Project Report (Project Term Aug-Dec 2022)

On

Digit Recogniser

Submited by : Yaswant Aditya & Saurav

Regno. : 11902268 & 11915238

Section : KM033

Course code : INT248

https://github.com/saurav8931/digit-recognition-python

Submitted to

Niharika Thakur

School of computer science and engineering

DECLARATION

We hereby declare that the project work entitled **Digit Recognizer** is an authentic record of our own work carried out as requirements of Project for the award of **B.Tech degree in CSE** from Lovely Professional University, Phagwara, under the guidance of **Niharika Thakur**, during August to December 2022. All the information furnished in this project report is based on our own intensive work and is genuine.

Name of Student 1 : Yaswant aditya **Signature of student1 :** Yaswant

Registration Number: 11902268

Name of Student 2 : Saurav **Signature of student2 :** Saurav

Registration Number : 11915238

ABSTRACT

Handwritten character recognition is one of the practically important issues in pattern recognition applications. The reliance of humans over machines has never been so high such that from object classification in photographs to adding sound to silent movies everything can be performed with the help of deep learning and machine learning algorithms. Likewise, Handwritten text recognition is one of the significant areas of research and development with a streaming number of possibilities that could be attained.

The main purpose of this project is to build an automatic handwritten digit recognition method for the recognition of handwritten digit strings. To accomplish the recognition task, first, the digits will be segmented into individual digits. Then, a digit recognition module is employed to classify each segmented digit completing the handwritten digit string recognition task. The applications of digit recognition include postal mail sorting, bank check processing, form data entry, etc. The heart of the problem lies within the ability to develop an efficient algorithm that can recognize handwritten digits and which is submitted by users by the way of a scanner, tablet, and other digital devices.

1. INTRODUCTION

Machine learning and deep learning plays an important role in computer technology and artificial intelligence. With the use of deep learning and machine learning, human effort can be reduced in learning, predictions and many more areas. This project presents recognizing the handwritten digits (0 to 9) from the famous MNIST dataset.

Handwritten digit recognition has gained so much popularity from the aspiring beginner of machine learning and deep learning to an expert who has been practicing for years. Developing such a system includes a machine to understand and classify the images of handwritten digits as 10 digits (0–9). Handwritten digit recognition with models trained on the MNIST data set is a popular “Hello World” project for deep learning as it is simple to build a network that achieves over 90% accuracy for it. There are also many existing open source implementations of MNIST models on the Internet, making it a well-documented starting point for machine learning beginners.

The goal of this project is to create a model that will be able to recognize and determine the handwritten digits from its image by using the concepts of Convolution Neural Network. Though the goal is to create a model which can recognize the digits, it can be extended to letters and an individual’s handwriting. The major goal of the proposed system is understanding Convolutional Neural Network and applying it to the handwritten recognition system.

2. Methodology

2.1 Basic steps in constructing a Machine Learning model :

2.1.1 - Data Collection

• The quantity & quality of your data dictate how accurate our model is

• The outcome of this step is generally a representation of data (Guo simplifies to specifying a table) which we will use for training

• Using pre-collected data, by way of datasets from Kaggle, UCI, etc., still fits into this step

2.1.2 - Data Preparation

• Wrangle data and prepare it for training

• Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.)

• Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data

• Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis

• Split into training and evaluation sets

2.1.3 - Choose a Model

• Different algorithms are for different tasks; choose the right one

2.1.4 - Train the Model

• The goal of training is to answer a question or make a prediction correctly as often as possible

• Linear regression example: algorithm would need to learn values for m (or W) and b (x is input, y is output)

• Each iteration of process is a training step

2.1.5 - Evaluate the Model

• Uses some metric or combination of metrics to "measure" objective performance of model • Test the model against previously unseen data 17

• This unseen data is meant to be somewhat representative of model performance in the real world, but still helps tune the model (as opposed to test data, which does not)

• Good train/eval split? 80/20, 70/30, or similar, depending on domain, data availability, dataset particulars, etc.

2.1.6 - Parameter Tuning

• This step refers to hyperparameter tuning, which is an "artform" as opposed to a science

• Tune model parameters for improved performance

• Simple model hyperparameters may include: number of training steps, learning rate, initialization values and distribution, etc.

2.1.7 - Make Predictions

• Using further (test set) data which have, until this point, been withheld from the model (and for which class labels are known), are used to test the model

2.2 Methodologies for Handwritten Digit Recognition System

We used MNIST as a primary dataset to train the model, and it consists of 70,000 handwritten raster images from 250 different sources out of which 60,000 are used for training, and the rest are used for training validation. Our proposed method mainly separated into stages, pre-processing, Model Construction, Training & Validation, Model Evaluation & Prediction. Since the loading dataset is necessary for any process, all the steps come after it.

