On the Transformation of Latent Space in Autoencoders

Jaehoon Cha  
jaehoon.cha@xjtlu.edu.cn

Kyeong Soo Kim  
kyeongsso.kim@xjtlu.edu.cn

Sanghyuk Lee  
sanghyuk.lee@xjtlu.edu.cn

Department of Electrical and Electronic Engineering  
Xi’an Jiaotong-Liverpool University  
Suzhou, 215123, P. R. China

Abstract

Noting the importance of the latent variables in inference and learning, we propose a novel framework for autoencoders based on the homeomorphic transformation of latent variables — which could reduce the distance between vectors in the transformed space, while preserving the topological properties of the original space — and investigate the effect of the latent space transformation on learning generative models and denoising corrupted data. The experimental results demonstrate that our generative and denoising models based on the proposed framework can provide better performance than conventional variational and denoising autoencoders due to the transformation, where we evaluate the performance of generative and denoising models in terms of the Hausdorff distance between the sets of training and processed—i.e., either generated or denoised—images, which can objectively measure their differences, as well as through direct comparison of the visual characteristics of the processed images.

1 Introduction

Data compression/restoration and generating new data based on the learned distribution from a training dataset have been extensively studied in the context of machine learning, especially with artificial neural networks. In their early stage, the wake-sleep algorithm was used to produce a good density estimator by training a stack of layers so that each of the layers can correctly represent activities above and below it [5].

Recently, Autoencoders (AE) have been gaining huge attention from researchers not only for data compression/restoration but also as generative models. AE is originally studied to extract salient features through its bottleneck structure which reduces the dimensionality of the input data [4]. AE is also studied as efficient generative models [17, 18, 11, 15, 10, 2]. In particular, Variational Autoencoder (VAE) is introduced as a stochastic variational inference and learning algorithm [8]. The encoder network of VAE approximates the posterior distribution given the input data and infers good values of latent variables. Then, the decoder network generates a distribution of input data over the latent variables. Because VAE takes latent variables during the generation phase, we began to realize the importance of the latent space and regularizers and investigate ways on how to explore the latent space during training and generating processes.

Preprint. Under review.
Note that the training objectives and the use of the reparameterization trick with Gaussian latent variables in conventional VAE may result in a regularizer with poor inference quality and thereby provide models which are not able to properly capture the dependencies in the original data due to the assumption of independent latent variables \[10\][19][16]. In this paper, therefore, we propose a novel framework of Latent space Transformation in Autoencoder (LTAE) based on the idea of mapping the latent space through transformation technique, which requires two new steps compared to the conventional AE: First, we reduce the distance between any two vectors in the latent space through the proposed transformation technique, which acts as a regularizer. Second, we explore the area of the latent space which does not correspond to any input vector through adding noise to the output from the encoder network in order to deal with unseen data. The LTAE framework also provides a better connection from inputs to latent variables to outputs by eliminating the reparameterization trick used in conventional VAE.

The advantages of the proposed LTAE framework are two-fold: First, this framework is so flexible that it can be applicable to both generative and denoising models. Second, the framework could improve the performance of the resulting models compared to conventional AEs.

Note that for reconstruction applications, the LTAE framework can be applied to Denoising Autoencoder (DAE), which was invented to extract more useful features by introducing a new training principle of denoising partially corrupted input data \[17\][18]. The introduced noise enables DAE to find useful features in a more robust way and results in good performance when reconstructing corrupted data. Denoising Latent space Transformation in Autoencoder (DLTAE)—i.e., DAE based on the LTAE framework—introduces noise at two different spaces in training, i.e., the input space and the latent space, to further enhance the robustness of a resulting model. Due to the transformation in DLTAE, it is also capable of generating data by taking variables in the transformed latent space with the decoder network. The generated images are clearer than those by VAE and LTAE, because DLTAE can capture more salient features by the noise introduced at two different spaces.

We also propose the use of the Hausdorff distance \[12\] as an objective measure of the performance of generative and denoising models, which is frequently used in computer vision and pattern recognition to measure the extent to which each point of a model set lies near some point of an image set and vice versa and thereby provide a degree of resemblance between the two \[6\]. Note that, however, we extend the application of the Hausdorff distance to the measurement of the similarity between two sets of images (i.e., the set of training images and that of processed images), rather than the similarity between two individual images/shapes, in this paper.

