A

SYNOPSIS

of

MINOR PROJECT

on

# Handwritten Digit Recognition

तमसो मा ज्योतिर्गमय्

Darkness to Light

*Submitted by*

*Yashvi Chittora(22EGICS158)*

**Project Guide**<br>
**Mr. Ritesh Kumar Jain**

**Head of Department**<br>
**Dr. Mayank Patel**

**Geetanjali Institute of Technical Studies, Dabok , Udaipur (Raj.)**
**Department of Computer Science and Engineering**
**October,2023**

## Problem Statement:

The goal of this project is to develop an advanced handwritten digit recognition system using neural networks. Traditional systems are limited to recognizing pre-scanned images of handwritten digits. Our enhanced system not only recognizes scanned images but also includes a graphical user interface (GUI) for real-time digit entry and recognition, providing a more interactive and versatile solution.

## Brief Description:

This project focuses on creating a neural network-based system for recognizing handwritten digits. The system utilizes the MNIST dataset, a well-known dataset of handwritten digit images, to train and test the neural network. Additionally, the project incorporates a GUI that allows users to write digits on the screen and receive real-time recognition feedback.

## Objective and Scope:

## Objective:

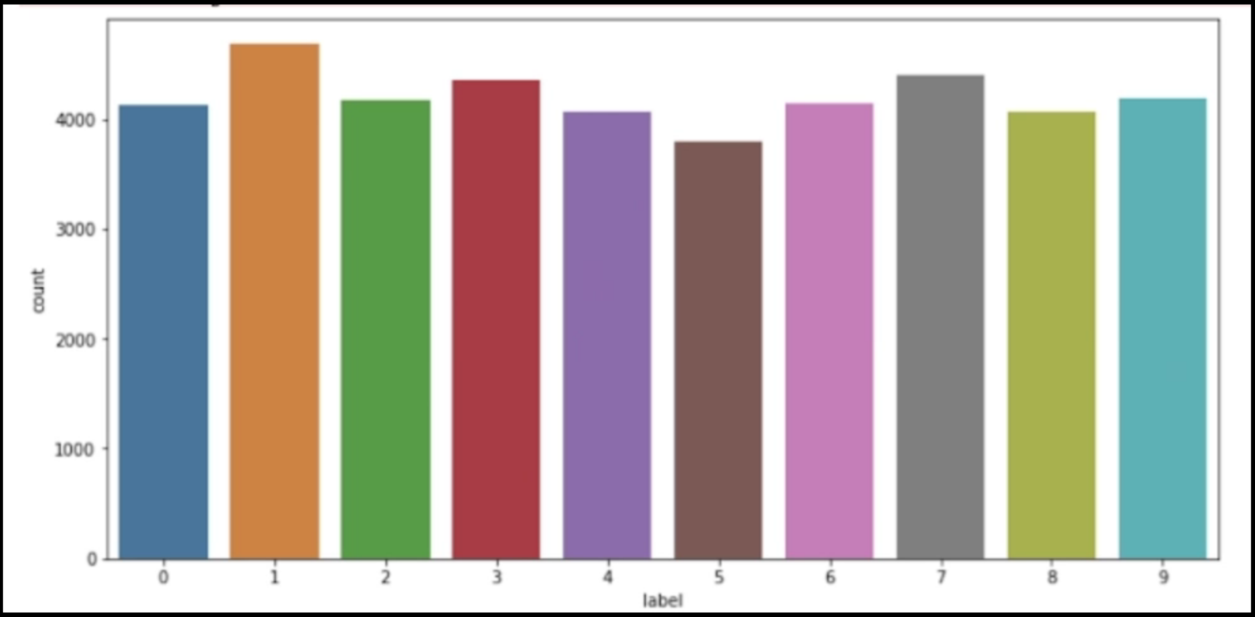
- Develop a robust handwritten digit recognition system using a neural network.
- Extend the functionality to include real-time digit entry and recognition via a GUI.

## Scope:

- Use the MNIST dataset for training and testing the neural network.
- Implement a three-layer neural network with one hidden layer for digit recognition.
- Create an interactive GUI for real-time digit input and recognition.

## Methodology:

1. Data Preprocessing:
   - Load the MNIST dataset.
   - Normalize the pixel values to the range [0, 1].
   - Split the data into 60,000 training examples and 10,000 testing examples.

