Isaac Kobby Anni

CS7300_F24_MidTerm_Practicals

October 2024

1 Super-resolution with autoencoder

1.1 Data Preprocessing

The task uses the MNIST dataset, which consists of grayscale images of handwritten digits with a resolution of 28x28 pixels. The images were downsampled to a resolution of 14x14 pixels to simulate a low-resolution scenario. For noise we have Gaussian or salt-and-pepper noise to enhance robustness during training. However, in this implementation, only downsampling was applied. All images were normalized to values between 0 and 1 to improve training stability.

1.2 Model Design

A Convolutional autoencoder was chosen because it is well-suited for image data due to its ability to capture spatial hierarchies and features effectively.

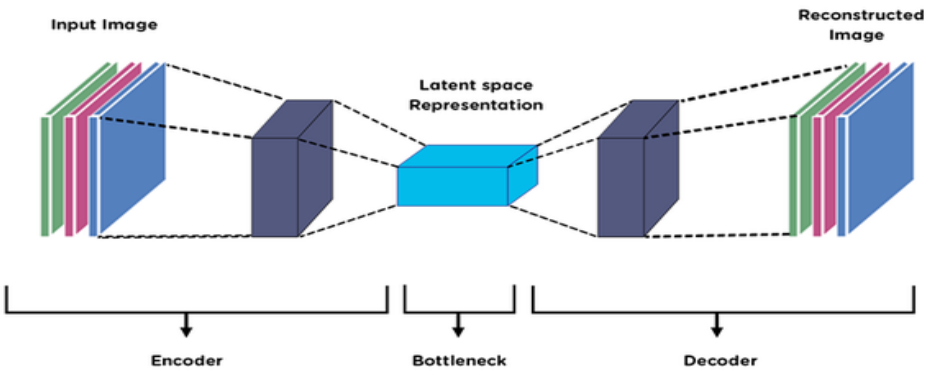


Figure 1: Convolutional autoencoder architecture; *image by Deepak Birla*

The encoder compresses the 14×14 low-resolution image into a smaller latent space by applying a sequence of convolutional layers. The layers progressively reduce the spatial dimensions from 14×14 to 4×4, which captures key patterns and structures within the images.

The decoder reconstructs the images back to a 40×40 size using transposed convolutions (upsampling) and and then crop it to the original 28×28 resolution. The ReLU activation function was used to

introduce non-linearity and help the model learn complex patterns, except for the output layer, where Sigmoid is applied to ensure pixel values range between 0 and 1.

1.3 Training the Model

The Mean Squared Error (MSE) loss was used to minimize the difference between the high-resolution targets and the model's outputs. This is suitable for pixel-level image reconstruction tasks. The Adam optimizer was selected for its adaptive learning rate capabilities, improving convergence. The model was trained for 5 epochs, as shown in the training curve plot below. The training and validation loss curves indicate that the model's loss consistently decreases, showing good convergence without significant signs of overfitting. The small gap between the training and validation loss further confirms that the model generalizes well.

