DATA ANALYTICS AND PREDICTIVE MODELLING

OPTICAL CHARACTER RECOGNITION

MEGHA S RAO

S201113400005

NIIT

BANGALORE

2020

ACKNOWLEDGEMENT

The satisfactory and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned the efforts with success.

I would like to proudly thank management of NIIT for providing such a healthy environment for the successful completion of project.

I would like to express my gratitude to Lopamudra Bera, Faculty Advisor for DAPM, NIIT, for her constant support and encouragement.

I would like to thank Vijeeth Bissa, IT Head, NIIT – Malleshwaram, for his constant support and encouragement

Finally, I would hereby acknowledge and thank my family who has been a source of inspiration and instrument in the successful completion of the project work.

MEGHA S RAO

TABLE OF CONTENTS

INTRODUCTION

PROJECT BACKGROUND

Optical Character Recognition (OCR) is one of the most widely implemented types of data entry methods. OCR is a complex technology that converts image containing text into formats with editable text. OCR allows you process scanned books, screenshots and images with text into editable documents like TXT, DOC or PDF files. This technology is widely used in many areas. The most advanced OCR system can handle any type of images, even such complex ones as scanned magazine pages with images and columns or photos from a mobile phone.

This project involves implementing OCR in the Python programming language for the conversion of image data to characters. Then, feature set and image data set are fed to an Artificial Neural Networks (ANNs) for character recognition. ANN recognizes various 8-bit ASCII English character with good recognition rates.

WHAT IS OCR?

OCR involves digitized scanning and recognition of written or printed text. Here the text is first photo scanned, analyzed, and is finally translated into character codes. This machine-encoded text can be easily searched and edited electronically. Next to keypunching, Optical Character Recognition is the oldest data entry technique in existence. Long before the first key-to-disk system of CRT was used, Optical Character Readers were entering data in commercial and government EDP installations.

OCR has greatly improved the process of data entry. This software tool enables quick conversion of scanned documents to searchable text files. Today, the need for the documents to be scanned is on a constant rise as it enables these documents to be viewed conveniently when required. The scanned documents can also be shared easily through the electronic medium.

CAUSES AND EFFECTS OF OCR

Although there are numerous advantages of **OCR**, it mainly helps businesses in increasing the effectiveness and efficiency of the work. Its ability to quickly search through enormous content is extremely helpful, particularly in office settings, which deal with high document inflow and high-volume scanning.

- ***Higher Productivity***

OCR software helps businesses to achieve higher productivity by facilitating quicker data retrieval when required. The time and effort which the employees were required to put in for extracting relevant data can now be channelized to focus on core activities. Besides, employees do not have to make numerous trips to central records room to access the required documents, as they can access them without getting up from their desks.

- ***Cost Reduction***

Opting for OCR will help businesses on cutting down on hiring professionals to carry out data extraction, which is one of the most important benefits of OCR data entry methods. This tool also helps in trimming various other costs, such as copying, printing, shipping, etc. Therefore, OCR eliminates the cost of misplaced or lost documents and offers higher savings in the form of reclaimed office space, which would otherwise be used for storing paper documents.

- ***High Accuracy***

One of the major challenges of data entry is inaccuracy. Automated data entry tools such as OCR data entry result in reduced errors and inaccuracies, resulting in efficient data entry. Besides, problems like data loss can also be successfully tackled by OCR data entry. As there is no manpower involved, the issues such as keying in wrong information accidentally or otherwise can be eliminated.

- ***Increased Storage Space***

OCR can scan, document, and catalogue information from enterprise-wide paper documents. This simply means that the data can now be stored in an electronic format in servers, eradicating the need for maintaining huge paper files. In this way, OCR data entry serves as one of the best tools to implement "Paperless" approach across the organization.

- ***Superior Data Security***

Data security is of utmost importance for any organization. Paper documents are easily prone to loss or destruction. Papers can be misplaced, stolen, or destroyed by natural elements such as moisture, pests, and fire. However, this is not the case with data that is scanned, analyzed, and stored in digital formats. Furthermore, the access to these digital documents can also be minimized to prevent mishandling of the digitized data.

- ***100% Text-searchable Documents***

One of the huge advantages of OCR data processing is that it makes the digitized documents completely text searchable. This helps professionals to quickly lookup numbers, addresses, names, and various other parameters that differentiate the document being searched.

