DL lab 7 -Autoencoders

1. Upload the Autoencoder (AE) jupyter notebook file (i.e., lab\_7\_AE\_FFNN.ipynb) to google colab root directory.
   * A white background with black text

     Description automatically generatedTrain the model with 30 epochs.
   * A screenshot of a computer code

     Description automatically generatedWrite the code implementation to calculate the loss (Mean Squared Error) for the test dataset.
   * Write the code implementation to plot the train and validation loss against number of epochs.

A computer code with many colored text

Description automatically generated with medium confidence

A graph of a loss

Description automatically generated with medium confidence

1. When above AE is used without activation functions, it is called a linear AE. Explain the relationship between linear AE and principal component analysis (PCA). Write the answer in a word file.

A linear autoencoder (AE) without activation functions is closely related to Principal Component Analysis (PCA) because both techniques aim to reduce the dimensionality of data while preserving as much information as possible.

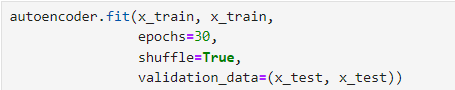
PCA is a statistical method that transforms data into a set of uncorrelated variables called principal components. These components capture the directions of maximum variance in the data, and PCA projects the data onto a lower-dimensional space, retaining the most important features.

A linear autoencoder, when used without activation functions, essentially learns to do the same thing. The encoder part of the AE tries to compress the input into a lower-dimensional latent space, and the decoder tries to reconstruct the original input from this compressed version.

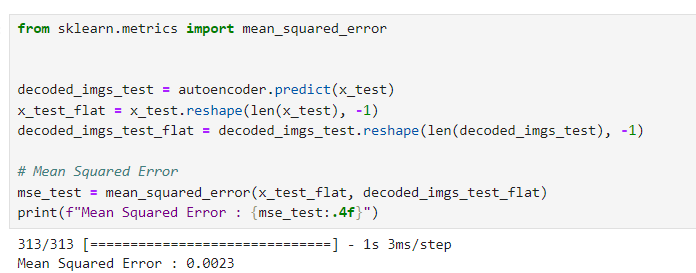
The key connection is that both linear AE and PCA are trying to find a lower-dimensional representation of the data. In fact, when trained with a linear activation function and squared error loss, the linear AE will learn a transformation similar to what PCA does: projecting the data onto its principal components.

However, while PCA computes this transformation analytically (using linear algebra), a linear autoencoder learns it through training. Despite this difference in approach, the outcome of both methods is quite similar in terms of the reduced representation of the data.

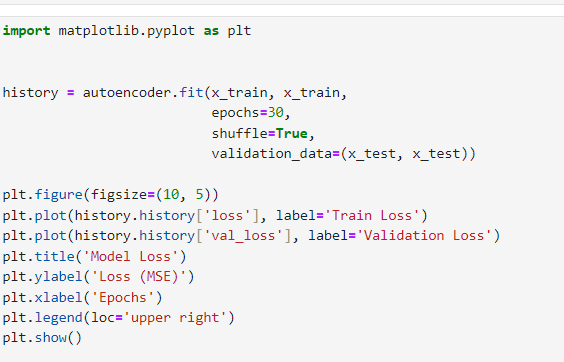
1. Upload the Vanilla CNN AE jupyter notebook file (i.e., lab\_7\_AE\_Vanilla\_CNN.ipynb) to google colab root directory.
   * Train the model with 30 epochs.

