unidirectional AEs. These AEs trained over 300 epochs.

Dimension of Latent Variable	**Peak Signal-to-Noise Ratio** (PSNR) ↑		**Structural Similarity** (SSIM) ↑		**# of Parameters** ↓	
	Unidirectional AE	**Bidirectional AE**	**Unidirectional AE**	**Bidirectional AE**	**Unidirectional AE**	**Bidirectional AE**
64	20.78 ± 0.04	**21.13 ± 0.10**	0.618 ± 0.002	**0.637 ± 0.004**	7,470,339	**3,735,907**
128	23.32 ± 0.03	**23.48 ± 0.03**	0.770 ± 0.001	**0.778 ± 0.001**	7,601,539	**3,801,507**
324	26.80 ± 0.15	**27.00 ± 0.13**	0.892 ± 0.002	**0.896 ± 0.003**	8,003,339	**4,002,407**
512	27.12 ± 0.02	**27.66 ± 0.01**	0.899 ± 0.001	**0.910 ± 0.001**	8,388,739	**4,195,107**

The denoising experiments used noise-corrupted input images. The additive-noise denoising used noisy input images $x = y + n$ where n that came from the Gaussian probability density $\mathcal{N}(\mu = 0, \sigma)$ for clean image y. The multiplicative (speckle) noise denoising models used the noisy input image $x = y * n$ where n came from $\mathcal{N}(\mu = 1, \sigma)$.

C. Results

BAEs outperformed unidirectional AEs on image compression and image denoising. BAEs also reduced the number of parameters by about 50%. The simulations used two performance metrics: peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [36].

The simulations showed that fully connected BAEs outperformed fully connected unidirectional AEs on both metrics. BAEs also reduced the number of trainable parameters by about 50%. The simulations found these bidirectional benefits for image compression on the digit MNIST handwritten dataset. Figure 4 shows the bidirectional benefits of an increase in PSNR on image compression and image denoising (additive and multiplicative). The simulations used four different dimensions for the latent variable. Figure 4a compares the BAEs with unidirectional AEs. Table I shows the bidirectional benefits in PSNR and in the reduced number of network parameters. These bidirectional benefits also extended to image denoising.

Figure 4b shows that BAEs performed better than unidirectional AEs on denoising additive Gaussian noise. Figure 4c shows a similar bidirectional benefit for denoising when the contaminating noise was multiplicative Gaussian noise. Figure 5 shows that we can more easily separate the compressed features from BAEs than those from unidirectional AEs. This favors BAEs for extracting discriminative features for recognition or classification.

We used the t-distributed stochastic neighbor embedding (t-SNE) method to visualize the encoded features. This method uses a statistical approach to map the high-dimensional representation of data $\{\mathbf{x}_i\}_{i=1}^N$ to their respective low-dimensional representation $\{\mathbf{y}_i\}_{i=1}^N$ based on the similarity of the datapoints [24]. It is a two-step method. The first step defines the conditional probability $p_{j|i}$ and the joint probability p_{ij} over the high-dimensional space. The conditional probability is proportional to the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. It uses a Gaussian probability density with mean $\mathbf{x}_i$:

$$p_{j|i} = \frac{\exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i^2\right)} \tag{26}$$

for all $j \neq i$ where σ_i is the variance of the Gaussian with mean $\mathbf{x}_i$. The term $p_{i|i} = 0$. The joint probability is

