Neural Networks – Project 1. Autoencoders

Jorge Chamorro Pedrosa - 100496527, Mario Coronado Fernández - 100496637

1. Network Creation and Validation.

The objective was to optimize a basic autoencoder by tuning key parameters from the statement – projected dimensions and no. of layers – and intrinsic of a basic autoencoder – hidden layer dimensions and learning rate –.

We tried different combinations for {3, 5} hidden layers for {15, 30, 50 and 100} projected dimensions and compare the results using MSE loss as our metric, since between the we measure differences between original and reconstructed images for each pixel.

1.1 MNIST Dataset

Projected Dimensions	*3 Hidden Layers*	*5 Hidden Layers*
15 Dimensions	*0.061*	*0.082*
30 Dimensions	*0.044*	*0.078*
50 Dimensions	*0.040*	*<u>0.077</u>*
100 Dimensions	*<u>0.039</u>*	*0.078*

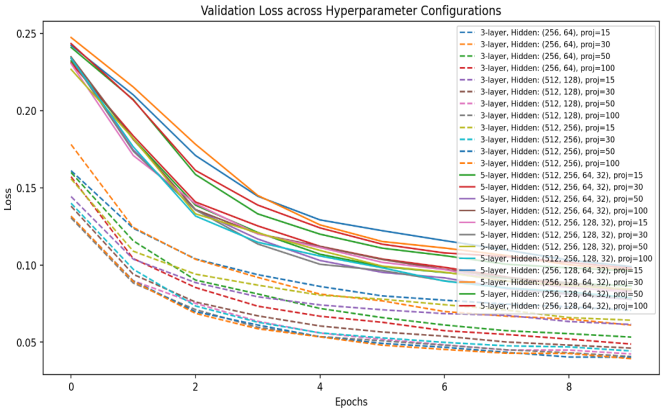

Table 1.1 & Figure 1.1: Mse Validation Loss for different architectures with 10 epochs

As we can clearly observe the best architecture for MNIST dataset is composed by 3 hidden layers {512, 128} and 100 projected dimensions, giving us a validation loss of 0.039.

1.2 FMINST Dataset

Projected Dimensions	*3 Hidden Layers*	*5 Hidden Layers*
15 Dimensions	*0.055*	*<u>0.071</u>*
30 Dimensions	*0.049*	*0.070*
50 Dimensions	*<u>0.047</u>*	*0.069*
100 Dimensions	*0.047*	*0.069*

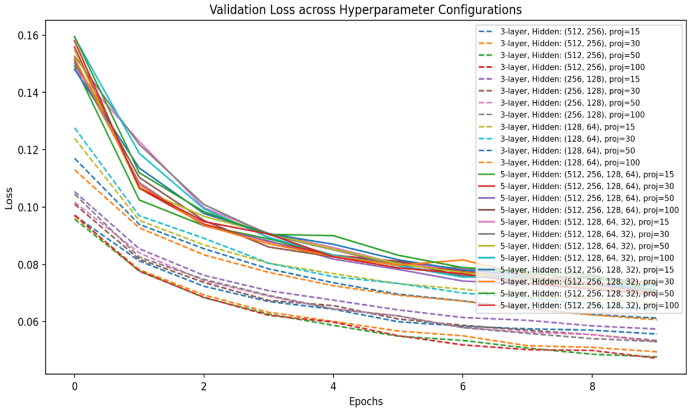

Table 1.2 & Figure 1.2: Mse Validation Loss for different architectures with 10 epochs

From our experiments, it is evident that simpler architectures with three hidden layers often outperform their five-layer counterparts. This suggests that for relatively simple image datasets like MNIST and FMNIST, excessive depth may introduce unnecessary complexity without meaningful performance gains.

For computational reasons, a subset of the training and validation data was used for initial hyperparameter optimization.

Once we found the best configuration for the subset, the model was retrained using the full dataset to assess final performance:

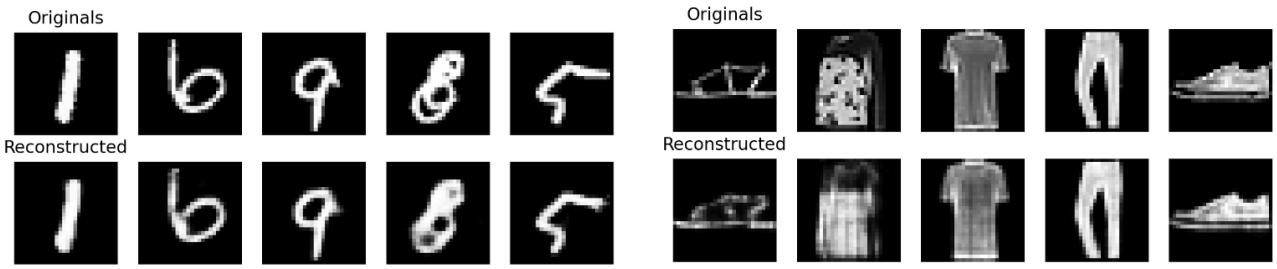

Figure 1.3: Reconstruction of both Mnist (left) and Fminst (right) datasets with the best architecture found for both of them

When trained with the complete train set, the final validation loss obtained after 30 epochs was:

Dataset	*Training Loss*	*Validation Loss*
MNIST	*0.013*	*0.016*
FMNIST	*0.024*	*0.027*

Table 1.3: Training and Validation Loss obtained for both datasets after 30 training epochs with the best architectures founded.

