Convolutional Autoencoders and Denoising Autoencoders on Fashion-MNIST: A Comparative Analysis

Student Name: Gabriel Lucky Lotanna
Student ID: 24070357
Module: Machine Learning and Neural Networks

1. Introduction

Autoencoders are neural networks that compress data into a compact form and then reconstruct it. Rather than predicting class labels, they instead reproduce their input. This forces the model to discover structure and regularities in the data. As a result, autoencoders are useful for dimensionality reduction, feature learning, anomaly detection, and denoising.

This report investigates convolutional autoencoders (CAEs) and convolutional denoising autoencoders (CDAEs)using the Fashion-MNIST dataset. The work is implemented in Python using Keras/TensorFlow, and all figures are generated from the accompanying notebook, conv_autoencoder_fashionmnist_research-2.ipynb. The aims of the project are to:

- Design and train a convolutional autoencoder to reconstruct Fashion-MNIST images
- Extend the model to a denoising autoencoder that removes additive noise.
- Analyse reconstruction quality, reconstruction-error distributions and latent-space structure
- Present results through clear, publication-style visualisations
- Critically evaluate the strengths and limitations of both models.

The report is structured as follows. Section 2 provides a brief background on autoencoders and related work. Section 3 describes the dataset and preprocessing steps. Section 4 outlines the model architectures and training setup. Section 5 presents experimental results linked to Figures 1–8. Section 6 discusses the findings, and Section 7 concludes with possible extensions.

In this tutorial, you will learn how a convolutional autoencoder processes Fashion-MNIST images and how adding noise affects the reconstruction quality.

2. Background and Related Work

An autoencoder consists of two main components: an encoder that maps an input xx to a latent representation zz, and a decoder that reconstructs an approximation x^x^ from zz. The model is trained to minimise a reconstruction loss, commonly mean squared error (MSE):

L(x,x^)= // x−x^ // 2L(x,x^)= // x−x^ // 2

By constraining the latent space to be lower-dimensional than the input, or by applying regularisation, the network is encouraged to learn informative features rather than simply copying the data. Hinton and Salakhutdinov (2006) showed that

autoencoders can perform competitive nonlinear dimensionality reduction and improve downstream classification.

Traditional autoencoders are fully connected networks and treat an image as a flat vector. This ignores spatial locality and is inefficient for image data. Convolutional autoencoders use convolutional and pooling layers. These extract local patterns and preserve spatial structure (Masci et al., 2011). They are well-suited to vision tasks and usually produce sharper reconstructions.

A denoising autoencoder reconstructs a clean target from a corrupted input. Noise is injected, for example, by adding Gaussian noise. The model learns to reverse the corruption (Vincent et al., 2008). Denoising autoencoders thus learn robust representations and can act as effective image denoisers.

In this project, convolutional versions of both standard and denoising autoencoders are implemented on Fashion-MNIST, allowing a direct comparison between clean reconstruction and denoising performance.

3. Dataset and Preprocessing

The experiments use the Fashion-MNIST dataset. It contains 70,000 grayscale images of clothing items from 10 classes (e.g. T-shirt, trousers, coat, boots). Each image has a resolution of 28×28 pixels. Fashion-MNIST features more complex textures and shapes than handwritten digits, making it a more challenging benchmark.

The dataset is loaded via the Keras API and split into 60,000 training and 10,000 test images. Preprocessing steps are minimal but important:

- Pixel values are converted to float32. They are then normalised to the range [0,1] by dividing by 255.
- A singleton channel dimension is added. Images now have shape (28, 28, 1), which is required by convolutional layers.

Figure 1 displays a grid of 12 random training samples to provide intuition for the data's variability and visual complexity.

Figure 1 – Example Fashion-MNIST training images.

