

Handwritten Digit Recognition

## Rahma Mohamed Makram

## Rana Mohamed Hussein Mohamed Bekheet

June 14, 2022

## Sarah Gamal El-Deen

Icon

Description automatically generated with medium confidence

## Group No. 22

## Project Name: Handwritten Digit Recognition

## Solwan Shokry Ahmed

# Problem Description

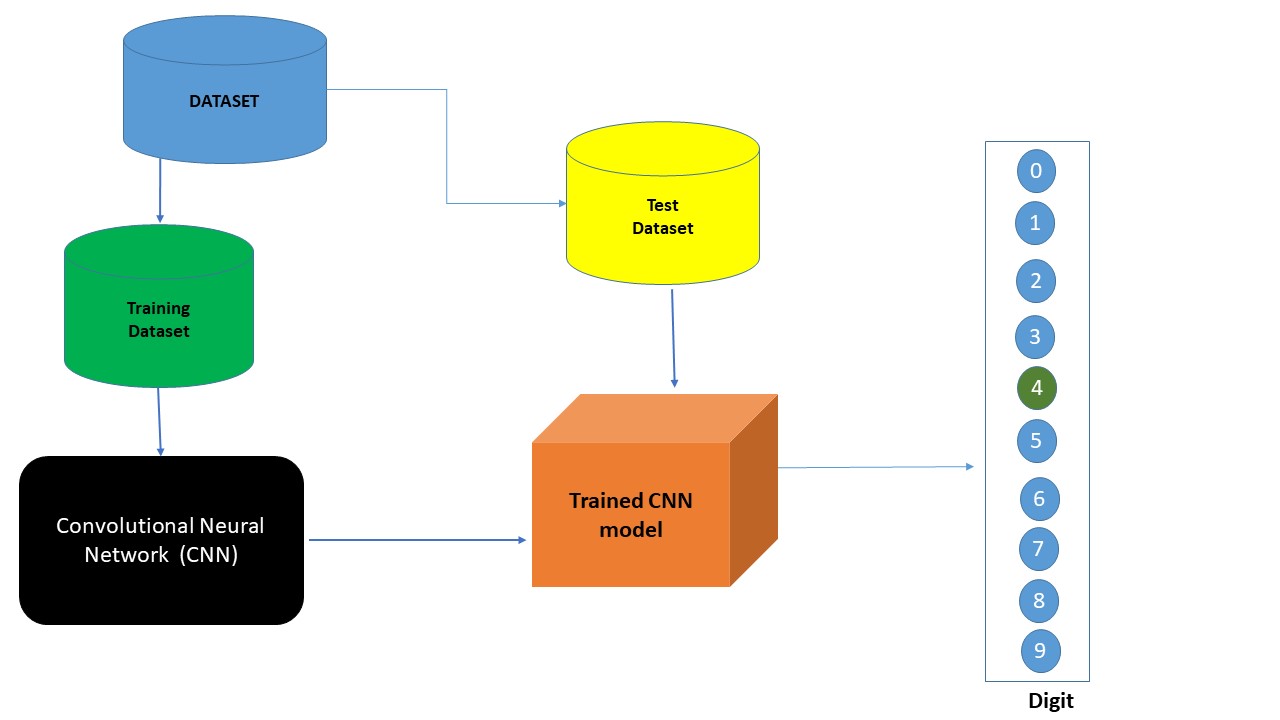
Optical character recognition or optical character reader is the electronic or mechanical conversion of images of typed, handwritten, or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image. The handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavors. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

# Proposed Solution

## Block diagram

Diagram

Description automatically generated



## Implementation

## Tools used:

1. Jupyter Notebook.
2. Computer vision library: OpenCV
3. Network used: Convolutional Network
4. Dataset: MNIST Dataset
5. Framework used: TensorFlow
6. API : Keras
7. GUI : Tkinter library.

Description:

Dataset: The MNIST database of handwritten digits contains 28x28 grayscale images of the 10 digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. Keras is a deep learning API written in Python and MNIST is a dataset provided by this API.

**Training Set**

* 60000 images
* Each image is of the shape 28 x 28 (rows x columns)
* 60000 labels defining the digit that corresponds to the respective image

**Test Set**

* 10000 images
* Each image is of the shape 28 x 28 (rows x columns)
* 10000 labels defining the digit that corresponds to the respective image

Training Model (Convolutional neural network CNN) :

* Input (28x28x1 matrix) -> Conv(32 3x3 filters, relu) -> Maxpool(2x2) -> Conv(64 3x3 filters, relu) -> Maxpool(2x2) -> Conv(64 3x3 filters, relu) -> Flatten to a column vector for FCN -> FCN (64 outputs, relu) -> FCN (10 outputs, softmax)
* Training Accuracy = **~99.4%**
* Test Accuracy = **~99.2%**

1. **Steps for Digit Detection:**
2. Import the libraries and load the dataset.

Import all the modules that we are going to need for training our model.

