NAME: KSHITIJ VINOD SALI
CLASS: BE-A ROLL NO.:
SUBJECT: LP-IV: DEEP LEARNING

LAB ASSIGNMENT - 03

Title: Building an End-to-End Image Classification Model.

Course Outcome: CO3- Apply deep learning techniques like CNN, RNN Auto encoders to solve real world problems.
CO4- Evaluate the performance of the model build using deep learning.

Date of Completion:
Assessment Grade/ Marks:
Assessor's Sign with Date:

Problem Statement: Build the image classification model by dividing the model into following 4 stages:
a) Loading and preprocessing the image data.
b) Defining the model's architecture.
c) Training the model
d) Estimating the model's performance.

Blooms Taxanomy Catagory: Create, Evaluate.

Requirements: Anaconda with Python 3.9.12,

Jupyter Notebook, Python Frameworks and Python Libraries, Dataset

Theory:

Convolutional Neural Network (CNN) is an architectural design specially designed for processing grid-like data such as images. Unlike an MLP, which treats an image as long vector of pixels & thus discards all spatial information, a CNN is designed to preserve & learn from spatial relationships between pixels. A CNN arranges its neurons in three dimensions: width, height & depth.

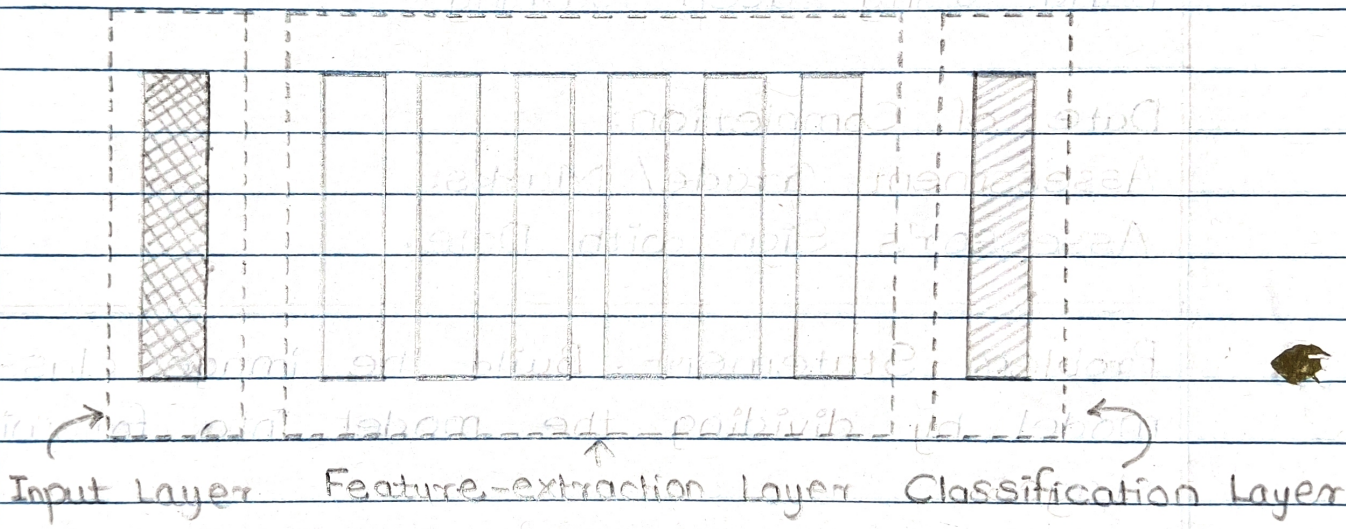

Fig: High-level general CNN architecture

i) Input Layer: This layer holds raw pixel data of image. (For MNIST, this is 28x28 matrix) For CNN, this is represented as 3D volume, including a depth dimension for colour channels.

ii) Feature-Extraction Layer: This is core of CNN, where hierarchical features are learned automatically. It typically consists of a repeating sequence of layers:

a) Convolutional Layer - This layer applies a set of learnable filters to input image. Each filter slides across width & height of input, computing a dot product to create a 2D activation map of that filter's response.

b) ReLU Activation - After convolution, a non-linear activation function like ReLU ($f(x) = \max(0, x)$) is applied to the feature maps.

c) Pooling Layer - The pooling layer downsamples the feature maps, reducing their spatial dimensions. This reduces number of parameters & convolutional complexity & it also makes learned features more robust to small translational in input image.

iii) Classification Layer: After several convolution & pooling layers, high-level feature maps are flattened into 1D vector. This vector is fed into one or more fully connected layers, similar to MLP, ~~from~~ A Final layer uses softmax activation function to produce class probabilities.

Algorithm and Steps:

i) Loading & Preprocessing the Image Data:

(x-train, y-train), (x-test, y-test) = mnist.load-data()

```
# Scale pixel values
x_train = x_train / 255.0
x_test = x_test / 255.0

# Reshape data to include channel dimension
x_train = x_train.reshape ((x_train.shape, 28,
28, 1))
x_test = x_test.reshape ((x_test.shape, 28,
28, 1))
```

ii) Defining the Model's Architecture: A sequential model is defined, but with convolutional & pooling layers.

```
model = Sequential ()
```

iii) Training the Model: The model is compiled with the SGD optimizer and sparse_categorical_crossentropy loss function. It is then trained for 10 epochs on the training data.

```
optimizer = SGD (learning_rate = 0.01, momentum = 0.9)
model.compile (
    optimizer = optimizer,
    loss = "sparse_categorical_crossentropy",
    metrics = ["accuracy"]
)
model.fit (x_train, y_train, epochs = 10, batch_size = 32)
```

iv) Estimating the Model's Performance: The trained model is evaluated on unseen test set to measure its final performance. The accuracy is calculated using Scikit-learn's accuracy_score.

```
predictions = np.argmax(model.predict(x_test)
axis=-1)
print(accuracy_score(y_test, prediction))
```

Additionally, the performance over time is visualized by plotting the training & validation accuracy & loss curves. These plots are essential for diagonising issues like overfitting & for confirming the model training has converged.

Inference: Thus, we have implemented the Image classification model using CNN. With the code we can see that sufficient accuracy has been met. Throughout, the epochs, our model accuracy increases and loss decreases that is good since our model gains confidence with our prediction.