Representation Learning and Exploration of Latent Space Embeddings

Gopal Krishna
Khoury College of Computer Sciences
Northeastern University
Boston, MA
krishna.g@northeastern.edu

Gourav Beura
Khoury College of Computer Sciences
Northeastern University
Boston, MA
beura.g@northeastern.edu

Heet Sakaria
Khoury College of Computer Sciences
Northeastern University
Boston, MA
sakaria.h@northeastern.edu

***Abstract*—To overcome the high dimensionality of data, learning latent feature representations for clustering has been widely studied recently. However, it is still challenging to learn "cluster-friendly" latent representations due to the unsupervised fashion of clustering. In this project, we will explore how to utilize image embeddings and latent space of autoencoders to cluster images with similar embeddings together.**

***Index Terms*—latent space, embeddings, auto-encoders, clustering, mel spectrograms**

I. Introduction

The real-world data is often redundant with high dimensions. This poses challenges not only for computational efficiency but also hinders the modeling of the representation. To overcome the high dimensionality of the image data, learning latent representations for clustering and classification has been widely studied. The basic assumption here is that the high dimensional data has often a lower dimension embedding which is sufficient to represent the content of the original data. Now if we extend this concept to the image representation problem, we realize that there must exist a lower dimension space that should be sufficient to describe the content of our image dataset. This space is better known as the **latent space** in machine learning. It is a lower dimensional manifold of the high dimensional images where we expect all the similar instances of the dataset to lie in close proximity.

Representation learning is a method of finding a representation of the data – the features, the distance function, the similarity function – that dictates how the predictive model will perform. It works by reducing high-dimensional data into low-dimensional data, making it easier to find patterns, anomalies, and also giving us a better understanding of the behavior of the data altogether. It also reduces the complexity of the data, so the anomalies and noise are reduced. This reduction in noise can be very useful for supervised learning algorithms.

The issue with representation learning is that it's very difficult to get representations that can accurately capture the underlying distribution of data and separation between classes or clusters. Two major factors that usually occur in any data distribution are variance and entanglement. These two factors need to be eliminated to get a good representation from the data. Variance in the data can also be considered the sensitivity and entanglement is the way a vector in the data is connected or correlated to other vectors in the data. These connections make the underlying distribution and properties of data very complex and hard to decipher using machine learning model. Hence, it is important to learn to simplify a complex arrangement of data by creating models that are invariant and untangled.

Deep neural networks have been decent at learning representations that are invariant (insensitive) to nuisance such as translations, rotations, occlusions, and also "disentangled", or separating factors in the high-dimensional space of data. Information Bottleneck was introduced by Tishby et. al. [1]. It was introduced with a hypothesis that it can extract relevant information by compressing the amount of information that can traverse the full network, forcing a learned compression of the input data. This compressed representation not only reduces dimensions but reduces the complexity of the data as well [2]. The idea is that a network rids noisy input data of extraneous details as if by squeezing the information through a bottleneck, leaving only the features most relevant to general concepts.

Latent variables are random variables that cannot be observed directly, but it lays the foundation of how the data is distributed. Latent variables also give us a low-level representation of high-dimensional data. They give us an abstract representation of how the data is distributed. Mathematical models containing latent variables are by definition latent variable models. These latent variables have much lower dimensions than the observed input vectors. This yields a compressed representation of the data. Latent variables are basically found at the information bottleneck. Eventually, it is in this information bottleneck that we can find the abstract representation of the given high-dimensional input. The manifold hypothesis states that the high-dimensional data lies on the lower-dimensional manifold.

Autoencoders, therefore, are neural networks that can be trained for the task of representation learning. Autoencoders attempt to copy their input to its output through a combination of an encoder and a decoder. Usually, autoencoders are trained using a learning algorithm based on comparing the activation of the network of the input to the activation of the reconstructed input. Traditionally, these were used for the task

of dimensionality reduction or feature learning , but recently it has been given to understand that the autoencoders and latent variable models – models which leverage the concept of prior and posterior distribution, like Variational Autoencoders – can be used for building generative models, meaning models which can generate new data. It achieves this by compressing the information in an information bottleneck such that only important features are extracted from the entire dataset, and those extracted features or representations can be used to generate new data.