- ***Massively Improves Customer Service***

Several inbound contact centers often provide information that their customers seek. While some call centers provide customers with the information they need, the others will have to quickly access certain personal or order-related information of the customers to process their requests. Quick data accessibility becomes extremely important in such cases. OCR helps in systematically storing and retrieving the documents digitally at blazing speeds. With this, the waiting time is drastically reduced for the customers, thereby improving their experience.

- ***Makes Documents Editable***

Scanned documents need to be edited most of the time, particularly when some information must be updated. OCR converts data to any preferred formats such as Word, etc., which can be easily edited. This can be of great help when there are contents which have to be constantly updated or regularly changed.

- ***Disaster Recovery***

Disaster recovery is one of the major benefits of using OCR for data entry. When data is stored electronically in secure servers and distributed systems, it remains safe even under emergency situations. When there are sudden fire breakouts or natural calamity, the digitized data can be quickly retrieved to ensure business continuation.

PROJECT RELATED PROGRAMMING AND MATHEMATICAL ASPECTS

Introduction to Python Programming with its key features

Due to many programming languages, and every language has its own unique and different feature. It is this feature that matter the most in planning and choosing the best programming language for a project. Its better to know about Python Programming before we use this in our project.

- ***Easy to code and Read***

Python is easy to learn as the coding is not very difficult and seems like simple English language. This makes it readable and easier to write a program.

✓ **Easy to code:** as we have discussed above Python is much easier to code as compared to other programming languages such as C++, Java. It is easier to code in Python which allows the user to invest less time and effort to code a program. Anyone can learn Python syntax within a few hours because of its simple language. As we know mastering in anything takes time. To learn every modules, package, and advanced concepts will take time but you can surely master in it by concentrating and effort. Thus, it can be concluded Python is programmer-friendly.

✓ **Easy to Read:** Even though it is a high-level programming language Python seems to be easy to use and the code is almost like English. Just by looking at it you will be able to know what it is supposed to do, even if you are not a programmer.

- ***High-Level Programming Language***

  Python is a high-level programming language which means there is no need to memorize the system architecture. There is also no requirement to manage memory. This feature makes Python more programmers friendly and is one of the key features of Python.

- ***Portable***

  Python programming can be written on a machine and worked on any machine without changing. This feature makes Python Portable. Say for example you have written a program for your windows machine, but you also want it to run in your Mac, nothing to worry about you can make it work by transferring the program no need of making any changes either. So we can say pick the code and run it in any machine without any changes or effort. In such a scenario, any feature that is dependent on the system should be ignored.

- ***Expressive***

  Python is more expressive than other languages. First, to know what is expressive you must dig into the example, say there are two programming language X and Y. X can make any programs that Y can make using local transformation. There are more programs which can be made using X that cannot be made by Y using local transformation. In this case, we can see X has more capability and hence we say X is more expressive than Y. Python helps you give your focus on the solution rather than investing the mind on the Syntax which is because it provides you with a myriad of constructs. This is an amazing feature which can clear your mind about 'why you should learn this particular system?'

- ***Object Oriented***

  Python is known as Object Oriented programming language because it can model the real world. Python basically focuses on an object and combines functions and data. Oppositely a language which is more procedure-oriented revolves around the function that is coded and reused. In Python programming, both Procedure-oriented and Object-oriented language is supported. This is one more key feature of Python. Unlike Java, it also underpins various inheritance. A class is an abstract data type and holds no values. Class is a blueprint, for such objects.

- ***Free and Open Source***
  Python programming language can be downloaded freely. It is an open source, which means the public has access to the source code. You can do whatever you want to download, change, use and distribute it. This is called FLOSS which means Free/Libre and Open Source Software.

- ***Interpreted***
  In languages such as C++ and Java, you have to first compile it and then run it; you must know this if you are familiar with any one of the languages. But in Python, there is no requirement of compiling it before running it. In Python, there is a conversion of the source code to a form that is called bytecode.
  This means the Python code can be run without stressing over connecting to libraries and other different things.
  The source code is run line by line and not all at a time, which is why Python is said to have an Interpreted feature. This feature makes debugging your code easier. But interpreting makes it slightly slower than Java. However, that should not matter compared to the advantages it has to offer.

- ***Extensible***

Python can be extended to other languages which make it extensible language. This feature allows you to write some of the Python codes ion other languages C++ or Java.

- ***Embeddable***

In the above feature extensible we got to know that other languages code can be used in Python source code. Now, Embeddable means we can also put our Python code in different languages source code, like C++. This enables the users to coordinate scripting capabilities into the program of other languages.

