USN NUMBER: 1RVU22BSC097
NAME: SHREYA SINHA

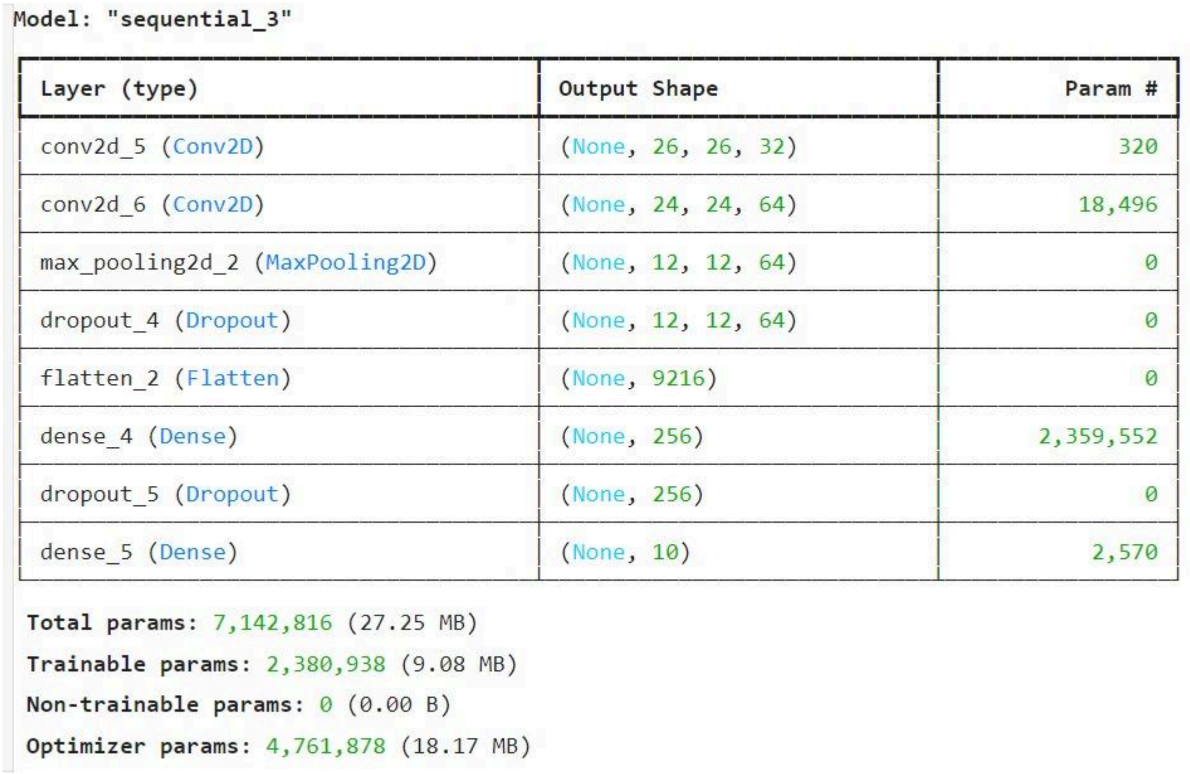
Ex No: 4 **Date: 28-08-24**	**Handwritten Digit Recognition using Convolutional Neural Network (CNN)**

Objective: To develop a Convolutional Neural Network (CNN) model for recognizing handwritten digits using the MNIST dataset.

Descriptions:Handwritten digit recognition is a well-known problem in the field of computer vision, where the goal is to classify images of handwritten digits into one of 10 classes (0-9). In this lab, we utilize a CNN model to achieve high accuracy in digit classification.

Model:The CNN model is designed with the following architecture:

1. **Convolutional Layers:**
   - **Two convolutional layers are used to extract features from the input images.**
   - **The first layer has 32 filters of size 3x3, and the second has 64 filters of the same size.**
2. **Pooling Layer:**
   - **A MaxPooling layer with a pool size of 2x2 is used to reduce the spatial dimensions of the feature maps.**
3. **Dropout Layer:**
   - **A dropout layer with a dropout rate of 25% is included to prevent overfitting.**
4. **Flatten Layer:**
   - **The feature maps are flattened into a 1D vector for input into the dense layers.**
5. **Dense Layers:**
   - **Two fully connected (Dense) layers are added. The first has 256 units with ReLU activation, followed by a dropout layer with a 50% dropout rate. The second Dense layer has 10 units (corresponding to the 10 classes) with softmax activation.**
6. **Compilation:**
   - **The model is compiled using the categorical cross entropy loss function, the Adadelta optimizer, and accuracy as the evaluation metric.**



```
Model: "sequential_3"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_5 (Conv2D)                    │ (None, 26, 26, 32)          │             320 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_6 (Conv2D)                    │ (None, 24, 24, 64)          │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 12, 12, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_4 (Dropout)                  │ (None, 12, 12, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten_2 (Flatten)                  │ (None, 9216)                │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_4 (Dense)                      │ (None, 256)                 │       2,359,552 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_5 (Dropout)                  │ (None, 256)                 │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_5 (Dense)                      │ (None, 10)                  │           2,570 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 7,142,816 (27.25 MB)
 Trainable params: 2,380,938 (9.08 MB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 4,761,878 (18.17 MB)
```

Building the Parts of the Algorithm:

1. **Define the model structure:**
   - Number of input features: 28x28x1 for grayscale images.
   - Number of output classes: 10 (digits 0-9).
2. **Initialize the model's parameters:**
   - Weights and biases are initialized automatically by the Keras framework.

Accuracy:

1. Test loss: 0.7110332250595093
2. Test accuracy: 0.836899995803833

Implementation:

1. **Import Libraries:**
   - Necessary libraries such as `tensorflow`, `keras`, `numpy`, and `matplotlib` are imported.
2. **Data Loading and Preprocessing:**
   - The MNIST dataset is loaded, and data is preprocessed by normalizing the pixel values and converting labels to one-hot encoded vectors.
3. **Model Creation:**
   - The CNN model is created using Keras Sequential API.
4. **Model Training:**

- The model is trained on the training set using a batch size of 128 and for 10 epochs.

5. **Model Evaluation:**
   - The trained model is evaluated on the test set, and the test loss and accuracy are printed.
6. **Model Saving:**
   - The trained model is saved to disk as `mnist.h5`.
7. **GUI for Prediction:**
   - A simple GUI using `Tkinter` is created to predict digits drawn by the user.

GitHub Link:
https://github.com/Shreyasinha7/Lab-4--Handwritten-digit-recognition-using-CNN.git