II. Related Work

The exploration of latent space embeddings and representation learning has a long history in the field of artificial intelligence and machine learning. Early work in this area focused on developing methods for discovering the underlying structure of data and representing it in a compact and meaningful way. This research has been driven by a desire to improve the performance of machine learning algorithms on a wide range of tasks, including classification, clustering, and dimensionality reduction.

One of the earliest approaches to latent space embedding was the development of principal component analysis (PCA) by Karl Pearson in 1901 [3]. PCA is a linear dimensionality reduction technique that seeks to find the directions in which the data vary the most, and represents the data in a new coordinate system defined by these directions. This approach has been widely used in many fields, including neuroscience, where it has been used to discover the underlying structure of neural activity patterns.

One common approach to exploring latent space embeddings is through the use of visualization techniques. For example, researchers have used two-dimensional scatter plots to visualize the relationships between different data points in the latent space. This can help to identify clusters or patterns in the data, and can be useful for tasks such as clustering or classification.

One well-known example of latent space embeddings is the t-SNE algorithm, which was introduced by van der Maaten and Hinton in 2008 [4]. t-SNE is a non-linear dimensionality reduction technique that maps high-dimensional data points into a lower-dimensional space by preserving the local structure of the data. This allows for the visualization of complex data structures and the identification of clusters and patterns in the data. t-SNE has been widely used in various fields, such as biology, neuroscience, and social network analysis.

An important contribution to the field of latent space embedding was the development of autoencoders by Hinton and Salakhutdinov in 2006 [5]. Autoencoders are neural network models that learn to compress and reconstruct data using a bottleneck architecture, where the input data is first encoded into a low-dimensional latent representation and then decoded back to its original space. This approach has been used in many applications, including image and speech recognition, where it has been shown to outperform other methods.

One type of autoencoder that has been widely used for latent space exploration is the vanilla autoencoder. This type of autoencoder uses a simple architecture, consisting of a single hidden layer that acts as the latent space. The encoder and decoder are trained together to minimize the reconstruction error between the input data and the reconstructed output. The resulting latent space is a compact representation of the input data, which can be used for various downstream tasks such as dimensionality reduction and data visualization.

Another type of autoencoder that has been used for latent space exploration is the convolutional autoencoder [6]. This type of autoencoder uses convolutional layers in the encoder and decoder, which allows it to learn spatial relationships between the input data. Convolutional autoencoders are particularly useful for image data, where they can learn hierarchical representations that capture the spatial structure of the input images.

A recent popular approach to learning latent space embeddings is variational autoencoders (VAEs) [7]. VAEs are a probabilistic extension of autoencoders, where the encoder and decoder are trained to model the data distribution in the latent space. This allows the VAE to generate new data samples from the latent space, and to learn more robust and disentangled representations. VAEs have been used in a wide range of applications, including image generation, natural language processing, and music generation.

Another area of research in this field focuses on the use of generative models to generate new data points in the latent space. For example, researchers have used generative adversarial networks (GANs) [8] to learn the distribution of data in the latent space and generate new data points that are similar to the original data. This can be useful for tasks such as data augmentation or generating new data for unsupervised learning.

Recent research has come up with a new type of autoencoder. A denoising autoencoder [9] is a type of autoencoder neural network that is trained to remove noise from input data. It does this by learning a representation or encoding of the input data that is robust to noise and then using this representation to reconstruct the original, noise-free input. This is accomplished through the use of a denoising function, which adds noise to the input data during training in order to improve the model's ability to remove it from the reconstructed output. Denoising autoencoders have been used for various tasks, including image denoising and anomaly detection.

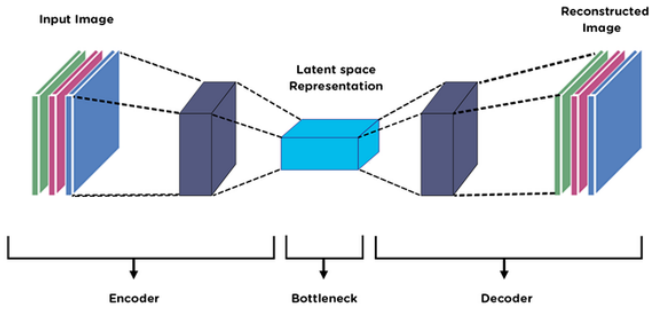
More recent research has focused on developing methods for learning disentangled representations of data, where the underlying factors of variation are explicitly separated in the latent space. This approach is motivated by the idea that disentangled representations are more interpretable and can improve the generalization ability of machine learning models.

