NEURAL NETWORKS: PROJECT 1
Authors:

Marina Gómez Rey (100472836)
Ángela Durán Pinto (100472766)
María Ángeles Magro Garrote (100472867)

Abstract
This code lab explores deep autoencoder implementations on MNIST and FMNIST databases. It compares 3-layer and 5-layer architectures, varying projected dimensions (15, 30, 50, 100), and explores different regularization techniques, including Lasso and Dropout regularization on the encoder's output. The best architecture is extended to denoising autoencoder using Gaussian noise with tunable variance. Performance is assessed using PSNR, and visual comparisons are made. The study aims to understand the impact of architecture, dimensionality reduction, and regularization on autoencoder performance in image tasks.

Introduction
Autoencoders, a type of neural network, are highly valued for their ability to efficiently learn representations of data. In this lab, our main goal is to implement deep autoencoders using the MNIST and FMNIST datasets, which contain handwritten digits and images of fashion products, respectively. We will explore how different architectural setups and regularization techniques affect the performance of autoencoders in tasks such as image reconstruction and denoising.

Autoencoders consist of two main components: an encoder network, responsible for compressing input data into a lower-dimensional representation, and a decoder network, tasked with reconstructing the original input from the compressed representation. By incorporating constraints during training, such as regularization, autoencoders can effectively learn meaningful representations and remove noise from corrupted data. In this project, we specifically compare two widely used regularization techniques:

- **Lasso regularization** penalizes the absolute value of the weights in the network, encouraging sparsity in the learned representations. This regularization term is integrated into the loss function. Throughout our experimentation, we tested several values of the regularization parameter lambda and saved the one that yielded the best results.

- **Dropout regularization** randomly deactivates a fraction of neurons during each training iteration, compelling the network to learn redundant representations and reducing the risk of overfitting.

Other regularization techniques were considered, such as **early stopping**, but due to the fact of lack of significant overfitting, it was not possible for implementation, as it would never stop the iterations.

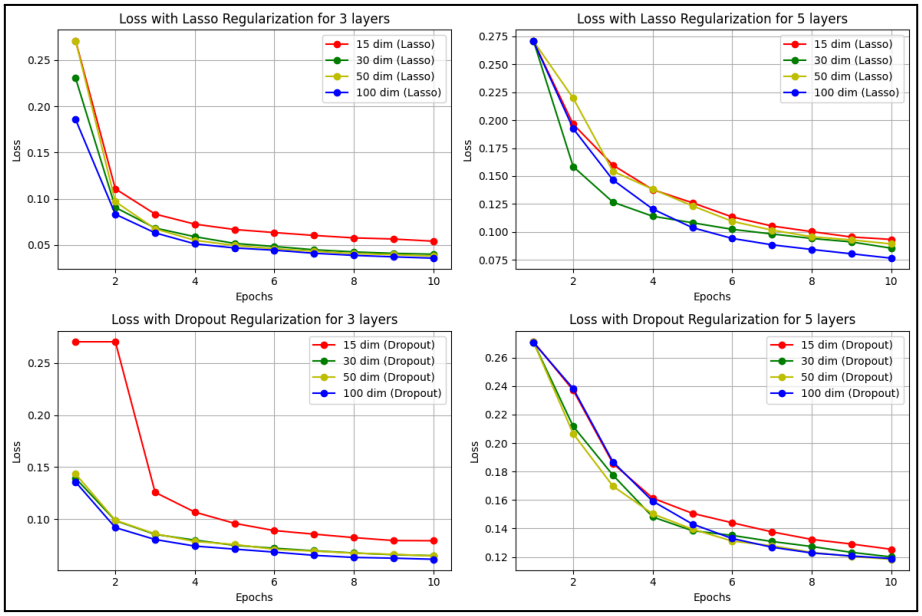
Moreover, we investigated denoising autoencoders, which are trained to reconstruct clean data from noisy inputs. By introducing Gaussian noise with adjustable variance, we simulate noisy inputs and evaluate the denoising performance of the autoencoders. We assess the quality of the denoised images using the Peak Signal-to-Noise Ratio (PSNR), which measures the ratio of the maximum possible power of a signal to the power of the noise corrupting the signal. Higher PSNR values indicate superior denoising performance, demonstrating the autoencoder's ability to effectively eliminate noise while preserving critical features of the input images.

Code & Results
The initial setup comprises:

- **Input size**: 784 pixels (28 pixels height × 28 pixels width), representing the dimensions of the input images.
- **Number of epochs**: 10, specifying the number of iterations over the entire dataset during training.
- **Batch size**: 128, indicating the number of samples from the dataset processed in each iteration before updating the neural network weights.
- **Learning rate**: 0.001, determining the step size for the gradient descent optimization algorithm.
- **Lasso regularization**: 0.00001, which penalizes large weight values, promoting sparsity in model parameters. A smaller lasso_lambda allows more flexibility in weights, while a larger value increases regularization strength, favoring simpler models. Thus, lasso_lambda regulates the trade-off between model complexity and sparsity. Several values for this parameter were tried but finally, the best results were obtained with this one.
- **Configurations**: A list of tuples representing different configurations for the neural network architecture, with each tuple specifying the number of layers and units in each layer.
- **Noise variances list for testing**: [0.1, 0.25, 0.5, 0.75, 1], used to test the denoising autoencoder model for various levels of noise variance.

After this, the first step was to determine which type of regularization technique to utilize: dropout regularization or Lasso regularization. Two autoencoder classes were created, each tailored to its respective regularization technique. The key distinction between them lies in how they handle regularization during training.

