Universidad Carlos III de Madrid
Bachelor's in Data Science and Engineering

Autoencoders

Neural Networks

Name:
Carlos Casanova Romero
Ricardo Vazquez Alvarez

Professor:
Pablo Martinez Olmos

March 11, 2024

Dense Neural Network

In this project, our aim is to find the most suitable auto-encoder for the MNIST and FMNIST databases using linear neural networks. Our exploration involves experimenting with auto-encoders of different architectures, specifically ones with 3 layers in both encoding and decoding, and others with 5 layers both encoding and decoding stages. Throughout this process, we go through various data transformations, validation strategies, and regularization techniques to optimize the training process.

Initially, we download the MNIST and FMNIST datasets and preprocess the data by converting it to tensors. Notably, we do not normalize the images as the pixel intensities fall within the optimal range of 0 to 1. This range aligns with the sigmoid activation function to be used in the final layer of decoders. Subsequently, we partition the data into training, testing, and validation sets for both datasets.

Looking for efficiency, we implement a function to allocate models and data to either the GPU or CPU, depending on the availability of GPU resources. This approach allows us to use CUDA and maximize training speed.

Architecture

When defining the Autoencoder model, we opt for simplicity by utilizing the Sequential library. The model begins with input and output layers set to 28x28, corresponding to the image size.

- For the 3 layer neural network we use the first hidden layer to be of 256 units, followed by a second hidden layer with 128 units.
- For the 5 layer encoding and decoding network, we define the hidden dimensions to be of 512, 256, 128 and 64 units.

These architecture choices were determined to be optimal for both the MNIST and FMNIST databases through a process of trial and error. Throughout the model, we employ a ReLU activation functions for each transformation. In the final layer, we utilize the Sigmoid activation function to constrain output values between 0 and 1, mirroring the pixel intensities.

For training the Autoencoder, we employ the Mean Squared Error (MSE) loss function as our criterion, coupled with the Adam optimizer using a default learning rate of 0.001.

Latent Dimension

Determining the appropriate latent dimension between the Encoder and Decoder poses a challenge during the model training phase. To address this challenge, we experiment with various latent dimensions: 15, 30, 50, and 100.

During training and validation, we iterate through each latent dimension and train the model for 20 epochs. We compare the training and validation loss scores to identify the latent dimension that minimizes the loss function most effectively for the validation set. This iterative process allows us to discern the optimal latent dimension that best captures the underlying features of the data while avoiding overfitting.

1.1 Regularization

Regularization is a technique used in order to mitigate the loss produce by overfitting a model. Firstly, what was done in this project is to see the evolution of the performance of the Neural Network as a function of epochs, but without regularization (in the validation and test set). In order to validate whether or not we did produce overfitting, the results should show a much worse performance in the validation set in comparison to the training set performance (if there is overfitting). Another clear indication of overfitting is when the validation set performance worsens as we increase epochs in training.

We then compared the results to the Regularized Models which included the **dropout technique** after each ReLU function in the model. This dropout function is a specific form of regularization that, provided its hyper-parameter p (probability), it assigns this probability to each node and essentially makes it so that each node has this probability of deactivating (dropping out). This allows for the creation of a more diverse set of networks within the layer which is done after each epoch (training stage).

To further enhance our regularization methods, we decided to implement L2 regularization using the Adam optimizer with the added weight decay. However, this approach significantly deteriorated the model's performance. Consequently, we opted to discontinue the use of this regularization method and instead continue only with the dropout technique.

The following Table 1.1 shows the validation loss obtained for each model type (Regularized and Non Regularized, for the different layers) and latent dimensions (values spanning from 15 to 100). Then in the last column one can see the best latent dimension obtain (lowest testing loss). The top table is for the MNIST data set, meanwhile the bottom is for the FMNIST data set.

1.2 Results

The following table shows the validation set loss results after training for 20 epochs in the Non Regularized models, meanwhile for the Regularized models it is for 15 epochs. It is important to note that the Naive Neural Networks used converged in a much more stable fashion to their respective losses meanwhile the regularized versions did not and were more sporadic. This evolution of the loss can be seen in the graphs shown in the **Appendix, Table 3.1**.