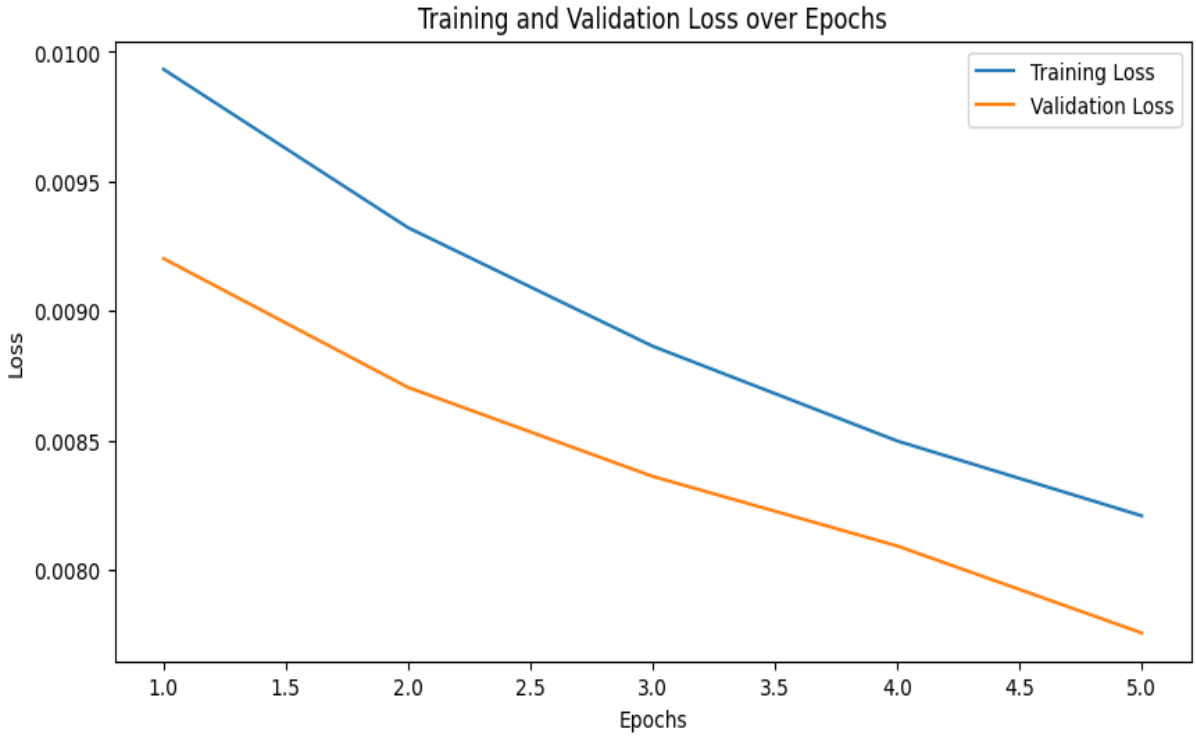


Figure 2: Training curve for convolutional autoencoder

1.4 Results and Visualization

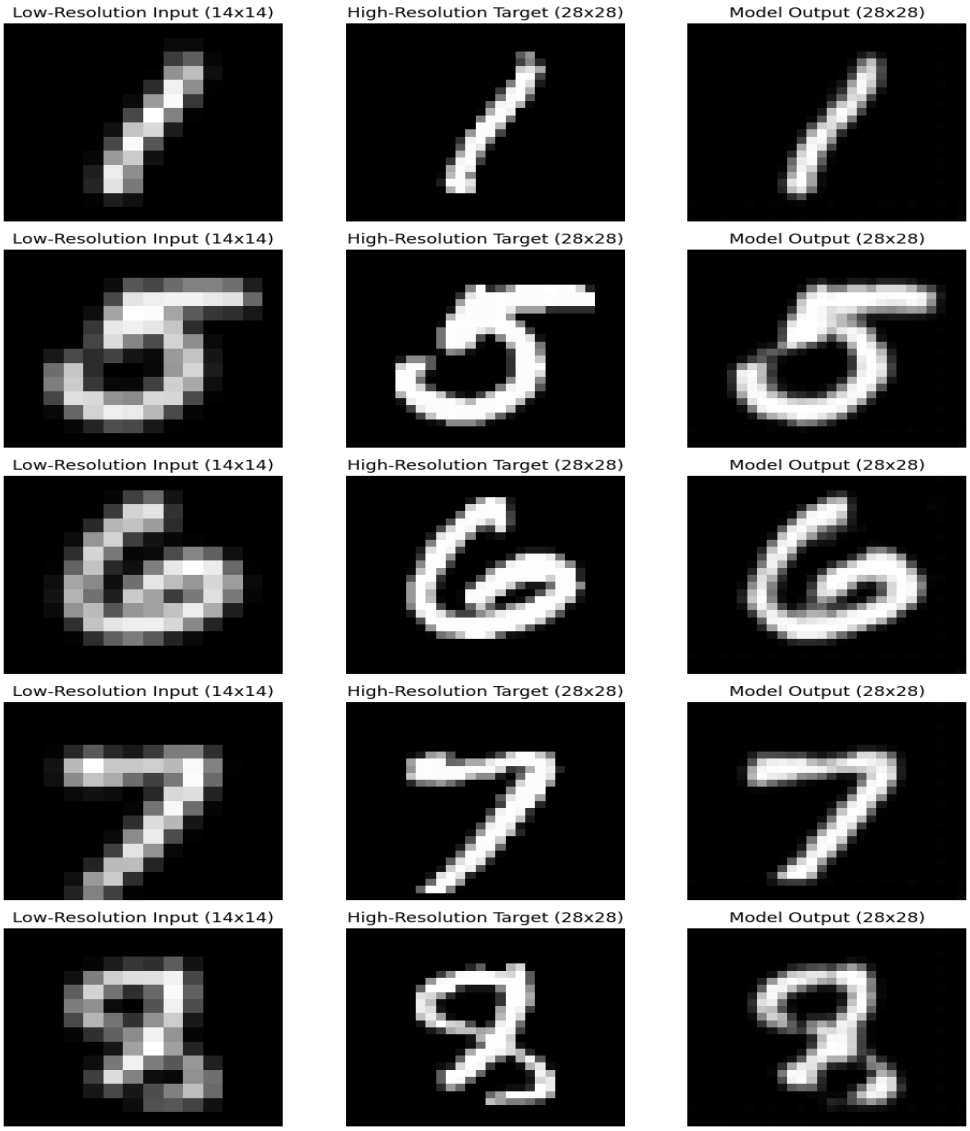


Figure 3: Results for convolutional autoencoder

The output visualization, shown above, shows five (5) examples of low-resolution inputs, high-resolution targets, and the model's reconstructed images. The reconstructed images closely resemble the targets, indicating that the autoencoder successfully enhanced the resolution. There is some noticeable blurring, which is common in autoencoders due to the pixel-level reconstruction approach. Further tuning or using a denoising autoencoder could help sharpen the outputs.

1.5 Insights from this implementation

- **Effectiveness:** The convolutional autoencoder approach demonstrated effectiveness in enhancing image resolution, producing outputs closely matching the high-resolution targets.

- **Model Choice Justification:** Convolutional layers were essential for learning spatial hierarchies, making them a better choice compared to fully connected layers, especially for image data.
- **Room for Improvement:** Future iterations could explore contractive or variational autoencoders to introduce regularization and encourage the model to learn more robust features.
- Alternatively, adding different noise during training could help the model generalize better and reduce blurring artifacts in the output.

2 Dimensionality Reduction Using PCA and Autoencoders

2.1 Data Preprocessing

The dataset used is Fashion MNIST, consisting of grayscale images of fashion items like shirts, shoes, and bags, etc, all at a resolution of 28x28 pixels. The dataset was split into training and test sets. No additional noise was added, and the images were normalized to have values between 0 and 1 for consistent scaling across all samples.

