

PROJECT REPORT
(Project Term January-May 2022)

Handwritten Digit Recognition

Submitted by

Ravinshu Makkar

Registration Number: 11908152

Course Code: INT 247

Under the Guidance of

Dr. Sagar Pande

School of Computer Science and Engineering



L OVELY
P ROFESSIONAL
U NIVERSITY

DECLARATION

I hereby declare that the project work entitled “Handwritten Digit Recognition” is an authentic record of our own work carried out as requirements of Project for the award of B. Tech degree in Computer Science & Engineering from Lovely Professional University, Phagwara, under the guidance of Dr. Sagar Pande, during January to May 2022. All the information furnished in this project report is based on our own intensive work and is genuine.

Project Group Number: 2295255

Name of Student 1: Ravinshu Makkar

Registration Number: 11908152

A handwritten signature in blue ink that reads "Ravinshu". The signature is written in a cursive style with a horizontal line underneath the name.

Date: 26/3/2022

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Project under my guidance and supervision. The present work is the result of their original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. The Project is fit for the submission and partial fulfillment of the conditions for the award of B. Tech degree in Computer Science & Engineering from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Designation

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date:

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher Dr. Sagar Pande who gave us the golden opportunity to do this wonderful project on the topic Handwritten Digit Recognition, which also helped me in doing a lot of research and I come to know about so many things. I am thankful to them.

It helped me increase my knowledge and skills. Thanks again to all who supported.

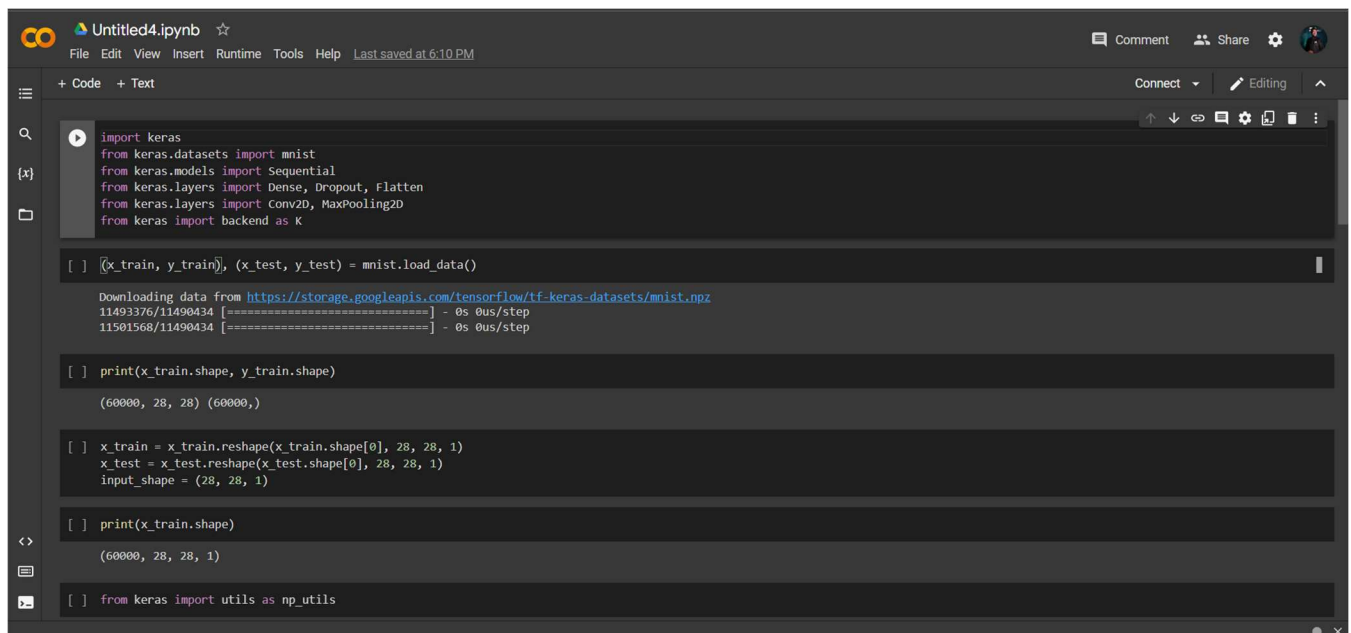
TABLE OF CONTENTS

Title Page.....	(i)
Declaration.....	(ii)
Certificate.....	(iii)
Acknowledgement.....	(iv)
Table of Contents.....	(v)

