Autoencoders with Multi-Layer Perceptron: Feature Extraction and Classification

GitHub Repository: https://github.com/itzTarun219/AE_MLP

Table of Contents:

1. Introduction

Deep learning and machine learning have revolutionized several domains, more so image classification. Autoencoder-based Feature Extraction is one of the successful methods wherein we utilize an unsupervised neural network to discover a compressed representation of the input. Here in this tutorial, we describe how Autoencoders can be utilized to enhance the accuracy of a Multilayer Perceptron (MLP) classifier by dimensionality reduction and helpful feature extraction. We apply this process with the MNIST handwritten digits dataset.

2. Understanding Autoencoders

2.1 Structure of Autoencoders

Autoencoders have two main components: the encoder and the decoder. The encoder transforms the input data into a lower dimension, often referred to as the "latent space" or "bottleneck." The representation captures the salient features of the input and discards the redundant information. The decoder transforms this compressed representation back to the input data in an attempt to minimize the input-output difference. The training process is achieved by reducing the reconstruction error, usually through a loss function such as Mean Squared Error (MSE) or Binary Cross-Entropy, depending on the nature of the input data (O'Reilly, 2019).

2.2 Types of Autoencoders

There are several types of Autoencoders, each designed for specific tasks:

- **Vanilla Autoencoders:** It is a basic structure with only one hidden layer. It learns to encode the input data given into a lower dimensional space and then it decodes it back to the original space.
- **Denoising Autoencoders:** It is trained to reconstruct the original inputs from the corrupted version. By intentionally adding noise to the input data while training it, denoising autoencoders learn to extract the various features that are much less sensitive to noise.
- **Variational Autoencoders (VAE):** Introduce a probabilistic approach to encoding. This allows for generating new data points by sampling from the distribution learnt, making VAEs more particularly useful in generative tasks.
- **Space Autoencoders:** The space autoencoders are sparsity constraint on the hidden layer, encouraging the model to learn the presentation where a small number of neurons are active at any given time which can lead to more interpretable features.

3. Multi-Layer Perceptron’s (MLPs)

3.1 Structure of MLPs

MLPs consist of an input layer, one or more hidden layers, and an output layer. And each layer is also fully connected to the next, allowing for the complex mappings from inputs to the outputs (O'Reilly, 2019).

- **Input Layer:** The input layer receives the input features and for each neuron in this layer it corresponds to a feature in the input data.
- **Hidden Layers:** One or more layers wherein the actual processing occurs. Each neuron in a hidden layer performs a weighted sum of its inputs and an activation function. The number of hidden layers and how many neurons each of them has can have a strong impact on the model's performance.
- **Output Layer:** The output layer produces the final output of the network. For classification tasks, the output layer typically uses a SoftMax activation function to produce probabilities for each class.

3.2 Activation Functions

The Common Activation functions include:

- **ReLU(Rectified Linear Unit):** $(f(x) \ = \backslash max(0, x))$, it is widely used for its simplicity and effectiveness. Also, it helps mitigate the vanishing gradient problem which allows for faster convergence during training (François Chollet, 2021).
- **Sigmoid:** $(f(x) \ = \backslash frac\{1\}\{1 + e^{\{-x\}}\})$, the sigmoid function takes the input to a range between 0 to 1. It is often used in binary classification tasks (François Chollet, 2021).
- **SoftMax:** It is used in the output layer for the multi-class classification, which converts the raw output scores into probabilities that sum to 1, making it easier to interpret the model’s predictions.

4. Combining Autoencoders and MLPs

4.1 Feature Extraction with Autoencoders

Autoencoders can effectively lower the input data dimensionality while preserving the most important features. By training an Autoencoder on the MNIST dataset, we are able to learn a compressed form of the images. This compressed form preserves the important features of the digits, such as shapes and strokes, and removes noise and unnecessary information.
Once the Autoencoder is trained, we can use the encoder part to transform the original images into their encoded representations. The encoded features can then be fed into an MLP for classification.

4.2 Classification with MLPs
After features have been extracted, an MLP can be trained over the extracted features to achieve classification tasks. This two-step process is likely to yield better results than employing raw input data.

