**PART I: Build, Train, and Test CNN on MNIST Dataset (60 Points) BY Yog Chaudhary**

**Project Report – CNN on MNIST with TensorFlow.**

The basic libraries were imported into Python. These include NumPy, pandas, matplotlib, and TensorFlow. The dataset has 55,000 values for our training dataset and 10,000 for our testing dataset. This allowed us to analyze the MNIST dataset correctly. It is worth noting that each image uses the “one-hot” format. The images are hand-written representations of the numbers 0- 9 and have been size normalized. This allows us to stay in a 1D array to analyze the dataset more easily. In part 1, we are required to construct a DL model using two convolutional layers (CONV1 and CONV2), each with corresponding ReLU and Max-Pooling layers, to process and rescale/reshape the images into flat arrays. The composition of each layer is described below. Below I have listed all the attributes of the design of the first model:

**Design of CNN model:**

**Convolution layer 1:**

Convolution 4D shape: [batch, Height, Width, depth] = [ 1, 28, 28, 32]

2D data size: 28 x28

Input shape: 28 x 28 x 1 (Depth = 1 in channel)

Output shape: 28 × 28 × 32 (Depth = 32 out channels)

Filter/Kernel/Window size = 5 x 5

Filter shape - weight shape: [5, 5, 1, 32]

Stride = 1

Stride shape: (1, 1, 1, 1]

Padding = SAME

Activation Function Layer: ReLU

**ReLu Layer 1:**

No Filter (only process data, not extract/learn features)

Input shape: 28 x 28 x 32

Output shape: 28 x 28 x 32

**Pooling Layer:**

Pooling Layer 1

Pooling method: Max pooling

Filter/Kernel/Window size - 2 x2

Filter/Kernel/Window shape: [1, 2, 2, 1]

Stride = 2

Stride shape: [1, 2. 2, 1]

Padding - SAME

Input channels: 32 inputs

Input shape: 28 x 28 x 32

Output channels: 32 outputs

Output shape: 14 x 14 x 32

**Convolution Layer 2**

Convolution shape: [batch, H, W, depth] = [ 1, 14, 14, 64 ]

2D data size: 14 x 14

Input shape: 14 x 14 x 32 (Depth = 1 in channel)

Output shape: 14 x 14 x 64 (Depth = 32 out-channels)

Filter/Kernel/Window size = 5 x 5

Filter shape = weight shape: [5 , 5, 32, 64]

Stride = 1

Stride shape: [1, 1, 1, 1]

Padding = SAME

Activation Function Layer: ReLU

**ReLu of the convolutional layer 2**

Input shape: 14x14x64

Output shape: 14x14x64

**Pooling Layer 2**

Pooling method: Max pooling

Filter/Kernel/Window size = 2 x2

Filter/Kernel/Window shape: [1, 2, 2, 1]

Stride = 2

Stride shape: [1, 2, 2, 1]

Padding - SAME

Depth input channels: 64 inputs

Input shape: 14 x 14 x 64

Depth output channels: 64 outputs

Output shape: 7 x 7 x 64

**Fully Connected layers:**

Fully Connected Layer 1 (FC 1)

FC\_1 shape: [ inputs (in \_channels), outputs (out\_channels)]

(7\*7 x64, 1024)

Fully Connected Dropout Layer (FC 2)

FC\_2 shape: [ inputs (in channels), outputs (out channels)]

= [1024. 1024]

Fully Connected Final Output Layer (FC 3)

FC 3 shape: [inputs (in channels), outputs (out channels) ]= [1024, 10]

**The difference between the 2 convolution layers is as follows.**

* All dimensions—height, breadth, and input—went from 28 to 14 respectively.
* There will be twice as many output channels as input channels to learn everything we want to.
* There are 32 input channels and 64 output channels in the filter variant, respectively.

**Screenshot Jupyter Notebook Below**

On steps 0-200

**A screenshot of a computer

Description automatically generated**

On Steps 4700-4900

**A screenshot of a computer

Description automatically generated**

**Report:**

The outcomes indicate a gradual enhancement in accuracy as we experimented with different CNN layers in the project. Furthermore, our findings demonstrate that modifying the filter dimensions at both the input and output layers can effectively enhance efficiency, as evidenced by the consistent increase in accuracy across the steps we implemented for the fully connected layers.