Figure 1 – Example Fashion-MNIST training images

4. Model Architectures and Training Setup

4.1 Convolutional Autoencoder (CAE)

The convolutional autoencoder operates directly on the 28×2828×28 image grid.
The encoder consists of:

- Conv2D layer with 32 filters of size 3×33×3, ReLU activation, padding='same'
- MaxPooling2D layer with 2×22×2 window, reducing spatial size to 14×1414×14
- Conv2D layer with 64 filters, again 3×33×3, ReLU activation
- MaxPooling2D layer with 2×22×2, giving a latent feature map of size 7×7×647×7×64

This 7×7×64 tensor acts as the latent representation. Keeping it as a structured feature map rather than flattening it into a vector lets the decoder use spatial information.

The decoder mirrors the encoder:

- Conv2D layer with 64 filters, ReLU
- Upsampling2D layer to upscale back to 14×1414×14
- Conv2D layer with 32 filters, ReLU
- Upsampling2D layer to 28×2828×28
- The final Conv2D layer uses 1 filter and a sigmoid activation to produce reconstructed images in [0,1].

The CAE is compiled with the Adam optimiser and MSE loss. Training uses:

- 20 epochs
- Batch size of 256
- 20% of the training set is reserved for validation

4.2 Convolutional Denoising Autoencoder (CDAE)

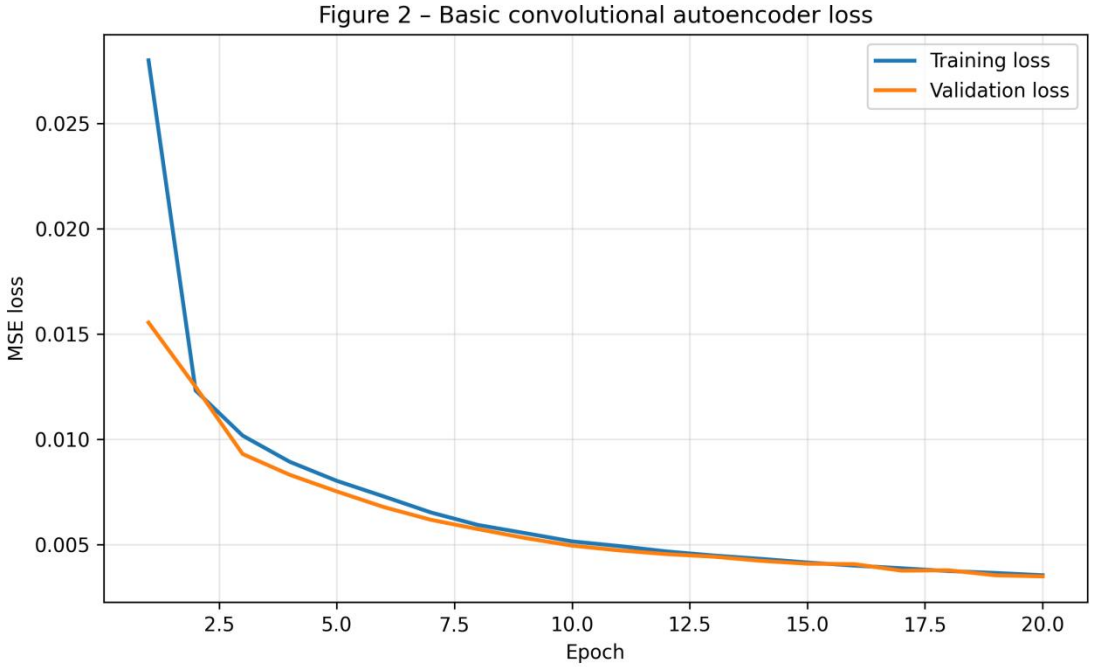
he CDAE uses the same architecture as the CAE but trains on noisy–clean pairs. Gaussian noise with a noise factor of 0.4 is added to the inputs, and the model learns to reconstruct the clean images. It is trained for 20 epochs using the same batch size and loss function as the basic CAE.
All visualisations use clear colours and labels to support accessibility for readers.

5. Results

5.1 Basic Convolutional Autoencoder

Training behaviour.
Figure 2 shows the training and validation loss curves for the CAE.