2.2 Dimensionality Reduction Techniques

I used KMeans clustering to evaluate the performance of dimensionality reduction techniques (PCA and autoencoder) on the Fashion MNIST dataset. KMeans is an unsupervised clustering algorithm, making it a suitable choice when working with reduced-dimensionality data. Since both PCA and autoencoder are unsupervised techniques, using another unsupervised method like KMeans helps maintain consistency in evaluating the results without relying on labeled data. In this implementation, KMeans serves as a baseline clustering method for assessing the effectiveness of the dimensionality reduction methods.

2.2.1 Principal Component Analysis (PCA) implementation:

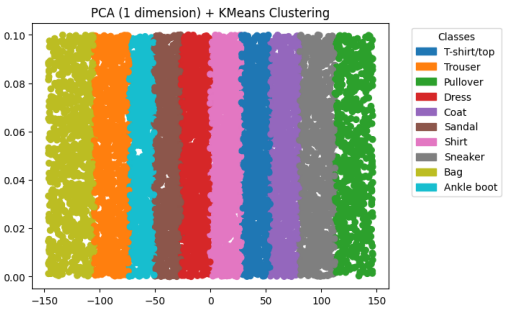
Used as a baseline dimensionality reduction technique. PCA was applied to reduce the dimensions of the dataset to 1, 2, and 3. The clustering performance of the reduced dimensions was evaluated using the Silhouette Score, a metric that measures the consistency of clusters. Table 1 below summarizes the results for the score pca and autoencoder side by side. For each dimensionality reduction (1, 2, and 3), the data was clustered using KMeans and plotted.

2.2.2 Autoencoder Implementation

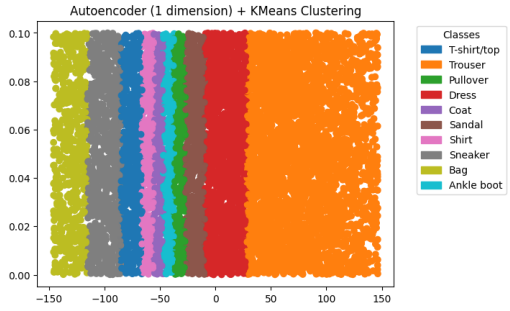
Convolutional Autoencoder: A neural network designed to learn efficient encodings from the images without supervision. A Convolutional Autoencoder was developed to automatically learn compressed representations of the Fashion MNIST data. The encoder compresses the data to a lower dimension, while the decoder reconstructs the data to its original shape. The same experiment as with PCA was conducted, reducing the dimensions to 1, 2, and 3 and visualizing the clustering results.