Auto-encoder type	Latent Dimensions				Best Latent dimension
	15	30	50	100	
Non-Regularized 3 layers	0.0236	0.0186	0.0262	0.0171	100
Non-Regularized 5 layers	0.0449	0.0315	0.0352	0.0379	30
Regularized 3 Layers	0.0299	0.115	0.0331	0.1154	15
Regularized 5 Layers	0.0638	0.1142	0.1240	0.1269	15

Auto-encoder type	Latent Dimensions				Best Latent dimension
	15	30	50	100	
Non-Regularized 3 layers	0.0221	0.0179	0.0165	0.0172	50
Non-Regularized 5 layers	0.0261	0.0222	0.0250	0.0223	30
Regularized 3 Layers	0.0266	0.0311	0.0327	0.0314	15
Regularized 5 Layers	0.0613	0.0712	0.2049	0.0716	15

Table 1.1: Validation Loss for the chosen models and latent dimensions. For the MNIST data set (**Top**), FMNIST data set (**Bottom**). Using 20 epochs for the non regularized and 15 epochs for the regularized mdoels.

With the achieved results we determined that it was beneficial to use regularization. Obtaining the best models architectures: **MNIST db:** 3 layers with 15 latent dimensions. **FMNIST db:** 3 layers with 15 latent dimensions.

We utilized the test set as input to the trained models to visualize the results for the input images.

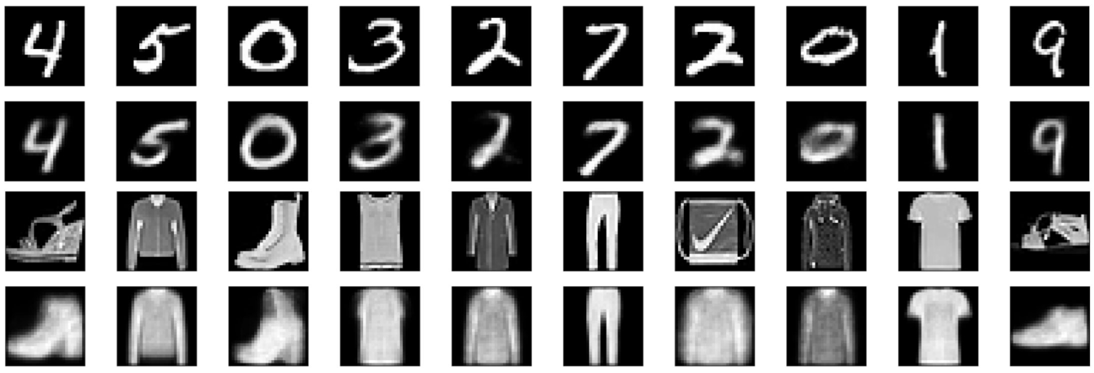


Figure 1.1: MNIST (top) and FMNIST (bottom) results after training the autoencoder and reconstructing the images with regularization (without noise).

Denoising Autoencoder

The objective of a Denoising Autoencoder is to reconstruct clean images from noisy input images. In defining the Autoencoder model for both the MNIST and FMNIST databases, we leverage our best-performing model obtained from previous processes to establish our denoising autoencoder. The key distinction lies in training the model with additional noise. To achieve this, we implement an add_gaussian_noise function, which introduces Gaussian noise to our training images. Consequently, the model learns to reconstruct these noisy images, enhancing its ability to denoise input data.

The metric used to measure performance is the Peak Signal-to-Noise Ratio (PSNR), which relies on the Mean Squared Error (MSE) loss score to quantify the reconstruction quality of images. A higher PSNR indicates better reconstruction quality.

The training will be conducted using the same architecture with regularization for 20 epochs. After training, we will visualize the results by assessing the reconstruction of images using the test set. This evaluation will enable us to observe how various types of noise affect the reconstruction process and the robustness of the linear denoising autoencoder.