2 Preliminaries

2.1 Notations and Basic Definitions

\(\mathbb{R}^d\) denotes a \(d\)-dimensional Euclidean space. Vectors are written in bold lowercase. If \(x\) is a vector, then, its \(i\)th element is denoted by \(x_i\). We use bold uppercase letters for matrices (e.g., \(A\)).

**Definition 2.1.** A nonempty set \(A\) in a metric space \((X, d)\) is said to be bounded if the diameter \(\text{diam}(A) < \infty\), where

\[
\text{diam}(A) \triangleq \sup_{x,y \in A} d(x, y). \quad (1)
\]

**Theorem 2.2.** A sequence \((x_n)\) in a normed space \(X\) is convergent if \(X\) contains an \(x\) such that

\[
\lim_{n \to \infty} \|x_n - x\| = 0. \quad (2)
\]

Then we write \(x_n \to x\).

**Theorem 2.3.** \[9\] Let \(B\) be a subset of a metric space \(X\) and let \(\varepsilon > 0\) be given. A set \(M \subseteq X\) is called an \(\varepsilon\)-net for \(B\) if for every point \(z \in B\) there is a point of \(M\) at a distance from \(z\) less than \(\varepsilon\). The set \(B\) is said to be totally bounded if for every \(\varepsilon > 0\) there is a finite \(\varepsilon\)-net \(M \subseteq X\) for \(B\), where “finite” means that \(M\) is a finite set.

**Theorem 2.4.** \[3\] A subset \(E\) of \(\mathbb{R}^n\) is totally bounded if and only if \(E\) is bounded.

**Definition 2.5.** \[12\] Let \(X\) and \(Y\) be topological spaces and \(f : X \to Y\) be a bijection, which is a one-to-one (injective) and onto (surjective) mapping. The function \(f\) is called a homeomorphism if \(f\) and the inverse function \(f^{-1} : Y \to X\) are continuous, and \(X\) and \(Y\) with a homeomorphism are called homeomorphic.
We explore this set by adding noise to the output from the encoder network in order to make the
where\[ \Phi \] is an activation function such as Softplus, sigmoid, hyperbolic tangent (tanh), rectified linear unit
\[ \text{ReLU} \], and leaky ReLU. Then, \[ \phi \] is trained to minimize the following loss function:
\[
\frac{1}{N} \sum_{x \in X_{in}} L(x, g(f(x; \phi); \theta)),
\]
where \( f(x; \phi) \in Z \), \( N \) is the number of input vectors and \( L \) is a loss function which could be either cross-entropy or \( L_2 \) loss.

Note that there is a set of vectors in the latent space \( Z \), which do not correspond to any input vector.
We explore this set by adding noise to the output from the encoder network in order to make the
original AE a generative model. In the LTAE framework, we introduce a transformation network and
a latent network between the encoder and the decoder network of the original AE to make it a
generative model. The latent network receives the outputs of the encoder network and injects them
to the decoder network. Due to a loss function between the latent network and the transformation
network, vectors are transformed in the latent network. We denote by \( Z_L \) a space of the transformed
vectors through the latent network. By reducing the distances between output vectors in \( Z_L \) without
changing their topological properties, the interpolation between output vectors during the generative
phase can be easier and more meaningful. In this section, a method to make \( Z_L \) and to deal with
unseen vectors, which are possibly lie on the sparse spaces on \( Z \) or \( Z_L \), is described.

3 Latent Space Transformation in Autoencoder

3.1 Continuity of the Original Autoencoder

Let us assume that one layer of a neural network consists of a set of matrix multiplication, addition,
and an activation function. We define a function \( h : X \rightarrow Y \), given by
\[
h(x) = f(Ax + b)
\]
where \( \dim(X) = n \), \( \dim(Y) = r \), \( A : \mathbb{R}^n \rightarrow \mathbb{R}^r \) is a matrix operator, \( b \) is a vector of \( r \) components, and \( f \) is an activation function such as Softplus, sigmoid, hyperbolic tangent (tanh), rectified linear unit (ReLU), and leaky ReLU. Then, \( x_n \rightarrow x \) implies \( h(x_n) \rightarrow h(x) \) from the fact that
\[
\|h(x_n) - h(x)\| = \|f(Ax_n + b) - f(Ax + b)\|
\leq \|Ax_n - x\|
\leq \|A\| \|x_n - x\|,
\]
where \( \|A\| \) is the matrix norm.

\[1\] If we consider the encoder network as a function, the latent space corresponds to the image of the function.
because matrix multiplication is bounded and all the activation functions considered satisfy
\[ \|f(x_n) - f(x)\| \leq \|x_n - x\|. \tag{10} \]

Due to the fact that a composite of continuous functions is continuous and a network with consecutive layers is a composite of layers, the continuity preserves through the layers.