Figure 2 – Training and validation MSE loss for the basic convolutional autoencoder.

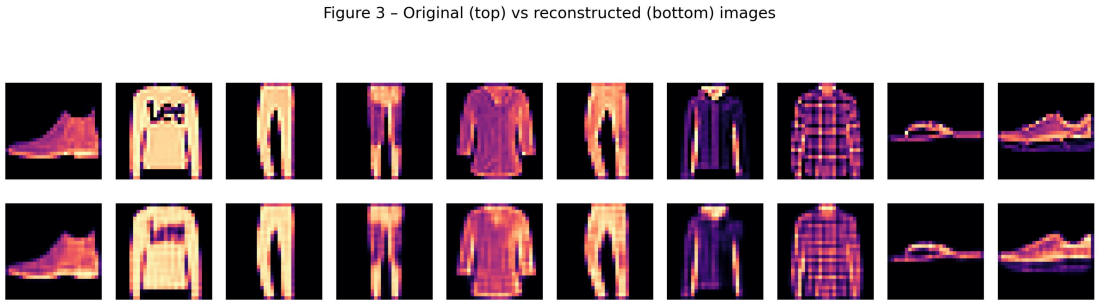


Both curves decrease smoothly over 20 epochs and remain close to each other. This indicates stable optimisation and limited overfitting. The final validation loss is only slightly higher than the training loss, suggesting the model generalises reasonably well to unseen data.

Reconstruction quality.

Figure 3 compares ten original test images with their reconstructions.

Figure 3 – Original (top row) vs reconstructed (bottom row) Fashion-MNIST test images using the CAE.

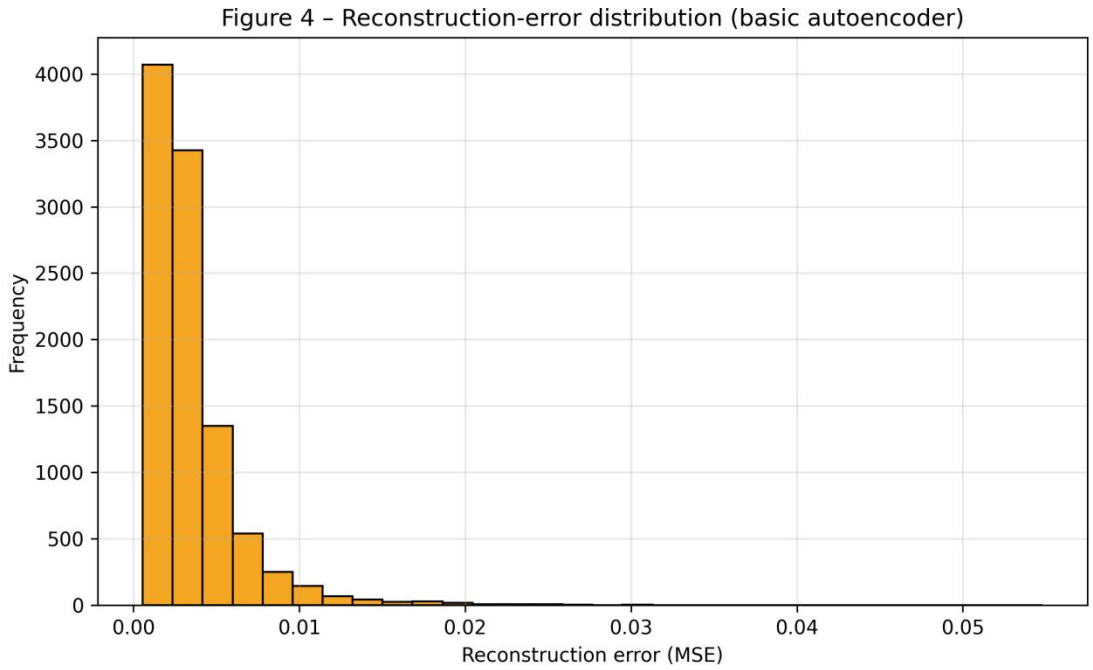


The reconstructions preserve the global silhouettes and class-specific shapes (e.g., the outlines of shoes, trousers, and shirts). The fine-grained texture is somewhat smoothed, as expected, because the bottleneck compresses information and the loss function penalises pixel-wise deviations rather than perceptual differences. Nevertheless, the outputs remain visually recognisable, showing that the CAE has learned a meaningful latent representation.