2.1 Results

The training results for the MNIST database demonstrated significant improvement over the epochs, showing no signs of overfitting. The PSNR increased from around 14 at the start to approximately 17 in the validation set, using a training noise level of 0.5

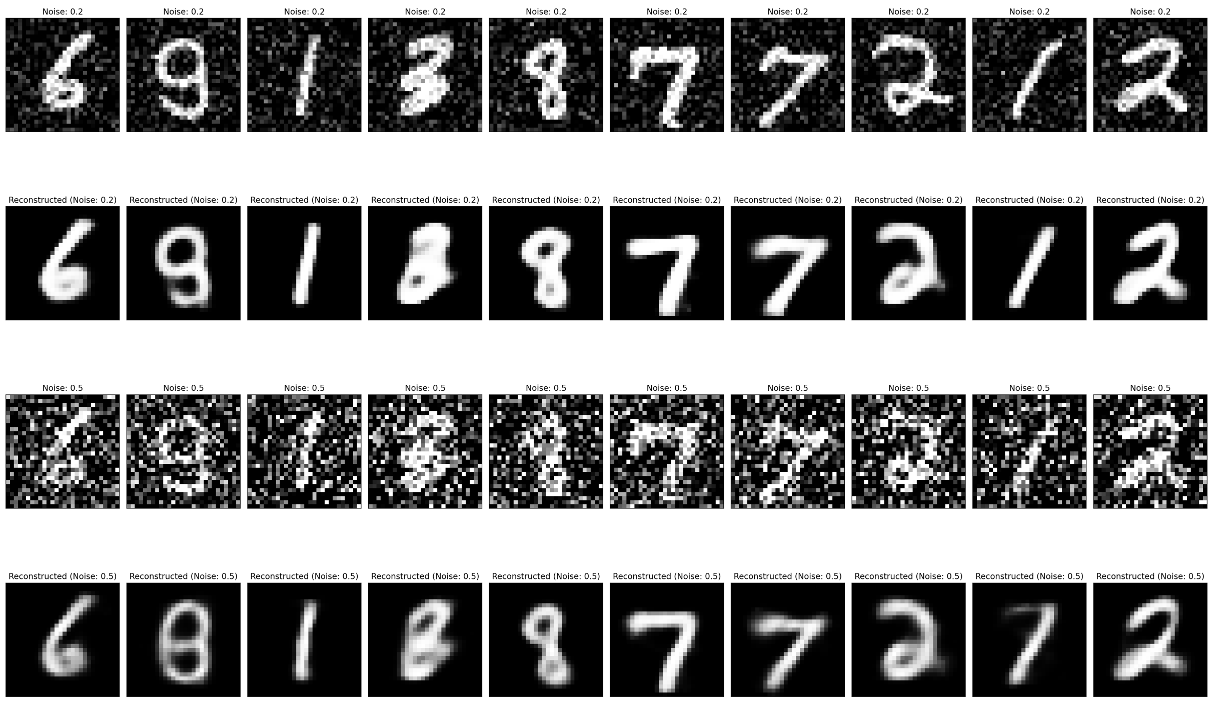

Noise: 0.2
Noise: 0.2
Noise: 0.2
Noise: 0.2
Noise: 0.2
Noise: 0.2
Noise: 0.2
Noise: 0.2
Noise: 0.2
Noise: 0.2
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Reconstructed (Noise: 0.2)
Noise: 0.5
Noise: 0.5
Noise: 0.5
Noise: 0.5
Noise: 0.5
Noise: 0.5
Noise: 0.5
Noise: 0.5
Noise: 0.5
Noise: 0.5
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)
Reconstructed (Noise: 0.5)

Figure 2.1: MNIST denoising results after training the best model. First two rows are for noise of 0.2 (noisy image is above). The last two rows are for added noise of 0.5.

For the FMNIST database, the PSNR exhibited lower values, hovering around 15 in the validation set throughout all epochs. While less detailed, the image remains accurate.