Now, we let \( f_\theta : X_{in} \rightarrow Z \) and \( g_\theta : Z \rightarrow X_{out} \) be composite functions of hidden layers from \( \Theta \) where \( f_\theta(x)=f(x; \phi) \) and \( g_\theta(z)=g(z; \theta) \). If \( x'=g_\theta(f_\theta(x)) \) and \( y'=g_\theta(f_\theta(y)) \) for any \( x, y \in X_{in} \), then
\[ \|x' - y'\| \leq c_y \|f_\theta(x) - f_\theta(y)\| \leq c_f \|x - y\|. \tag{11} \]

where \( c_f \) and \( c_y \) denote the product of the norms of projection matrices in the encoder and the decoder networks, respectively. From (11), therefore, we can expect that, if the distance between latent vectors is small, the distance between the resulting outputs from the decoder network in a generative model — i.e., \( g_\theta(z) \) and \( g_\theta(z') \) — is small. In fact, this is the major reason we introduce the latent space in this way, due to the continuity between the latent space and the output space, \( g \).

### 3.2 Mapping of Unseen Vectors

Let \( Z = U \cup V \), where \( U \) is a subset of \( Z \) which consists of \( f_\theta(x) \) for all \( x \in X_{in} \) and \( V = Z - U \). Because \( U \) is a subset of \( \mathbb{R}^m \), where \( m \) is a dimension of the latent space, and bounded, it is a totally bounded. Therefore, for every \( \varepsilon > 0 \) there is a finite \( \varepsilon \)-net \( M_\varepsilon \) for \( \hat{U} \). Let \( M_\varepsilon = \{m_{\varepsilon}^{(1)}, m_{\varepsilon}^{(2)}, \ldots, m_{\varepsilon}^{(K)}\} \). Then there is a collection of open balls \( \mathcal{B} = \bigcup_{i=1}^{K} B_d(m_{\varepsilon}^{(i)}, \varepsilon) \) such that \( U \subseteq \mathcal{B} \), where \( B_d(m_{\varepsilon}^{(i)}, \varepsilon) \triangleq \{z \in \mathbb{R}^m \mid d(m_{\varepsilon}^{(i)}, z) < \varepsilon\} \) and \( d \) is a given metric or a metric induced by norm on \( Z \). Then, there is \( m_{\varepsilon}^{(j)} \) for all \( u^{(j)} \in U \) such that \( B_d(m_{\varepsilon}^{(j)}, \varepsilon) \subseteq B_d(u^{(j)}, 2\varepsilon) \) and it implies
\[ Z = \hat{U} \cup \hat{V}, \tag{12} \]

where \( \hat{U} = \bigcup_{j=1}^{K} B_d(u^{(j)}, 2\varepsilon) \), \( \hat{V} = Z - \hat{U} \) and \( K \leq J \leq N \).

Note that, unlike \( U \), \( \hat{U} \) in (12) now includes latent vectors corresponding to both unseen vectors and observed vectors in the input space, i.e., \( \hat{U} \supseteq U \). \( \hat{V} \) in (12), on the other hand, does not include any vectors from \( U \) and, as a result, does not have any information on the observed data. Our approach to mapping of unseen vectors, therefore, is to locate a latent vector \( z \) of an unseen vector within an open ball in \( \hat{U} \) (i.e., \( z \in B_d(u, 2\varepsilon) \)). \( \exists u \in U \) through the transformation technique described in Section 3.3.

### 3.3 Transformation

Note that \( \hat{U} \) is not appropriate for an input space of a generative model, because \( \text{diam}(\hat{U}) \) is big and clusters of vectors in the latent space are far away from each other in general, which makes it difficult to interpolate. In this section, we define the transformation network and the latent network in order to transform vectors in \( Z \) into \( Z_L \).

