ISSN 2959-6157

A Comparative Study of Different Autoencoders Architectures

Zelin Zhu

Abstract:

The paper is aimed to present a comparative study of different autoencoder architectures. The introduction section provides the background of autoencoder architectures and display the motivation for conducting this study.

In the second part, I would like to systematically introduce the method I used to design and implement the experiment. This section begins with data preparation, where I describe the procedures for data acquisition and normalization to ensure the dataset's suitability for analysis. After that, I have roughly described the implement logics for each autoencoder architecture. At last, the visualization techniques are used to display the results and it can help to emphasize how these graphical representations contribute to understanding the experiment's outcomes.

In the last part, I would like to make some deepen analysis on these three autoencoder types based on the images we got on the methodology part. After that, I will discuss the advantages and disadvantages of each autoencoders, which can help us choose the most reliable autoencoder according to different dataset and conditions.

Keywords: autoencoder architectures, dataset, machine learning, visual comparison

Introduction

Background of Autoencoder Architectures

Autoencoders are neural network-based models based on unsupervised learning method that are used to find the underlying relationship between data and display data in a smaller dimension. An Autoencoder is mainly including three parts which is Encoder, Bottleneck and Decoder. Encoder is used to compresses the input data to a lower dimensional encoding and bottleneck layer is the central part of the whole autoencoder process representing the core features of the input, while the decoder reconstructs the original input from this compressed representation.

One of the key advantages of autoencoder is that it is effective in reducing the dimensionality of data. It is useful for facing a large amount of data and reduce it can improve the efficiency and lower the cost in computing. For another advantage, autoencoder can also be trained to reconstruct more cleaner data from noisy input. Through the process of denoising data, autoencoders can significantly enhance the robustness of models against the noisy input.

The main architectures of autoencoder

There are three Autoencoder architectures will be discussed in this paper, which is Standard Autoencoder, Denoising Autoencoder, Sparse Autoencoder. These Autoencoder architectures has their corresponding use cases. Each architecture serves specific rules within different fields, showcasing their versatility and effectiveness for tailored applications.

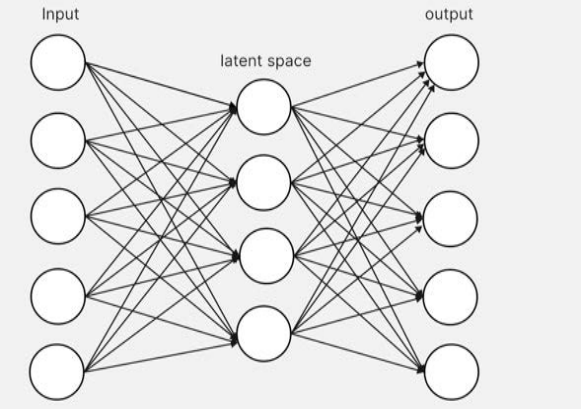

Figure 1: Standard Autoencoder Architecture

Standard Autoencoder(Figure 1) is the simplest architecture applied on the less complexity datasets and its main purpose is reducing the dimensionality and extract some features in relatively smaller datasets. The structure of Standard Autoencoder is simple, and its

straightforward structure is useful for further exploration on more complex autoencoder models.