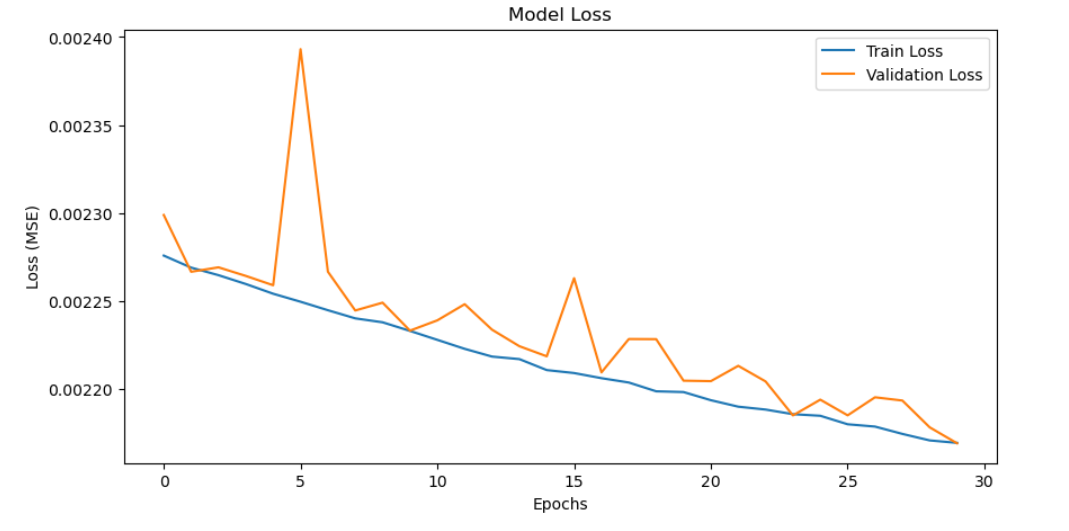


* + Write the code implementation to calculate the loss (Mean Squared Error) for the test dataset.



* + Write the code implementation to plot the train and validation loss against number of epochs.



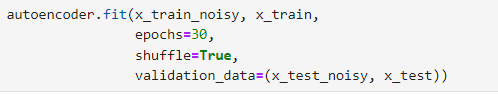


1. Observe the model performance improvements between the above two models and give reasons for the observed improvements.

The second model exhibits a much lower Mean Squared Error (MSE) of 0.0023, indicating that it reconstructs input data with greater accuracy compared to the first model, which has a higher MSE of 0.0086. Additionally, the second model shows a steeper decrease in training loss across epochs, suggesting faster convergence to an effective solution. The lower validation loss further highlights its ability to generalize better to unseen data, as it captures the essential features of the dataset while minimizing the risk of overfitting.

**Possible Reasons for Improvement**

* **Model Architecture**  
  The second model, a Vanilla CNN, incorporates convolutional layers, which are particularly well-suited for image data. These layers are efficient in capturing spatial hierarchies and local patterns, leading to better feature extraction compared to simpler models that rely on fully connected layers.
* **Network Depth**  
  With a deeper architecture consisting of multiple convolutional layers, the second model is able to learn more complex data representations. This additional depth enhances the model’s ability to reconstruct the input data more accurately.
* **Strides and Pooling**  
  The use of strided convolutions in the second model allows for downsampling of the input images in the encoder. This technique helps the model focus on the most important features by reducing noise and irrelevant details, ultimately improving reconstruction quality.
  + Train the model with 30 epochs.



* + Write the code implementation to calculate the loss (Mean Squared Error) for the test dataset.

