Handwritten Digit Recognition using Convolutional Neural Networks with Keras

A

PROJECT REPORT

Submitted for

IT 3140 (Soft Computing)

Internal Assessment Component

Submitted by

Sarthak Ahuja - 199301246

Shivansh Syal - 199301241

Dept. of Computers and Communication

Submitted To

Prof. (Dr.) Sumit Srivastava

Professor

Dept. of Information Technology

MANIPAL UNIVERSITY JAIPUR

Information Technology

MANIPAL UNIVERSITY JAIPUR

JAIPUR-303007

RAJASTHAN, INDIA

December 2021

ABSTRACT

The handwritten digit recognition is the ability of computers to recognize human handwritten digits. A number of algorithms are used To recognise pictures, it was created in the field of computer vision. Our effort are focused on developing a model that will be able to predict future events. to recognise and identify a handwritten digit from its picture more precisely. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different ways. We intend to implement a handwritten digit recognition app using the MNIST dataset and a special type of deep neural network that is Convolutional Neural Networks. We have also implemented a GUI in the form of an interactive window to draw digits on canvas which then uses the CNN model to recognize the drawn image.

INTRODUCTION

A Convolutional Neural Network or CNN is a Deep Learning Algorithm which is very effective in handling image classification tasks. It is able to capture the Temporal and Spatial dependencies in an image with the help of filters or kernels. Convolutional Neural Networks (CNN) have recently emerged as one of the most appealing methodologies, and they have played a key role in a number of recent successes and demanding machine learning applications, such as ImageNet object identification, image segmentation, and face recognition. As a result, we chose CNN for our classification jobs of recognising handwritten numbers,

The MNIST database is a collection of handwritten digits. It may be used to train different image processing systems. In the field of machine learning, the database is also commonly utilised for training and testing. There are 60,000 training and 10,000 testing examples in this programme.

Each picture has a predetermined size. The photos are 28*28 pixels in size. It's a database for people who want to experiment with machine learning and pattern recognition techniques on real-world data with minimal pre-processing and formatting. This database will be used in our experiment.
Convolutional Neural Networks Convolutional neural networks are deep artificial\sneural networks. It may be used to classify photographs (for example, naming what they see), cluster them based on similarities (photo search), and recognise objects within scenes. It can be used to recognise faces, people, street signs, tumours, platypuses, and a variety of other visual data. A CNN's main building piece is the convolutional layer. The parameters of the layer are made up of a series of learnable filters (or kernels) with a narrow receptive field but that span the whole depth of the input volume. Each filter is convolved over the width and height of the input volume during the forward pass, computing the dot product and providing a two-dimensional activation map for that filter. As a consequence, the network learns when it sees a certain sort of feature at a given spatial location in the input.

TECHNOLOGIES INVOLVED

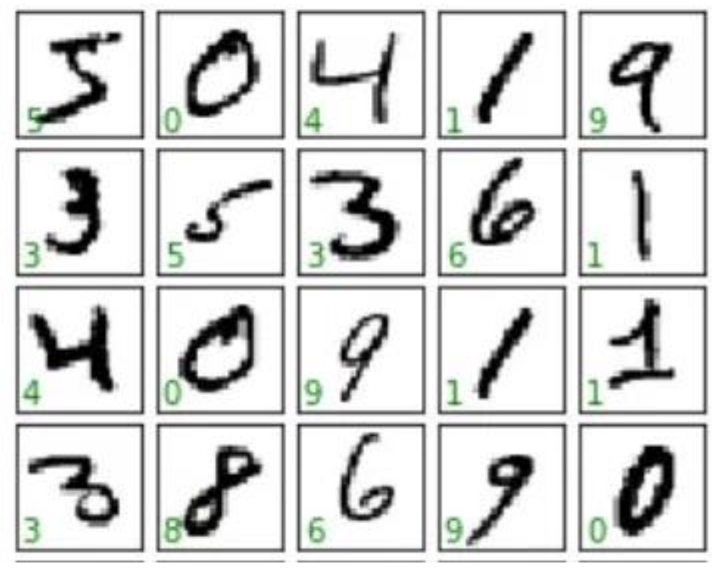
- Keras
- CNN
- TensorFlow
- numPy
- Tkinter (for GUI)
- MNIST database
- CV2

METHODOLOGY

Deep Learning has emerged as a key tool for self-perception issues such as picture interpretation, human speech recognition, and robot exploration of the world. We want to use the Convolutional Neural Network idea for digit recognition. The suggested approach aims to understand CNN and apply it to a handwritten digit recognition system. The features maps are extracted from the 2D photos using a Convolutional Neural Network.