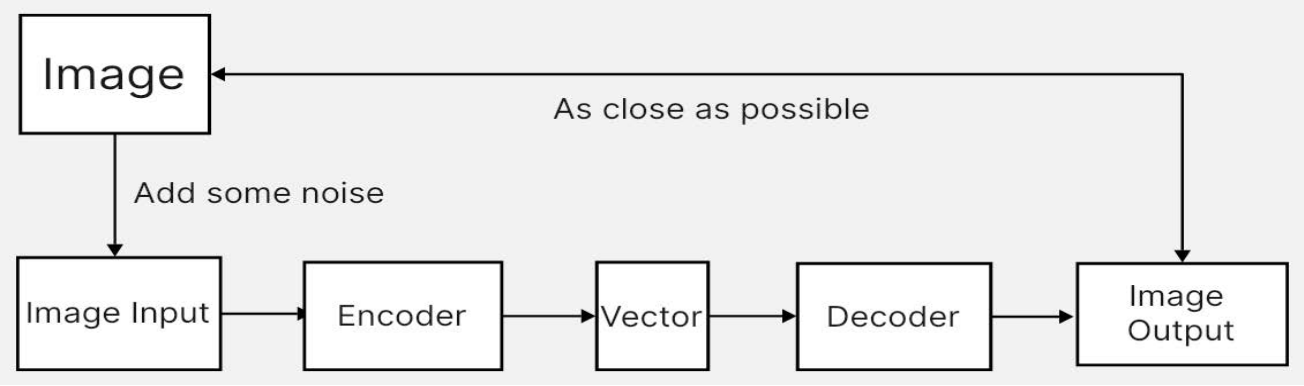

Figure 2: Denoising Autoencoder Architecture[2]

The basic principle of Denoising Autoencoder (Figure 2) is adding some noise to the input to corrupt data and mask some of the value and then Denoising Autoencoders learn to reconstruct the original input corrupted by noise. They are renowned for their ability to enhance the robustness and generalization of features learned. This property is effective in managing real-world data which contains different kinds of noises. Especially, Denoising Autoencoders is useful in images denoising tasks where the goal is to remove noise from images.

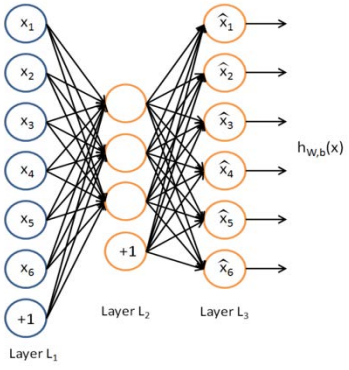

Figure 3: Sparse Autoencoder Architecture[3]

Sparse Autoencoder is also an autoencoder designed based on neural network. The purpose of Sparse Autoencoder is reconstructing the input data from learned Sparse representation. This type of autoencoder typically contains more hidden units than the input but only a few are allowed to be active at once[1]. This property is named the sparsity of network[1]. Compared with some other traditional autoencoder, the sparse autoencoder introduce a sparsity penalty term to induce the activation of hidden layer neurons to be sparser. Sparse autoencoders can capture the important features of the input data better by enforcing sparse activations of the hidden layer neurons.

Motivation of Conducting this Study

Recently, Autoencoders have acquired significant attention in the field of machine learning because of their ability to learn efficient codlings of data in an unsupervised way. These architectures are useful in various applications including dimensionality reduction, data denoising, improve the calculation efficiency. Autoencoders play an important role in the theoretically understanding and practical implementations in machine learning.

The main reason for doing the experiment arises from the need to deeply understand and evaluate the performance of different autoencoder architectures. There is existing various Autoencoder Architectures ranking from basic model like standard autoencoder architecture to more complex model like sparse or denoising Autoencoders Architectures. The motivation of conducting this study is used to compare these architectures and identifying the strengths and weaknesses of each architecture. In this way, we can choose the most suitable model for specific tasks.

This study is aimed to contribute to the field by offering insights into the practical applications and theoretical underpinnings of various autoencoder model and the ultimately goal is to foster a deeper understanding of autoencoders and to advance them in solving a real-world problem effectively.

Methodology

Methodology for Standard Autoencoder

In the study of standard autoencoder, we explored the feature representations of the Fashion MINIST dataset using an autoencoder. The dataset consists of 60000 training images and 10000 test images, each of 28*28 pixels, representing 10 different categories of apparel. Initially, the image data was normalized to [0, 1] range and reshaped into 784-dimensional vectors[4]. The design of the autoencoder included an input layer, an encoding layer, and a decoding layer, with the encoding layer compressing the input into a two-dimensional representation. I have employed the ‘relu’ activation function and ‘adam’ optimizer[4], with mean squared error as the loss function. The

model was trained over 20 epochs with a batch size of 64. After training, the encoded test set was visualized using scatter plots to illustrate the distribution in the low-dimensional feature space. Additionally, we compared the original images with their reconstructions by the decoder to access the quality of the model's reconstruction.