Overview of project	6
Introduction	10
Goal of Project	12
Feature Extraction	13
Image Correction	14
OCR	15
Tools and setup	16
Environment	18

Overview of the project:

The main goal of this project is to be able to interpret photos containing handwritten numbers and use basic image correlation algorithms to identify those digits. Normally, these images are represented and read as matrices, with each element representing a pixel. The image correlation technique analyses these matrices utilizing algorithms in order to get the match that reflects the digit we're seeking to figure out. To do this I have used MNIST 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 keras import backend as K

(x_train, y_train), (x_test, y_test) = mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [=====] - 0s 0us/step
11501568/11490434 [=====] - 0s 0us/step

print(x_train.shape, y_train.shape)

(60000, 28, 28) (60000,)

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)

print(x_train.shape)

(60000, 28, 28, 1)

from keras import utils as np_utils
```

1. INTRODUCTION

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. The aim of this project is to apply Machine Learning methodologies in order to improve the performance of a trained model based on solving the purpose of recognising digits from handwriting.

In this project, we used the dataset from “MNIST,” trained it with the same dataset and tested it with test data and train data to get a better result.

The moved to an image to work with it and changed the dimension of the image to 28/28 to match the “MNIST” dataset and then analyse it with the rest. It predicts with information gained from all the datasets which have over 60,000 images and give the most accurate output to determine which number is visible on the screen.

It is easy for the human brain to process images and analyse them. When the eye sees a certain image, the brain can easily segment it and recognize its different elements. The brain automatically goes through that process, which involves not only the analysis of this images, but also the comparison of their different characteristics with what it already knows in order to be able to recognize these elements. (“School of Science Engineering HANDWRITTEN DIGITS ...”) There is a field in computer science that tries to do the same thing for machines, which is Image Processing.

Image processing is the field that concerns analysing images to extract some useful information from them. This method takes images and converts them into a digital form readable by computers, it applies certain algorithms on them, and results in a better-quality image or with some of their characteristics that could be used in order to extract some important information from them. (“School of Science Engineering HANDWRITTEN DIGITS ...”)

Image processing is applied in several areas, especially nowadays, and various software has been developed that use this concept. Now we have self-driven cars which can detect other cars and human beings to avoid accidents. Also, some social media applications, like Facebook, google photos that can do facial recognition, thanks to this

technique. Furthermore, some software uses it in order to recognise the characters in some images, which is the concept of optical character recognition (OCR), which we will be discussing and discovering in this project. ("School of Science Engineering HANDWRITTEN DIGITS ...")

One of the narrow fields of image processing is recognizing characters from an image, which is referred to as Optical Character Recognition (OCR). This method is about reading an image containing one or more characters or reading a scanned text of typed or handwritten characters and be able to recognize them. ("School of Science Engineering HANDWRITTEN DIGITS ...") A lot of research has been done in this field in order to find optimal techniques with a high accuracy and correctness. "The most used algorithms that proved a very high performance are machine learning algorithms like Neural Networks and Support Vector Machine." ("School of Science Engineering HANDWRITTEN DIGITS ...")



(OCR)

"One of the main applications of OCR is recognizing handwritten characters." ("School of Science Engineering HANDWRITTEN DIGITS ...") In this project, we will focus on building a mechanism that will recognize handwritten digits. We will be reading images containing handwritten digits extracted from the MNIST database and try to recognize which digit is represented by that image. For that we will use basic Image Correlation techniques, also referred to as Matrix Matching. This approach is based on matrices manipulations, as it reads the images as matrices in which each element is a pixel. ("School of Science Engineering HANDWRITTEN DIGITS ...") It overlaps the image with all the images in the reference set and find the correlation between them in order to be able to determine the digit it represents.

The goal of this project:

"Apply and manipulate the basic image correlation techniques to build program and keep polishing and enhancing in order to investigate to which extent it can get improved." ("School of Science Engineering HANDWRITTEN DIGITS ...") This would allow us to see how far we can go, in terms of accuracy and performance, but using just the very simple and basic techniques of matrix matching and without going into complicated methods of machine learning.

Artificial Neural Network (ANN)

An Artificial Neural Network (ANN) is a system that imitate the biological neural network in the brain of a human being. It's a machine learning algorithm, which means it uses data to learn how to respond to various inputs. The ANN can be thought of as a box that accepts one or more inputs and produces a single output. There are multiple connecting connections inside the box. The data is entered into the software, which then passes through the ANN's layers and nodes to produce an output using a transfer function.

For OCR, Artificial Neural Networks are employed, and they have shown to have a very high accuracy rate. The ANN would "recognize a character based on its topological properties, such as shape, symmetry, closed or open areas, and number of pixels" in this scenario. ("School of Science Engineering HANDWRITTEN DIGITS ...") This type of algorithm's excellent accuracy is due to its capacity to learn from a training set of characters with comparable characteristics.

Support Vector Machine:

SVM stands for Support Vector Machine and is a machine learning algorithm. Pattern classifiers with good performance are known as SVMs. SVMs strive to minimize the "upper bound of the generalization error" whereas Neural Networks aim to minimize the training error. This technique's learning method is based on classification and regression analysis. This type of classifier has shown to be particularly effective in the recognition of very complicated characters, such as those found in the Khmer language.

Feature Extraction

Feature extraction is a pattern recognition approach. The basic concept behind feature extraction is to analyse photos and generate some characteristics from them that may be used to identify each individual element. Curvatures, holes, and edges, for example, are examples of these properties. The holes inside the digits (for example, the eight, the six, and maybe the two) as well as the angles between some straight lines (for example, the one, the four, and the seven) are examples of these features in digit recognition. When an unknown image needs to be recognized, its characteristics are compared to these in order to classify it.

Image Correction

The technique of image correlation is used to recognize characters in images. In order to analyse the photos, this method, also known as Matrix Matching, employs mathematical algorithms. The images are read as matrices with this methodology, with each element representing a pixel, making it easier to modify them using mathematical methods. The image that needs to be identified is loaded as a matrix and compared to the reference set's images. The test image is superimposed on each image in the reference set to evaluate how it compares to each one and determine which one best depicts it. The judgement can be determined by looking at the pixels that match and the ones that are missing from either image.

More about OCR:

It is easy for the naked eye to recognize a character when spotted in any document; however, computers cannot identify the characters from an image or scanned document. In order to make this possible, a lot of research has been done, which resulted in the development of several algorithms that made this possible. One of the fields that specialize in character recognition under the light of Image Processing is Optical Character Recognition (OCR). In Optical Character Recognition, a scanned document or an image is read and segmented in order to be able to decipher the characters it contains. The images are taken and are preprocessed to get rid of the noise and have unified colors and shades, then the characters are segmented and recognized one by one, to finally end up with a file containing encoded text containing these characters, which can be easily read by computers. (“Handwritten Digit Recognition for Banking System - IJERT”) Optical Character Recognition dates to the early 1900s, as it was developed in the United States in some reading aids for the blind. In 1914, Emanuel Goldberg was able to implement a machine able to convert characters into “standard telegraph code”. (“School of Science Engineering HANDWRITTEN DIGITS ...”) In the 1950s, David Shepard, who was at that time an engineer at the Department of Defense, developed a machine that he named Gismo, which can read characters and translate them into machine language. In 1974, Ray Kurzweil decided to develop a machine that would read text for blind and visually impaired people under his company, Kurzweil 4 Computer Products. There are several software and programs, nowadays, which use OCR in several different applications. In 1996, the United States Postal Services were able to develop a mechanism, HWAI, which recognizes handwritten mail addresses. (“School of Science Engineering HANDWRITTEN DIGITS ...”)

1.1.1 Tools and Setup

The main goal of this project is to be able to interpret photos containing handwritten numbers and use basic image correlation algorithms to identify those digits. Normally, these images are represented and read as matrices, with each element representing a pixel. The image correlation technique analyses these matrices utilizing algorithms in order to get the match that reflects the digit we're seeking to figure out. Because this research will

mostly include matrices and large numerical computations, it is critical to evaluate the tools that will provide us with a proper environment in which to do these calculations.

1) Octave:

Octave is a free and open-source software that uses a high-level programming language. It is compatible with MATLAB and has the same functionality. It provides a very easy and appropriate interface for performing some mathematical calculations. It includes certain tools for solving math issues, such as some popular linear algebra problems.

When it comes to these operations, it is also quite efficient in terms of resource usage, such as time and memory. It is also quite simple to use when dealing with matrices, as it includes many functions and operations that make manipulating them less expensive. We will be dealing with photos as matrices in this project, with each element representing a pixel, so selecting a tool that will make our computations easier and more efficient in terms of time and memory resources is critical. MATLAB and Octave are both simple to learn and use, and they provide an ideal environment for this type of research.

2) MNIST Database

The MNIST database (Modified National Institute of Standards and Technology database) is a massive dataset with hundreds of thousands of handwritten numbers. Because the NIST set was divided into those written by high school students and those written by Census Bureau workers, this dataset was created by combining different sets within the original National Institute of Standards and Technology (NIST) sets to create a training set containing a variety of types and shapes of handwritten digits.

It is required to have a good dataset with a big number of handwritten digits in order to develop our recognizer and test its performance. This dataset should enable us to identify the obstacles and limitations of the image correlation technique, as well as encourage us to seek out ways and rules to improve it and measure its correctness. We chose this dataset to test our program since it has proven to be extremely reliable and important in the field.

3) Feasibility Study:

From a technical standpoint, because this project involves a lot of numerical computations, utilizing Octave is a good idea because it will make the application run faster. This software will also offer us with libraries for reading and manipulating photos, making the implementation process much easier.

The MNIST Database was chosen as the dataset to use in the project's testing. Thousands of handwritten numbers have been used in the creation of programs with a similar goal in this database. This dataset is free to use by the general public. It is also incredibly useful for our project, since it will save us time by allowing us to use it as a test set without having to make one ourselves.

Setup:

To accomplish this task, I used many predefined libraries to work with and with the help of jupyter notebook I created a training dataset file along with recognizer file with the extension “ipynb”.

Things to import:

- TensorFlow
- Keras
- OpenCV-python
- matplotlib

and more such libraries for the help of creating a better classifier and working with the model and recognizing the perfect desired number.

With the help of keras I loaded the model in my project to work with it and it helped to predict the digit. Right now, this works with a single digit but there is the future scope of getting more digits in a single file or image. CV binary can be used to see this image in black & white in front of us to better recognize it. This can be used to work on making beautiful things such as handwriting recognition and could help to get things from paper to machine and handle it in a more reliable form.

1.1.1. Environment

Creating an environment was important as without it this wouldn't work properly. Now to do so I installed python along with many pip imports to make this work smoothly and as fast as possible. After the perfect environment is ready the thing was to code the image as needed in binary, so the computer read it properly to predict the number and provide me with the output. To make the shape of proper size in the best NumPy (in my case 28,28).

Now after we have got the best Environment with everything ready it's time to make the image perfect ARRAY and to use it in our Machine learning project. Then we can continue to work upon it by changing it to black and white or to a different size. In this OpenCV helped and we got the image to work with then we moved on to prediction using the model created before.

Creating the reference and test set

The creation of the reference and test sets, which will both be employed in the implementation phase, is one of the most important tasks in the project. The test set will be used to assess the program's performance and determine its success or mistake rate. It will come from the MNIST dataset, which contains the handwritten digits that we want to recognize and identify. The reference set is used to compare the test images and determine which digit each one represents. It will be made with a variety of typefaces.

Data Sets:

Before we begin the design and execution of the software, we must first determine the type of data we will be using. That is why, before generating the reference and test sets, we needed to look at its format to understand how it is represented.

Data Set Format:

The MNIST dataset I downloaded contains 60,000 photographs of handwritten numbers ranging from zero to nine, all clustered together in one file. Each graphic is 28 by 28 pixels in size and represents a digit. I've noticed that the photographs in the file aren't grouped in any pattern or sequence. The images are represented as matrices, with the pixels represented by the elements. Each image also has a label that identifies the digit it represents. This label came in handy later when it came time to create the test set.

Furthermore, the data was used without preprocessing because there was no noise or serious issues to deal with.



Figure 6.1.1. Example of the MNIST dataset [15]

Reference Set:

To be able to recognize the digit represented by a given image, it must be compared to other images containing recognized digits. To do so, you'll need to construct a reference set that includes all these photographs.

That is to say, each image we would want to recognize is to be compared to the images in the reference set. The image that best portrays the correct number is the one with the highest match. Because handwritten digits vary from person to person, the reference collection must include digits in a variety of fonts. As a result, we used the online image editor pixlr.com to make six photos of each digit, each with a different font. Images in the reference set are the same size as those in the MNIST dataset, i.e., 28 by 28 pixels.



Test Set:

To analyse its performance and quantify its success rate, the program to be developed must be tested against some images containing handwritten digits. As a result, it's critical to construct a test set. The test set is an example of an image comprising handwritten digits that must be compared to the photos in the reference set in order to be identified.

The file from the MNIST database was used to create this set. There were 60,000 photos in the original file, each representing a distinct digit. This made it impossible to find each number using the label during the program's testing. We opted to keep several photos from each digit in a different file to make it easier to access each digit we desire. As a result, we have 20 photographs of each digit stored in 10 distinct files. That is, the final test set consisted of ten files, each of which represented a number and contained 20 photographs of that digit. Octave was used to extract these photos from the original file by reading them and their labels.

We had to make some changes to the elements of all the matrices representing the test set in order to make manipulating the matrices/images easier. The black pixels were left alone because they were initially represented as zeros. The white ones, on the other hand, each had a separate non-zero number, thus we all became ones.

Implementation:

The goal of this project is to create a system that can recognize handwritten digits from the MNIST dataset. Image Correlation, commonly known as Matrix Matching, is the method we've chosen. The purpose of this research is to see how good we can make the accuracy rate without employing advanced techniques like machine learning by using the basic and simple notions of these methods.

Importing the necessary files:

```
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 keras import backend as K
```

Downloading the necessary dataset and dividing it with train and test data:

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Output for the downloading of the dataset:

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz  
11493376/11490434 [=====] - 0s 0us/step  
11501568/11490434 [=====] - 0s 0us/step
```

To check the shape of dataset for training and testing:

```
print(x_train.shape, y_train.shape)  
  
(60000, 28, 28) (60000,)
```

Changing the shape of the dataset we got.

```
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)
```

Verify the new shape:

```
print(x_train.shape)  
  
(60000, 28, 28, 1)
```

Processing the data under keras module:

```
y_train = keras.utils.np_utils.to_categorical(y_train, 10)  
y_test = keras.utils.np_utils.to_categorical(y_test, 10)
```

Dividing the data for test and training use:

```
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')

x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

Initialising required variables:

```
batch_size = 1280
num_classes = 10
epochs = 10
```

Passing the required activation function for the building of model:

```
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'))
```

Using tensorflow module to building the whole model:

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

hist = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(x_test, y_test))
print("The model has successfully trained")
```

```
Epoch 1/10
47/47 [=====] - 38s 806ms/step - loss: 2.3156 - accuracy: 0.0984 - val_loss: 2.3086 - val_accuracy: 0.1436
Epoch 2/10
47/47 [=====] - 38s 803ms/step - loss: 2.3133 - accuracy: 0.1005 - val_loss: 2.3062 - val_accuracy: 0.1536
Epoch 3/10
47/47 [=====] - 38s 802ms/step - loss: 2.3114 - accuracy: 0.1041 - val_loss: 2.3038 - val_accuracy: 0.1596
Epoch 4/10
47/47 [=====] - 38s 801ms/step - loss: 2.3087 - accuracy: 0.1072 - val_loss: 2.3014 - val_accuracy: 0.1662
Epoch 5/10
47/47 [=====] - 38s 801ms/step - loss: 2.3059 - accuracy: 0.1113 - val_loss: 2.2990 - val_accuracy: 0.1712
Epoch 6/10
47/47 [=====] - 38s 805ms/step - loss: 2.3040 - accuracy: 0.1129 - val_loss: 2.2966 - val_accuracy: 0.1753
Epoch 7/10
47/47 [=====] - 38s 805ms/step - loss: 2.3016 - accuracy: 0.1181 - val_loss: 2.2941 - val_accuracy: 0.1809
Epoch 8/10
47/47 [=====] - 38s 804ms/step - loss: 2.2992 - accuracy: 0.1204 - val_loss: 2.2917 - val_accuracy: 0.1843
Epoch 9/10
47/47 [=====] - 38s 803ms/step - loss: 2.2962 - accuracy: 0.1273 - val_loss: 2.2893 - val_accuracy: 0.1867
Epoch 10/10
47/47 [=====] - 38s 803ms/step - loss: 2.2947 - accuracy: 0.1259 - val_loss: 2.2869 - val_accuracy: 0.1909
The model has successfully trained
```