The Advantages of this approach will include:

- **Dimensionality Reduction:** It reduces the input size and can lead to faster training times and lower computational expenses.
- **Improved Generalization:** It focuses on the most informative features; the model will not be overfit as much noise in the data.
- **Enhanced Performance:** It combines the strengths of Autoencoders and MLPs can lead to enhanced classification performance.

5. Experimental Setup

5.1 Dataset Selection
For this tutorial, we are using the MNIST dataset, which consists of 70,000 images of handwritten digits (0-9). The dataset is divided into 60,000 images and 10,000 test images each image resulting 28*28 pixels, which is in total of 784 features when flattened. And this dataset is widely used for the benchmarking and classification algorithms (TensorFlow, 2025).

5.2 Implementation
We will implement with the following steps to be followed:

- We Load and preprocess the MNIST dataset.
- Building an Autoencoder to extract features.
- Training an MLP on the extracted features from the autoencoders.
- Evaluating the performance of the MLP.

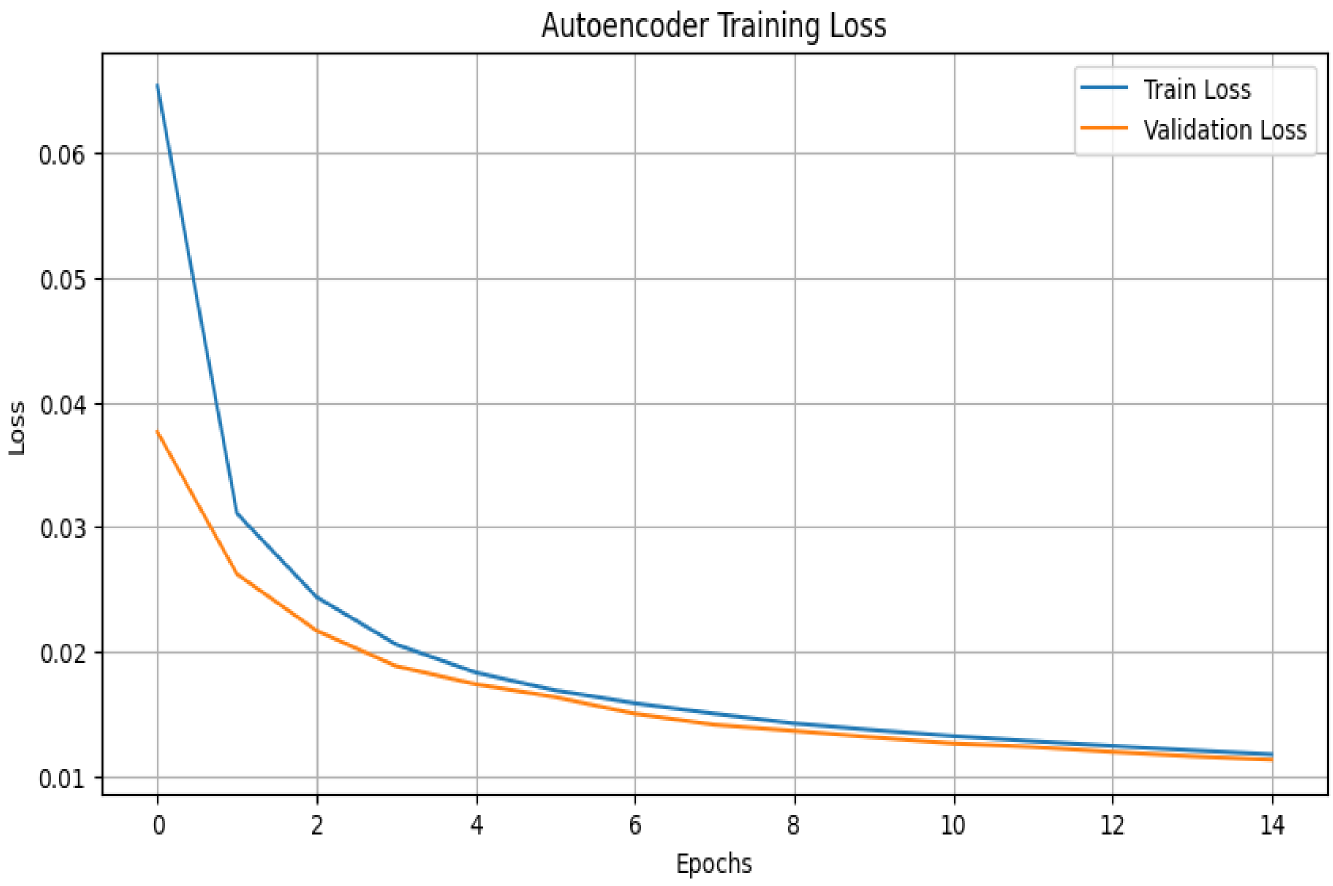
6. Results and Discussion

6.1 Performance Metrics
To evaluate the performance of our MLP, we will use the following metrics:

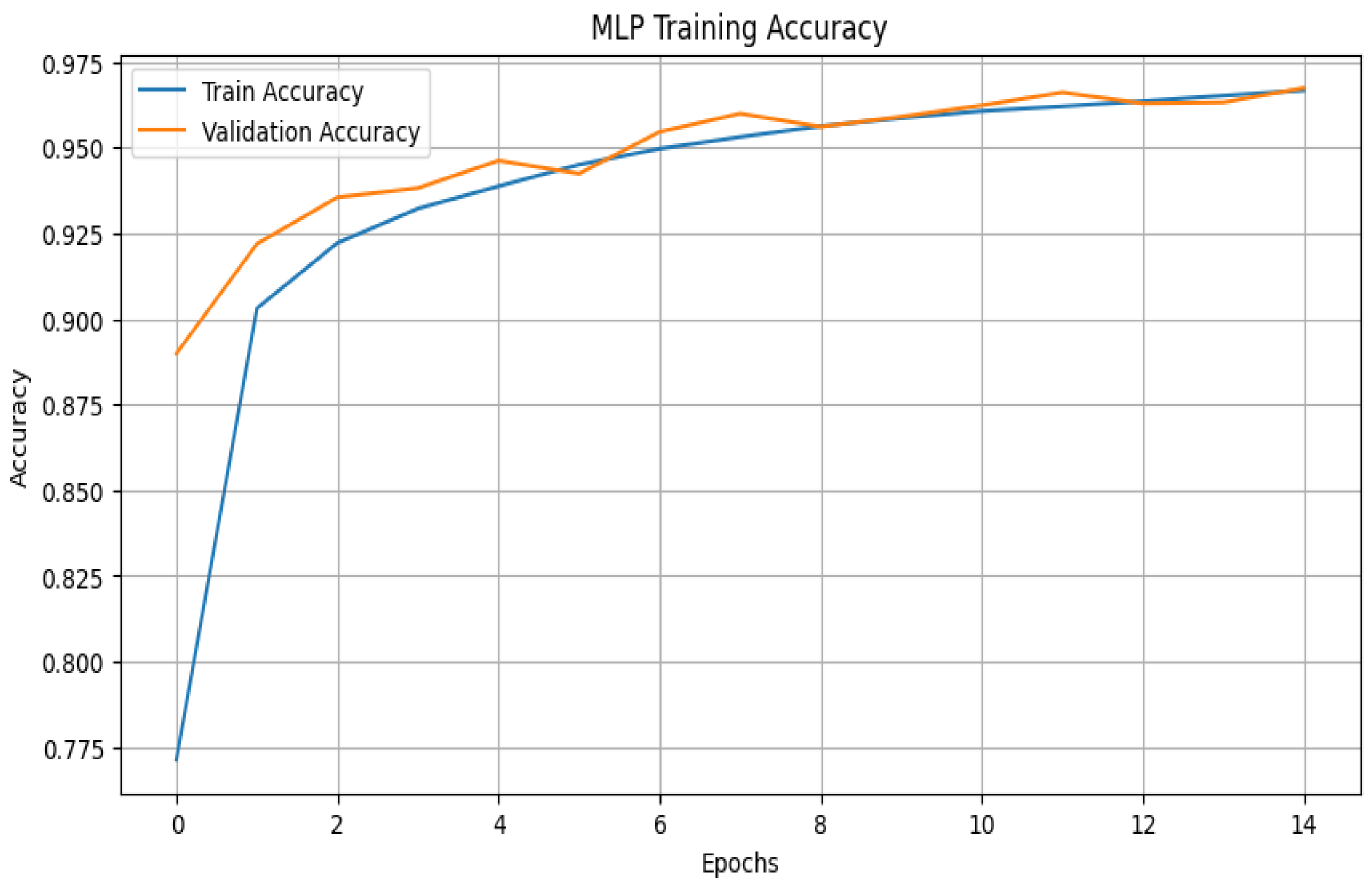
- **Accuracy:** It is the percentage of correctly classified instances. And it is calculated as the number of correct predictions divided by the total number of predictions.
- **Loss:** It is the difference between predicted values and the actual values. A Lower loss indicates better model performance.