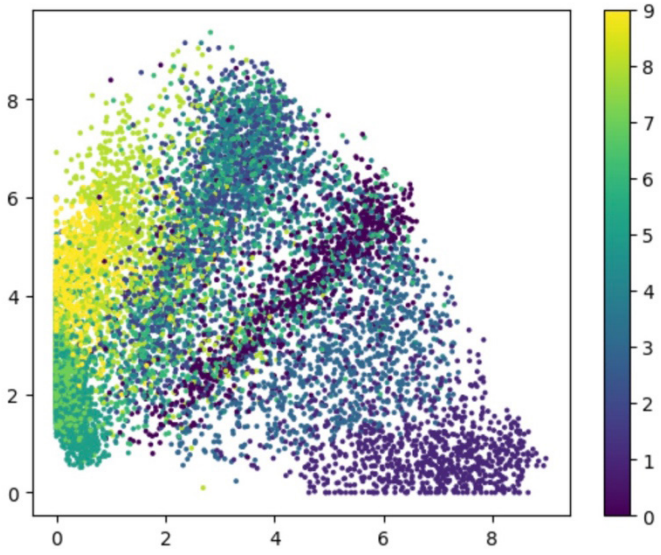

Figure 5: The Scatter Plot of the Standard Autoencoder Architecture

Figure5 illustrates the two-dimensional feature representation of the Faison MINIST test dataset obtained through our autoencoder model. Each point on the plot corresponds to an individual image, with different colors indicating the image's category, numbered from 0 to 9[4]. The plot reveals the distribution of different categories in this reduced feature space. Although some categories seem to be cluster, there is obvious overlap between most of these different categories. This overlap suggests a certain level of information loss inherent in compressing complex data into two dimensions. It can also imply the visual similarities between categories presented in the original dataset.

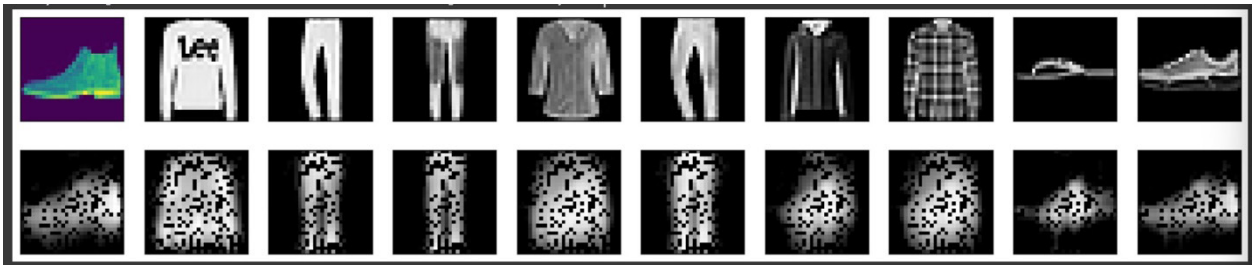

Figure 6: The Different between the original image and reconstructed image

Figure 6 displayed a detailed visual comparison between original and reconstructed images from the Fashion MNIST dataset using the simplest autoencoder model. The upper row of the figures displays a selection of ten original images. Each images represent different apparel categories including footwear, outerwear, and accessories. The autoencoder displays the corresponding reconstructed images below these original images.

The comparison reveals that the simple autoencoder can generally maintain the basic outlines and structural elements of the images. However, there is a noticeable degradation in the finer details and textures. The effect is particularly pronounced in images that originally had complex patterns or multiple colors, where the reconstructed images appear to be significantly blurred and details are less distinct.

The different impact of the reconstruction process across various types of apparel suggests that the autoencoder performs better with images having distinct and simple geometrical shapes. For example, the outlines of shoes and bags are relatively well-preserved, indicating the model's efficiency in capturing prominent features. Conversely, there is a greater challenge in some complex items such as checked shirts and detailed dresses because their intricate patterns and subtle color variations are not being as clearly replicated.

Overall, these observations are critical for accessing the capabilities and limitations of our autoencoder model,

especially in applications involving image compression or dimensionality reduction. This phenomenon indicates that there is still existing the need for further refinements in the model architecture to enhance its ability to reconstruct more complex images faithfully.