- ***Large and Standard Library***

Python has a large library inbuilt that can be used while you are coding a program so that you don't have to write your own code for every single thing. In Python, the library implies a gathering of modules where you will locate all sort of stuff. Modules are records, much the same as books which contain functions that should be imported in your program. For example, on the off chance that you need to take a stab at something else for the graphics you can go to and check the Python Imaging Library.

- ***GUI programming***

In Python, you can create basic GUI's by using Tk.

- ***Dynamically Typed***

Python is dynamically-typed. This implies the Type for esteem is chosen at runtime, not progress of time. This is the reason we don't have to determine the sort of information while pronouncing it.

What is Artificial Intelligence?

What if your computer could wash your dishes, do your laundry, cook you dinner, and clean your home? I think I can safely say that most people would be happy to get a helping hand! But what would it take for a computer to be able to perform these tasks in the same way that humans can?

The famous computer scientist Alan Turing proposed the Turing Test as a way to identify whether a machine could have intelligence indistinguishable from that of a human being. The test involves a human posing questions to two hidden entities, one human, and the other a machine, and trying to identify which is which. If the interrogator is unable to identify the machine, then the machine is considered to have human-level intelligence.

While Turing's definition of intelligence sounds reasonable, at the end of the day what constitutes intelligence is fundamentally a philosophical debate. Computer scientists have, however, categorized certain types of systems and algorithms into branches of AI. Each branch is used to solve certain sets of problems. These branches include the following examples, as well as many others:

- Logical and probabilistic deduction and inference based on some predefined knowledge of a world. e.g. Fuzzy inference can help a thermostat decide when to turn on the air conditioning when it detects that the temperature is hot and the atmosphere is humid
- Heuristic search. e.g. Searching can be used to find the best possible next move in a game of chess by searching all possible moves and choosing the one that most improves your position
- Machine learning (ML) with feedback models. e.g. Pattern-recognition problems like OCR.

In general, ML involves using large data sets to train a system to identify patterns. The training data sets may be labelled, meaning the system's expected outputs are specified for given inputs, or unlabeled meaning expected outputs are not specified. Algorithms that train systems with unlabeled data are called *unsupervised* algorithms and those that train with labelled data are called *supervised*. Many ML algorithms and techniques exist for creating OCR systems, of which ANNs are one approach.

What are Artificial Neural Networks (ANNs)?

An ANN is a structure consisting of interconnected nodes that communicate with one another. The structure and its functionality are inspired by neural networks found in a biological brain. Hebbian Theory explains how these networks can learn to identify patterns by physically altering their structure and link strengths. Similarly, a typical ANN (shown in Figure 15.1) has connections between nodes that have a weight which is updated as the network learns. The nodes labelled "+1" are called *biases*. The leftmost blue column of nodes is *input nodes*, the middle column contains *hidden nodes*, and the rightmost column contains *output nodes*. There may be many columns of hidden nodes, known as *hidden layers*.