Reconstruction-error distribution (basic AE).

To quantify reconstruction performance, the per-image MSE between input and reconstruction is computed on the test set. Figure 4 displays the histogram of these errors.

Figure 4 – Distribution of reconstruction errors for test images using the basic CAE.



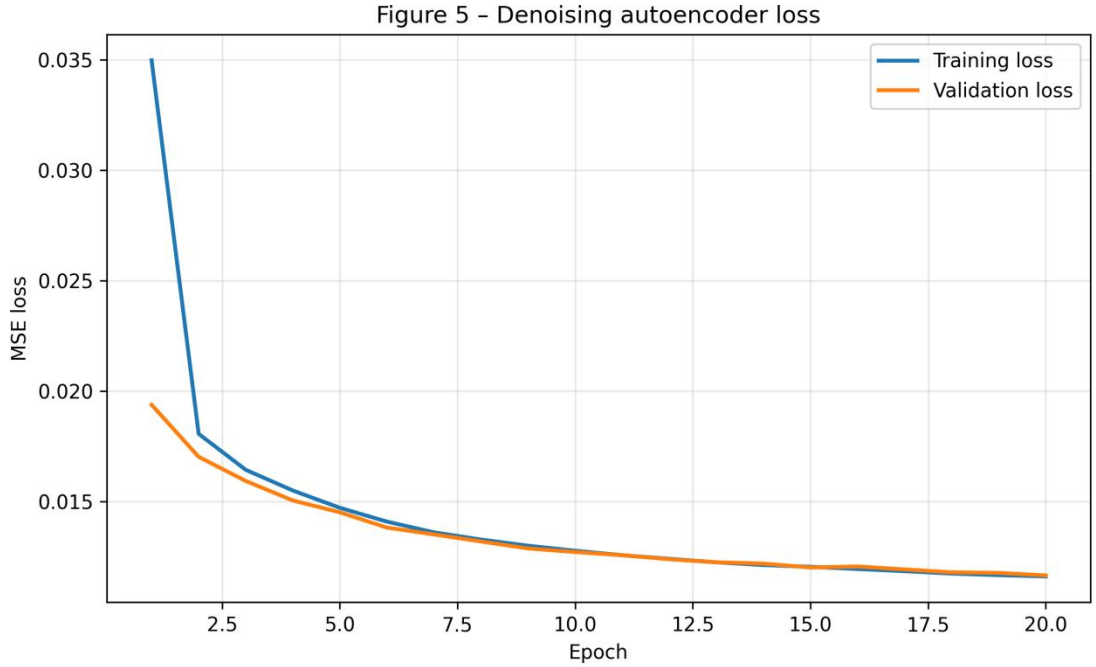
The distribution is right-skewed. Most images have very low error, while a smaller number show higher error. This suggests the model reconstructs typical, easy images well but struggles more with unusual shapes or borderline examples between classes. There are no extreme outliers. This supports the visual impression that reconstructions are generally reasonable.

5.2 Convolutional Denoising Autoencoder

Training behaviour.

The denoising autoencoder faces a more challenging task: mapping noisy inputs back to clean images. Figure 5 shows its training and validation loss.

Figure 5 – Training and validation loss for the convolutional denoising autoencoder.

