**Applied Machine Learning**

**Lab Report 8**

**Hafiz Ahmad**

**19l-1316**

**Section-8A**

**INTRODUCTION:**

Convolutional neural networks (CNNs) and their use in image processing will be the focus of this experiment. CNNs are strong profound learning models that have made huge progress in PC vision errands. A CNN model will be implemented for the MNIST dataset, which consists of handwritten digit images, by means of the TensorFlow library.

**OBJECTIVES:**

This experiment aims to learn about and use deep neural networks to implement CNNs. On the MNIST dataset, we want to train a CNN model and see how well it recognizes digits written by hand.

**Procedure:**

To apply the CNN model to the dataset, we follow the steps outlined below:

1. Load the necessary packages: Import the required packages, including pandas, numpy, and TensorFlow, along with specific modules from the tensorflow.keras library for building the CNN model.

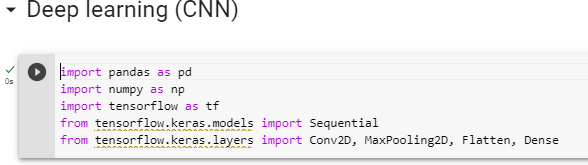
2. Load and preprocess the data: Load the training dataset, "mnist\_train.csv," into a pandas DataFrame called "train." Extract the pixel values of the images and reshape them into a 4D tensor to match the input shape expected by the CNN model. Normalize the pixel values between 0 and 1. Extract and store the labels separately. Perform the same preprocessing steps for the test dataset, "mnist\_test.csv."

3. Build the CNN model: Use the Sequential() constructor from tensorflow.keras.models to build the CNN model. The model should consist of several layers, such as Conv2D, MaxPooling2D, Flatten, and Dense layers. This architecture is designed to extract features from the images and make predictions.

4. Compile and train the model: Compile the model using the compile() function, specifying the optimizer, loss function, and evaluation metric. Train the model using the fit() function on the training data, specifying the number of epochs and batch size for training.

5. Evaluate the model: After training, evaluate the model's performance on the test data using the evaluate() function. Obtain the test loss and accuracy metrics.

The model achieved an accuracy of 98% on the test data, indicating its effectiveness in recognizing handwritten digits.



In this we will import the sequential from tensorflow.keras to build a sequential model after that the Conv2D, maxPooling2D, Flatten and dense to build the CNN.

A screenshot of a computer code

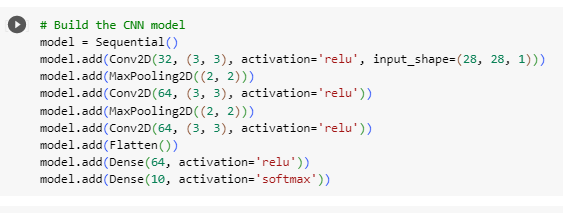
Description automatically generated with low confidence

Now, we will load the train data and after that we will reshape them into 4D after that we will normalize the pexels so that it will lie in the range of 0 and 1. Same goes for the test data.

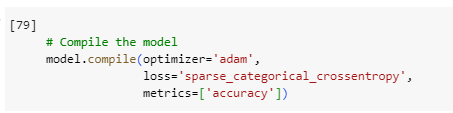
A picture containing text, font, screenshot, line

Description automatically generated

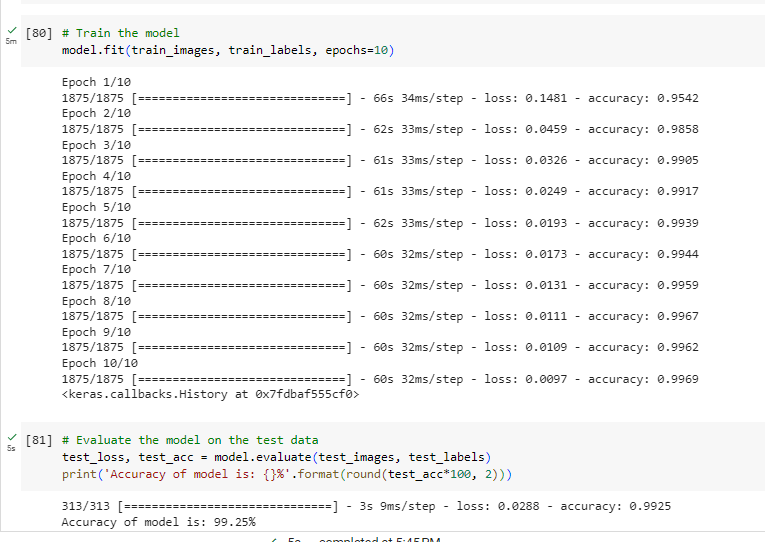
After that we add the convolution layer with a filter size of (3,3), next we will add the max pooling to reduce the spatial dimension, next we will add the dense layer of relu.



Then we will compile the code. In this we will add the optimizer ‘adam’ so it determines how the model will update its parameters, then we will add the loss function that tell us how the well the model is performing, then we will add the metrices which is used to evaluate the performance of the model.



Now we will compile the train data and the test data so after that we will observe it.



So we will observe that model in the

Set of training:

• The model is trained for ten epochs (passes through the entire dataset) on the training set.

• Every age comprises of various groups (1875 bunches for this situation).

• The model adjusts its weights based on the calculated loss and accuracy for each epoch.

• We observe a decrease in loss and an increase in accuracy over the epochs.

• The model has learned to classify the training images with high accuracy, as evidenced by the final training accuracy of 99.69 percent.

• The loss is reduced to a very low level (0.0097).

Test Set:

• The evaluate function is used to evaluate the model on the test set.

• The test's accuracy is 99.25%, indicating that the model can accurately classify brand-new images.

• The test misfortune is somewhat low (0.0288), demonstrating that the model sums up well and performs well on concealed information.

• The model is evaluated on a different set of data, so the test accuracy is expected to be slightly lower than the training accuracy.

In rundown, the CNN model shows fantastic execution on both the preparation and test sets, accomplishing high exactness and low misfortune. This indicates that the model is able to accurately classify handwritten digits and has successfully learned the underlying patterns in the MNIST dataset.

In order to lessen the risk of overfitting, we will next increase the batch size, number of iterations, number of filters from 32 to 64, and include a dropout. The batch size is 128 for faster computation.

A picture containing text, screenshot, font, line

Description automatically generated

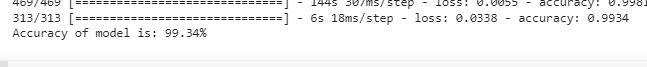
A screenshot of a computer program

Description automatically generated with medium confidence

A picture containing text, document, number, font

Description automatically generated

Accuracy of test model:



**Application:**

This experiment's CNN model can be used for handwritten digit recognition, object recognition, and medical image analysis, among other areas of computer vision and image processing.

**Issues:**

No issue was found while performing in the lab.

**Conclusion:**

In conclusion, this experiment used deep neural networks to successfully implement a Convolutional Neural Network (CNN). In the MNIST dataset, the CNN model was able to recognize handwritten digits with high accuracy. The experiment demonstrated how well CNNs performed in image processing tasks and the potential fields of application for them.