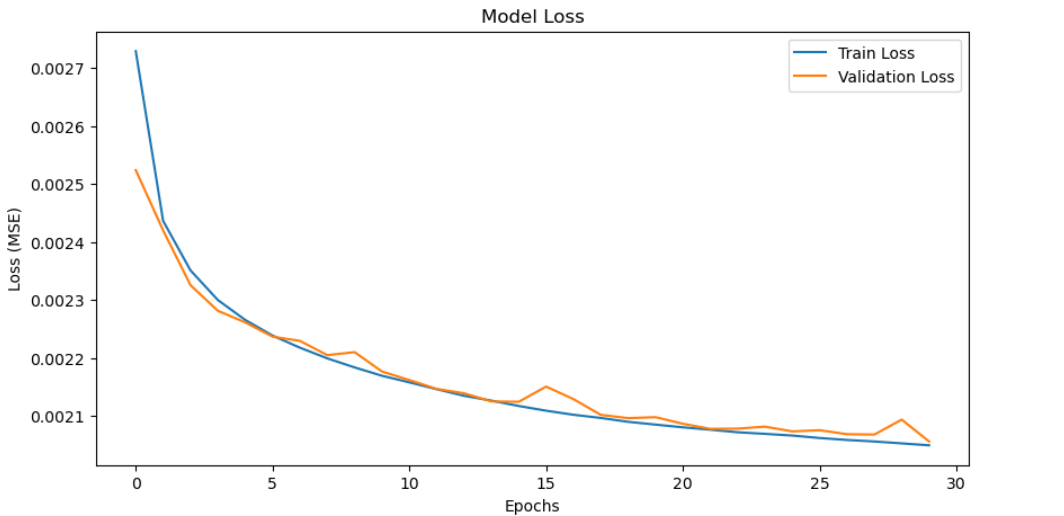
A screenshot of a computer program

Description automatically generated

* + Write the code implementation to plot the train and validation loss against number of epochs.

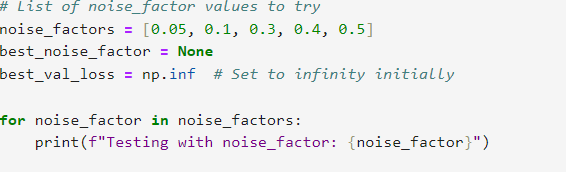
A screenshot of a computer code

Description automatically generated



* + Experiment with “noise\_factor” value and use the best value you find in the final implementation. (Pay attention to how this value affect the images by observing the noise added images in the code.)

Lower values introduce less noise, while higher values introduce more noise, which make reconstruction harder.



The training results show how different noise factors affect the performance of the model. Here’s a summary of the validation losses for each noise factor:

* **Noise factor 0.05:** Validation Loss: 0.00237
* **Noise factor 0.1:** Validation Loss: 0.00351
* **Noise factor 0.3:** Validation Loss: 0.00980
* **Noise factor 0.4:** Validation Loss: 0.01414 (at epoch 13, then cut off)

The best performance, based on the lowest validation loss, is with the **noise factor of 0.05**. The model trained with this noise factor has a validation loss of **0.00237**, making it the most effective configuration tested.

1. Observe the model performance improvements between the Image De-noising AE and the Vanilla CNN AE.
   * Explain the reasons for the observed improvements.

The Image Denoising Autoencoder (Denoising AE) generally performs better than the Vanilla CNN Autoencoder when it comes to reconstructing clean images from noisy inputs. This is due to its specialized training on noisy data, which enables the model to develop strong feature extraction and noise suppression capabilities. By learning to model the distribution of clean images, the Denoising AE is able to filter out noise effectively during the reconstruction process, resulting in lower Mean Squared Error (MSE) values. In contrast, the Vanilla CNN, which is trained only on clean images, struggles to handle noise, leading to higher reconstruction errors and reduced ability to generalize to real-world noisy scenarios.

1. Explain the differences between AE and Variational AE (VAE).

Autoencoders (AE) and Variational Autoencoders (VAE) both aim to encode data into a lower-dimensional latent space and reconstruct it, but they differ fundamentally in approach. AEs are deterministic models that encode the input into fixed latent variables and then decode it back to the original form. The goal of an AE is to minimize reconstruction error, but its latent space is unstructured, which can make interpolation and generation of new data challenging. On the other hand, VAEs are probabilistic models that encode the input into a distribution (typically Gaussian) rather than fixed values, sampling from this distribution to reconstruct the data. This makes VAEs more flexible for generating new data, as they can sample from the latent space to create novel inputs similar to the training data.

Additionally, VAEs introduce latent space regularization using a KL divergence loss, enforcing a structure on the latent space and encouraging it to follow a standard normal distribution. This structured latent space allows for better generative capabilities, making VAEs ideal for tasks that require creating new data, such as image or text generation. In contrast, AEs are typically better suited for tasks like dimensionality reduction and denoising, where the focus is on accurately reconstructing the original input rather than generating new examples.