Firstly, we will train a CNN (Convolutional Neural Network) on MNIST dataset, which contains a total of 70,000 images of handwritten digits from 0-9 formatted as 28×28-pixel monochrome images.Then, we will preprocess the

input data by reshaping the image and scaling the pixel values between 0 and 1.
After that, we will design the neural network and train the model. After the model is trained, we will save it for future use and using the GUI,our model will process the image to identify the digit and return a series of 10 numbers corresponding to the ten digits with an activation on the index of the proposed digit.



MNIST DATASET

Rather than having a fully connected layer of neurons, the convolutional neural network examines the mapping of picture pixels with the neighbourhood space. In signal and image processing, the convolutional neural network is a useful tool. Even in computer vision disciplines like handwriting recognition, natural object categorization, and segmentation, CNN has shown to be a far superior tool than any other previously used methods. The overall goal could be to create a machine learning model that could recognize people’s handwriting.

ARCHITECTURE OF THE PROPOSED MODEL

When learning deep learning with a neural network, one quickly understands how vital CNN is for picture categorization. Convolutional Neural Networks are a type of multi-layer neural network that can identify visual patterns directly from pixel pictures with little or no preprocessing. Almost all CNN architectures follow the same fundamental design concepts of applying convolutional layers to the input in a sequential manner, occasionally downsampling the spatial dimensions (Max pooling), and increasing the number of feature maps. Fully linked layers, activation functions, and loss functions (e.g., cross entropy or

softmax) are also included. Convolutional layers, pooling layers, and fully linked layers, however, are the most significant of CNN's processes.
As a result, before presenting our proposed model, we'll quickly go over these layers.
The first layer to extract features from pictures is the convolutional layer. Because pixels are only connected to those in their immediate vicinity, The connection between distinct sections of a picture can be preserved via convolution. Convolution is the process of reducing the size of a picture while maintaining the connection between pixels by filtering it with a smaller pixel filter. When we use a 3x3 filter with 1x1 stride (1-pixel shift at each step) to apply convolution to a 5x5 picture, we get a 3x3 output (64 percent reduction in complexity)

Pooling layers are commonly included after each convolution layer in CNN to minimise the spatial size of the features maps. Overfitting can also be mitigated by pooling layers. By picking the maximum, average, or total values inside these pixels, we choose a pooling size to decrease
the number of parameters. One of the most prevalent pooling strategies is Max Pooling, which may be shown as follows:

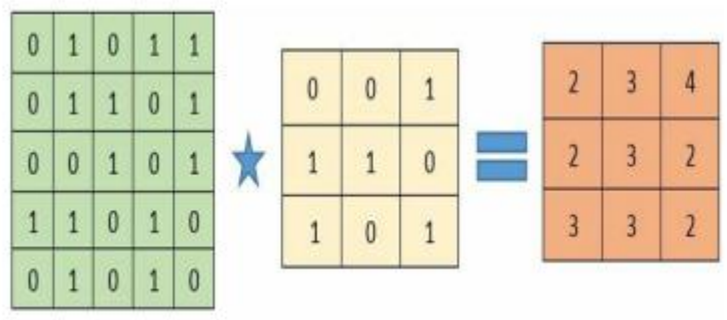


Fig. 1: Convolution operation

Any design in which each parameter is linked to one another to identify the relationship and influence of each parameter on the labels is referred to as a fully connected network. We can build a fully connected network to identify the photos in the end since convolution and pooling layers minimise time-space complexity.