Methodology for Denoising Autoencoder

In the Denoising Autoencoder experiment, we utilized an autoencoder model to explore feature representations within the Fashion MINIST dataset and assessed the model's performance in handling noisy data. The dataset used in this experiment is the same to the study of simple autoencoder, which contains 60000 training images and 10000 test images, each with a dimension of 28*28 pixels, representing ten different categories of apparel. Initially, the image data was normalized to the [0,1] range and then reshape it into 784 dimensional vectors[4]. Subsequently, through adding some noise from a normal distribution (noise factor of 0.5), we simulated potential data quality issues encountered in real-world applications.

The architecture of the autoencoder includes multiple encoding and decoding layers, with the encoding layers gradually reducing the dimensionality to 2 dimensions, and the decoding layers progressively restoring it back to the original 784 dimensions. We used the 'relu' activation function for nonlinear transformations, and the final layer employs 'tanh' to match the normalization range of the data[4]. The model was trained using the 'adam' optimizer[4], with mean squared error as the loss function, over a training period of 20 cycles.

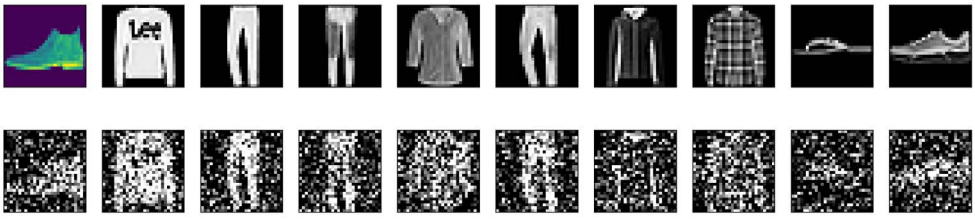

Figure 7: After adding some Noise to the Autoencoder

Figure 7 illustrates the impact of noise on images from Fashion MINIST dataset. The upper row of the figure displays a selection of original images, showcasing various apparel items in clean form. The lower row demonstrates the same images after the introduction of Gaussian noise, providing a clear visualization of how noise can degrade image quality.

Detailed Analysis

First, the first row in figure 7 is the original images from the Fashion MINIST dataset, representing different categories of apparel such as shoes, tops, and pants, each depicted in a

28*28-pixel resolution. These images serve as a baseline for accessing the impact of noise on visual clarity.

The second row presents images that have been subjected to Gaussian noise. This manipulation simulates potential real-world interferences, illustrating significant visual degradation, particularly in terms of detail and texture preservation. In the noise-added images, the noise is mainly manifested as random black and white speckles that cause interference in the details and structure of the image. The effect of noise is especially noticeable in areas with more detail such as the texture and edges of the garment.

The reason why we add some noise to the image input is that we hope to increase robustness of the autoencoder. In the real world, data is often affected by noise for a variety of reasons. For example, images may be corrupted during capture, transmission, or storage. By adding noise to the training process, these real-world situations can be simulated, providing the model with a training environment that is closer to reality.

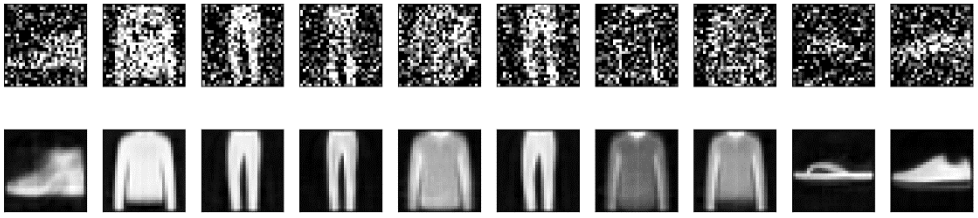

Figure 8: Decoding the Noised Images

The images in the top row have been intentionally degraded with Gaussian noise to simulate typical real-world disturbances that images might suffer during acquisition or transmission. The added noise blurs details and alters the texture and edges of fashion items, making them difficult to recognize.

The bottom row shows the result of the denoising self-encoder's processing. It is clear that the autoencoder re-

moves a great deal of noise effectively, restoring clarity and detail to the image. While some slight traces of noise may still be present, key features such as contours, shapes and major details are well preserved.

And then, we can make a denoising quality assessment. The denoising quality was assessed by comparing the noisy images with their reconstructed versions. The comparison results show that the self-encoder is able to recover most of the original content, demonstrating the efficiency of the model in filtering noise while retaining key features. This is particular evident in the recovery of details and the maintenance of overall structure.