2.2.1 Import the libraries:

Libraries required are Keras, Tensor flow, Numpy, Matplotlib and pygame

1. Keras : Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models. 18 It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks. Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

2. TensorFlow: TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow tutorial is designed for both beginners and professionals. Our tutorial provides all the basic and advanced concept of machine learning and deep learning concept such as deep neural network, image processing and sentiment analysis. TensorFlow is one of the famous deep learning frameworks, developed by Google Team. It is a free and open source software library and designed in Python programming language, this tutorial is designed in such a way that we can easily implements deep learning project on TensorFlow in an easy and efficient way. Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems. It can run on single CPU systems, GPUs as well as mobile devices and largescale distributed systems of hundreds of machines.

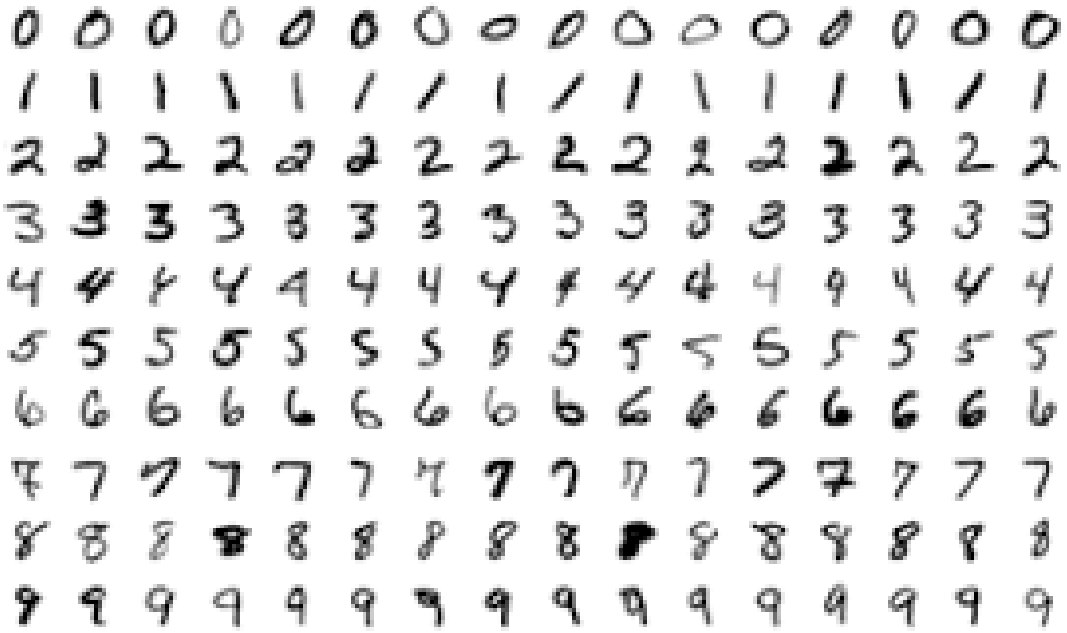
3. Numpy: NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. Numpy which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. It is an open source project and you can use it freely. NumPy stands for Numerical Python. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

4 Matplotlib : Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

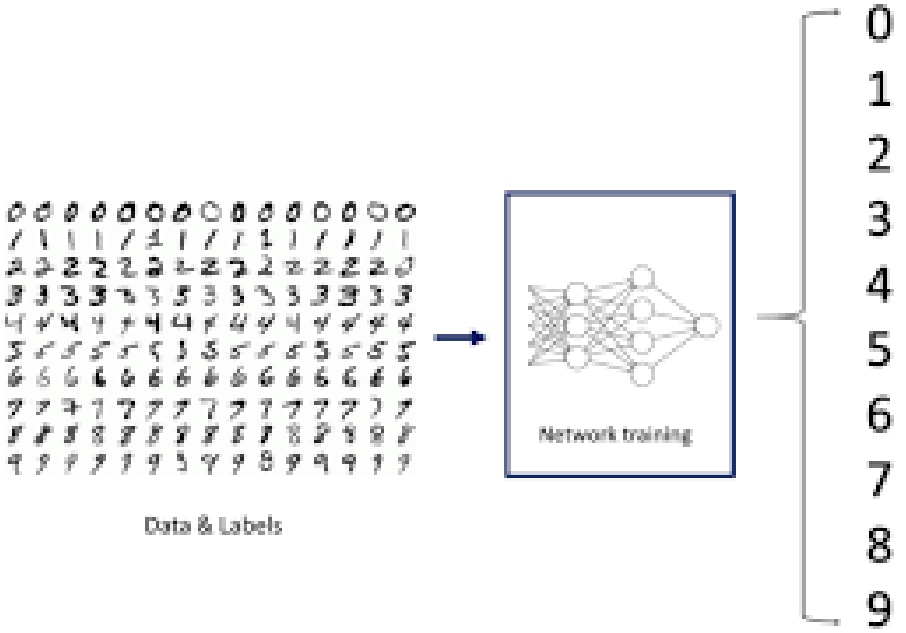
5. Pygame : Pygame is a Python wrapper for the SDL library, which stands for Simple DirectMedia Layer. SDL provides cross-platform access to your system's underlying multimedia hardware components, such as sound, video, mouse, keyboard, and joystick. pygame started life as a replacement for the stalled Py-SDL project. The cross-platform nature of both SDL and pygame means you can write games and rich multimedia Python programs for every platform that supports them!