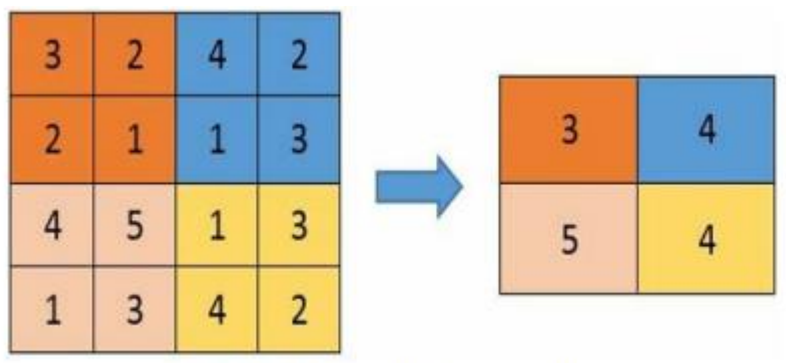


Fig. 2: Max pooling operation

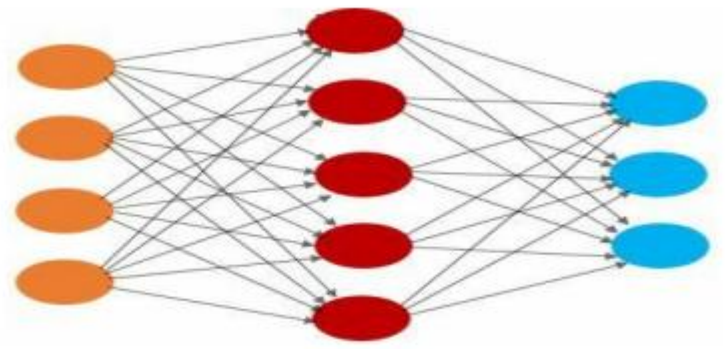


Fig. 3: Fully connected layers

We believe it is now appropriate to provide an overview of our proposed convolutional neural network. It is comparable to existing handwritten digit recognition designs [1,6,8,10,11], but it has improved performance by changing a number of filters, neurons, and activation functions. It is made up of seven layers.
The Model's Explanation
A basic convolutional neural network is made up of layers, each of which uses a differentiable function to translate one volume of activations to another. To construct the network, we need three different sorts of layers. Convolutional layers, pooling layers, and completely linked layers are the three types. Our network architecture will be built by stacking these layers. Below, we'll go through the specifics in further depth.

STEPS FOR MAKING THE PROJECT

1) Import the libraries and load the dataset

```
import os
import cv2
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
```

2) Decide if to load an existing model or to train a new one

```
if not os.path.isfile('handwritten_digits.h5'):
```

3) Loading the MNIST dataset and Splitting it into Train and Test

```
# Loading the MNIST data set with samples and splitting it
    mnist = tf.keras.datasets.mnist
    (X_train, y_train), (X_test, y_test) = mnist.load_data()
```

4) Process the data and normalizing it

```
# Normalizing the data (making length = 1)
    X_train = tf.keras.utils.normalize(X_train, axis=1)
    X_test = tf.keras.utils.normalize(X_test, axis=1)
```

5) Defining the model architecture

```
# Create a neural network model
# Add one flattened input layer for the pixels
# Add two dense hidden layers
# Add one dense output layer for the 10 digits
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu))
    model.add(tf.keras.layers.Dense(units=128, activation=tf.nn.relu))
    model.add(tf.keras.layers.Dense(units=10, activation=tf.nn.softmax)
```

6) Compiling the model

```
# Compiling and optimizing model
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
```

7) Fitting the model

```
 # fitting the model
    model.fit(X_train, y_train,
          batch_size=128,
          epochs=10,
          verbose=1,
          validation_data=(X_test, y_test))
```

8) Evaluating the model on test data and saving it

```
# Evaluating the model
    val_loss, val_acc = model.evaluate(X_test, y_test)
    print(val_loss)
    print(val_acc)

    # Saving the model
    model.save('handwritten_digits.h5')
```

9) Loading Custom Images and predicting them

```
image_number = 1
while os.path.isfile('digits/digit{}.png'.format(image_number)):
    try:
        img = cv2.imread('digits/digit{}.png'.format(image_number))[:, :, 0]
        img = np.invert(np.array([img]))
        prediction = model.predict(img)
        print("The number is {}".format(np.argmax(prediction)))
        plt.imshow(img[0], cmap=plt.cm.binary)
        plt.show()
        image_number += 1
    except:
        print("Error reading image! Proceeding with next image...")
        image_number += 1
```