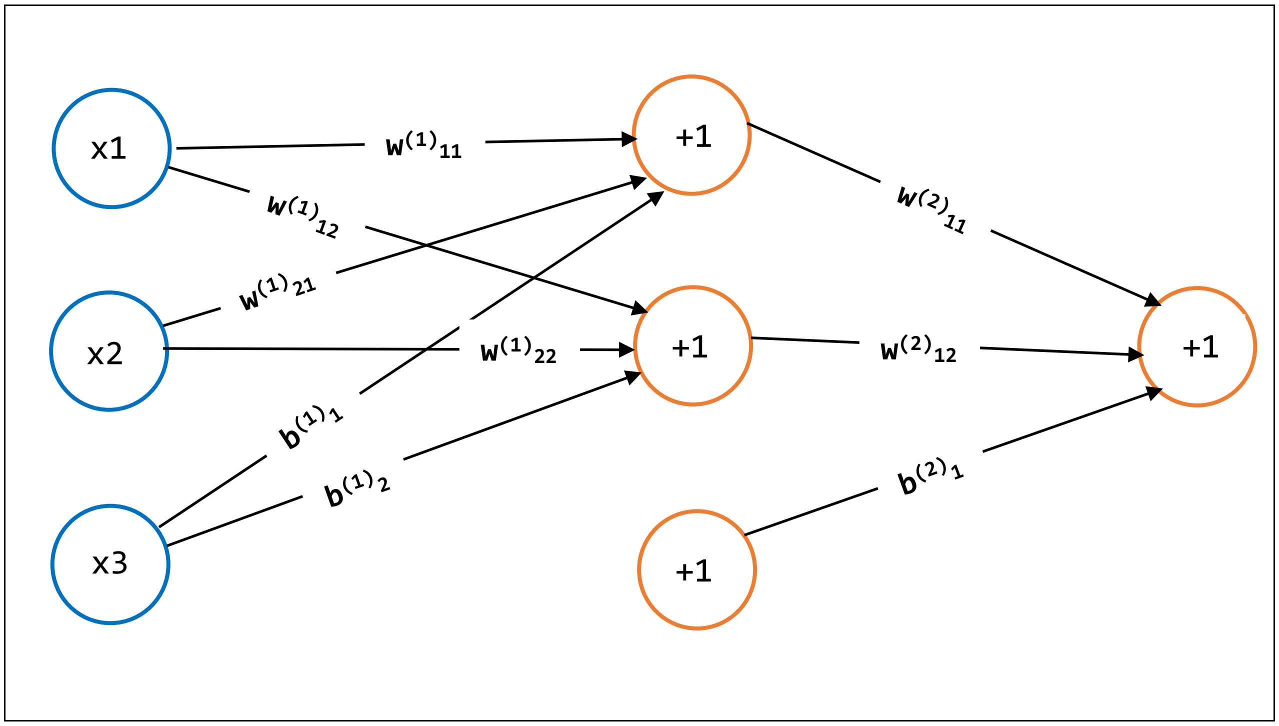


Fig: An Artificial Neural Network

The values inside all of the circular nodes in Figure 15.1 represent the output of the node. If we call the output of the nnth node from the top in layer LL as a n(L)n(L) and the connection between the iith node in layer LL and the jjth node in layer L+1 as $w^{(L)}{}_j$ i , then the output of node $a^{(2)}{}_2$ is:

$$a^{(2)}{}_2 = f(w^{(1)}{}_{21}\, x_1 + w^{(1)}{}_{22}\, x_2 + b^{(1)}{}_2)$$

where f(.) is known as the *activation function* and bb is the *bias*. An activation function is the decision-maker for what type of output a node has. A bias is an additional node with a fixed output of 1 that may be added to an ANN to improve its accuracy. We'll see more details on both of these in Designing a Feedforward ANN.

This type of network topology is called a *feedforward* neural network because there are no cycles in the network. ANNs with nodes whose outputs feed into their inputs are called recurrent neural networks. There are many algorithms that can be applied to train feedforward ANNs; one commonly used algorithm is called *backpropagation*. The OCR system we will implement in this chapter will use backpropagation.

Gradient Descent

This is a Machine Learning optimization technique to find the most optimal set of parameters for a given problem. Plotting here, some cost function measurements of the error of the learning system. In supervised learning, feeding your model with a group of parameters in some ways to tune your model. We need to identify different values in the parameters. So, it produces optimal results.

Hence, in gradient descent we pick some points at random. These represent, some set of parameters to your model. We measure the area of the point that produces on our system. We gradually move down the curve trying a different set of parameters here. New set of parameters might produce minimum error than the previous one. Then we continue the same and then we hit the bottom of the gradient. Here least error is observed pointing out that we hit the bottom of the curve. This is how we train our model to get optimal results.