2.3 Loading The Data Set:

2.3.1 MNIST Data Set:



Modified National Institute of Standards and Technology (MNIST) is a large set of computer vision dataset which is extensively used for training and testing different systems. It was created from the two special datasets of National Institute of Standards and Technology (NIST) which holds binary images of handwritten digits. The training set contains handwritten digits from 250 people, among them 50% training dataset was employees from the Census Bureau and the rest of it was from high school students. However, it is often attributed as the first datasets among other datasets to prove the effectiveness of the neural networks.



The database contains 60,000 images used for training as well as few of them can be used for cross validation purposes and 10,000 images used for testing. All the digits are grayscale and positioned in a fixed size where the intensity lies at the center of the image with 28×28 pixels. Since all the images are 28×28 pixels, it forms an array which can be flattened into 28*28=784 dimensional vector. Each component of the vector is a binary value which describes the intensity of the pixel.

2.4 Pre-Processing

Data pre-processing plays an important role in any recognition process. Data pre-processing is a data mining technique which is used to transform the raw data in a useful and efficient format. To shape the input images in a form suitable for segmentation pre-processing is used. Data pre-processing is a necessary step before building a model with these features. It usually happens in stages.

• Data quality assessment

• Data cleaning

• Data transformation

• Data reduction

2.4.1 Data quality assessment:

A Data Quality Assessment is a distinct phase within the data quality life-cycle that is used to verify the source, quantity and impact of any data items that breach pre-defined data quality rules. The Data Quality Assessment can be executed as a one-off process or repeatedly as part of an ongoing data quality assurance initiative. The quality of your data can quickly decay over time, even with stringent data capture methods cleaning the data as it enters your database. People moving house, changing phone numbers and passing away all mean the data you hold can quickly become out of date. A Data Quality Assessment helps to identify those records that have become inaccurate, the potential impact that inaccuracy may have caused and the data's source. Through this assessment, it can be rectified and other potential issues identified.

2.4.2 Data cleaning:

Data cleaning is one of the important parts of machine learning. It plays a significant part in building a model. It surely isn't the fanciest part of machine learning and at the same time, there aren't any hidden tricks or secrets to uncover. However, proper data cleaning can make or break your project. Professional data scientists usually spend a very large portion of their time on this step. Because of the belief that, "Better data beats fancier algorithms". If we have a well-cleaned dataset, we can get desired results even with a very simple algorithm, which can prove very beneficial at times. Obviously, different types of data will require different types of cleaning. However, this systematic approach can always serve as a good starting point.

2.4.3 Data transformation

In fact, by cleaning and smoothing the data, we have already performed data modification. However, by data transformation, we understand the methods of turning the data into an appropriate format for the computer to learn from. Data transformation is the process in which data is taken from its raw, siloed and normalized source state and transform it into data that's joined together, dimensionally modelled, de-normalized, and ready for analysis. Without the right technology stack in place, data transformation can be time-consuming, expensive, and tedious. Nevertheless, transforming the data will ensure maximum data quality which is imperative to gaining accurate analysis, leading to valuable insights that will eventually empower data-driven decisions. Building and training models to process data is a brilliant concept, and more enterprises have adopted, or plan to deploy, machine learning to handle many practical applications. But for models to learn from data to make valuable predictions, the data itself must be organized to ensure its analysis yield valuable insights.

2.4.4 Data reduction:

Data reduction is a process that reduced the volume of original data and represents it in a much smaller volume. Data reduction techniques ensure the integrity of data while reducing the data. The time required for data reduction should not overshadow the time saved by the data mining on the reduced data set. Data Reduction Techniques: Techniques of data deduction include dimensionality reduction, numerosity reduction and data compression.

1. Dimensionality Reduction:

a. Wavelet Transform

b. Principal Component Analysis

c. Attribute Subset Selection

2. Numerosity Reduction:

a. Parametric

b. Non-Parametric

3. Data Compression:

When you work with large amounts of data, it becomes harder to come up with reliable solutions. Data reduction can be used to reduce the amount of data and decrease the costs of analysis. After loading 23 the data, we separated the data into X and y where X is the image, and y is the label corresponding to X. The first layer/input layer for our model is convolution. Convolution takes each pixel as a neuron, so we need to reshape the images such that each pixel value is in its own space, thus converting a 28x28 matrix of greyscale values into 28x28x1 tensor. With the right dimensions for all the images, we can split the images into train and test for further steps. After loading the data, we separated the data into X and y where X is the image, and y is the label corresponding to X. The first layer/input layer for our model is

convolution. Convolution takes each pixel as a neuron, so we need to reshape the images such that each pixel value is in its own space, thus converting a 28x28 matrix of greyscale values into 28x28x1 tensor. With the right dimensions for all the images, we can split the images into train and test for further steps.