10) Create GUI

Here, We create a GUI using Tkinter and win32gui. We have created a function predict_digit() that takes the image as input and then uses the trained model to predict the digit. We create a canvas where we can draw by capturing the mouse event and with a button, we trigger the predict_digit() function and display the results.

```
from keras.models import load_model
from tkinter import *
import tkinter as tk
import win32gui
from PIL import ImageGrab, ImageOps
import numpy as np
model = load_model('handwritten_digits.h5')
```

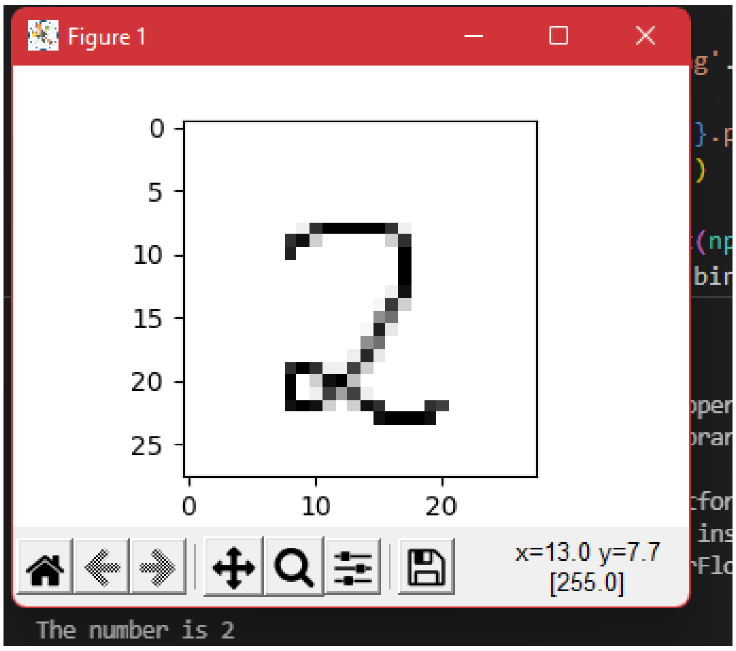
```
def predict_digit(img):
    #resize image to 28×28 pixels
    img = img.resize((28,28))
    #convert rgb to grayscale
    img = img.convert('L')
    #inverting the image
    img = ImageOps.invert(img)
    img = np.array(img)
    #reshaping to support our model input and normalizing
    img = img.reshape(1,28,28,1)
    img = img/255.0
    #predicting the class
    res = model.predict([img])[0]
    return np.argmax(res), max(res)

class App(tk.Tk):
    def __init__(self):
        tk.Tk.__init__(self)
        self.x = self.y = 0
        # Creating elements
        self.canvas = tk.Canvas(self, width=300, height=300, bg = "white",
cursor="cross")
        self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))
        self.classify_btn = tk.Button(self, text = "Recognise", command
=         self.classify_handwriting)
        self.button_clear = tk.Button(self, text = "Clear", command =
self.clear_all)
        # Grid structure
        self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
        self.label.grid(row=0, column=1,pady=2, padx=2)
        self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
        self.button_clear.grid(row=1, column=0, pady=2)
        #self.canvas.bind("<Motion>", self.start_pos)
        self.canvas.bind("<B1-Motion>", self.draw_lines)
    def clear_all(self):
        self.canvas.delete("all")
    def classify_handwriting(self):
        HWND = self.canvas.winfo_id() # get the handle of the canvas
        rect = win32gui.GetWindowRect(HWND) # get the coordinate of the canvas
        im = ImageGrab.grab(rect)
        digit, acc = predict_digit(im)
        self.label.configure(text= str(digit)+', '+ str(int(acc*100))+'%')
    def draw_lines(self, event):
        self.x = event.x
        self.y = event.y
        r=8
        self.canvas.create_oval(self.x-r, self.y-r, self.x + r, self.y + r,
fill='black')
```