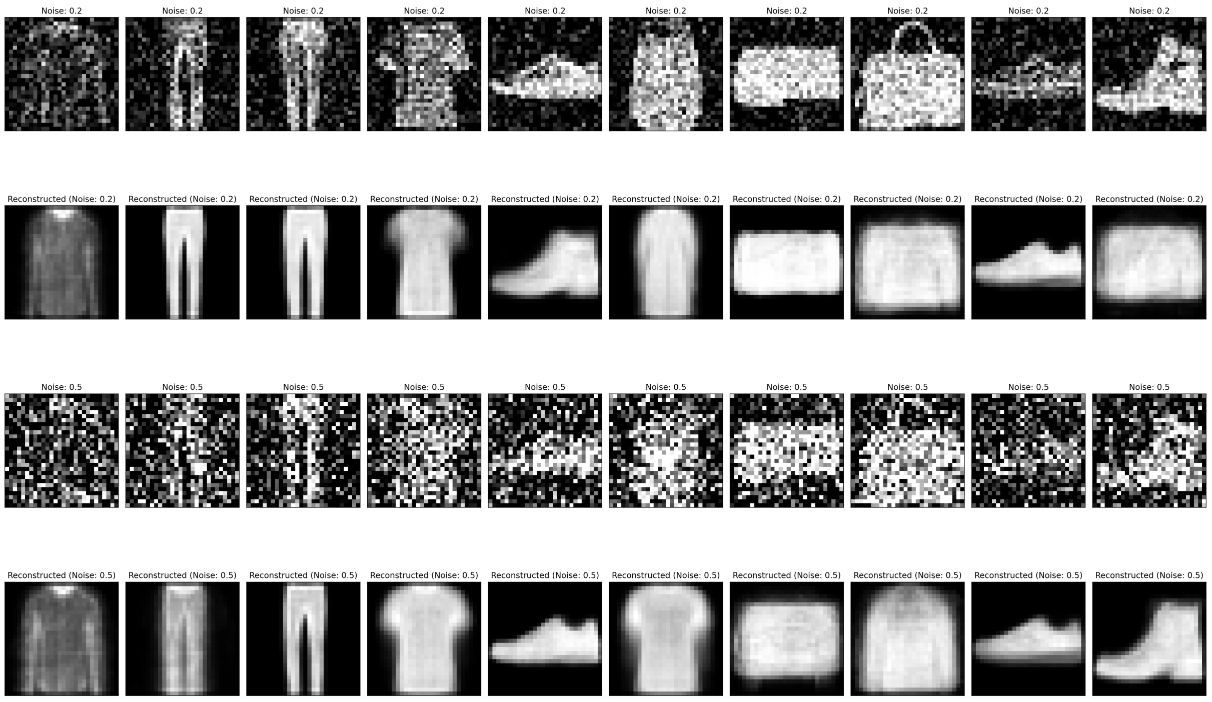
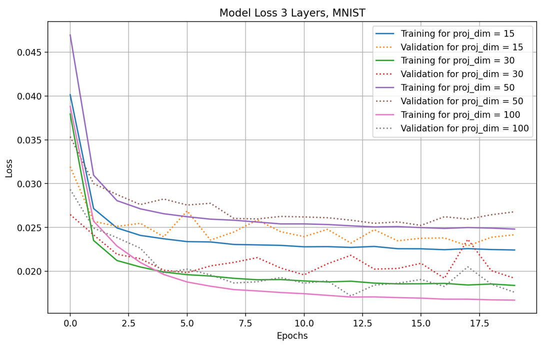
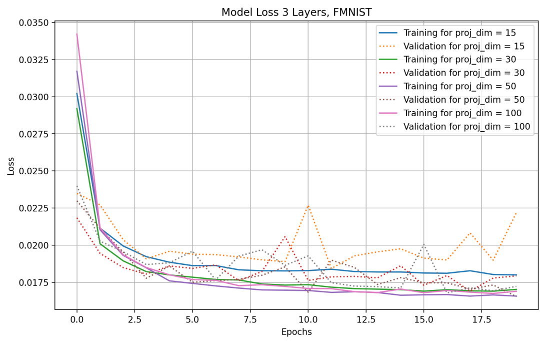


Figure 2.2: FMNIST denoising results after training the best model. Showing the noisy images first, then the reconstructed images after. First two rows are for noise of 0.2 and last two rows are for noise of 0.5.