2.5 Data Encoding:

This is an optional step since we are using the cross-categorical entropy as loss function. We have to specify the network that the given labels are categorical in nature. The raw data can contain various different types of data which can be both structured and unstructured and needs to be processed in order to bring to form that is usable in the Machine Learning models. Since machine learning is based on mathematical equations, it would cause a problem when we keep categorical variables as is. Many algorithms support categorical values without further manipulation, but in those cases, it's still a topic of discussion on whether to encode the variables or not. After the identification of the data types of the features present in the data set, the next step is to process the data in a way that is suitable to put to Machine Learning models. The three popular techniques of converting Categorical values to Numeric values are done in two different methods.

1. Label Encoding.

2. One Hot Encoding.

3. Binary Encoding.

Encoding variability describes the variation of encoding of individually inside a category. When we talk about the variability in one hot encoding, the variability depends on the time of implementation in which it decides the number of categories to take that do have sufficient

impact on the target. Other encoding methodologies do show a significant variability which is identified at the time of validation.

2.6. CONVOLUTION NEURAL NETWORK:

In simpler words, CNN is an artificial neural network that specializes in picking out or detect patterns and make sense of them. Thus, CNN has been most useful for image classification. A CNN model has various types of filters of different sizes and numbers. These filters are essentially what helps us in detecting the pattern. The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data. Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a “convolution“. 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. Our model is composed of feature extraction with convolution and binary classification. Convolution and max pooling are carried out to extract the features in the image, and a 32 3x3 convolution filters are applied to a 28x28 image followed by a max-pooling layer of 2x2 pooling size followed by another convolution layer with 64 3x3 filters.

In the end, we obtain 7x7 images to flatten. Flatten layer will flatten the 7x7 images into a series of 128 values that will be mapped to a dense layer of 128 neurons that are connected to the categorical output layer of 10 neurons. The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product.

A dot product is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the “scalar product” Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom. The output from multiplying the filter with the input array one time is a single value. As the filter is applied multiple times to the input array, the result is a two-dimensional array of output values that represent a filtering of the input. As such, the two-dimensional output array from this operation is called a “feature map”.

2.7 Training & Validation:

After the construction of the model the model has to be compiled to train it with the available data set. Optimizers are used to compile the model. Compiling the model takes three parameters: optimizer, loss and metrics. Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function. The optimizer controls the learning rate. We will be using ‘adam’ as our optmizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer. We will use ‘categorical_crossentropy’ for our loss function. This is the most common choice for classification. A lower score indicates that the model is performing better. To make things

even easier to interpret, we will use the ‘accuracy’ metric to see the accuracy score on the validation set when we train the model. The idea behind training and testing any data model is to achieve maximum learning rate and maximum validation. Better Learning rate and better validation can be achieved by increasing the train and test data respectively. 35 Once the model is successfully assembled, then we can train the model with training data for 100 iterations, but as the number of iteration increases, there is a chance for overfitting. Therefore we limit the training up to 98% accuracy, as we are using real-world data for prediction, test data was used to validate the model.

2.7.1 ADAM Optimizer :

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the ‘gradient descent with momentum’ algorithm and the ‘RMSP’ algorithm. The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. Adam was presented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper (poster) titled “Adam: A Method for Stochastic Optimization“.

Why ADAM?

1. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

2. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

3. Adam is relatively easy to configure where the default configuration parameters do well on most problems.

2.8 Model Evaluation & Prediction:

For real-world image classification prediction, we need to do a little image pre-processing on the real-world images as model training was done with greyscale raster images. The steps of image pre-processing are :

1. Loading image

2. Convert the image to greyscale

3. Resize the image to 28x28

4. Converting the image into a matrix form

5. Reshape the matrix into 28x28x1

After pre processing, we predict the label of the image by passing the pre-processed image through the neural network. The output we get is a list of 10 activation values 0 to 9, respectively. The position having the highest value is the predicted label for the image.

These structures are called as Neural Networks. It teaches the computer to do what naturally comes to humans. Deep learning, there are several types of models such as the Artificial Neural Networks (ANN), Autoencoders, Recurrent Neural Networks (RNN) and Reinforcement Learning. But there has been one particular model that has contributed a lot in the field of

computer vision and image analysis which is the Convolutional Neural Networks (CNN) or the ConvNet.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

Methods for evaluating a model's performance are divided into 2 categories: namely, holdout and Cross-validation. Both methods use a test set (i.e data not seen by the model) to evaluate model performance. It's not recommended to use the data we used to build the model to evaluate it. This is because our model will simply remember the whole training set, and will therefore always predict the correct label for any point in the training set. This is known as overfitting.

Holdout:

The purpose of holdout evaluation is to test a model on different data than it was trained on. This provides an unbiased estimate of learning performance. In this method, the dataset is randomly divided into three subsets:

1. Training set is a subset of the dataset used to build predictive models.

2. Validation set is a subset of the dataset used to assess the performance of the model built in the training phase. It provides a test platform for fine-tuning a model's parameters and selecting the best performing model. Not all modeling algorithms need a validation set.