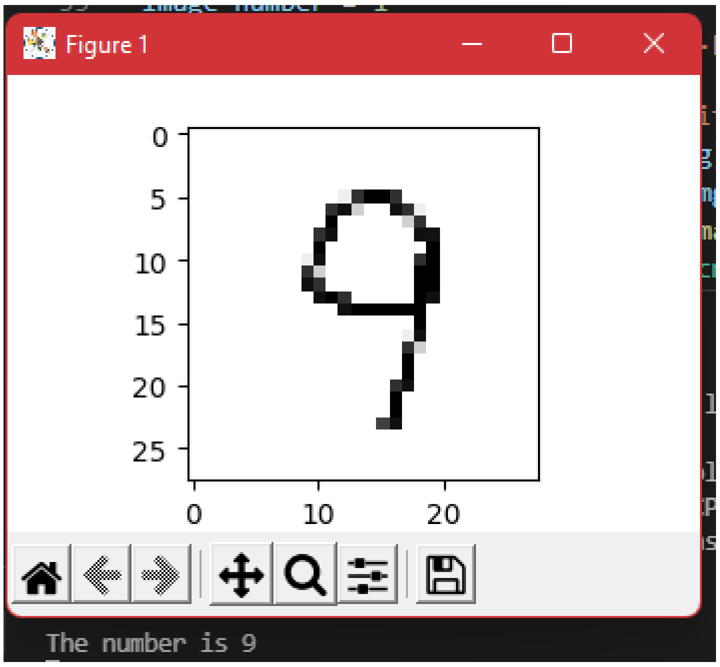
```
app = App()
mainloop()
```

EXPERIMENTAL RESULTS AND ANALYSIS

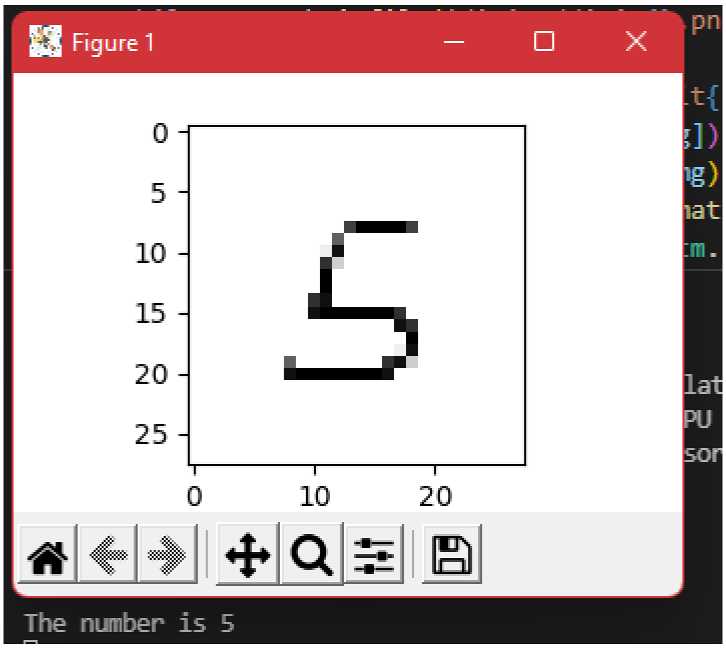
Predicitons for Custom Images :



```
The number is 2
```