Through the latent network, \( \text{diam}(\hat{U}) \) becomes smaller and clusters of vectors in \( Z \) get closer. The transformation network and the latent network are located between the encoder and the decoder network. The outputs of the encoder network go to the transformation network and the latent network. Let \( X \in X_{in} \) be a set of input vectors at an iteration. Figure 1 shows the architecture of the LTAE. In the transformation network, each element of \( z \) in \( Z \) is transformed by the standard normalization, \( \text{diam}(\hat{U}) \) becomes smaller and clusters of vectors in \( Z \) get closer.

\[
(z_N)_{i} = \frac{z_{i} - \mu(Z_{i})}{\sigma(Z_{i})}, \tag{13}
\]

where \( \mu(\cdot) \) and \( \sigma(\cdot) \) denote the mean and the standard deviation, or by the min-max normalization,
\[
(z_N)_{i} = \frac{z_{i} - \min(Z_{i})}{\max(Z_{i}) - \min(Z_{i})}. \tag{14}
\]

\[ ^{2}\text{Note that we choose the standard normalization and the min-max normalization for a simple transformation case. There is no limitation of transformation methods. Any transformation technique that makes vectors close without changes of topological properties can be used.} \]
Figure 1: Architecture of the Latent space Transformation Autoencoder. $Z, Z_N$ and $Z_L$ denote spaces of the outputs of the encoder, the transformation and the latent networks. $\epsilon$ is a set of noise which has the same size with the $Z_L$. Vectors in $Z$ go to the latent network and the transformation network. Vectors in $Z_L$ are learned to form as similar as possible with corresponding vectors in $Z_N$ and to recover vectors in $X'$ as similar as possible to corresponding vectors in $X$.

where $Z_i$ is a set of the $i$th element of all vectors in $Z$ and $z_N=((z_N)_i) \in Z_N$.

We want to train the parameters of the latent network to compute the normalization process of the transformation network. Because both normalization methods require subtract first and then division, we define two $m$-dimensional row vectors $\alpha$ and $\beta$ of the latent network such that $z_L$ is calculated by

$$z_L = \alpha \bigodot (z \bigoplus \beta),$$

(15)

where $\bigodot$ and $\bigoplus$ denote element-wise multiplication and addition. Then, the $L_2$ loss between $Z_L$ and $Z_N$ is calculated so that $\alpha$ and $\beta$ are learned to make $Z_L$ and $Z_N$ similar. Note that we cannot use probability-based loss functions like cross-entropy for the loss between $Z_L$ and $Z_N$ because a range of values of vectors are larger than $[0, 1]$.

Now, our goal is to train a neural network consisting of an encoder network $f$ and a decoder network $g$ with weights and biases $\phi$ and $\theta$, and the transformation network output $Z_N$ and the latent network output $Z_L$ to minimize the following loss function:

$$\frac{1}{N} \sum_{x \in X} \tilde{L}(x; g(z; \theta)) + \|z_N - z_L\|_2,$$

(16)

where $z=u+\epsilon$ for a given $\epsilon>0$ and $\|\epsilon\|<2\epsilon$, $u=f(x; \phi)$, and $\tilde{L}$ is either cross-entropy or $L_2$ loss.

4 Analysis

The LTAE aims that clusters in the latent space get closer to one another and thereby makes it easy to learn unseen vectors in the latent space so that any vector in a specific subset of the latent space can have matched outputs. The latent space and the transformed latent space share the same topological properties because of the homeomorphism between the two spaces. The latent network transforms vectors in the latent space into the transformed latent space, where unseen vectors lie nearby observed vectors since $\operatorname{diam}(Z_L)$ becomes small. All possible input vectors of the decoder network during the generation process are sampled according to the transformation used during the training process.

4.1 Homeomorphism

In topology, two homeomorphic spaces are considered to be topologically equivalent. This means that, if topological space $X$ and $Y$ are homeomorphic, all topological properties of $X$ (e.g., compactness, connectedness, or Hausdorff) are preserved in $Y$. 
Note that the equation (15) can be rewritten as a function \( f: \mathbb{Z} \rightarrow \mathbb{Z} \): For \( z \in \mathbb{Z} \) and \( z_l \in \mathbb{Z}_L \), \( f \) is defined as \( z_l = f(z) = \prod_{i=1}^{m_i} f_i(z_i) \), where \( f_i(z_i) = \alpha_i(z_i + \beta_i) \) and \( \prod \) denotes the Cartesian product. Because \( f \) is both continuous and bijection and has a continuous inverse function (i.e., homeomorphism), \( \mathbb{Z} \) and \( \mathbb{Z}_L \) are homeomorphic and topologically equivalent. With the transformation network, \( \alpha \) is trained to get close with \( 1/\sigma(Z(i)) \) and \( 1/(\max(Z(i)) - \min(Z(i))) \), and \( \beta \) is trained to get close with \( -\mu(Z(i)) \) and \( -\min(Z(i)) \) in the standard normalization and the min-max normalization, respectively, while preserving the topological properties of \( \mathbb{Z} \).

