جامعة الأخوين

AL AKHAWAYN UNIVERSITY

Fall 2021

Intro. To Artificial

IntelligenceCSC 4301

Khaoula HASHAS

Houssam MOULOUE

Supervised by: Dr. Tajjeddine RACHIDI

Handwritten Digit Recognition using Python

Introduction:

Understanding human written digits is hard task for computers. The aim of this project is to create a handwritten digit recognition app that uses convolutional neural networks. Convolutional neural networks are basically a class of artificial neural network that specializes in analyzing visual imagery.
To train our model. We used the MNIST dataset found on the Keras library in python. This is a type of supervised learning since we used images of handwritten digits that are paired with their respective numeral name. To achieve our goal we trained the model with 60000 images then we tested it on 10000 other images to measure its efficiency with random handwritten numbers.
In general, we used python as our main programming language combined with NumPy, TensorFlow, Keras and Pillow libraries. Moreover, we used a simple GUI that contains a canvas to draw digits on to make our project more user friendly.
The Handwritten Digit Recognition in Python was developed using Python Deep Learning, This we are going to implement a handwritten digit recognition app using the MNIST dataset. We will be using a special type of deep neural network that is Convolutional Neural Networks. In the end, we are going to build a GUI in which you can draw the digit and recognize it straight away.

To accomplish this project, we must first employ appropriate technology (described below), and then import and install the relevant modules and libraries. Following that, we put our data into the application and separate it into training and test data. The training data is what will be used to train the model, while the test data should not be viewed until the test.

What is Handwritten Digit Recognition?

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.

Python Deep Learning Project:

To make machines more intelligent, the developers are diving into machine learning and deep learning techniques. A human learns to perform a task by practicing and repeating it repeatedly so that it memorizes how to perform the tasks.
The neurons in his brain then fire automatically, allowing them to do the task they have learnt rapidly. This is also quite similar to deep learning. It employs a variety of neural network topologies to solve various issues.

Programming language:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

TensorFlow is an amazing information stream in machine learning library made by the Brain Team of Google, made open source in 2015.It is intended to ease the use, and broadly relevant to both numeric and neural system issues just as different spaces. Fundamentally, TensorFlow is a lowlevel tool for doing entangled math and it targets specialists who recognize what they are doing to construct exploratory learning structures, to play around with them and to transform them into running programs. For the most, it can be considered as a programming framework in which one can entitle to calculations as graphs. Nodes in the graph speak the math activities, and the edges contain the multi-dimensional information clusters (tensors) related between them.

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

The MNIST dataset:

The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset. This is probably one of the most popular datasets among machine learning and deep learning enthusiasts. The MNIST dataset contains 60,000 training images of handwritten digits from zero to nine and 10,000 images for testing. Therefore, the MNIST dataset has 10 different classes. The handwritten digits images are represented as a 28×28 matrix where each cell contains grayscale pixel value.

The MNIST dataset has been the target of so many research done in recognizing handwritten digits.

This allowed the development and improvements of many different algorithms with a very high performance, such as machine learning classifiers.

In order to be able to implement our recognizer and test its performance, it is necessary to have a suitable dataset, which contains a large number of handwritten digits. This dataset should be able to allow us to discover the challenges and limitation of the image correlation technique and push us to look for ways and rules to enhance it and assess its accuracy. We have opted for

this dataset to be used for testing our program since it has proved a great reliability and importance in the field.

Implementation Steps:

1. Import the libraries and load the dataset
2. Preprocess the data
3. Create the model
4. Train the model
5. Evaluate the model
6. Create Graphical User Interface to predict digits

Dataset Format:

The dataset that I have downloaded from the MNIST database contains 60,000 images of handwritten digits, from zero to nine, all grouped in one file. Each of the images is of size 28 by 28 pixels and represents a digit. I have noticed that there is no pattern or order to the way the images were organized in the file. The images are represented as matrices, of which the elements represent the pixels. In addition, each image has a label that indicates the digit represented. This label was very helpful later on in order to be able to create the test set. Furthermore, the data did not contain noise or any major problems to deal with, that is why it was used without preprocessing it.

Example of the MNIST dataset

Building Python Deep Learning Project on Handwritten Digit Recognition:

The following are the steps to putting the handwritten digit recognition project into action:

1. Install the necessary libraries for this project using this command:

```
pip install numpy, tensorflow, keras, pillow,
```

2. Import the libraries and load the dataset

```
import keras

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D


from tensorflow import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D


from keras import backend as K

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(x_train.shape, y_train.shape)
```

First, we are going to import all the modules that we are going to need for training our model. The Keras library already contains some datasets and MNIST is one of them. Therefore, we can easily import the dataset and start working with it. The **mnist.load_data()** method returns us the training data, its labels and also the testing data and its labels.

3. Preprocess the data

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

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).

4. Create the model

```
batch_size = 128
num_classes = 10
epochs = 10

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=
['accuracy'])
```

Now we will **create our CNN model** in Python data science project. 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 offer fitting of the model.