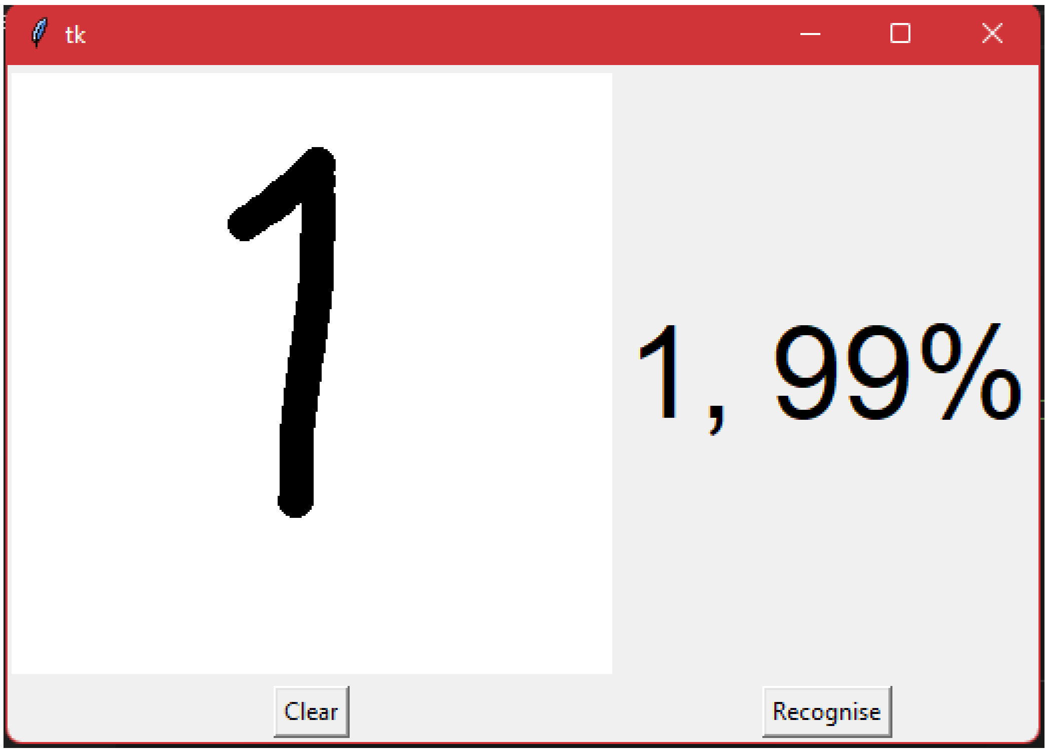


```
The number is 9
```

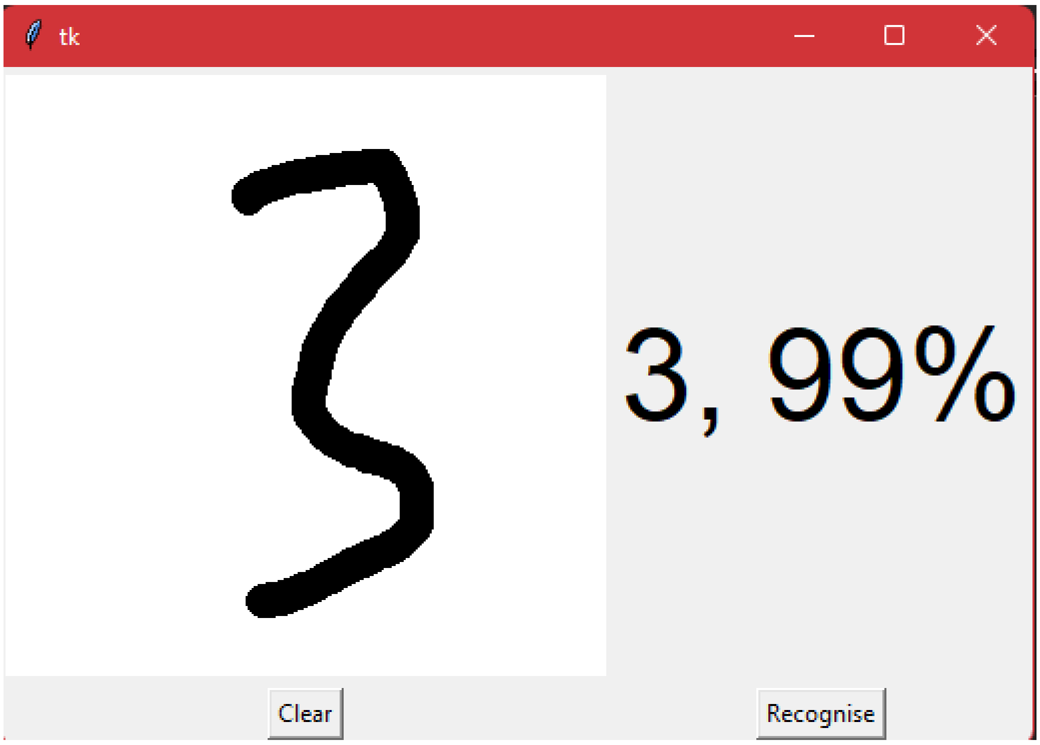


