**Applied Machine Learning**

**Lab Report 7**

**Hafiz Ahmad**

**19l-1316**

**Section-8A**

**INTRODUCTION:**

The aim of this experiment is to investigate neural networks, with a particular emphasis on the Perceptron Learning algorithm and Multi-Layer Perceptron Learning. Neural networks are computational models that draw inspiration from the human brain and possess the ability to learn and make predictions by identifying intricate patterns within data. To accomplish this, we will employ the TensorFlow library and develop a neural network model using the Sequential API. The dataset employed in this experiment is the MNIST dataset, which consists of handwritten digits.

**OBJECTIVES:**

The objective of this experiment is to gain a comprehensive understanding of Perceptron Learning and Multi-Layer Perceptron Learning algorithms within neural networks. We aim to train a neural network model on the MNIST dataset and evaluate its accuracy in recognizing handwritten digits.

**Procedure:**

To direct this analysis, we start by stacking the important bundles, including pandas, numpy, and TensorFlow, for information control and brain network demonstrating. Additionally, for data visualization, we import matplotlib. The training dataset, "mnist\_train.csv," is then loaded into a "train" pandas DataFrame. This dataset comprises of transcribed digit pictures alongside their relating marks. Then, we set up the preparation information by removing the pixel values from the "train" DataFrame and normalizing them by separating by 255. The standardized pixel values are put away in the "train\_images" DataFrame, while the marks are put away in the "train\_labels" Series. Using the reshape() function, we reshape the "train\_images" DataFrame into a 3D tensor of shape representing the number of images (28, 28). Using tf.convert\_to\_tensor(), this tensor is then transformed into a TensorFlow tensor. Moving on to the test dataset, the "mnist\_test.csv" file is loaded into a "test" pandas DataFrame. The trained model is tested using images of handwritten digits in this dataset. The "label" column is removed from the "test" DataFrame in order to prepare the test data, and the remaining pixel values are stored in the "test\_images" DataFrame. The relating marks are extricated and put away in the "test\_labels" Series. In a similar manner, we transform the "test\_images" DataFrame into a TensorFlow tensor by reshaping it into a 3D tensor. With the information arranged, we continue to assemble the brain network model utilizing the Consecutive() constructor from the tensorflow.keras.models module. The model comprises of various Thick layers, each characterized by the quantity of units and initiation capability. Using the compile() function, we specify the Adam optimizer, the SparseCategoricalCrossentropy loss function, and accuracy as the evaluation metric after the model has been constructed. The fit() function is then used to train the model on the training images and labels. For the purpose of iterating over the training data, the number of epochs is set to 10. Subsequent to preparing, the model's presentation is assessed on the test pictures and names utilizing the assess() capability. Variables are used to store test loss and accuracy. For improved readability, the model's accuracy is printed, rounded to two decimal places, and multiplied by 100.

We begin by adding the libraries required to complete our experiment in the first step.

Then, we load the CSV document and preprocess the information by normalizing every pixel esteem somewhere in the range of 0 and 1 by partitioning them by 255. The images are then reshaped and transformed into TensorFlow tensors.

The preprocessed training images are now represented by the "train\_images" variable. The test data undergo the same preprocessing procedures.

The Multi-Layer Perceptron model is our next option. We flatten the data, create an output layer with 10 units representing output probabilities and a hidden layer with 200 units and the ReLU activation function.

The accuracy on the test data was 97.95%, while the accuracy on the training data was 99.6 percent.

In outline, in light of the given exactness esteems, the model seems to have performed well on both the preparation and test information, demonstrating great speculation. This suggests that the model can accurately predict new, unobserved examples and has successfully learned the data's underlying patterns.

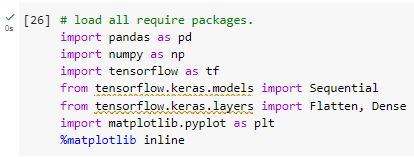
By presenting the sigmoid initiation capability and 200 units, the precision of the model somewhat lessens yet performs well.