4.2 Layer Transformation

The layer transformation makes vectors in the latent space located within a small and dense region. The method seems similar to batch normalization and Layer normalization because the method calculates the standard normalization and the min-max normalization \([7, 1]\). The main difference between the layer transformation from batch normalization and the layer normalization is that \( \alpha \) and \( \beta \) are learned to normalize each output of the encoder network. The transformation network transforms vectors in the latent space using statistical features of outputs of the encoder network during the training and thereby the range of the latent vectors is determined by the statistical features. The main advantage of the layer transformation is that clusters of vectors get close. As a result, distances between observed vectors in the latent space get smaller and so it becomes easy to interpolate sparse spaces between all observed vectors because if the vectors in the latent space are widely distributed during the training process, many of them will not result in outputs close to those corresponding to observed vectors in the input space during the generation process. On the other hand, it would be easier for unseen vectors in the input space to have outputs close to those corresponding to observed vectors in the input space during the generation process, if the vectors corresponding to the train dataset are close to one another in the latent space.

4.3 Similarity Measure of Sets of Images

In evaluating the performance of generative models for image synthesis, two major requirements, which are seemingly contradictory to each other, should be taken into account: Generated images should have visual characteristics similar to those of some training images, but, at the same time, differ from the training images \([13]\). In order to meet these requirements, we propose the Hausdorff distance \([12]\) as a metric capturing the similarity between two sets of images.

Let \( U \) and \( V \) be two different sets of images and \( u \) and \( v \) be individual images belonging to \( U \) and \( V \), respectively. Due to taking the maximum of both \( \sup_{u \in U} \inf_{v \in V} d(u, v) \) and \( \sup_{v \in V} \inf_{u \in U} d(v, u) \) in the definition of the Hausdorff distance given in (4), the difference of two sets of images can be properly measured by taking into account the two major requirements. The Hausdorff distances of three different types are illustrated in the Figure 2.

Note that, unlike the conventional use of the Hausdorff distance as a similarity measure between two individual images/shapes (e.g., \([6]\)), we use it to measure the similarity between two sets of images (i.e., the set of training images and that of processed images) to objectively evaluate the performance of generative and denoising models.
5 Experiment

We trained the LTAE model of images from the MNIST dataset. The encoder and the decoder each has two hidden layers with 500 hidden units for MNIST. The number of hidden units is chosen based on prior autoencoder literature [8]. A softplus rectifier is used for two hidden layers in the encoder and the decoder. A linear function is used for the output layer of the encoder and a sigmoid function is used for the output layer of the decoder. We use Cyclical Learning Rates (CLR) for with the base learning rate 0.001, the maximum learning rate 0.005, and step size of 5500 [14] with batch size of 100 and. The weights are initialized by Xavier initialization and the . The model is tested with different values of $\varepsilon$ and latent space dimension.

In this paper, we use the standard normalization and the min-max normalization transformation technique. $\varepsilon$ is added to variables at the transformed latent space so that the LTAE learns unseen data around the input data set while training, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ or $\varepsilon \sim \mathcal{U}(-\sigma, \sigma)$ according to the transformation technique. We use LTAE-S-$\sigma$ and LTAE-M-$\sigma$ to denote the LTAE with $\sigma$ by the standard normalization and the min-max normalization transformation technique, respectively.

5.1 Generative Models

In order to generate images, vectors are sampled with respect to the transformation techniques. Vectors are sampled from $\mathcal{N}(\mu, \sigma^2)$ in the LTAE-S, where $\mu$ and $\sigma$ are the mean and the standard deviation of vectors in $Z_L$, respectively. In the LTAE-M, vectors are sampled from $\mathcal{U}(m, M)$, where $m$ and $M$ are the minimum and the maximum of the vectors in $Z_L$, respectively.

5.2 DLTAE: Denoising and Generative Models

The transformation and the latent networks can be located between any layers, which makes it easy to combine LTAE with any AEs. We propose DLTAE by introducing noise as same as the DAE [17, 18]. In this paper, we take an example of the simple DAE case whose noise is injected only at the input space. The architecture of the DLTAE is the same as the LTAE and corruption data process is the same as the DAE: The corrupted input vector by added noise, i.e., $x + \hat{\epsilon}$, is injected to the encoder network. The output of the decoder network of $x + \hat{\epsilon}$ is compared with the original vector, $x$. The loss function of the DLTAE is the same with the Equation (16) except $u = f(x + \hat{\epsilon}; \phi)$ and $\hat{L}$ is the cross-entropy loss in our experiment In fact, compared to the DAE, the reduction of the introduced noise in DLTAE occurs at two different places, i.e., the encoder network related with the noise occurs at input space and the decoder network in regard to the noise at latent space. Due to the corruption of inputs and its same structure with the LTAE, the DLTAE can be used as a denoising model and a generative model at the same time.