```
The number is 5
```

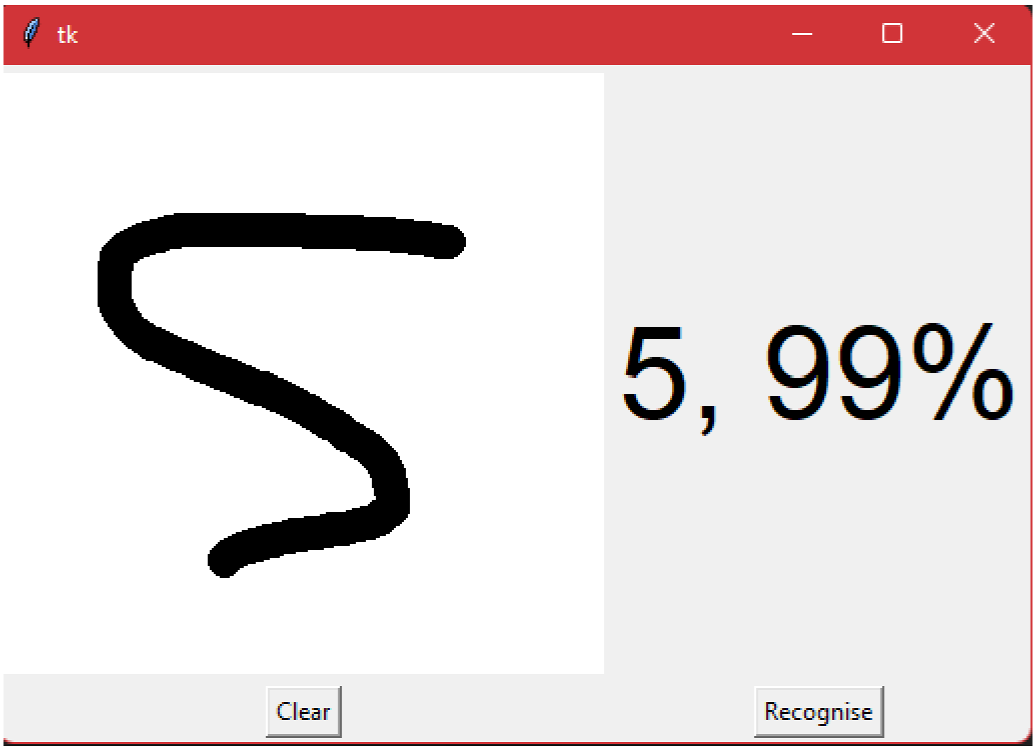
Predicitons for drawn Images :