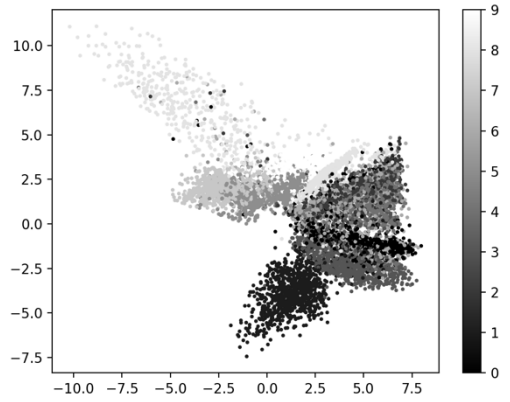

Figure 9: The Scatter Plot of the denoising Autoencoder Architecture

Figure 9 displays how the denoising autoencoder compresses the high-dimensional data (784 dimensions) into a two-dimensional space. This compression can help us to visualize the underlying structure and clustering behavior of different apparel types in a more easily interpretable form. Different colors in the scatter plot represent different categories of the Fashion MNIST dataset. For instance, similar types of apparel may cluster together, indicating that the autoencoder preserves some semantic meaning of the original data after encoding. The distances between clusters can provide insights into how distinctly the autoencoder has learned to represent different categories. Closer clusters suggest categories with visually similar characteristics, whereas widely separated clusters could indicate distinct apparel types. The denoising quality is evaluated by comparing the noisy images with their reconstructed versions. The comparison results show that the autoencoder is able to recover most of the original content, demonstrating the efficiency of the model in filtering noise while preserving key image features. This is particularly evident in the recovery of details and the maintenance of the overall structure.

Methodology for Sparse Autoencoder

In the sparse autoencoder development consisting of an encoder and a decoder. The encoder is composed of three fully connected layers with decreased neurons respectively, where the final layer compresses the data into a two-dimensional representation. Each layer employs the relu activation function and the first encoding layer includes L1 regularization to enhance the sparsity of the model[4]. The decoder mirrors the encoder's architecture, culminating in an output layer that utilizes the tanh activation function to reconstruct the original input data. The training of the autoencoder was performed using the Adam optimizer and MSE loss function. The model processed several epochs of training on the entire dataset with batch sizes of 64 samples. Additionally, we shuffled the data during training to decrease some related dependencies that could potentially influence the model's performance. To access the model's performance and the learned feature representations, we used two visualization techniques: Firstly, we encoded samples from the test set using the trained encoder and displayed the two-dimensional encoded results in a scatter plot to observe the distribution of different digit categories within the encoded space. Secondly, we selected several test images along with their reconstructions made by autoencoder. It provides a direct measure of the model's reconstruction quality through visually comparing and evaluating the similarity between the original and reconstruction images.

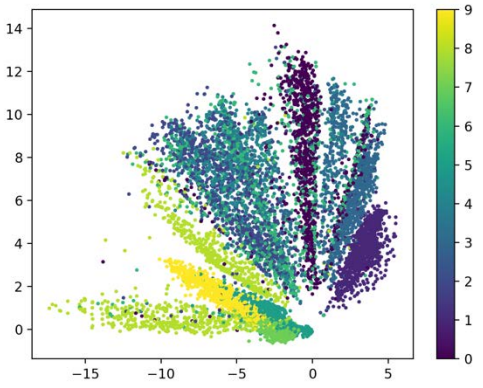

Figure 10: The Scatter Plot of Sparse Autoencoder

The images (Figure 10) represent a scatter plot of the two-dimensional encoded representations of images from the fashion minist dataset, which is achieved through a trained sparse autoencoder. Each point in the scatter plot corresponds to an individual image, and the color coding indicates the type of apparel represented by each image, as designated by the dataset labels from 0 to 9. This visualization not only demonstrates the clustering ability of

the self-encoder, but also illustrates how well the coded features discriminate between different apparel types.

Distinct clusters suggest that the autoencoder effectively captures and separates the intrinsic features of different apparel types in the dataset. For example, denser clustering may indicate apparel types with more consistent and distinctive characteristics, while more dispersed clustering may reflect greater variability within categories. Relative positions and distances between clusters can provide valuable insights into the relationships and similarities between different apparel categories. Categories that are positioned closer together might share visual or stylistic similarities that the autoencoder has learned to encode similarly.