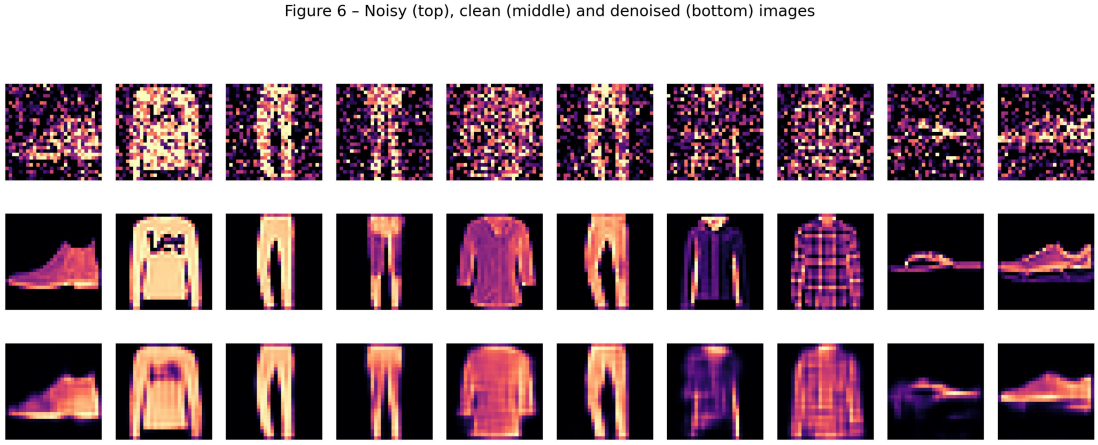


Both curves decrease steadily. The gap between the training and validation losses remains modest. The overall loss values are higher than in the basic CAE since the task is harder. Still, the shape of the curves shows successful learning without severe overfitting.

Noisy vs clean vs denoised images.

Figure 6 illustrates the denoising effect by showing, for a subset of test images, the noisy input, the clean target and the reconstructed (denoised) output.

Figure 6 – Example triplets of noisy inputs (top), clean targets (middle) and denoised outputs (bottom) from the CDAE.



The added Gaussian noise significantly corrupts the inputs, yet the CDAE recovers clear silhouettes and much of the original structure. In many cases, the denoised images visually resemble the clean targets, with noise largely removed from background regions. Some very fine details are not fully recovered, but the qualitative performance demonstrates that the model has learned to separate signal from noise.

5.3 Latent-Space Analysis

Autoencoders are prized for reconstruction and for the structure of their latent representations. To explore this (**see Figure 7**), the encoder part of the CAE is used to obtain latent feature maps, which are flattened and projected into two dimensions using PCA.

These 7×7×64 tensors are flattened and projected into two dimensions using principal component analysis (PCA).

Figure 7 – PCA projection of the CAE latent space, coloured by true class label.