III. Methodology

A. Image

1) Autoencoder: An autoencoder is a type of neural network that aims to learn a representation of data through a

Fig. 1: Architecture of Autoencoder



process of encoding and decoding. It consists of two main parts: an encoder and a decoder.

The encoder takes in the input data and maps it to a lower-dimensional representation, called the latent representation or the bottleneck, through a series of linear and non-linear transformations. This can be represented mathematically as:

$$\mathbf{z} = f(\mathbf{x})$$

where $\mathbf{x}$ is the input data and $\mathbf{z}$ is the latent representation.

The decoder then takes the latent representation and maps it back to the original dimensionality of the input data, producing a reconstruction of the input. This can be represented as:

$$\hat{\mathbf{x}} = g(\mathbf{z})$$

where $\hat{\mathbf{x}}$ is the reconstructed input.

The autoencoder is trained to minimize the reconstruction error between the input data and the reconstructed data, using an objective function such as the mean squared error:

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2$$

where n is the number of samples and $||\cdot||$ is the Euclidean norm.

The architecture of an autoencoder is shown in Figure 1.

In summary, an autoencoder is a neural network that learns a compact representation of the input data through a process of encoding and decoding, with the aim of minimizing the reconstruction error.

2) Triplet loss: : Triplet loss is a type of loss function that is commonly used in training deep neural networks for tasks such as face recognition or metric learning. It is based on the idea of maximizing the distance between samples from different classes while minimizing the distance between samples from the same class.

The triplet loss function is defined as the difference between the distance between a positive sample and a negative sample, and a pre-defined margin. This can be represented mathematically as:

$$\mathcal{L} = \max(d(\mathbf{a}, \mathbf{p}) - d(\mathbf{a}, \mathbf{n}) + \alpha, 0)$$

where $\mathbf{a}$ is the anchor sample, $\mathbf{p}$ is the positive sample, $\mathbf{n}$ is the negative sample, $d(\cdot)$ is a distance function such as the Euclidean distance, and α is the margin.

The triplet loss function encourages the distance between the positive and negative samples to be greater than the margin, while also encouraging the distance between the anchor and positive samples to be small. This allows the model to learn a compact and discriminative representation of the data.

In summary, triplet loss is a loss function that aims to maximize the distance between samples from different classes and minimize the distance between samples from the same class, in order to learn a compact and discriminative representation of the data.

3) Latent space: A latent space, also known as a latent feature space or embedding space, is an embedding of a set of items within a manifold in which items resembling each other are positioned closer to one another in the latent space. Position within the latent space can be viewed as being defined by a set of latent variables that emerge from the resemblances of the objects.

A latent variable is a random variable that cannot be observed directly, but it lays the foundation of how the data is distributed. Latent variables also give us a low-level representation of high-dimensional data. They give us an abstract representation of how the data is distributed.

Mathematical models containing latent variables are by definition latent variable models. These latent variables have much lower dimensions then the observed input vectors. This yields in a compressed representation of the data. Latent variables are basically found at the information bottleneck. Eventually, it is in this information bottleneck where we can find the abstract representation of the given high-dimensional input. The manifold hypothesis states that the high-dimensional data lies on the lower-dimensional manifold.