3. Test set, or unseen data, is a subset of the dataset used to assess the likely future performance of a model. If a model fits to the training set much better than it fits the test set, overfitting is probably the cause.

The holdout approach is useful because of its speed, simplicity, and flexibility. However, this technique is often associated with high variability since differences in the training and test dataset can result in meaningful differences in the estimate of accuracy.

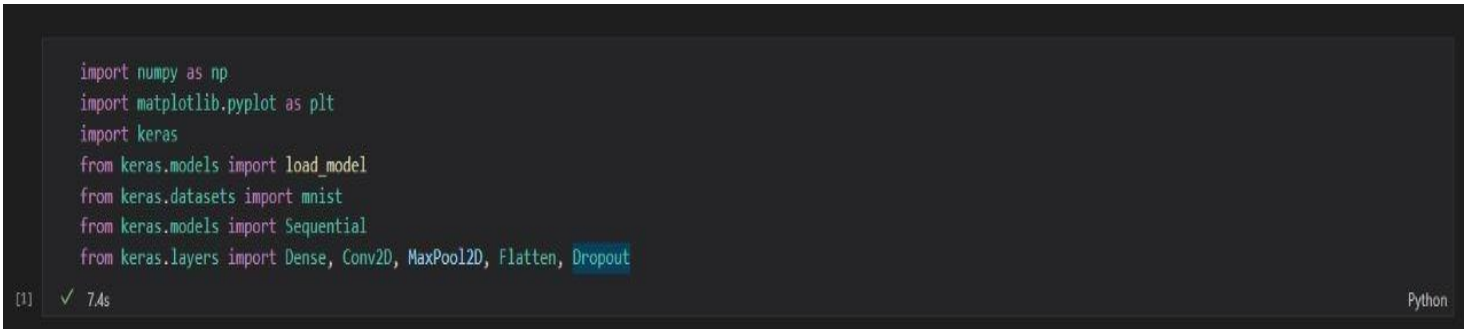
Cross-Validation:

Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis. The most common cross-validation technique is k-fold cross-validation, where the original dataset is partitioned into k equal size subsamples, called folds. The k is a user-specified number, usually with 5 or 10 as its preferred value. This is repeated k times, such that each time, one of the k subsets is used as the test set/validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get the total effectiveness of our model.

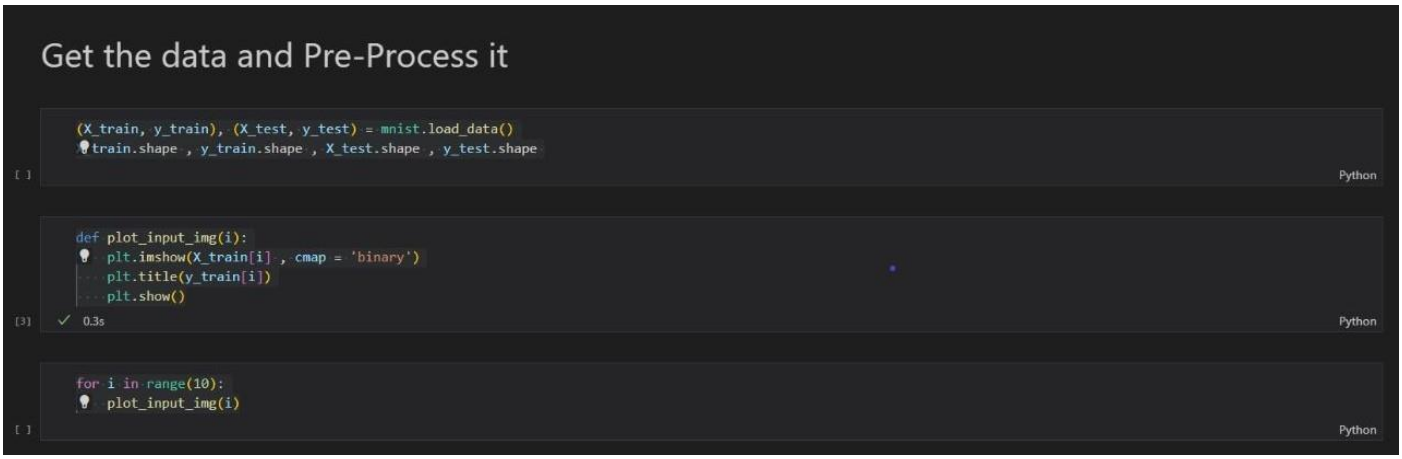
4. Implementation

4.1 Import the libraries and load the dataset: 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. So we can easily import the dataset and start working with it.

The load_data() method returns us the training data, its labels and also the testing data and its labels.