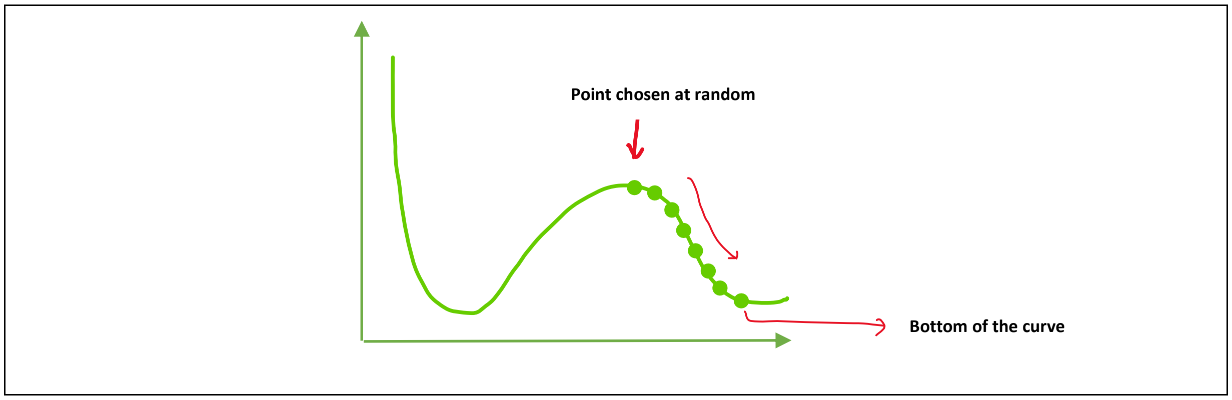


Fig: Gradient Descent

SOLUTION STATEMENT

The following OCR is explored in following ways:

1. OCR in Python language using ANN.
2. Develop a program that can read the image data and predict the characters.
3. It must also determine if a predicted character matches the actual one or not.

Consider we divide this in following parts:

PART I:
Visualizing characters in an OCR database
Developing an OCR application that recognizes text from image data and visualizes it by converting it into 2D images.

Part II:
Building an OCR Engine

- Develop OCR engine using ANN.
- Providing image dataset as input
- Split those into training and test data
- 70% training data (labelled)
- 30% test data (unlabeled)

PREREQUISITES

1. Working station with Windows latest version.
2. Anaconda Navigator – Includes latest version of **spyder (4+)** and following packages: OpenCV, neurolab previously installed.
3. Data File: **Letter.data** copied to your directory.

DATA AND CODE

Part I: *Visualizing characters in an OCR database*

Create a new python file and import the following packages:

```
import os
import sys
import cv2
import numpy as np
```

Define the input file containing the OCR data:

```
#Define the input file
input_f = 'letter.data'
```

Define the visualization and other parameters requires to load the data from the file:

```
#Define Visualization and other parameters
img_resize_factor = 12
start, end = 6, -1
height, width = 16, 8
```

Iterate through the lines of that file until the user presses Esc key. Each line in tat file is tab separated. Read each line and scale it up to 255.

```
#Iterate until the user presses Esc key
with open(input_f, 'r') as f:
    for line in f.readlines():
        #Read the data
        data = np.array([255*float(x) for x in line.split('\t')[start:end]])
```

Reshape the 1D array into 2D image:

```
#Reshape the data into 2D image
img = np.reshape(data, (height, width))
```

Scale the image for visualization:

```
#Scale the image
img_scaled = cv2.resize(img, None, fx=img_resize_factor, fy=img_resize_factor)
#Also printing the line:
print(line)
```

Display the image:

```
#Display the image
cv2.imshow('Img', img_scaled)
```

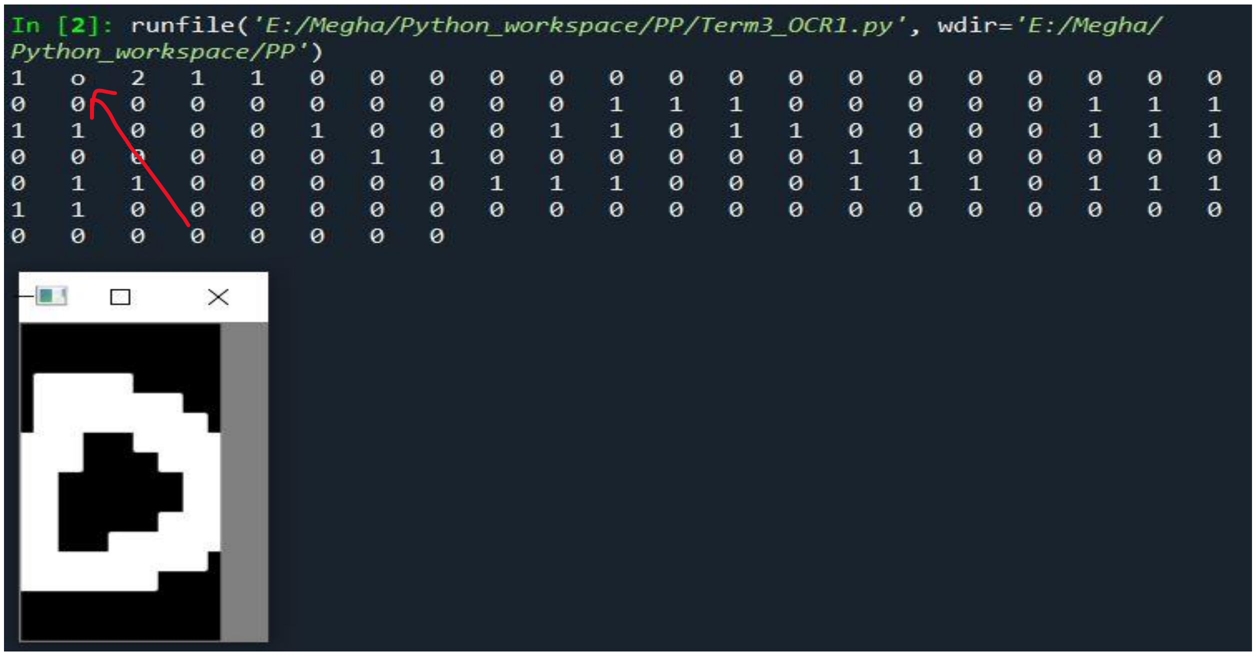
Check if the user has pressed the Esc key:

```
#Check
c = cv2.waitKey()
        if c == 27:
            break
```