2. Regularization Techniques.

Now we tried different regularization techniques to see their effect on the performance of our best architecture. Concretely we tried L1 (lasso), L2 (ridge) and dropout regularization techniques.

MODEL	**MSE**
Non-Regularized (3 layers)	*0.040*
L1 Regularization (3 layers)	*0.078*
L2 Regularization (3 layers)	*0.041*
L1 & L2 Regularization	*0.077*

*Table 2.1: **(MSE)** validation loss for different autoencoder models with and without regularization. For MINIST Dataset*

Ridge regularization was a good try for our model, however our non-regularized model is slightly better. And if compared with L1 and L1 & L2 combined, our model outperforms both of them.

MODEL	**MSE Loss**
Non-Regularized (3 layers)	*0.047*
L1 Regularization (3 layers)	*0.065*
L2 Regularization (3 layers)	*0.049*
L1 & L2 Regularization	*0.064*

*Table 2.2: **(MSE)** validation loss for different autoencoder models with and without regularization. For FMNIST Dataset*

In FMINST dataset occurs exactly the same as in the other dataset, L2 regularization is a very solid option but our non-regularized architecture continues being the best selection.

Dropout is effective in preventing overfitting. However, in autoencoders, this technique worsens the quality of the model. This is because, as we can see in the basic autoencoder, it does not exhibit overfitting. Our results indicate that L2 regularization slightly improves performance, while L1 regularization significantly worsens it. This suggests that forcing sparsity in the latent space (L1) may overly constrain the model.

3. Denoising Autoencoder.

A denoising autoencoder was trained by injecting Gaussian noise into the input images. The effectiveness of denoising was evaluated based on PSNR:

- Lower noise variance resulted in better reconstructions, as expected.
- At higher noise levels, the model struggled to recover finer details, leading to lower PSNR values.

The training results for the MNIST dataset demonstrated significant improvement over the epochs showing no signs of overfitting. The PNSR reflected this by increasing from 17 to 22 at a noise level of 0.2.

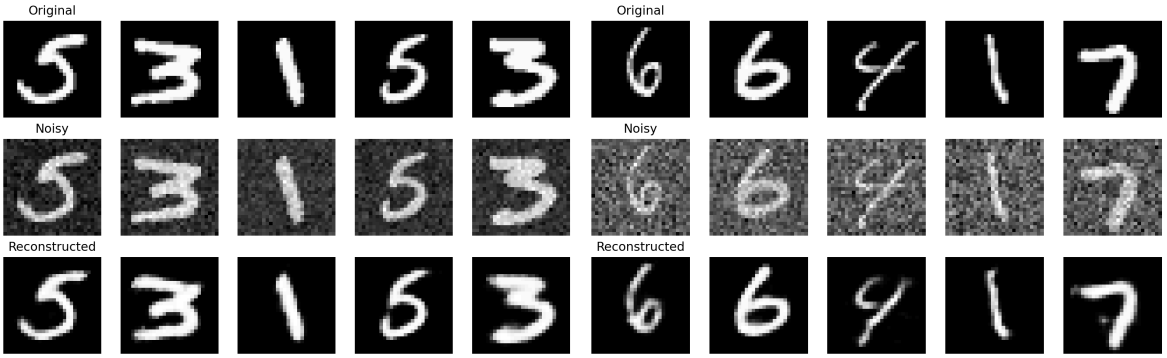

Figure 3.1: MNIST denoising results after training the best model. The left represents noise 0.2, and in the right we have it for 0.5.

For the FMNIST dataset, the PNSR doesn't exhibit much lower values, hovering around the same values in the validation set through all the epochs (20 dB). This is surprising since it has higher details in its images. While less detailed, reconstructions remain accurate.

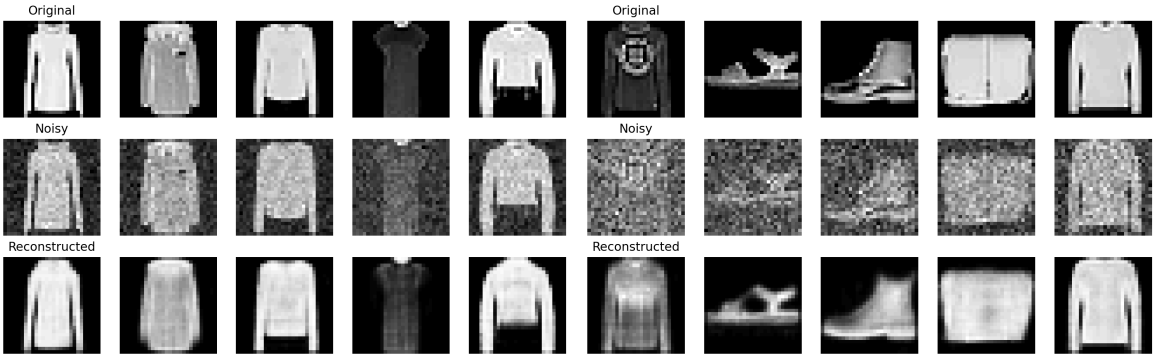

Figure 3.2: FMNIST denoising results after training the best model. Left figures are for noise 0.2 and the right ones are for added noise 0.5.