---

3 Available at http://www.cs.nyu.edu/~roweis/data.html

4 It is not the limitation of the DLTAE: Any corruption process can be applied to the DLTAE.
Table 1: The Hausdorff distance between training images and generated images

| Compared with                  | VAE  | DLTAE-M | DLTAE-S |
|-------------------------------|------|---------|---------|
| Hausdorff distance ($L_2$-norm) | 9.364 | 7.4631  | 8.7427  |
| Hausdorff distance (cross-entropy) | 6.219 | 5.8758  | 5.6153  |

Table 2: The Hausdorff distance of corrupted images and reconstructed images with respect to training images

| Compared with                  | Corrupted images | DAE     | DLTAE-M | DLTAE-S |
|-------------------------------|------------------|---------|---------|---------|
| Hausdorff distance ($L_2$-norm) | 14.8508          | 6.2902  | 4.7362  | 4.6981  |
| Hausdorff distance (cross-entropy) | 5.7924          | 5.4895  | 5.3686  | 5.3655  |

5.3 Comparison with VAE for Generative Models

We take the VAE and calculate the Hausdorff distance by taking $L_2$-norm and cross entropy as $d$ in the equation (4) between the training image set and the generated image set for a comparison with the proposed model for a generative model. We train the VAE with the same number of hidden layers and units. The base learning rate and the maximum learning rate are set 0.0008 and 0.002, respectively, because the gradient decent diverges while training the VAE with the same learning rate condition mentioned in Section 5. Transformed latent space of the LTAE-M-0.06, the LTAE-S-0.02, and the VAE with 2 dimensional latent space are shown in Figure 3. The range of the transformed latent space are determined according to the transformation technique.

We take the DLTAE-M and DLTAE-S to compare with the VAE for a generation performance. First, we compare three models with MNIST data set. 100 samples are randomly picked up and then used to train three models with 2 dimensional latent space. We change the step size for CLR to 10 and iterations to 40000 because the number of the data set has been changed. Ten sets of 10000 images are generated and the mean of the Hausdorff distance between training images and each set is summarized in Table 1.

5.4 Comparison with DAE for Denoising Models

Even though the goal of the DAE is not reconstruction, we compare the reconstruction of the DLTAE with the DAE to check its denoising performance. The DAE is trained with the same number of hidden layers, units, and hyperparameters for CLR. While training, Gaussian random noise of $\mathcal{N}(0, 0.5^2)$ is added to an original input image. The corrupted image is injected to the three models. The sample outputs from the three models are shown in Appendix, and the Hausdorff distances are summarized in Table 2, which demonstrate that the reconstruction images by the DLTAE is more similar to the original images by capturing salient features like the DAE.

6 Conclusions

We have proposed a novel framework for AE based on the homeomorphic transformation of latent variables through new latent and transformation networks installed between the encoder and the decoder networks of AE: unlike the conventional VAEs based on the reparameterization trick with independent Gaussian latent variables, the proposed framework allows more flexibility in handling the latent space while maintaining the direct connection from inputs to latent variables to outputs. We have investigated the effect of the transformation in both learning generative models and denoising corrupted data. The experimental results with the images from the MNIST dataset show that the proposed framework could generate a model working as both a generative model and a denoising model with much improved performance.
References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Antonia Creswell and Anil Anthony Bharath. Denoising adversarial autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, (99):1–17, 2018.

[3] Theodore W Gamelin and Robert Everist Greene. *Introduction to topology*. Courier Corporation, 1999.

[4] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[5] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[6] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):850–863, Sep. 1993.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[9] Erwin Kreyszig. *Introductory functional analysis with applications*, volume 1. Wiley New York, 1978.

[10] Lars Maaløe, Casper Kaas Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

[11] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[12] James R. Munkres. *Topology: A first course*, volume 23. Prentice-Hall Englewood Cliffs, NJ, 1975.

[13] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[14] Leslie N Smith. Cyclic learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE, 2017.

[15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[16] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

[17] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

[18] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[19] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.