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{27}$$

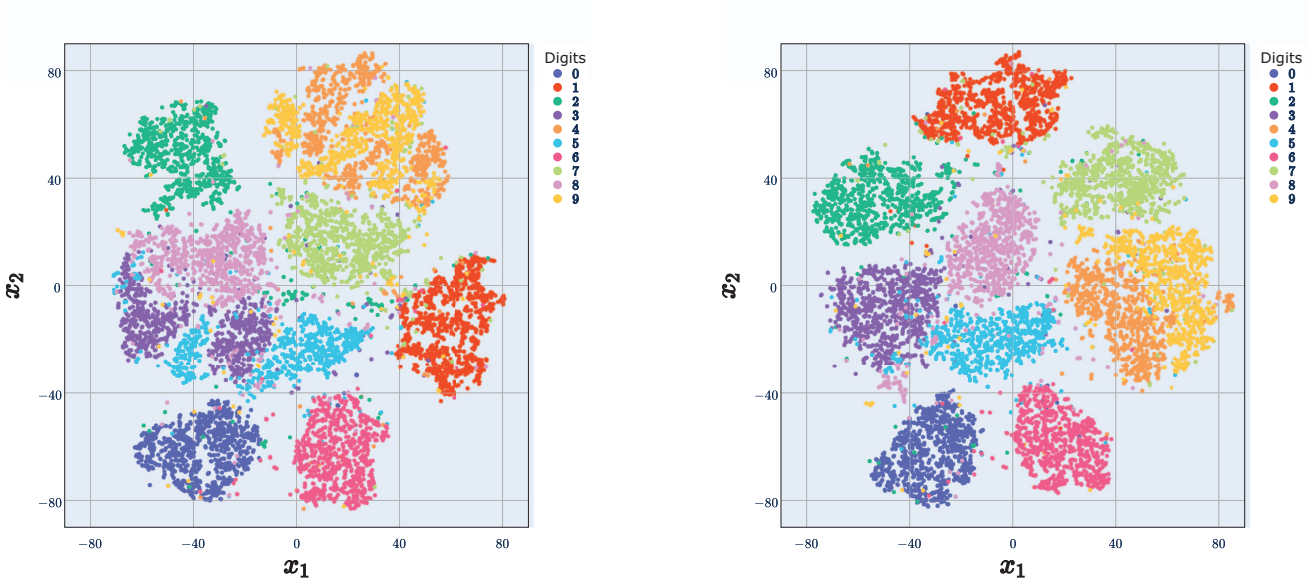
where N is the number of datapoints.

The second step maps the high-dimensional representation $\mathbf{x}_i$ to its corresponding low-dimensional representation $\mathbf{y}_i$ in $\mathbb{R}^d$ (d is typically 2 or 3). It uses a heavy-tailed student's-t density with one-degree of freedom (which equals the Cauchy density) to model the low-dimensional joint distribution. So the joint probability q_{ij} of the low-dimensional representations $\mathbf{y}_i$ and $\mathbf{y}_j$ has the form

$$q_{ij} = \frac{\left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2\right)^{-1}}{\sum_k \sum_{l \neq k} \left(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2\right)^{-1}} \tag{28}$$

TABLE IV: Image denoising with convolutional autoencoders on the CIFAR-10 image dataset: Bidirectional AEs always slightly outperformed their corresponding unidirectional AEs. We tested the AEs on additive noise and on multiplicative (speckle) noise. The AEs used a latent variable of dimension 324. These AEs trained over 300 epochs. The unidirectional AEs each used 8M trainable parameters and the bidirectional AEs each used 4M trainable parameters.

Metric	**Standard deviation of noise σ**	**Additive Noise**		**Multiplicative (Speckle) Noise**	
		Unidirectional AE	**Bidirectional AE**	**Unidirectional AE**	**Bidirectional AE**
Peak-Signal-to-Noise-Ratio (PSNR) ↑	0.05	26.25 ± 0.12	**26.44 ± 0.09**	26.59 ± 0.10	**26.82 ± 0.11**
	0.1	25.21 ± 0.06	**25.39 ± 0.06**	26.15 ± 0.11	**26.37 ± 0.11**
	0.3	21.81 ± 0.03	**22.08 ± 0.02**	24.10 ± 0.08	**24.23 ± 0.06**
	0.5	20.01 ± 0.01	**20.17 ± 0.01**	22.48 ± 0.09	**22.57 ± 0.03**
Structural Similarity Measure (SSIM) ↑	0.05	0.874 ± 0.003	**0.878 ± 0.002**	0.885 ± 0.002	**0.891 ± 0.003**
	0.1	0.838 ± 0.002	**0.843 ± 0.002**	0.873 ± 0.003	**0.878 ± 0.003**
	0.3	0.680 ± 0.002	**0.697 ± 0.002**	0.803 ± 0.002	**0.807 ± 0.002**
	0.5	0.567 ± 0.001	**0.582 ± 0.001**	0.733 ± 0.002	**0.737 ± 0.002**