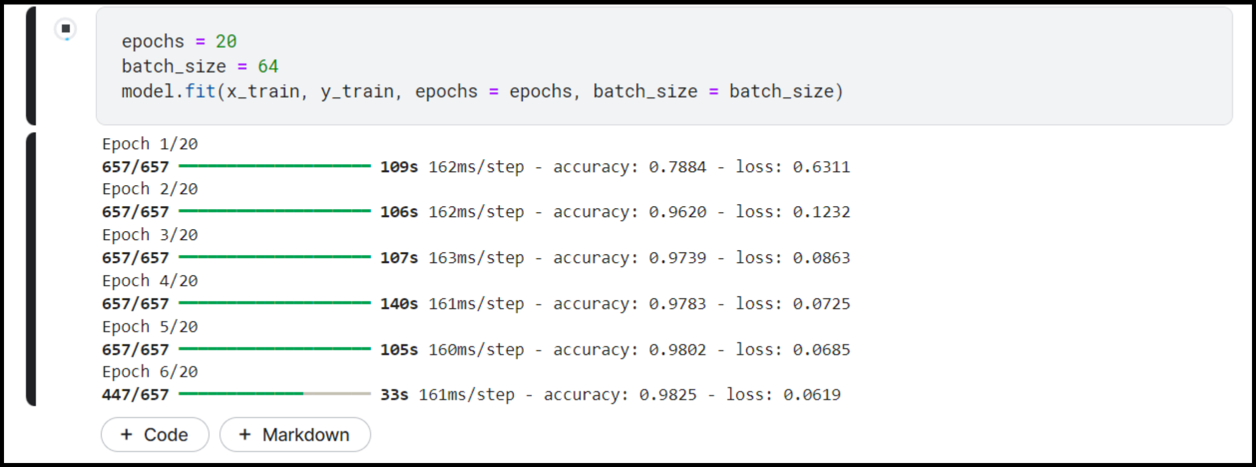


**2. Neural Network Architecture:**

- **Input Layer: Distributes features to the next layer.**
- **Hidden Layer: Contains 100 activation units (neurons) to introduce nonlinearity.**
- **Output Layer: Contains 10 output units, each representing a digit (0-9).**

**3. Training Process:**

- **Perform feedforward propagation to compute activations.**
- **Implement backpropagation to minimize the error between predicted and actual outputs.**
- **Use a regularization parameter (lambda) set to 0.1 to prevent overfitting.**
- **Optimize the network for 70 iterations to achieve the best performance.**



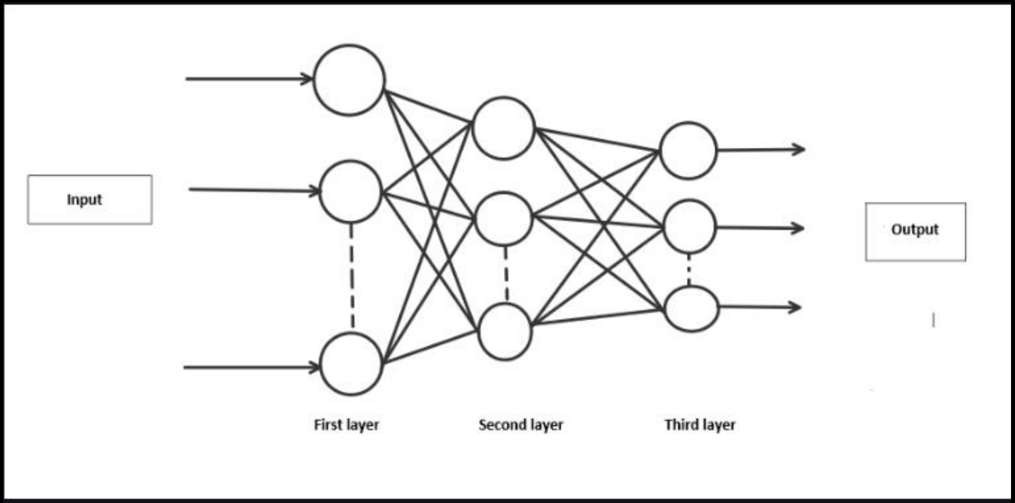
```
epochs = 20
batch_size = 64
model.fit(x_train, y_train, epochs = epochs, batch_size = batch_size)
```

```
Epoch 1/20
657/657 ━━━━━━━━━━━━━━━━━━━━ 109s 162ms/step - accuracy: 0.7884 - loss: 0.6311
Epoch 2/20
657/657 ━━━━━━━━━━━━━━━━━━━━ 106s 162ms/step - accuracy: 0.9620 - loss: 0.1232
Epoch 3/20
657/657 ━━━━━━━━━━━━━━━━━━━━ 107s 163ms/step - accuracy: 0.9739 - loss: 0.0863
Epoch 4/20
657/657 ━━━━━━━━━━━━━━━━━━━━ 140s 161ms/step - accuracy: 0.9783 - loss: 0.0725
Epoch 5/20
657/657 ━━━━━━━━━━━━━━━━━━━━ 105s 160ms/step - accuracy: 0.9802 - loss: 0.0685
Epoch 6/20
447/657 ━━━━━━━━━━━━━━━━━━━━ 33s 161ms/step - accuracy: 0.9825 - loss: 0.0619
```