1. Preprocess the data.

* The image data cannot be fed directly into the model, so we need to perform some operations and process the data to make it ready for our neural network. The dimension of the training data is (60000,28,28). The CNN model will require one more dimension, so we reshape the matrix to shape (60000,28,28,1), then the data is scaled, normalized and categoricalized.

1. Create the model.

* Now we will create our CNN model in Python. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures; this is the reason why CNN works well for image classification problems. The dropout layer is used to deactivate some of the neurons and while training, it reduces over fitting of the model.

1. Train the model.

* The model.fit() function of Keras will start the training of the model. It takes the training data, validation data, epochs, and batch size. It takes some time to train the model. After training, we save the weights and model definition in the ‘mnist.h5’ file.

1. Evaluate the model.

* We have 10,000 images in our dataset which will be used to**evaluate how good our model works**. The testing data was not involved in the training of the data therefore, it is new data for our model. The MNIST dataset is well balanced so we can get around 99% accuracy.

1. Obtain the image to be predicted from the user
2. Perform processing on the image and predict digit :

The image is converted to a grey scale image and color inverted, the image is introduced to the trained model to make predictions

In case of the web cam prediction the image processing also include:

* + 1. Gaussian Blur
    2. Thresholding
    3. Canny Edge detection
    4. Find contours
    5. Find Contour area

# How to use the application (manual)

1. Run the python notebook “Cvproject2”.
2. GUI will appear with 3 buttons and a note pad.

2.1. The “Recognize” button which you press after drawing the digit in the note pad which will show the detected number and its accuracy.

2.2. The “Clear” button which deletes the digit you drew and allow you to repeat step 1.

2.3. A “Live Detect” button which will:

2.3.1 Open the Camera.

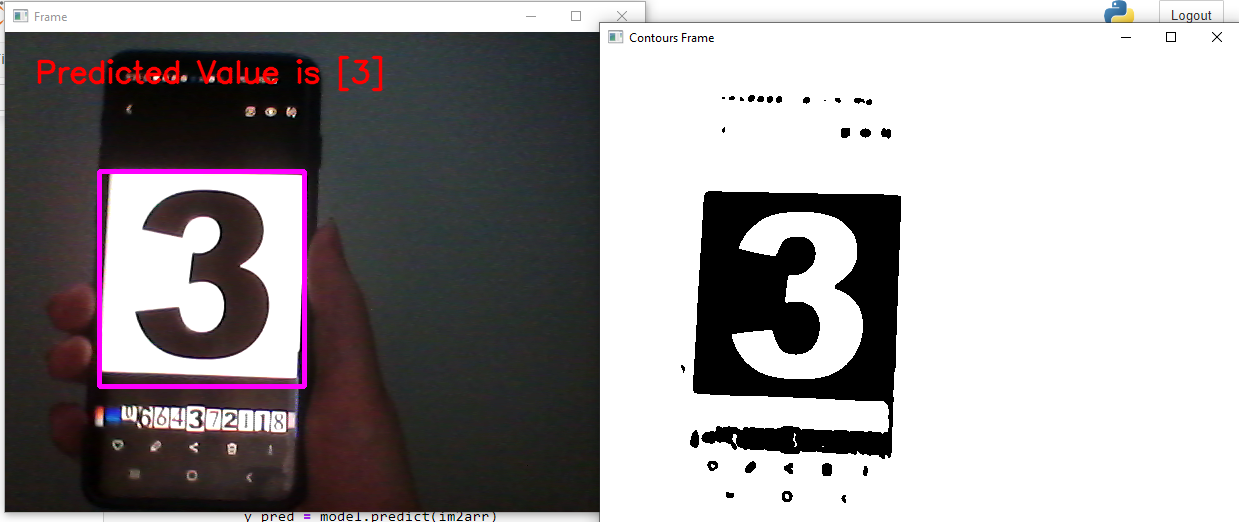
2.3.2 Then you should place a handwritten digit in front of it.

2.3.3 The predicted value of the digit will appear on the frame.

Graphical user interface, application

Description automatically generated





# How the work is distributed among team members

|  |  |
| --- | --- |
| Name | Task |
| Rahma Mohamed Makram | * GUI with tkinter library. * Choice of the deep learning model. * CNN model implementation and initial training and testing. * Model deployment on the Created GUI. * Project presentation. * Project report. |
| Rana Mohamed Hussein Mohamed Bekheet | * Searching for deep learning Models and application functionality testing. * Model deployment on the Created GUI. * Project report. * Project presentation. |
| Sarah Gamal El-Deen | * CNN Model accuracy enhancement and final training and testing. * CNN Model parameters’ fitting check and adjusting the network’s depth. * Project report. * Project presentation. |
| Solwan Shokry Ahmed | * GUI with tkinter library. * Choice of the deep learning model. * CNN model implementation and initial training and testing. * Model deployment on the Created GUI. * Project presentation. * Project report. |