4.2 Pre-process 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).



```
# pre process the images

#Normalizing the image to [0,1] range
X_train = X_train.astype(np.float32)/255
X_test = X_test.astype(np.float32)/255

#REshape / expand the dimentions of images tp (28,28,1)
X_train = np.expand_dims(X_train, -1)
X_test = np.expand_dims(X_test, -1)

# convert classes to one hot vectors
y_train = keras.utils.to_categorical(y_train)

y_test = keras.utils.to_categorical(y_test)
```

[5] ✓ 0.1s Python

```
model = Sequential()
model.add(Conv2D(32, (3,3), input_shape = (28,28,1) , activation = 'relu') )
model.add(MaxPool2D((2,2)))

model.add(Conv2D(64, (3,3) , activation = 'relu') )
model.add(MaxPool2D((2,2)))

model.add(Flatten())

model.add(Dropout(0.25))

model.add(Dense(10, activation="softmax"))
```

[6] ✓ 1.7s Python

```
model.summary()
```

[ ] Python

4.3 Create the model: 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. We will then compile the model with the Adadelta optimizer.

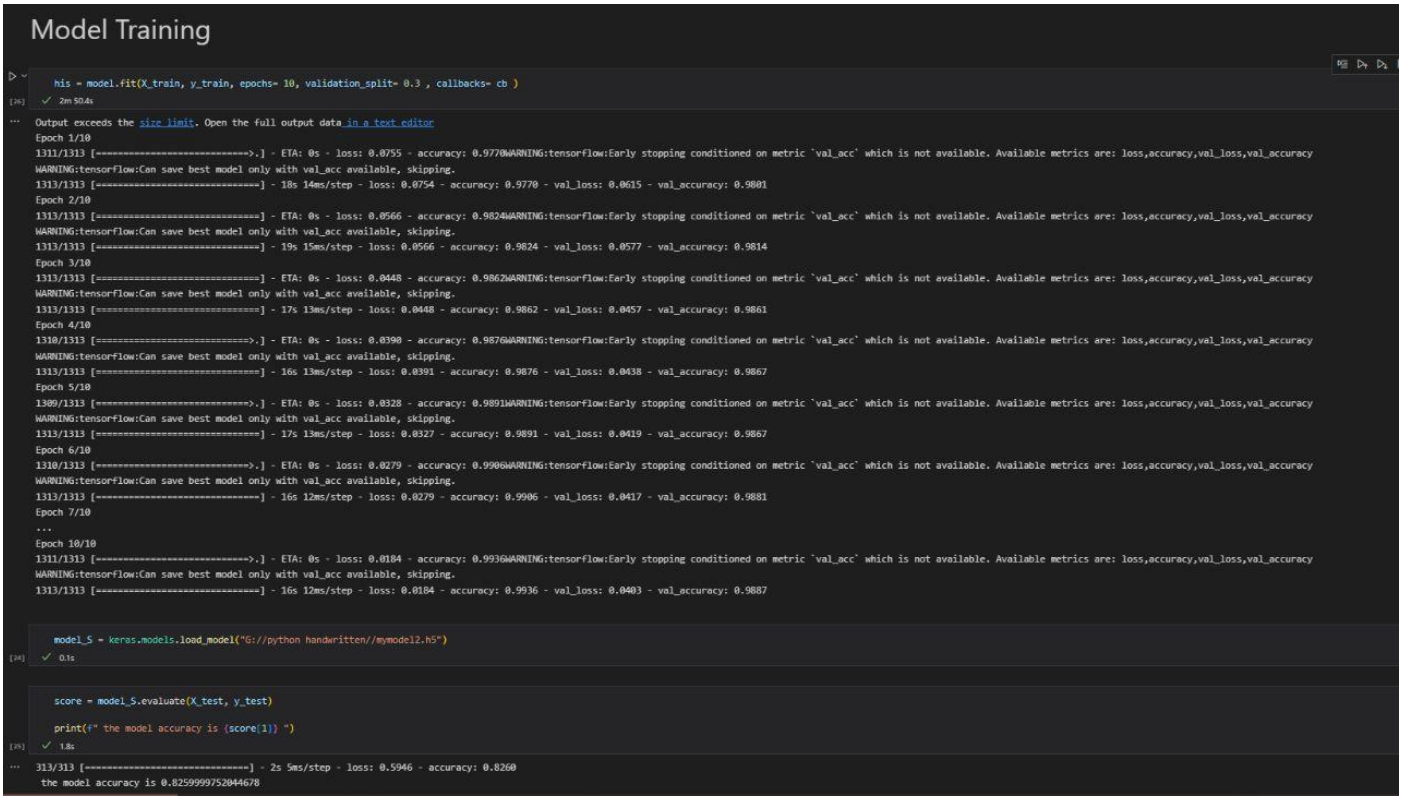
```
model.compile(optimizer= 'adam', loss = keras.losses.categorical_crossentropy , metrics=['accuracy'] )
```