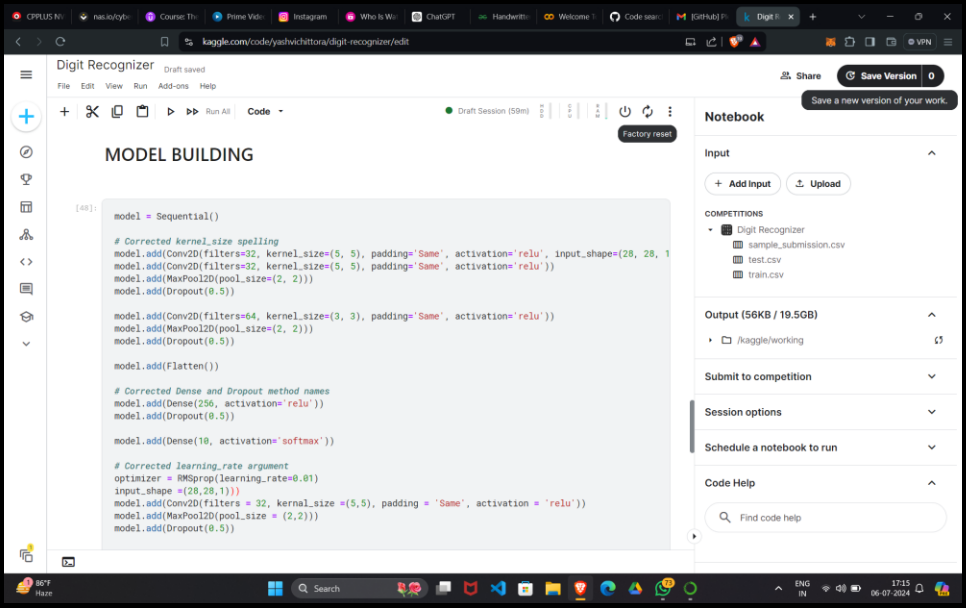
## 4. GUI Integration:

**- Develop a graphical user interface for real-time digit input.**

**- Integrate the trained neural network with the GUI for immediate recognition feedback.**



*Layers of Neural Network*



MODEL BUILDING

```
model = Sequential()

# Corrected kernel_size spelling
model.add(Conv2D(filters=32, kernel_size=(5, 5), padding='Same', activation='relu', input_shape=(28, 28, 1
model.add(Conv2D(filters=32, kernel_size=(5, 5), padding='Same', activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='Same', activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Flatten())

# Corrected Dense and Dropout method names
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(10, activation='softmax'))

# Corrected learning_rate argument
optimizer = RMSprop(learning_rate=0.01)
input_shape =(28,28,1)))
model.add(Conv2D(filters = 32, kernal_size =(5,5), padding = 'Same', activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout(0.5))
```

## Hardware and Software Requirements:

### Hardware:

- Standard computer with sufficient RAM (at least 8GB) and CPU (preferably multi-core).

### Software:

- Python 3.x
- TensorFlow and Keras libraries
- Matplotlib for visualization
- GUI toolkit (such as Tkinter or PyQt)

**Technologies:-**

**- Programming Language: Python**

**- Machine Learning Framework: TensorFlow and Keras**

**- Data Visualization: Matplotlib**

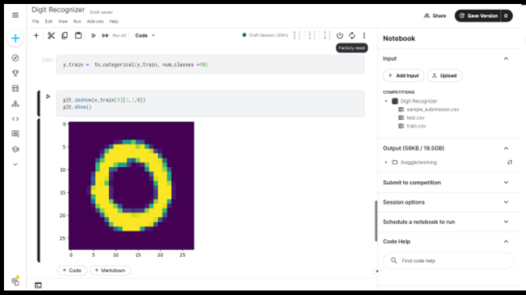
**- GUI Development: Tkinter or PyQt**

**Testing Techniques:**

**- Unit Testing: Ensure individual components (e.g., data preprocessing, model training) function correctly.**

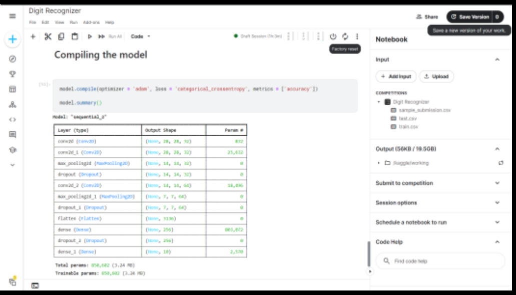
**- Integration Testing: Verify that the neural network and GUI components work seamlessly together.**