(a) Unidirectional autoencoder features

(b) Bidirectional autoencoder features

Fig. 5: t-distributed Stochastic Neighbor Embedding (t-SNE) features with autoencoder compression on MNIST handwritten digits: The compressed features from BAEs separate more easily than do those from unidirectional AEs. The figure shows the 2D projection of the latent-space representation of the compressed features with autoencoder networks. The dimension of the latent space was 144. The transformed features from the BAE separated better than did those for the unidirectional AE. (a) Unidirectional autoencoder features. (b) Bidirectional autoencoder features.

for all $i \neq j$ and $q_{ii} = 0$. The location of the low-dimension representation comes from minimizing the Kullback-Leibler divergence

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{29}$$

The t-SNE algorithm uses gradient descent to iteratively find the value of $\mathbf{y}_i$ that minimizes the $KL(P||Q)$.

Simulations also showed that convolutional BAEs outperformed convolutional unidirectional AEs on the CIFAR-10 dataset. Table III shows that convolutional BAEs slightly outperformed their corresponding unidirectional architecture for image compression. This included a slight increase in the PSNR and the SSIM as well as a reduction of about 50% in the number of parameters. Table IV shows a similar bidirectional benefit of a slight increase in the PSNR and a slight increase in the SSIM for the image denoising task.

V. Conclusions

Bidirectional autoencoders offer an efficient way to learn autoencoder mappings. The new bidirectional backpropagation algorithm allows a single network to perform encoding and decoding. The bidirectional architecture improved network performance and substantially reduced computing memory because it cut in half the number of trainable synaptic parameters. So it should have more pronounced bidirectional benefits on larger-scale models and aid hardware implementations. Preliminary simulations also found that these bidirectional benefits extended to variational autoencoders.

REFERENCES

[1] M. Dorosti and M. M. Pedram, "Improving english to persian machine translation with GPT language model and autoencoders," in *2023 9th International Conference on Web Research (ICWR)*. IEEE, 2023, pp. 214–220.

[2] D. Biesner, K. Cvejoski, and R. Sifa, "Combining variational autoencoders and transformer language models for improved password generation," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–6.

[3] T. Ge, J. Hu, X. Wang, S.-Q. Chen, and F. Wei, "In-context autoencoder for context compression in a large language model," *arXiv preprint arXiv:2307.06945*, 2023.

[4] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops*. IEEE Computer Society, 2014, pp. 496–503. [Online]. Available: https://doi.org/10.1109/CVPRW.2014.79

[5] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016. [Online]. Available: https://doi.org/10.1016/j.neucom.2015.08.104

[6] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[7] "Autoencoders based deep learner for image denoising," *Procedia Computer Science*, vol. 171, pp. 1535–1541, 2020, third International Conference on Computing and Network Communications (CoCoNet'19).

[8] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *IEEE International Conference on Data Mining Workshops, ICDM Workshops*. IEEE Computer Society, 2016, pp. 241–246. [Online]. Available: https://doi.org/10.1109/ICDMW.2016.0041

[9] K. Cho, "Boltzmann machines and denoising autoencoders for image denoising," in *1st International Conference on Learning Representations, ICLR Workshop*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3468

[10] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Simul.*, vol. 4, no. 2, pp. 490–530, 2005. [Online]. Available: https://doi.org/10.1137/040616024

[11] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao, "Coupled deep autoencoder for single image super-resolution," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 27–37, 2017. [Online]. Available: https://doi.org/10.1109/TCYB.2015.2501373

[12] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010. [Online]. Available: https://dl.acm.org/doi/10.5555/1756006.1953039

[13] A. Morales-Forero and S. Bassetto, "Case study: A semi-supervised methodology for anomaly detection and diagnosis," in *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2019, pp. 1031–1037.