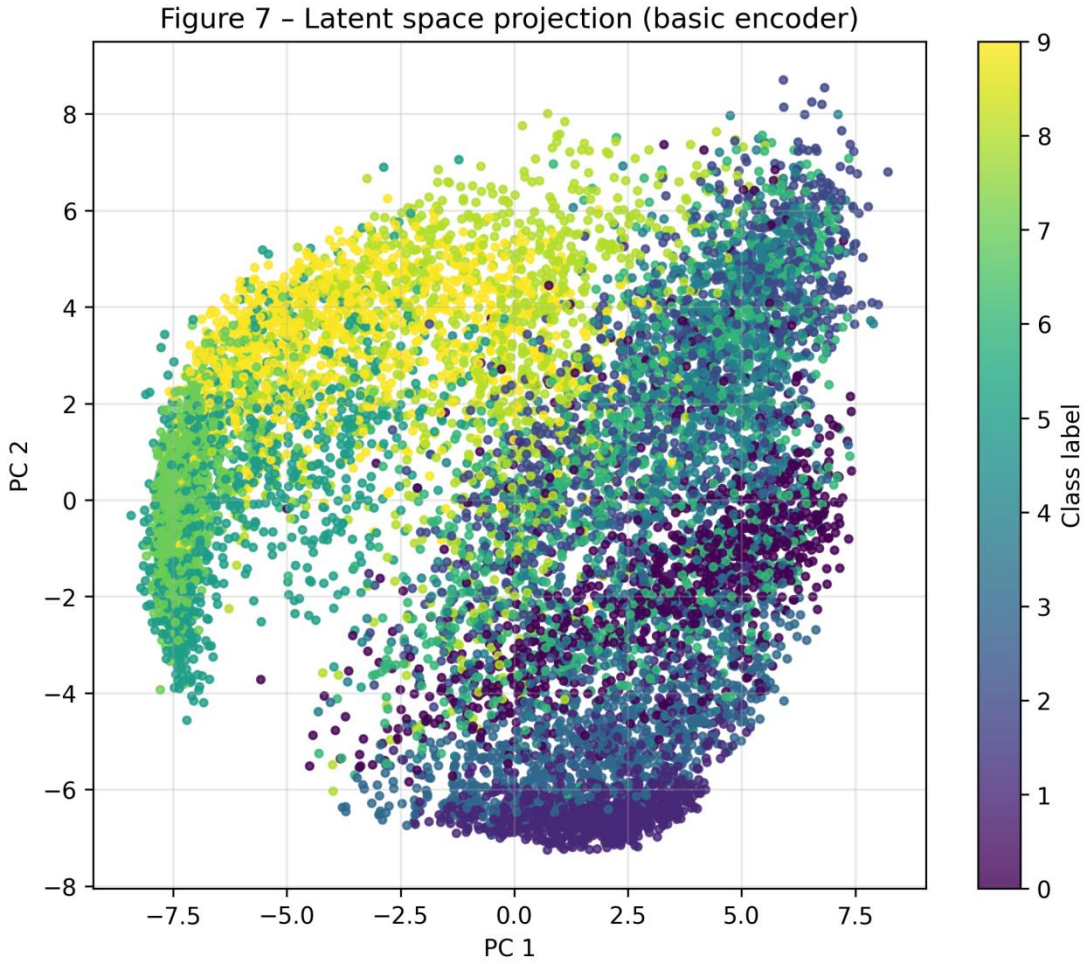
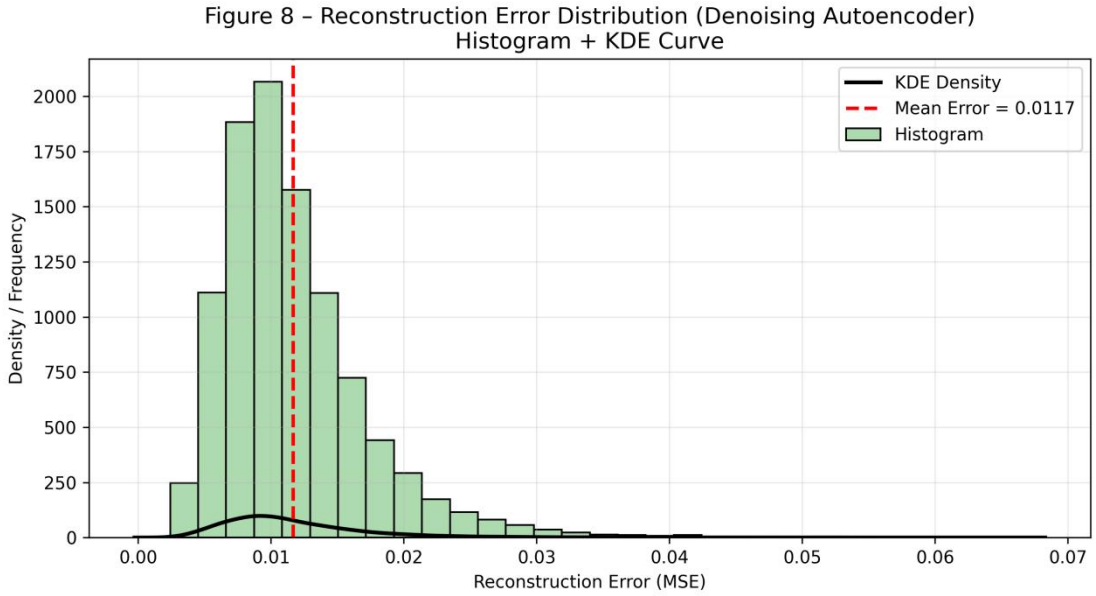


Figure 7 shows that, even though the autoencoder was trained without labels, different clothing categories form partially separated clusters in the 2D projection. This implies that the latent space has captured class-relevant information. Items with similar visual appearance (e.g. different types of footwear) tend to lie closer together, while more distinct categories occupy different regions.