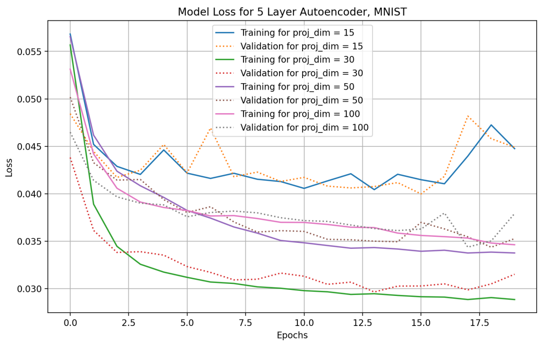
Appendix



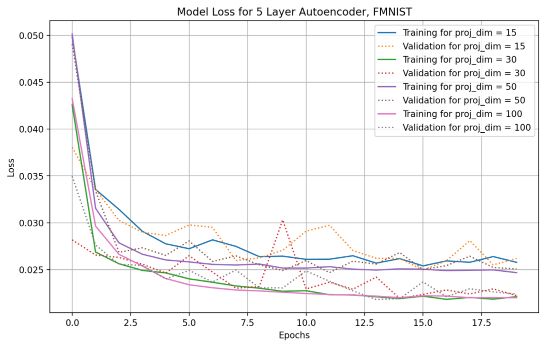
(a) Naive 3 layer MNIST results.



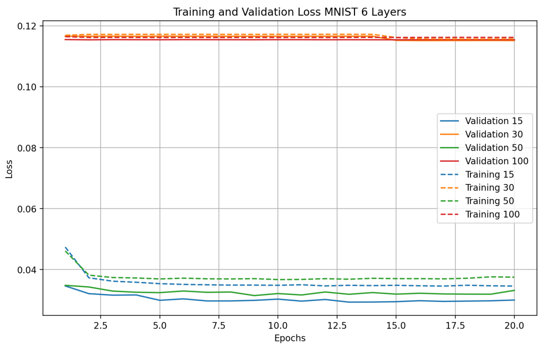
(b) Naive 3 layer FMNIST results.



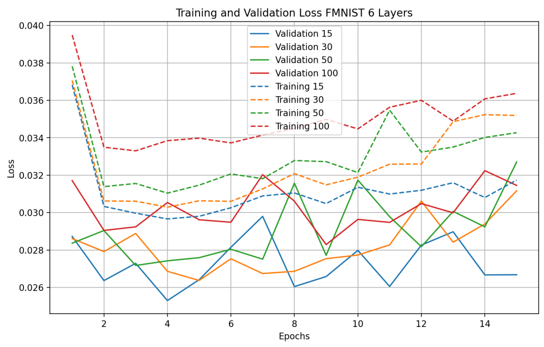
(c) Naive 5 layer MNIST results.



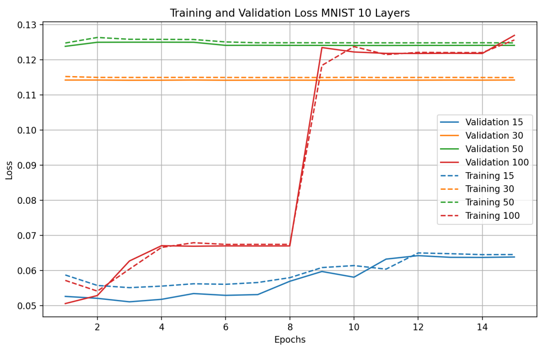
(d) Naive 5 layer fMNIST results.



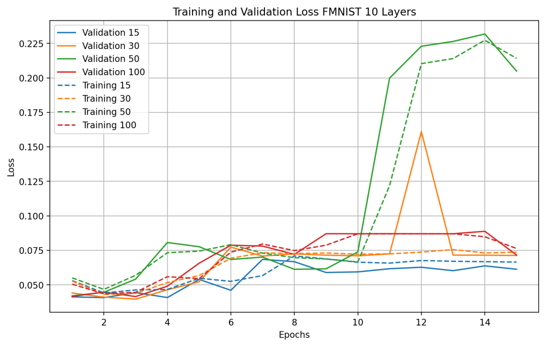
(e) Regularized 3 layer MNIST results.



(f) Regularized 3 layer FMNIST results.



(g) Regularized 5 layer MNIST results.



(h) Regularized 5 layer FMNIST results.

Figure 3.1: Training and Validation results per projected dimension of the Naive and Regularized models.