**- Performance Testing: Evaluate the accuracy and speed of the digit recognition system on both the MNIST test set and real-time input via the GUI.**



```
[38]: y_train =  to_categorical(y_train, num_classes =10)
```

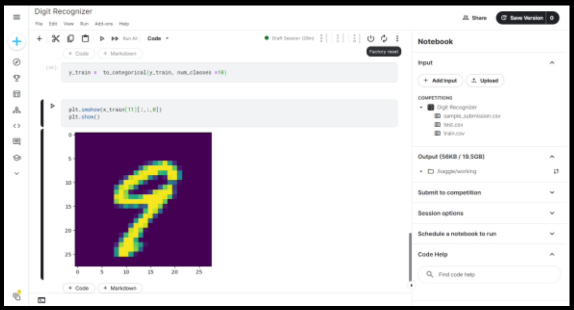
```
plt.imshow(x_train[1][:,:,0])
plt.show()
```



Compiling the model

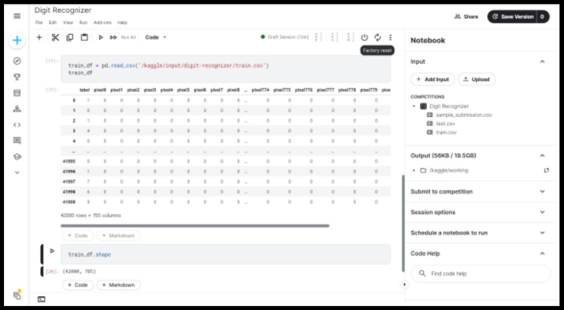
```
[51]: model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])

      model.summary()
```

```
Model: "sequential_2"
┌──────────────────────────────┬──────────────────────┬───────────┐
│ Layer (type)                 │ Output Shape         │   Param # │
├──────────────────────────────┼──────────────────────┼───────────┤
│ conv2d (Conv2D)              │ (None, 28, 28, 32)   │       832 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ conv2d_1 (Conv2D)            │ (None, 28, 28, 32)   │    25,632 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ max_pooling2d (MaxPooling2D) │ (None, 14, 14, 32)   │         0 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ dropout (Dropout)            │ (None, 14, 14, 32)   │         0 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ conv2d_2 (Conv2D)            │ (None, 14, 14, 64)   │    18,496 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ max_pooling2d_1 (MaxPooling2D)│ (None, 7, 7, 64)    │         0 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ dropout_1 (Dropout)          │ (None, 7, 7, 64)     │         0 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ flatten (Flatten)            │ (None, 3136)         │         0 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ dense (Dense)                │ (None, 256)          │   803,072 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ dropout_2 (Dropout)          │ (None, 256)          │         0 │
├──────────────────────────────┼──────────────────────┼───────────┤
│ dense_1 (Dense)              │ (None, 10)           │     2,570 │
└──────────────────────────────┴──────────────────────┴───────────┘
 Total params: 850,602 (3.24 MB)
 Trainable params: 850,602 (3.24 MB)
```



```
[28]: y_train =  to_categorical(y_train, num_classes =10)
```

```
plt.imshow(x_train[11][:,:,0])
plt.show()
```



```
[19]: train_df = pd.read_csv("/kaggle/input/digit-recognizer/train.csv")
      train_df
```

```
[19]:
        label pixel0 pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 pixel8 ... pixel774 pixel775 pixel776 pixel777 pixel778 pixel779 pixe
    0       1      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
    1       0      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
    2       1      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
    3       4      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
    4       0      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
  ...     ...    ...    ...    ...    ...    ...    ...    ...    ...    ... ...      ...      ...      ...      ...      ...      ...
41995       0      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
41996       1      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
41997       7      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
41998       6      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0
41999       9      0      0      0      0      0      0      0      0      0 ...        0        0        0        0        0        0

42000 rows × 785 columns
```

```
train_df.shape
```

```
[20]: (42000, 785)
```

## Project Contribution:

This project enhances the traditional handwritten digit recognition systems by incorporating real-time digit input and recognition capabilities. The integration of a GUI makes the system more user-friendly and practical for a wider range of applications, from educational tools to assistive technologies. By implementing a neural network with regularization and optimization techniques, the project ensures high accuracy and reliability in digit recognition tasks.

**Geetanjali Institute of Technical Studies, Dabok , Udaipur (Raj.)**
**Department of Computer Science and Engineering**
**October,2023**