Checking for loss and accuracy for this project:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 2.2869138717651367
Test accuracy: 0.19089999794960022
```

```
model.save('mnist.h5')
print("Saving the model as mnist.h5")
```

```
Saving the model as mnist.h5
```

After the model is created, we move to new file to recognise the digit



Below image shows the test file.

A screenshot of a Jupyter Notebook interface within Visual Studio Code. The notebook is titled 'Untitled.ipynb - Handwritten digit recognizer'. The left sidebar shows a file explorer with a folder named 'HANDWRITTEN DIGIT RECOGNIZER' containing files like 'final.h5', 'gui_digit_recognizer.py', 'mnist_model.h5', 'mnist.h5', 'train_digit_recognizer.py', 'try.png', and 'Untitled.ipynb'. The main editor area shows the following code cells:

```
[13]: from keras.models import load_model
      from PIL import imageGrab, Image
      import numpy as np
      import tensorflow as tf
      import cv2
      import matplotlib as plt

[14]: img = cv2.imread('try.png')

[15]: img.shape
      ... (164, 176, 3)

[16]: changedImg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

[17]: changedImg.shape
      ... (164, 176)

[18]: changedImg = cv2.resize(changedImg, (28,28), interpolation = cv2.INTER_AREA)
```

The bottom status bar indicates 'Jupyter Server: local' and the system clock shows '10:14 PM 3/30/2022'.

This file contains model predictions code and further required code to change image to desired shape and all.

1. Import the required modules and packages.

```
from keras.models import load_model
from PIL import ImageGrab, Image
import numpy as np
import tensorflow as tf
import cv2
import matplotlib as plt
```

2. Getting image ready for working on it.

```
img = cv2.imread('try.png')
```

For this I am using Open-Cv python package.

3. Checking the original shape of image

```
img.shape

(164, 176, 3)
```

4. Changing the image to black and white to work on it.

```
changedImg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

5. Changing the image shape to required dataset

```
changedImg = cv2.resize(changedImg,(28,28),interpolation = cv2.INTER_AREA)
```

- Using TensorFlow to normalize the image for further working on it.

```
changedImg = tf.keras.utils.normalize (changedImg,axis = 1)
```

- Changing the shape acc to dataset of model.

```
changedImg = np.array (changedImg).reshape (-1,28,28,1)
```

- Using model to predict the binary form of image and testing it against the many images and the finding the perfect match.

```
model = load_model('mnist.h5')
predictions = model.predict (changedImg)
```

[31] Python

... WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x00000188E39F2D00> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

- Printing the output.

```
print(np.argmax(predictions))
```

4

As that handwritten text was 4 and we got the output as 4.

Challenges and Limitations:

This was my first project involving Optical Character Recognition (OCR). As a result, I encountered numerous obstacles and issues while working on this project. To begin with, it took me a long time to grasp and become comfortable with all of the ideas, from image processing to OCR, as well as all of the techniques and algorithms utilized in it.

Furthermore, the data we were working with was extremely difficult to work with due to the way the digits were written. Some of the numbers were thicker or thinner than the others because they were rotated at an angle, and some of the digits were not well centered or written in a confusing manner.

Related Works

- <https://sisu.ut.ee/imageprocessing/book/1>.
- http://www.odl.ox.ac.uk/papers/OCRFeasibility_final.pdf
- H. F. Schantz, The history of OCR, optical character recognition. Manchester Centre, VT: Recognition Technologies Users Association, 1982.
- History of Computers and Computing, Internet, Dreamers, Emanuel Goldberg. [Online] Available:
<https://historycomputer.com/Internet/Dreamers/Goldberg.html>.