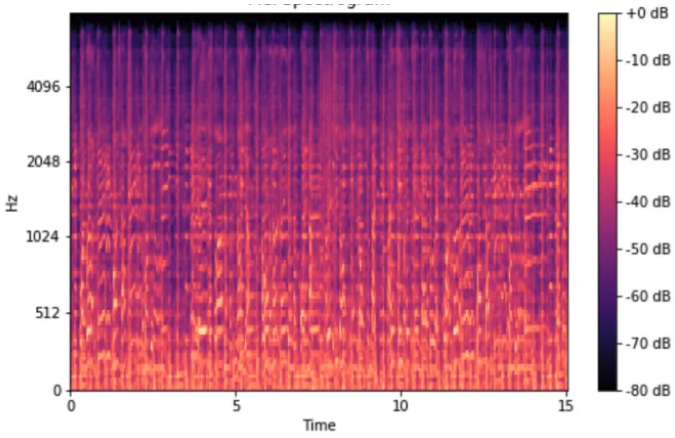
B. Audio

1) Mel spectogram: An audio signal is made up by multiple single-frequency sound waves. The amplitudes of the signal are recorded when taking samples over time. Fourier transform is a mathematical method that allows us to break down a signal into its constituent frequencies and amplitudes. It transfers the signal from the time domain to the frequency domain yielding a spectrum as the outcome. Every signal can be decomposed into a set of sine and cosine waves that sum up the original signal, making this possible. The Fast Fourier transform is an effective technique for analyzing a signal's frequency content. FFT is performed on overlapping windowed portions of the signal for non-periodic signals, yielding a spectrogram. A spectrogram is a bunch of FFTs stacked on top of each other. It is a way to visually represent a signal's loudness, or amplitude, as it varies over time at different frequencies.

Humans do not perceive frequencies on a linear scale. We are better at detecting differences in lower frequencies than

higher frequencies. The mel scale is a scale such that equal distances in pitch sounded equally distant to the listener. A mel spectrogram is obtained when the frequencies in a spectrogram are converted to the mel scale.

Fig. 2: Mel spectrogram



2) Convolutional autoencoder: A convolutional autoencoder is an autoencoder that uses convolutional layers instead of fully-connected layers in its encoder and decoder. This allows the model to learn spatial hierarchies of features from the input data, which can be useful for tasks such as image reconstruction or image translation.

The encoder of a convolutional autoencoder consists of a series of convolutional and pooling layers, which map the input data to a lower-dimensional latent representation. This can be represented mathematically as:

$$\mathbf{z} = f(\mathbf{x}) = \text{pool}(\text{conv}(\text{conv}(\mathbf{x})))$$

where $\mathbf{x}$ is the input data, $\mathbf{z}$ is the latent representation, $\text{conv}(\cdot)$ is a convolutional layer, and $\text{pool}(\cdot)$ is a pooling layer.

The decoder of a convolutional autoencoder consists of a series of upsampling and convolutional layers, which map the latent representation back to the original dimensionality of the input data, producing a reconstruction of the input. This can be represented as:

$$\hat{\mathbf{x}} = g(\mathbf{z}) = \text{conv}(\text{up}(\text{conv}(\mathbf{z})))$$

where $\hat{\mathbf{x}}$ is the reconstructed input, $\text{up}(\cdot)$ is an upsampling layer, and $\text{conv}(\cdot)$ is a convolutional layer.

The convolutional autoencoder is trained to minimize the reconstruction error between the input data and the reconstructed data, using an objective function such as the mean squared error:

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2$$

where n is the number of samples and $||\cdot||$ is the Euclidean norm.

For the audio modality the audio files were first converted to mel spectrograms and once the spectrograms against all the audios in the dataset were obtained a convolutional autoencoder was used for reconstruction in a similar way to previous section. The embeddings are generated by global average pooling the filters obtained at the bottleneck layer.

C. Text

1) LSTM: LSTMs are a type of recurrent neural network (RNN) that are particularly well-suited for modeling long-term dependencies in sequential data. These dependencies can be difficult for traditional RNNs to learn due to the vanishing gradient problem, which occurs when the gradients of the weights in the network become very small and the network is unable to learn effectively.

LSTMs address this issue by introducing additional "memory cells" and three "gates" that control the flow of information into and out of the cells. The gates are implemented as sigmoid neural networks and are trained to learn when to allow information to pass through and when to block it.

Here is the basic structure of an LSTM cell:

In this notation, x_t is the input at time step t, h_t is the hidden state at time step t, and i_t, f_t, and o_t are the input, forget, and output gates, respectively, at time step t. The memory cell $\tilde{c}_t$ is the "candidate" memory at time step t, and c_t is the actual memory at time step t.

The input gate controls the flow of input into the cell, the forget gate controls the flow of information from the previous cell, and the output gate controls the flow of information from the cell to the hidden state. The memory cell and the hidden state are both passed from one time step to the next and are used to maintain a long-term memory of the input sequence.