* Screenshot of results

A close up of a white background

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA white background with black text

Description automatically generated

**Result:**

With the design listed above, the test and trained the first model for steps from 1 to 5000 with 100 test iterations of a 50-element batch shows absolute accuracy of 99.04%. This also indicates the dropout layer to see if the predicted and actual values are equal. The testing step 0 model accuracy was 0.1968. However, the model progressed and improved in step 4900. The rate is 0.9904. This is a great result of the model that fits the problem and the associated dataset well.

**PART II: Redesign Convolutional Neural Network (30 Points)**

In the second model, we took the design from the original model and only made a few modifications. Here we are only using one convolutional layer with adjusted code, so for all layer 2 design elements I removed CONV2 and its associated ReLU, and sizes were adjusted accordingly. This required me to adjust how we reshape/flatten my code to refer to pooling layer outputs and array shapes.

We can see in the Design:

**Convolution layer 1:**

Convolution 4D shape: [batch, Height, Width, depth] = [ 1, 28, 28, 32]

2D data size: 28 x28

Input shape: 28 x 28 x 1 (Depth = 1 in channel)

Output shape: 28 × 28 × 32 (Depth = 32 out channels)

Filter/Kernel/Window size = 5 x 5

Filter shape - weight shape: [5, 5, 1, 32]

Stride = 1

Stride shape: (1, 1, 1, 1]

Padding = SAME

Activation Function Layer: ReLU

**ReLu Layer 1:**

No Filter (only process data, not extract/learn features)

Input shape: 28x28x32

Output shape: 28x28x32

**Pooling Layer:**

Pooling Layer 1

Pooling method: Max pooling

Filter/Kernel/Window size - 2 x2

Filter/Kernel/Window shape: [1, 2, 2, 1]

Stride = 2

Stride shape: [1, 2. 2, 1]

Padding - SAME

Input channels: 32 inputs

Input shape: 28 x 28 x 32

Output channels: 32 outputs

Output shape: 14 x 14 x 32

**Fully Connected layers:**

Fully Connected Layer 1 (FC 1)

FC\_1 shape: [ inputs (in \_channels), outputs (out\_channels)]

(7\*7 x64, 1024)

* **Screenshot below.**

**A screenshot of a computer

Description automatically generated**

On Steps 0-300

**A screenshot of a computer

Description automatically generated**

On Steps 4700 – 4900

**A screenshot of a computer program

Description automatically generated**

**Reports:**

The incremental improvement in accuracy has steadily risen because of employing both the ReLU activation layer and a solitary CNN layer within the project. Notably, when compared to alternative techniques involving alterations in filter shapes at the input and output stages, the accuracy of 0.9895 stands out significantly. This firmly demonstrates the capacity of these modifications to enhance overall efficiency and effectiveness.

* Results Screenshot:

A screenshot of a computer

Description automatically generatedA screen shot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA close up of a number

Description automatically generatedA screenshot of a white background

Description automatically generated**PART III: Update Number of Steps of Training CNN Model (10 Points)**

For part 3 we are required to modify the single convolution layer code from part 2 by changing the number of steps from 5000 to 3000. This simple change required no additional modifications for the model to execute successfully.

* **Screenshot below.**

On Step 0 - 300

**A screenshot of a computer

Description automatically generated**

On Step 2700 - 2900

**A screenshot of a computer

Description automatically generated**

**Report:**

The results using the ReLu activation layer show that the step increase accuracy has been improving based on the single CNN layer that we used in the project. When compared to the accuracy of all the other layers, the accuracy has also been shown to be significantly better with 0 showing an accuracy of 0.2201 and the final test result for step 2900 showing an accuracy of 0.9877, showing that filter shape changes at the input and output changes are fully capable of increasing efficiency based on the step-by-step evolution of accuracy based on the steps. As a result, accuracy can be increased overall by using a single CNN layer with the highest steps.

* **Screenshot Hare results in steps below.**

A black text on a white background

Description automatically generatedA white background with black text

Description automatically generatedA screenshot of a computer

Description automatically generatedA white background with black text

Description automatically generated