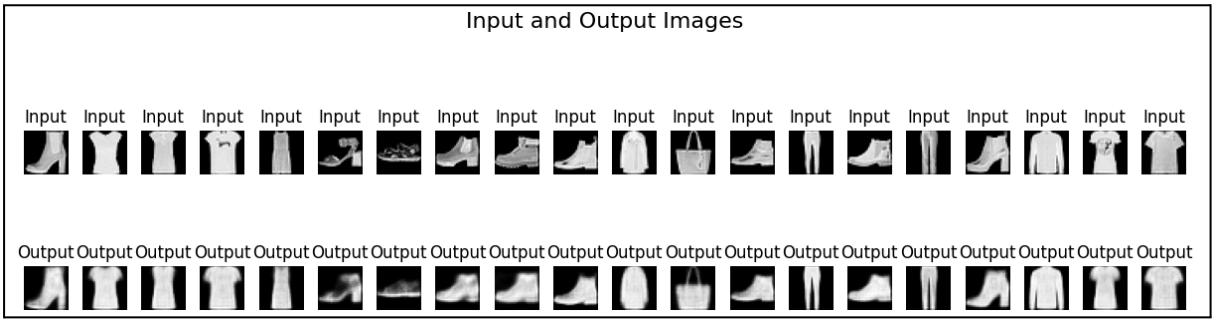
- For the dropout autoencoder, dropout layers were strategically inserted into both the encoder and decoder components, allowing for random deactivation of neurons to mitigate overfitting.
- The Lasso autoencoder directly incorporates L1 regularization into the loss function, penalizing large weights in the encoder's linear transformations. Both autoencoders underwent training using the provided training data loader, with the optimization process orchestrated by the training functions training_dropout and training_lasso.



Example of the loss evolution with the MNIST dataset and different regularization techniques (similar results were obtained with FMNIST)

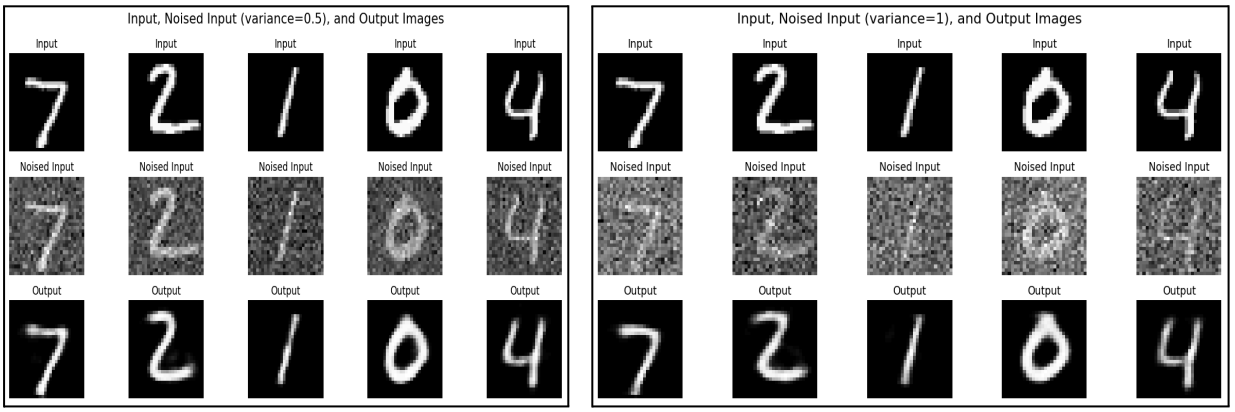
As observed, Lasso regularization works better for our model. This is also proved by comparing both losses of the best models obtained, which happened to be 3 layers and 100 projected dimensions in all cases.

	Lasso best model loss	Dropout best model loss
MNIST database	0.036	0.061
FMNIST database	0.040	0.062

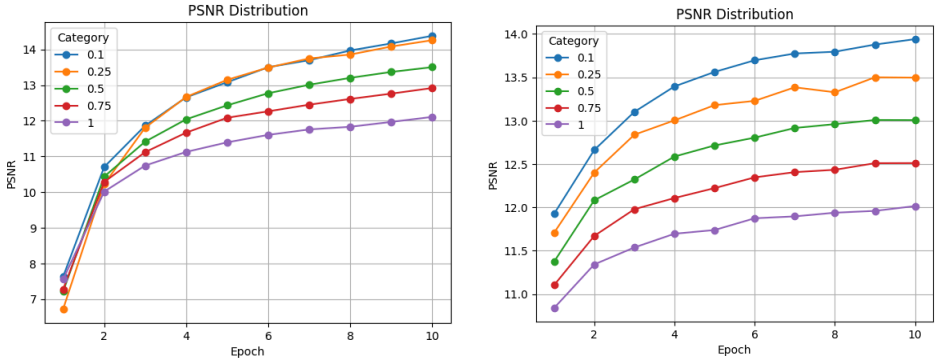


Output of random images from FMINST for the best lasso model (3 layers , 100 projected dimensions)

Once having chosen the best architecture trained with the best regularization technique, the denoising autoencoder was created to reconstruct clean images from noisy inputs. This process involved the implementation of several functions tailored to denoising tasks such as a "add_noise" to add Gaussian noise to input images, with the variance of the noise as a configurable parameter or "psnr" to calculate the Peak Signal-to-Noise Ratio (PSNR) between original and denoised images, serving as a quantitative metric to evaluate denoising quality. During training, PSNR values were computed to assess denoising performance and, evidently, better PSNR was obtained with images with less noise.



Output of MNIST random images with the denoising autoencoder at different variances



Evolution of PSNR among different variances (more variance, less PSNR)