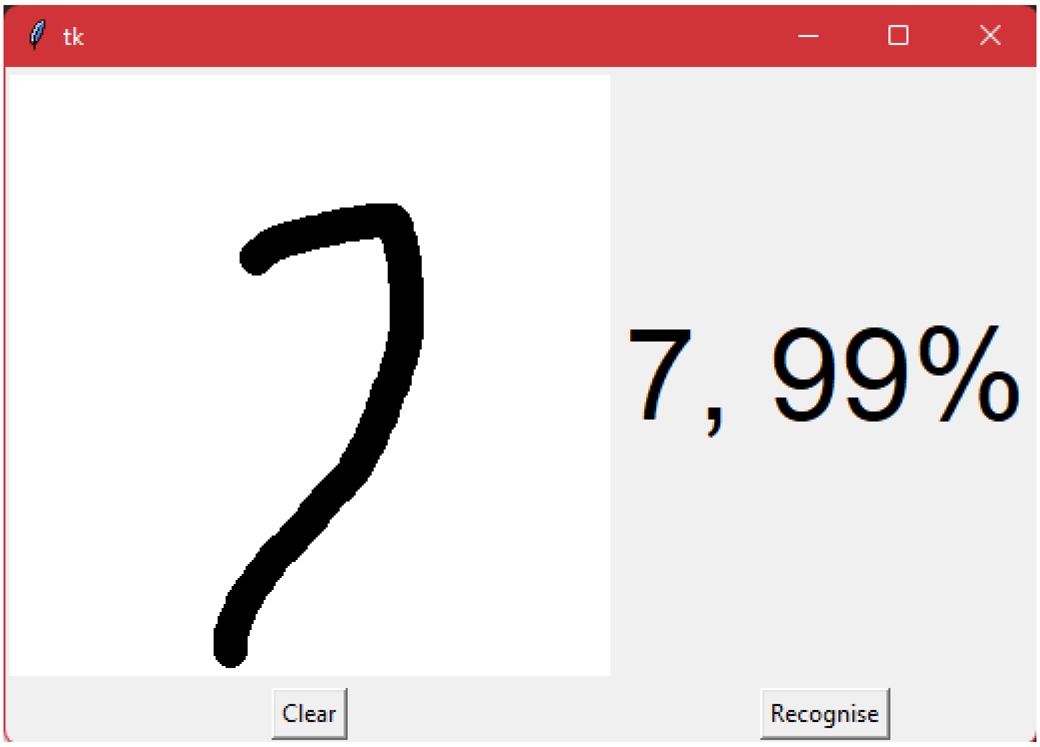


1, 99%



3, 99%





CONCLUSION AND FURTHER SCOPE

We show a model that can detect handwritten digits in this project. It can later be expanded to include character recognition and real-

time handwriting analysis. The identification of handwritten digits is the initial step in the large field of Artificial Intelligence and Computer Vision. As may be observed from the outcomes of the experiment, CNN outperforms other classifiers. With additional convolution layers and buried neurons, the findings may be made more accurate. It has the potential to fully eliminate the necessity for typing.
Digit recognition is an excellent model issue for learning about neural networks and provides a solid foundation for developing more complex deep learning approaches.

We intend to build a Text recognition system which can be used analyse and recognize digits as well as texts. We Also Intend to provide a better GUI with more customizations and using a webcam as an input to feed an image of a digit to our trained model in the future.

References

- Nimisha Jain, Kumar Rahul, Ipshita Khamaru. AnishKumar Jha, Anupam Ghosh (2017). “Hand Written Digit Recognition using Convolutional Neural Network (CNN)”, International Journal of Innovations & Advancement in Computer Science, IJIACS,ISSN 2347 – 8616,Volume 6, Issue 5.
- Dr.Kusumgupta2 ,"A Comprehensive Review On Handwritten Digit Recognition Using Various Neural Network Approaches", International Journal Of Enhanced Research In Management & Computer Applications, Vol. 5,No. 5, Pp. 22-25, 2016.
- Saeed AL-Mansoori, "Intelligent Handwritten Digit Recognition using Artificial Neural Network", Int. Journal of Engineering Research and Applications, vol. 5, no. 5, pp. 46-51, 2015.
- Nurul Ilmi, Tjokorda Agung Budi W and Kurniawan Nur R, “Handwriting Digit Recognation using Local Binary Pattern Varience and K-Nearest Neighbor,” 2016 Fourth International Conference on Information and Communication Technologies (ICoICT).
- Haider A. Alwzwazy1, Hayder M. Albehadili2, Younes S. Alwan3, Naz E. Islam4, "Handwritten Digit Recognition using Convolutional Neural

Networks", International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, no. 2, pp. 1101-1106, 2016.

- Tobias Kutzner, Mario Dietze, Ingrid Bönninger, Carlos M. Travieso, Malay Kishore Dutta, and Anushikha Singh, "Online Handwriting Verification with Safe Password and Incresing Number of Features,"2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN).
- Kaiming, He and Xiangyu, Zhang and Shaoqing, Ren and Jian Sun " Spatial pyramid pooling in deep convolutional networks for visual recognition‖ European", Conference on Computer Vision, arXiv:1406.4729v4 [cs.CV] 23 Apr 2015.
- Kussul, Ernst; Tatiana Baidyk (2004). "Improved method of handwritten digit recognition tested on MNIST database". Image and Vision Computing, 22(12): 971–981.
- "Handwritten Digit Recognition using various Neural Network Approaches", International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 2, pp. 78-80, 2015.
- Huimin Wu. CNN-Based Recognition of Handwritten Digits in MNIST Database.
- LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-Based Learning Applied to Document Recognition". Proceedings of the IEEE, 86(11): 2278–2324.