[8] ✓ 0.3s Python

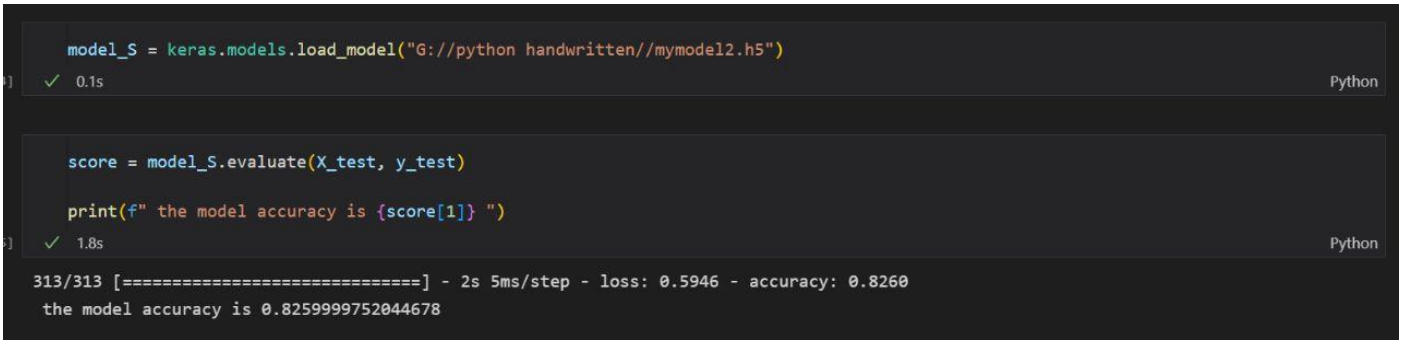
```
#Callbacks

from keras.callbacks import EarlyStopping, ModelCheckpoint

#Earlystopping

es = EarlyStopping(monitor='val_acc', min_delta= 0.01, patience= 4, verbose= 1)

#Model Check Point

mc = ModelCheckpoint("./mymodel2.h5", monitor="val_acc", verbose= 1, save_best_only= True)
cb = [es,mc]
```

[9] ✓ 0.2s Python

4.4 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.



4.5 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.



4.6 Create GUI to predict digits: 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 pygame library comes in the Python standard library. We have created a function program 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.