5.4 Reconstruction Error for the Denoising Autoencoder

Finally, reconstruction errors are analysed for the CDAE. For each test image, the MSE between the clean target and the denoised output is computed. Figure 8 provides a detailed view of the error distribution using both a histogram and a kernel density estimate (KDE).

Figure 8 – Reconstruction-error distribution for the CDAE, showing histogram, KDE curve and mean error.



The histogram reveals a concentration of errors around a relatively low value, with a long but thin tail towards higher errors. The KDE curve provides a smooth approximation of the distribution, making it easier to see the mode and the distribution's general shape. A dashed vertical line marks the mean error, which lies slightly to the right of the mode due to the skewness. Compared with Figure 4, errors are generally higher for the denoising model, which is expected given the additional difficulty of reconstructing clean images from heavily corrupted inputs. However, the distribution is still narrow enough to confirm that the CDAE performs effectively.

6. Discussion

The experiments highlight several important points about convolutional and denoising autoencoders on the Fashion-MNIST dataset.

Firstly, the **convolutional architecture** clearly benefits image reconstruction. By preserving spatial structure through convolution and pooling, the CAE produces sharper, more realistic outputs than a fully connected autoencoder would. The smooth training and validation curves (Figure 2) show that the model has sufficient capacity to fit the dataset without overfitting.

Secondly, the **denoising extension** significantly improves robustness. Training on noisy–clean pairs forces the CDAE to focus on essential structure and ignore noise. This is evident in Figure 6, where large amounts of Gaussian noise are removed while preserving the overall shape. At the same time, the higher error distributions (Figure 8 compared with Figure 4) remind us that the denoising task is inherently more difficult.

Thirdly, the **latent-space analysis** confirms that the encoder has learned meaningful structure. The unsupervised latent features, when projected with PCA, form clusters that align reasonably well with class labels (Figure 7). This suggests that the autoencoder could be used as a feature extractor for downstream tasks such as classification or clustering.

There are, however, some limitations. The models were trained with relatively simple architectures and a fixed noise level, so there is room to explore deeper networks, alternative regularisation techniques (e.g. dropout or sparsity constraints) and more realistic noise patterns. Additionally, MSE is a purely pixel-wise metric and does not always align with human perception; perceptual losses or adversarial training could potentially produce sharper, more detailed reconstructions.

7. Conclusion

This project implemented and evaluated a convolutional autoencoder and a convolutional denoising autoencoder on the Fashion-MNIST dataset. The CAE achieved good reconstruction quality on clean images, while the CDAE demonstrated strong denoising ability, recovering clear silhouettes from heavily corrupted inputs. Analysis of reconstruction-error distributions and PCA projections of the latent space showed that both models learned meaningful and structured representations.

Overall, the results confirm that convolutional autoencoders are powerful tools for unsupervised representation learning on image data. The denoising variant, in particular, offers robustness to noise and could be valuable in applications such as image restoration or anomaly detection. Future work could extend the approach to variational autoencoders, deeper convolutional stacks, or real-world datasets such as medical images, where unsupervised feature learning is especially valuable.

References

Masci, J., Meier, U., Cireşan, D. and Schmidhuber, J. (2011)

'Stacked convolutional auto-encoders for hierarchical feature extraction', Proceedings of the International Conference on Artificial Neural Networks (ICANN), pp. 52–59.

Vincent, P., Larochelle, Y., Bengio, Y. and Manzagol, P. (2008)

'Extracting and composing robust features with denoising autoencoders', Proceedings of the 25th International Conference on Machine Learning (ICML), pp. 1096–1103.