6.2 Visualizations
The Visualizations play a crucial role in understanding model performance (The Matplotlib, 2025). We will include the following plots:

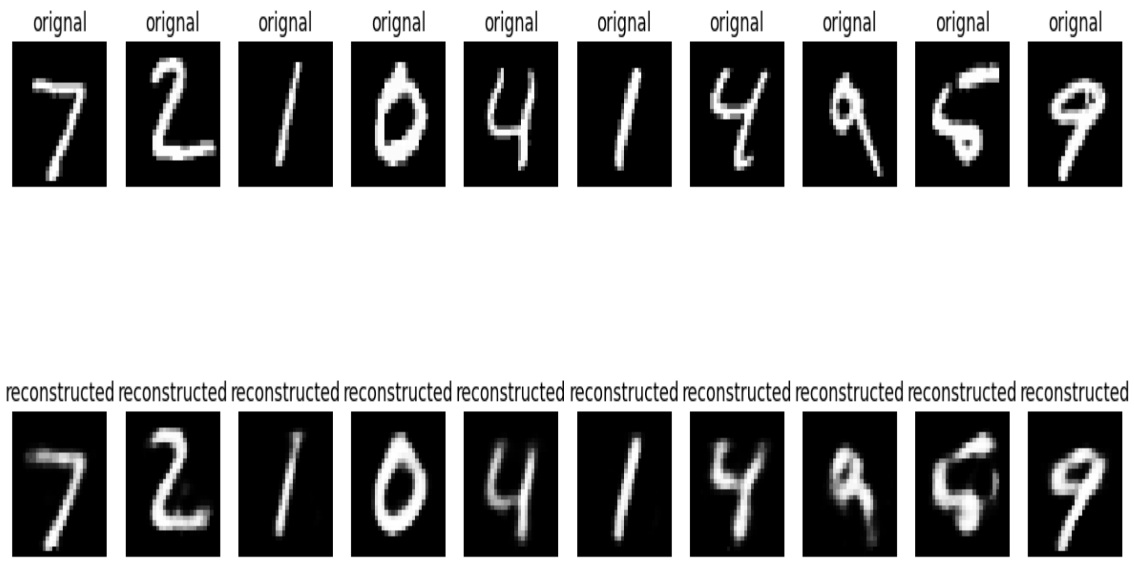
- **Autoencoder training and validation loss over epochs:** In the Line graph it shows how the loss decreases over epochs, which indicates the autoencoders adaptability and learning process.



- **MLP Training Accuracy vs Epochs:** This Line graph is showing the increase in accuracy during training, and showing how well the model learns the extracted features.



- **Displaying Original and Reconstructed images:** A side-by-side comparisons of the input images and their reconstructions helps us access how well the autoencoder captures the essential details.



7. Conclusion:

In this tutorial, we showed how to use Autoencoders for feature extraction and MLPs for classification. This approach can greatly improve classification performance by taking advantage of the strengths of both models. By leveraging learned compressed representations we have achieved the improved performance while reducing the computational complexity. And this model is widely applicable beyond MNIST as well which includes medical imaging, anomaly detection, and natural language processing.

References

François Chollet, 2021. *Deep Learning with Python, Second Edition.* [Online]
Available at: https://www.manning.com/books/deep-learning-with-python-second-edition
O'Reilly, 2019. *Deep Learning with TensorFlow 2 and Keras - Second Edition.* [Online]
Available at: https://learning.oreilly.com/library/view/deep-learning-with/9781838823412/Text/Preface.xhtml#_idParaDest-6
TensorFlow, 2025. *Introduction to TensorFlow.* [Online]
Available at: https://www.tensorflow.org/learn
The Matplotlib, 2025. *Plot types.* [Online]
Available at: https://matplotlib.org/stable/plot_types/index.html