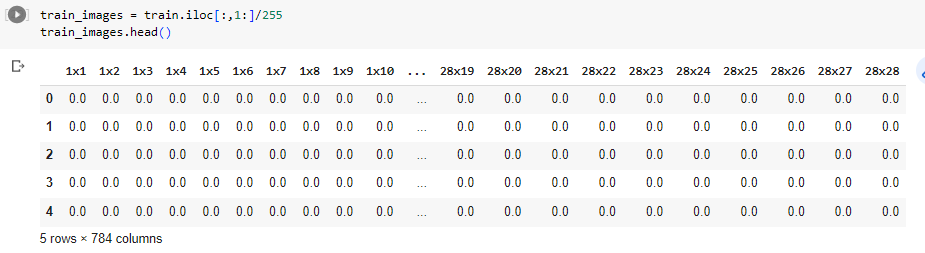
The model's performance is further enhanced as additional activation functions are added following.

In the subsequent step, we observe the effect on accuracy by increasing the number of iterations and adjusting the unit values.

After 27 training epochs, the model had a training accuracy of 99.86%. The test precision is 98.09%. The model's ability to recognize handwritten digits with high accuracy is demonstrated by these accuracy metrics.

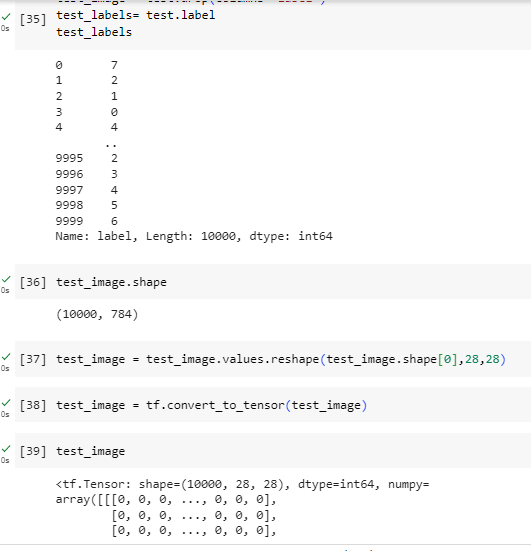
In the first step we will add the necessary libraries for the completion of our experiment.

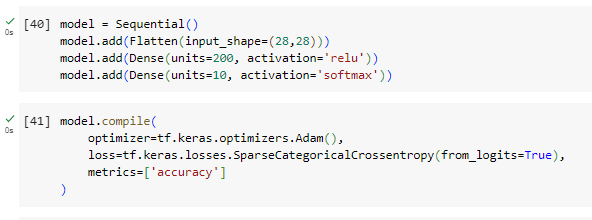




A screenshot of a computer program

Description automatically generated with medium confidence





A screenshot of a computer

Description automatically generated with medium confidence

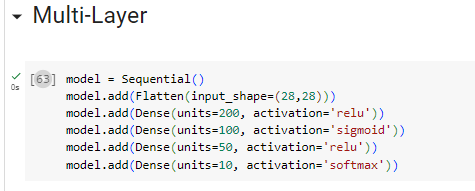
A picture containing text, font, line, screenshot

Description automatically generated

A screenshot of a computer program

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with medium confidence



A screenshot of a computer

Description automatically generated with medium confidence



A screenshot of a computer program

Description automatically generated with medium confidence

**Application:**

NN, including the Perceptron Learning and Multi-facet Perceptron models investigated in this examination, have many applications in different fields. The following are a few examples of how this experiment can be used:

Recognition of Handwritten Digits: The prepared brain network model can be applied to undertakings including manually written digit acknowledgment, like optical person acknowledgment (OCR), robotized structure handling, and digit-based verification frameworks.

Classification of Images: For image classification tasks, neural networks, including Multi-Layer Perceptron models, are frequently utilized. This experiment's trained model can be used for more than just handwritten digits to classify images, such as for object recognition in computer vision applications.

**Issues:**

No issue was found while performing in the lab.

**Conclusion:**

In this experiment, we have effectively implemented the Perceptron Learning and Multi-Layer Perceptron Learning algorithms using neural networks. By training a neural network model on the MNIST dataset, we were able to assess its accuracy in accurately recognizing handwritten digits. The model achieved a high level of accuracy, demonstrating its efficacy in performing digit recognition tasks.