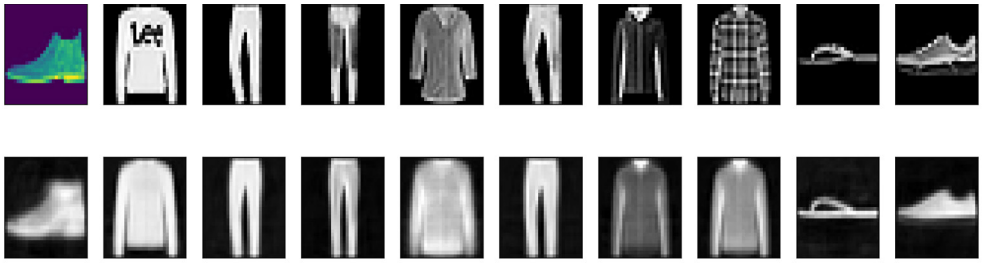

Figure 11: The Differences between Original Image and Reconstructed Image

After observing the Figure 11, we can see that the image contains two rows of visual displays of fashion items from the fashion minist dataset. The top row contains the original image, and the bottom row shows the reconstructed image after processing by sparse autoencoder. Each column corresponds to a specific item and each column demonstrates the effect of the autoencoder in recovering the original image after the encoding and decoding process.

Although the reconstructed images are slightly blurred in the bottom row, the general shape and key structural elements of the clothing and accessories are preserved. This suggests that the autoencoder, even after significant dimensionality reduction, is still able to capture the key visual features needed to recognize and differentiate between different fashion items.

There are also existing some visible differences in texture and detail definition between the detail and reconstructed images. These differences highlight areas where the model may have lost information, particularly in finer detail and edge definition. However, the overall integrity of the item shapes was well preserved, indicating that the main architectural features were effectively encoded and decoded.

Deepen Analysis for the Different Autoencoder Architecture and Get a Conclusion

In this analysis, we compare three different types of autoencoders: standard autoencoder, denoising autoencoder, sparse autoencoder. In the following, I will compare these three autoencoders based on the code and imaged displayed above and discuss their applications, strengths and weaknesses, and their respective characteristics.

First, Standard Autoencoder are often utilized for dimensionality reduction and feature learning in an unsupervised learning condition. Secondly, these models learn compressed representations of data which can be useful for tasks like weight initialization and noise reduction.

Their straightforward architecture is easy to implement and train, which is appropriate to revealing the underlying structure of data. However, they may struggle with noisy data and can't capture complex features as effectively as more specialized models.

Denoising Autoencoder excel in applications where data integrity is compromised by noise. These models enhance their generalization by training to recover the original data from noisy inputs. In this way, they are valuable in real-world applications because noise is prevalent in real-world. However, their training process is more complex and sensitive to the nature of noise, so it requires more higher precision to make sure optimal performance.

Sparse Autoencoder aims to enforce sparsity in data representations, making them suitable for tasks that require feature selection and compression sensing. The incorporation of sparsity-inducing regularizations like L1 encourages the model to focus on the most significant features, thereby improving interpretability. While they provide robust feature representations, tuning their regularization parameters is critical and challenging.

The images provided serve as visual evidence of the functionality of the respective encoders. Images processed through the Standard Autoencoder show basic reconstruction capabilities but may lack detail accuracy. In contrast, the denoising autoencoder effectively reduces noise as evidenced by the clear image, which shows significant improvement over the original image. Finally, although there is a significant data compression, the sparse autoencoder maintains key visual information, which demonstrates the effectiveness of its sparse constraints.

In conclusion, the choice between these autoencoder types should be guided by specific application needs and data characteristic.

References

[1]“ML | Auto-Encoders,” *GeeksforGeeks*, Jun. 21, 2019. https://www.geeksforgeeks.org/auto-encoders/
[2]from lecture of Lihong Yi
[3] A. Ng, “CS294A Lecture notes Sparse autoencoder.” Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf
[4]Nana0606, “autoencoder/autoencoder at master · Nana0606/autoencoder,” *GitHub*, 2018. https://github.com/Nana0606/autoencoder/tree/master/autoencoder.