[14] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, A. Rahman, J. D. Deng, and J. Li, Eds. ACM, 2014, p. 4. [Online]. Available: https://doi.org/10.1145/2689746.2689747

[15] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 665–674. [Online]. Available: https://doi.org/10.1145/3097983.3098052

[16] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless telecommunications symposium (WTS)*. IEEE, 2018, pp. 1–5. [Online]. Available: https://doi.org/10.1145/3097983.3098052

[17] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*. ACM, 2018, pp. 689–698. [Online]. Available: https://doi.org/10.1145/3178876.3186150

[18] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, "Sequential variational autoencoders for collaborative filtering," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019*. ACM, 2019, pp. 600–608. [Online]. Available: https://doi.org/10.1145/3289600.3291007

[19] S. Zhai and Z. Zhang, "Semisupervised autoencoder for sentiment analysis," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[20] H. Sagha, N. Cummins, and B. W. Schuller, "Stacked denoising autoencoders for sentiment analysis: a review," *WIREs Data Mining Knowl. Discov.*, vol. 7, no. 5, 2017. [Online]. Available: https://doi.org/10.1002/widm.1212

[21] C. Wu, F. Wu, S. Wu, Z. Yuan, J. Liu, and Y. Huang, "Semi-supervised dimensional sentiment analysis with variational autoencoder," *Knowl. Based Syst.*, vol. 165, pp. 30–39, 2019. [Online]. Available: https://doi.org/10.1016/j.knosys.2018.11.018

[22] O. Adigun and B. Kosko, "Bidirectional representation and backpropagation learning," in *International Joint Conference on Advances in Big Data Analytics*, 2016, pp. 3–9.

[23] ——, "Bidirectional backpropagation," *IEEE Transaction on Systems, Man, and Cybernetics: Systems, Man, and Cybernetics*, 2018.

[24] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[25] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[26] Z. Ma and A. Leijon, "Beta mixture models and the application to image classification," in *Proceedings of the International Conference on Image Processing, ICIP 2009, 7-10 November 2009, Cairo, Egypt*. IEEE, 2009, pp. 2045–2048. [Online]. Available: https://doi.org/10.1109/ICIP.2009.5414043

[27] G. Loaiza-Ganem and J. P. Cunningham, "The continuous bernoulli: fixing a pervasive error in variational autoencoders," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[28] B. Kosko, K. Audhkhasi, and O. Osoba, "Noise can speed backpropagation learning and deep bidirectional pretraining," *Neural Networks*, vol. 129, pp. 359–384, 2020.

[29] B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 49–60, 1988.

[30] O. Adigun and B. Kosko, "Noise-boosted bidirectional backpropagation and adversarial learning," *Neural Networks*, vol. 120, pp. 9–31, 2019. [Online]. Available: https://doi.org/10.1016/j.neunet.2019.09.016

[31] B. Kosko, "Bidirectional associative memories: unsupervised hebbian learning to bidirectional backpropagation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 103–115, 2021.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pretraining of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[33] O. Adigun and B. Kosko, "Deeper neural networks with non-vanishing logistic hidden units: NoVa vs. ReLU neurons," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021, pp. 1407–1412. [Online]. Available: https://doi.org/10.1109/ICMLA52953.2021.00227

[34] ——, "Deeper bidirectional neural networks with generalized non-vanishing hidden neurons," in *21st IEEE International Conference on Machine Learning and Applications, ICMLA 2022*. IEEE, 2022, pp. 69–76. [Online]. Available: https://doi.org/10.1109/ICMLA55696.2022.00017

[35] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.