5. Create GUI (Graphical User Interface) to predict digits:

Here is the full code for our gui_digit_recognizer.py file:

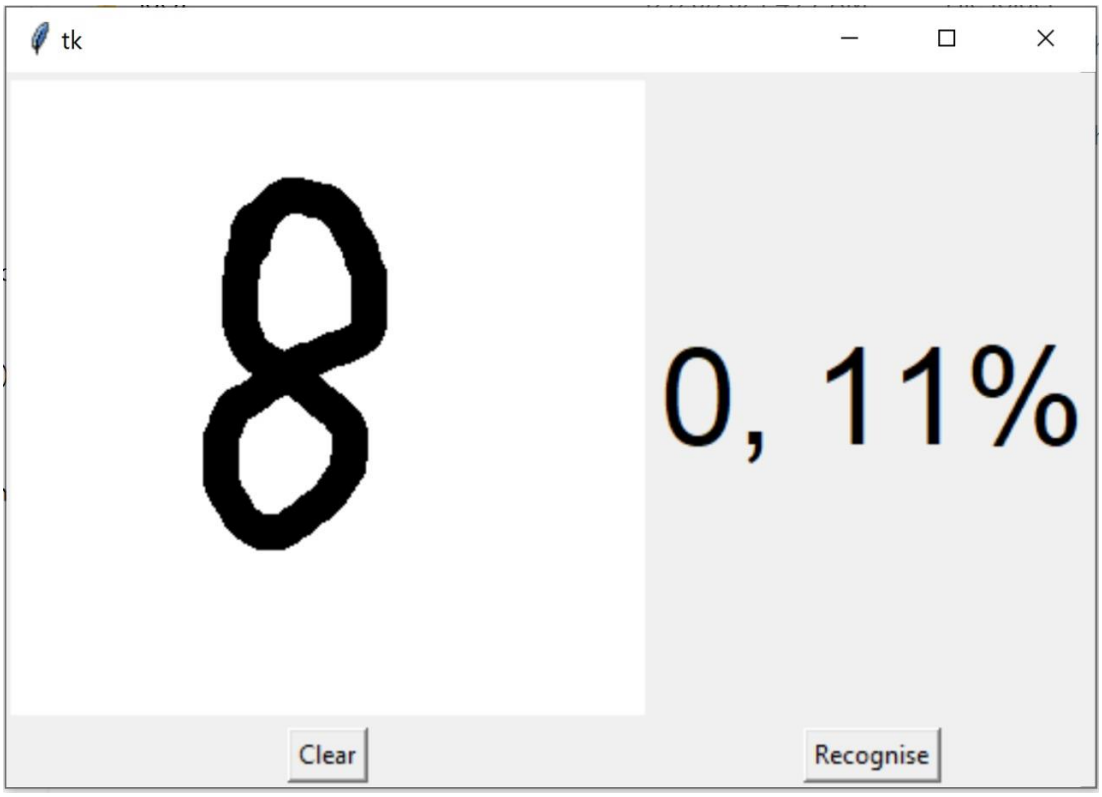
```
from keras.models import load_model
from tkinter import *
import tkinter as tk
import win32gui
from PIL import ImageGrab, Image
import numpy as np
model = load_model('mnist.h5')
def predict_digit(img):
    #resize image to 28x28 pixels
    img = img.resize((28,28))
    #convert rgb to grayscale
    img = img.convert('L')
    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()
```

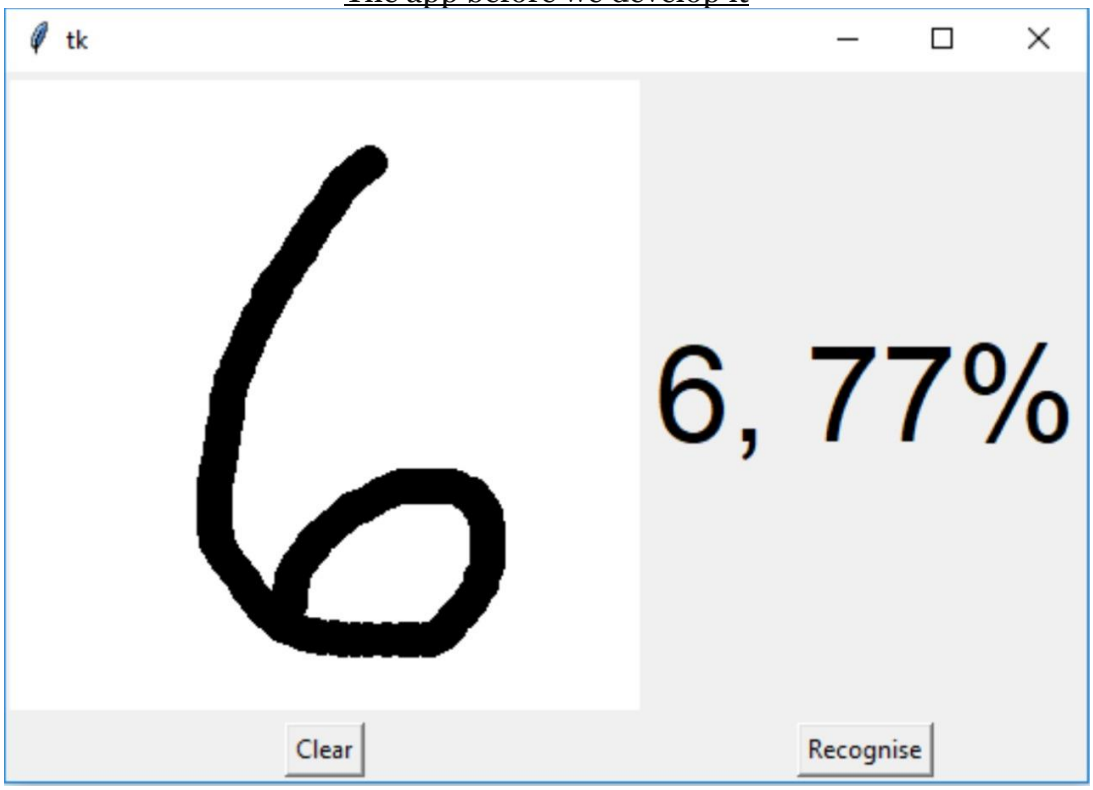
Now for the GUI, we have created a new file in which we build an interactive window to draw digits on canvas and with a button, we can recognize the digit. The Tkinter library comes in the Python standard library. We have created a function **predict_digit ()** that takes the image as input and then uses the trained model to predict the digit.