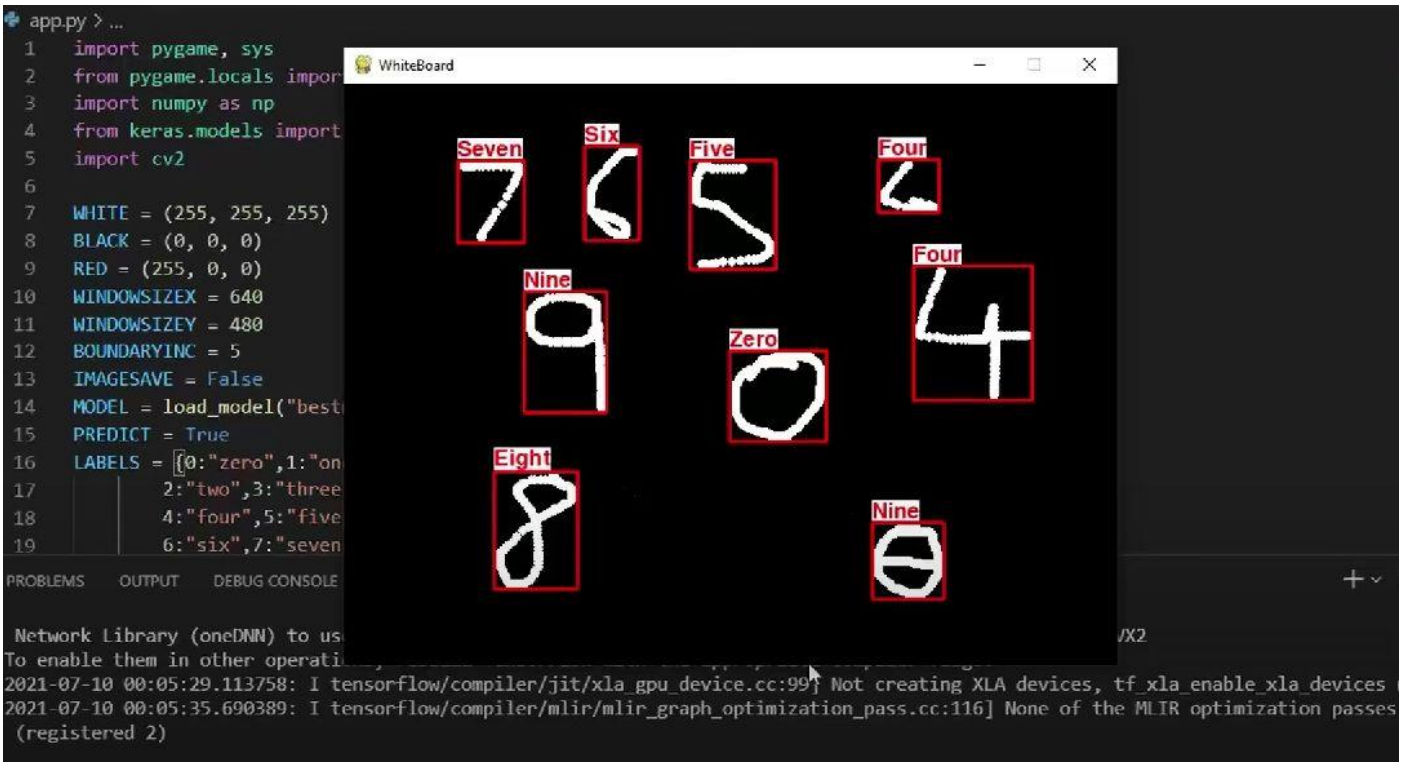
app.py > ...

```python
from tokenize import Number
from numpy import testing
from numpy.lib.type_check import imag
import pygame, sys
from pygame import image
from pygame.locals import *
import numpy as np
from keras.models import load_model
import cv2
from tensorflow.python.keras.backend import constant

WINDOWSIZEX = 640
WINDOWSIZEY = 480
BOUNDRYINC = 5
WHITE = (255,255,255)
BLACK = (0,0,0)
RED = (255, 0, 0)

IMAGESAVE = False
MODEL = load_model("mymodel2.h5")
LABELS = {0:"Zero", 1:"One",
          2:"Two", 3:"Three",
          4:"Four", 5:"Five",
          6:"Six", 7:"Seven",
          8:"Eight", 9:"Nine"}

# Initialize our pygame
pygame.init()

#FONT = pygame.font.Font("freesansbold.tff", 18)
DISPLAYSURF = pygame.display.set_mode((WINDOWSIZEX, WINDOWSIZEY))

pygame.display.set_caption("Digit Board")
```

```python
iswriting = False

number_xcord = []
number_ycord = []
imag_cnt = 1
PREDICT = True


while True:

    for event in pygame.event.get():
        if event.type == QUIT:
          pygame.quit()
          sys.exit()
        if event.type == MOUSEMOTION and iswriting :
            xcord, ycord = event.pos
            pygame.draw.circle(DISPLAYSURF, WHITE, (xcord, ycord), 4, 0)

            number_xcord.append(xcord)
            number_ycord.append(ycord)

        if event.type == MOUSEBUTTONDOWN:
            iswriting = True

        if event.type == MOUSEBUTTONUP:
            iswriting = False
            number_xcord = sorted(number_xcord)
            number_ycord = sorted(number_ycord)

            rect_min_x, rect_max_x = max(number_xcord[0]-BOUNDRYINC, 0), min(WINDOWSIZEX, number_xcord[-1]+BOUNDRYINC)
            rect_min_Y, rect_max_Y = max(number_ycord[0]-BOUNDRYINC), min(number_ycord[-1]+BOUNDRYINC, WINDOWSIZEX)

            number_xcord = []
            number_ycord = []

            img_arr = np.array(pygame.PixelArray(DISPLAYSURF))[rect_min_x:rect_max_x, rect_min_Y, rect_max_Y].T.astype(np.float32)

            if IMAGESAVE:
                cv2.imwrite("image.png")
                imag_cnt += 1
```

```
            if PREDICT:

                image = cv2.resize(ing_arr,(28,28))
                image = np.pad(image,(10,10), 'constant', constant_values = 0)
                image = cv2.resize(imag, (28,28))/255

                label = str(LABELS[np.argmax(MODEL.predict(image.reshape(1,28,28,1)))])

                textSurface = FONT.render(label, True, RED, WHITE)
                textRecoObj = testing.get_rect()
                textRecoObj.left, textRecoObj.bottom = rect_min_x, rect_max_Y

                DISPLAYSURF.blit(textSurface, textRecoObj)

            if event.type == KEYDOWN:
                if event.unicode == "n":
                    DISPLAYSURF.fill(BLACK)

        pygame.display.update()
```

4.5 Output :



Conclusion

Our project HANDWRITTEN DIGIT RECOGNITION deals with identifying the digits. The main purpose of this project is to build an automatic handwritten digit recognition method for the recognition of handwritten digit strings. The proposed system takes 28x28 pixel sized images as input. The same system with further modifications and improvements in the dataset and the model can be used to build Handwritten Character Recognition System which recognizes human handwritten characters and predicts the output.

REFERENCES

1. https://www.tensorflow.org/learn

2. https://www.geeksforgeeks.org/python-tkinter-tutorial/

3. https://medium.com/the-andela-way/applying-machine-learning-to-recognize handwrittencharacters-babcd4b8d705

4. https://nanonets.com/blog/handwritten-character-recognition/

5. https://www.geeksforgeeks.org/python-tkinter-tutorial/

6. https://medium.com/the-andela-way/applying-machine-learning-to-recognize handwrittencharacters-babcd4b8d705

7. https://nanonets.com/blog/handwritten-character-recognition/