2) Denoising Adversarial autoencoder: A denoising adversarial autoencoder (DAE) is a type of autoencoder that is trained to denoise data. An autoencoder is a type of neural network that is trained to encode data from a high-dimensional space into a lower-dimensional space, and then decode the data back into the original space. This process is called encoding and decoding, respectively.

A DAE is trained by adding noise to the input data, and then training the autoencoder to recover the original, clean data. This is done by minimizing the difference between the denoised data and the original data. The training process is guided by an adversarial loss function, which helps the autoencoder learn to denoise the data effectively.

The architecture of a DAE typically consists of two parts: an encoder and a decoder. The encoder maps the input data to a lower-dimensional space, while the decoder maps the lower-dimensional representation back to the original space. The encoder and decoder are typically implemented as neural networks, with the encoder being a feedforward network and the decoder being a transposed feedforward network.

The training process for a DAE involves two steps. First, the autoencoder is trained to denoise the data by minimizing

the difference between the denoised data and the original data. This is done using a reconstruction loss function, which measures the difference between the denoised data and the original data.

The second step is to train the adversarial network. The adversarial network is trained to discriminate between the original data and the denoised data, while the autoencoder is trained to fool the adversarial network by producing denoised data that is as close as possible to the original data. This is done using an adversarial loss function, which measures the difference between the original data and the denoised data.

The overall objective of a DAE is to learn a denoising function that maps noisy data to clean data. This is done by minimizing the sum of the reconstruction loss and the adversarial loss. The resulting denoising function can be used to remove noise from data, and has applications in a variety of fields, including image and signal processing.

The architecture of a DAE, where x is the input data, z is the lower-dimensional representation of the data, and $\hat{x}$ is the denoised data:

$$z = f_{enc}(x)$$
$$\hat{x} = f_{dec}(z)$$

The mathematical formulation of the objective function of a DAE, where L_{rec} is the reconstruction loss, L_{adv} is the adversarial loss, and θ represents the parameters of the autoencoder:

$$\min_{\theta} L_{rec}(x, \hat{x}) + L_{adv}(x, \hat{x})$$

IV. Experiments and Results

A. Experimental Set-Up

All the models and code was implemented in Python and PyTorch. For training and evaluation of deep learning models we used the GPU backend on Google Colab. The MNIST dataset was used for ablation studies which contains images of handwritten digits divided into 10 classes. It is a widely used dataset in computer vision applications. For studies on the audio modality we use the UrbanSound8k dataset. This dataset contains 8732 labeled sound excerpts of urban sounds from 10 classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. The text modality studies are done on a dataset called Yelp reviews. It contains 10000 samples of reviews from the Yelp website. The test dataset contains 1000 samples of positive reviews and 1000 samples of negative reviews. Every sample is a English sentence with not more than 16 words.

Once the autoencoders have been trained on their respective settings we get rid of the decoder. We pass each example through the encoder to generate the embeddings at the bottleneck layer and then use these embeddings for clustering and other analysis.

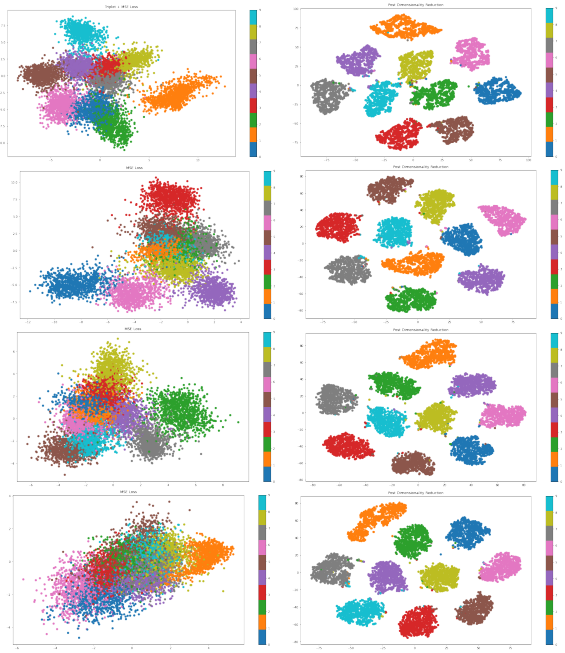


Fig. 3: Figure showing latent space representation (left images) and dimension reduced (right images) when embedding size varies.