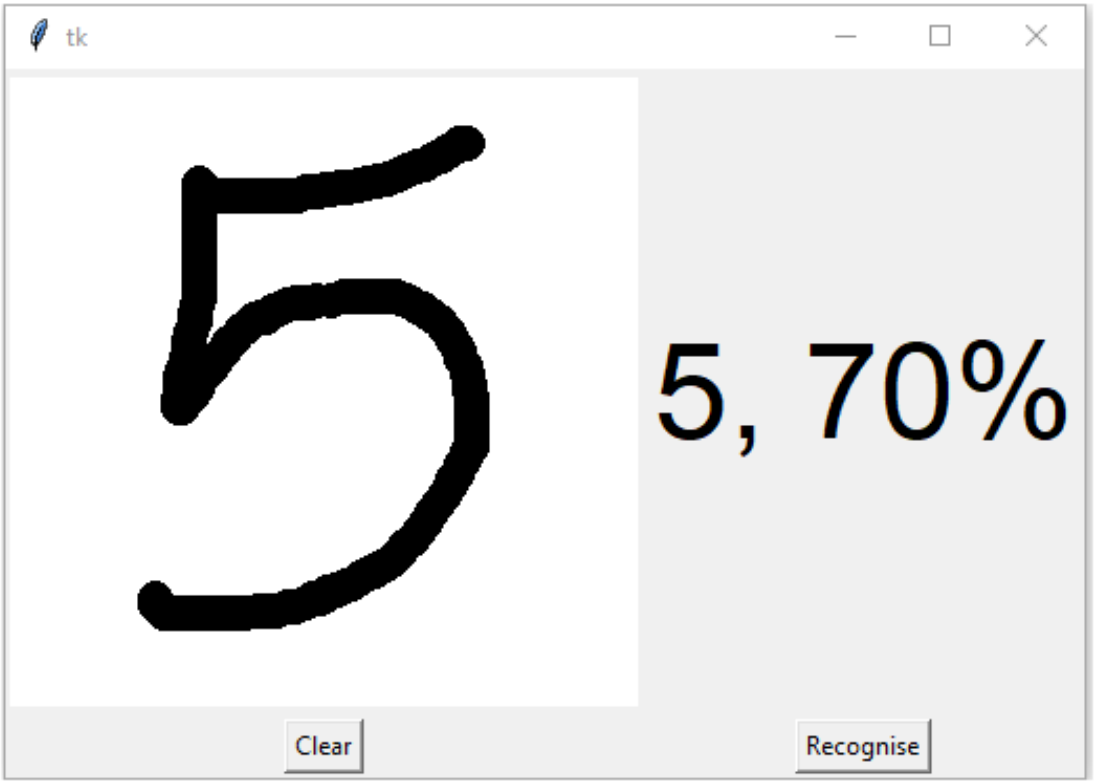
Then we create the App class, which is responsible for building the GUI for our app. We create a canvas where we can draw by capturing the mouse event and with a button, we trigger the predict_digit () function and display the results.

6. Screenshots:



The app before we develop it





The app after we develop it

We successfully developed a Python deep learning project based on a handwritten digit identification program. We created and trained a Convolutional neural network that is quite good for picture classification. Later, we develop the GUI, in which we put a digit on the canvas, then classify it and display the findings.

References:

https://ijisrt.com/wp-content/uploads/2019/06/IJISRT19JU358.pdf

https://projectworlds.in/artificial-intelligence-project-handwritten-digits-recognition/

http://athena.ecs.csus.edu/~shiroors/Project_Presentation.pdf

https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html