2.3 Comparison and Analysis

The autoencoder consistently outperformed PCA across all dimensions based on the Silhouette Score. This indicates that the autoencoder was more effective at capturing the underlying structure of the Fashion MNIST dataset and clustering it meaningfully. For both PCA and the Autoencoder, the clustering patterns in the reduced dimensions were plotted, results as shown below side by side for the two approaches.

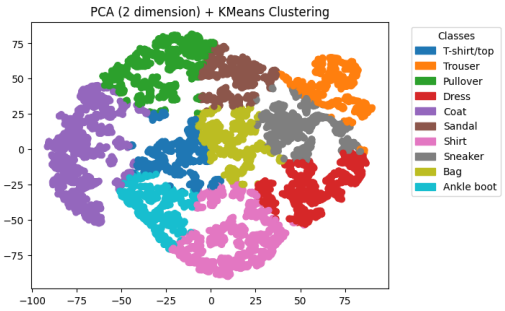


(a) PCA + KMeans Clustering (Dimension 1)

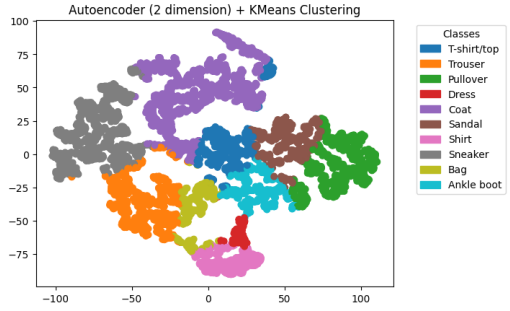


(b) Autoencoder + KMeans Clustering (Dimension 1)

Figure 4: Comparison of Clustering Results

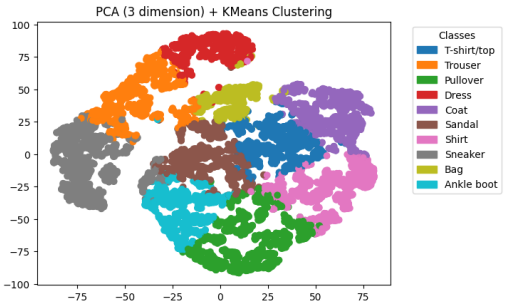


(a) PCA + KMeans Clustering (Dimension 2)

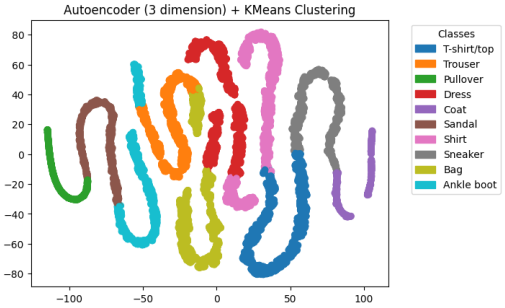


(b) Autoencoder + KMeans Clustering (Dimension 2)

Figure 5: Comparison of Clustering Results



(a) PCA + KMeans Clustering (Dimension 3)



(b) Autoencoder + KMeans Clustering (Dimension 3)

Figure 6: Comparison of Clustering Results

The PCA visualizations show more uniform separations but with less flexibility in the cluster shapes. In contrast, the autoencoder visualizations display more varied and organically shaped clusters, suggesting that the autoencoder captured more nuanced features of the data.

Dimensionality	**PCA Silhouette Score**	**Autoencoder Silhouette Score**
1	0.5317	0.6386
2	0.4028	0.5150
3	0.3476	0.5165

Table 1: Comparison of Silhouette Scores for PCA and Autoencoder

The higher Silhouette Score for the autoencoder in all dimensionalities suggests that the autoencoder learned a more compact and meaningful representation of the data. This implies that autoencoders are suitable for extracting latent features that better represent the original data's structure than linear methods like PCA.

2.4 Conclusion

While effective as a baseline, PCA's linear nature limits its ability to capture complex data patterns, as evidenced by lower Silhouette Scores in higher dimensions. Onn the other hand, autoencoder approach demonstrated its strength in non-linear transformations, allowing it to cluster the data more effectively. Its flexibility in learning complex features led to better clustering performance as shown in the visualizations and Silhouette Scores.

2.5 Future work

Additional experiments could explore tuning the architecture and hyperparameters of the autoencoder to further enhance performance. Another avenue could be combining PCA and autoencoders to exploit the strengths of both methods.