B. Ablation study

We conducted three studies. Part (1) impact of embedding size of latent space on clustering. Part (2) impact of triplet loss on the embedding clustering. Final study is about comparison of PCA and latent space.

1) Impact of embedding size of latent space on clustering: In this experiment, we visualized the impact of change in the embedding size on the clustering of different types of images in latent space. The details are shown in Fig. 3

2) Impact of triplet loss on the embedding clustering: In this experiment, we compared the impact of using triplet loss(contrastive loss) in conjunction with reconstruction loss (MSE Loss). The metric of comparision is Silhouette Score(SS). In this we use K-Means clustering on the embeddings generated by the autoencoders latent space.

Embedding Size	SS Triplet Loss	SS Without Triplet Loss
4	0.672 ± 0.004	0.534 ± 0.001
8	0.623 ± 0.002	0.532 ± 0.028
16	0.616 ± 0.003	0.523 ± 0.011
32	0.576 ± 0.003	0.513 ± 0.006

3) Comparison of PCA and latent space: something something something In this experiment, we compared the clustering of embeddings generated using PCA and autoencoder's latent space. The metric of comparision is Silhouette Score(SS).

Embedding Size	SS PCA	SS Autoencoder Latent Space
4	0.2444	0.672 ± 0.004
8	0.185	0.623 ± 0.002
16	0.132	0.616 ± 0.011
32	0.0988	0.576 ± 0.006

C. Other results

For experiments conducted on audio modality we get similar results to the ablation studies i.e. we pass the mel spectrograms for each audio in test set to create an embedding and then use these embeddings to cluster them into their respective classes. The cluster formed for audio spectrograms are well defined. In some cases there is some overlap in two class clusters because of the similar nature of the audio like for e.g. sharp and quick nature of both dog barks and gun shots.

Our hypothesis did hold true for data in the textual modality. The denoising autoencoder was not able to produce embeddings that are clearly separable. We think this is because textual data is more complex than other modalities. Additionally, the dataset that we used is very a realistic dataset which means that it contains a lot of abnormal sentences which may neither be positive nor negative. Such samples hinder the training of the model but also help generalize the model. We computed the silhouette score of our embeddings after performing PCA with different embedding sizes. In future, we would like to experiment with more robust models and cleaner dataset.

V. Conclusion

Hence, through our experiments and use of autoencoders we found necessary proof for our hypothesis. We observe that as we increase dimension of embedding space, the clusters get well defined. We also establish that by adding contrastive loss to train the encoder, we observe that it can represent the data better, compared to just using the reconstruction loss. We show that the supervised learning approach, performs better in capturing information correlation at lower dimension. It gives better separation than other simpler dimensionality reduction methods like PCA. In the future, the authors of this study would like to try the experiments on larger scale datasets and experiment with generative methods as a next step to this project.

References

[1] Tishby, Naftali; Pereira, Fernando C.; Bialek, William (September 1999). The Information Bottleneck Method (PDF). The 37th annual Allerton Conference on Communication, Control, and Computing. pp. 368–377.

[2] Tishby, N. Zaslavsky, N. (2015). Deep Learning and the Information Bottleneck Principle..CoRR, abs/1503.02406.

[3] K. Pearson, (1901). On Lines and Planes of Closest Fit to Systems of Points in Space (PDF). Philosophical Magazine 2 (11): 559–572. doi:10.1080/14786440109462720

[4] Laurens van der Maaten, Geoffrey Hinton; 9(86):2579-2605, 2008.

[5] Hinton, G.E. Salakhutdinov, R.R.. (2006). Reducing the Dimensionality of Data with Neural Networks. Science (New York, N.Y.). 313. 504-7. 10.1126/science.1127647.

[6] Marco Maggipinto, Chiara Masiero, Alessandro Beghi, Gian Antonio Susto, A Convolutional Autoencoder Approach for Feature Extraction in Virtual Metrology

[7] Diederik P. Kingma; Max Welling, An Introduction to Variational Autoencoders, now, 2019.

[8] Goodfellow, Ian, et al. "Generative adversarial networks." Communications of the ACM 63.11 (2020): 139-144.

[9] Shen, Tianxiao, et al. "Educating text autoencoders: Latent representation guidance via denoising." International conference on machine learning. PMLR, 2020.