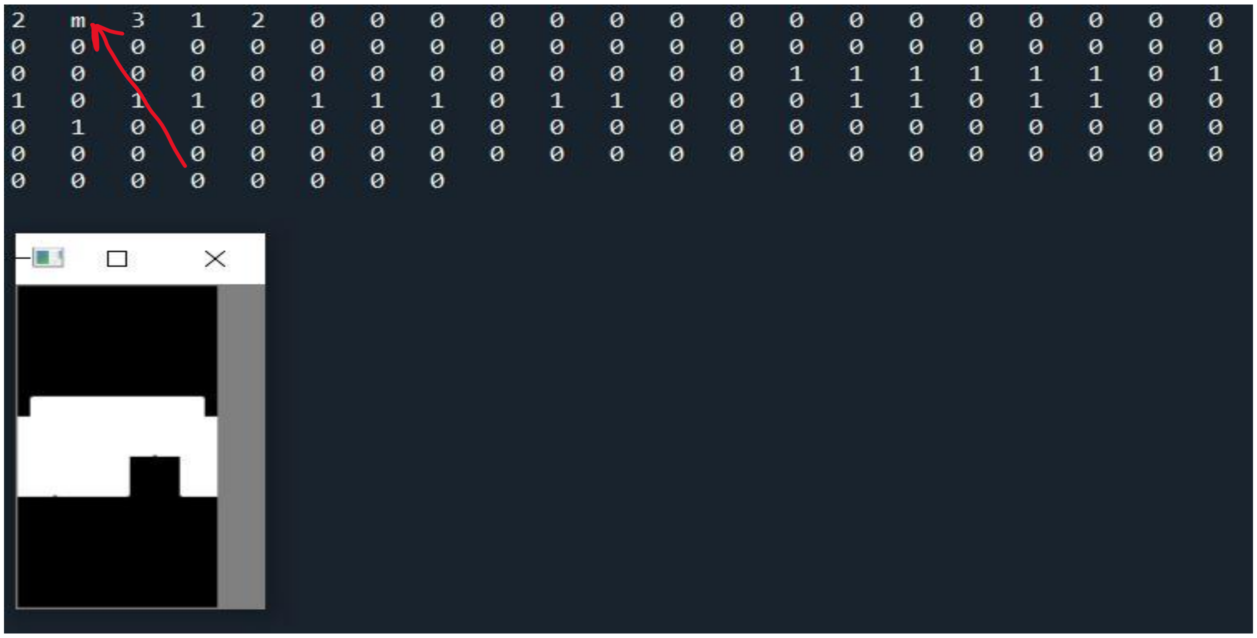
Once you run the code, we will observe an output screenshot as shown below:

Red arrow – Predicted

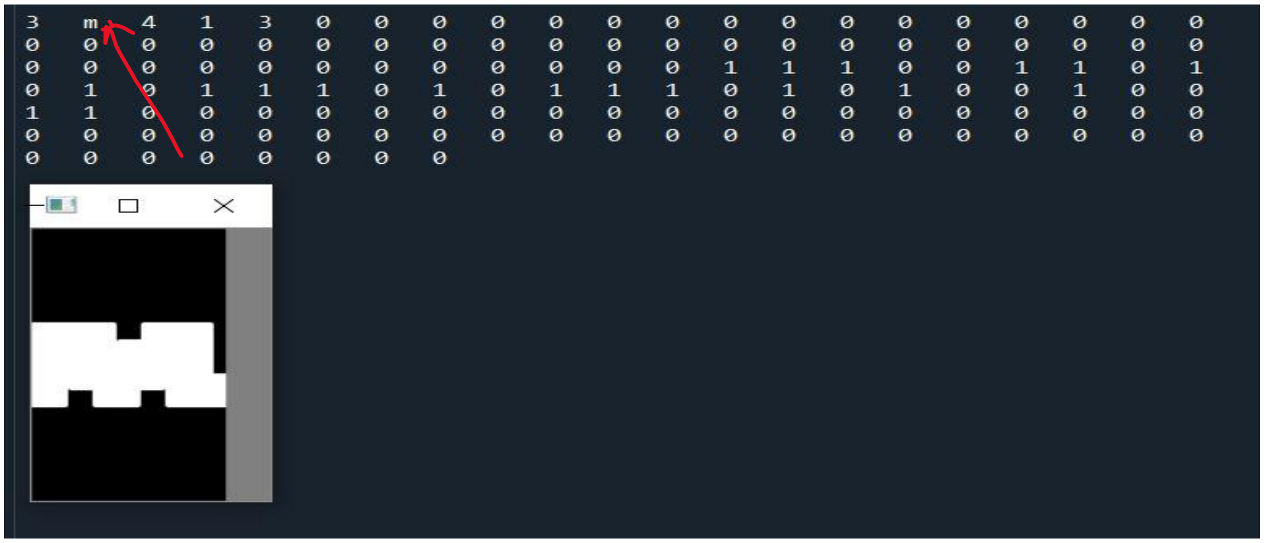
Screenshot – Orignal



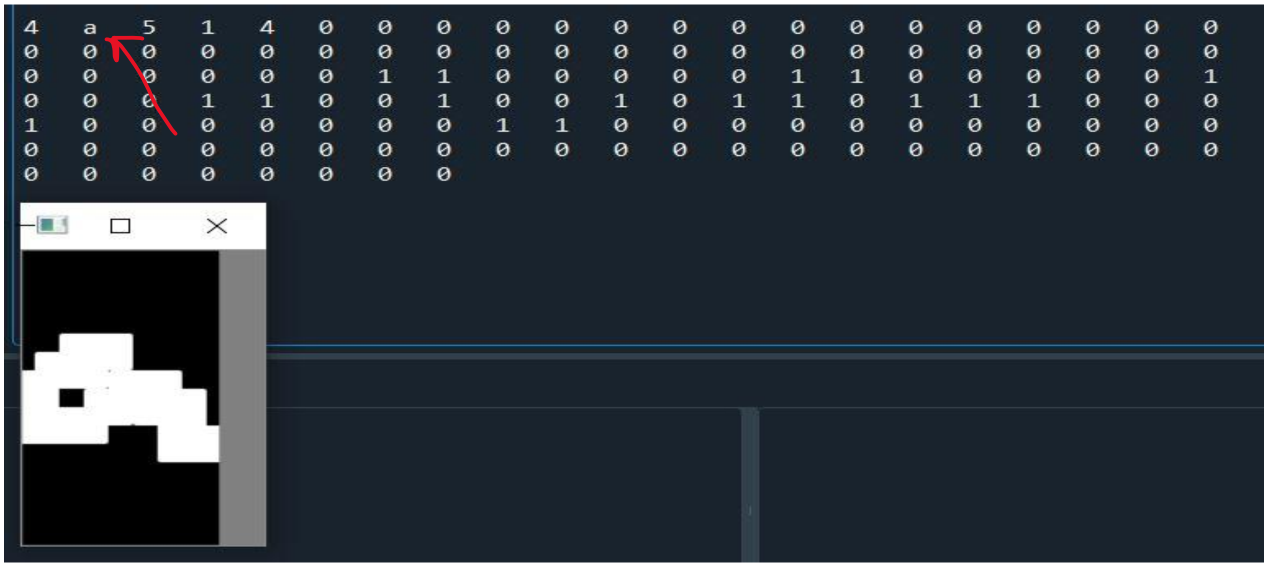
```
In [2]: runfile('E:/Megha/Python_workspace/PP/Term3_OCR1.py', wdir='E:/Megha/
Python_workspace/PP')
1   o   2   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0   1   1   1
1   1   0   0   0   1   0   0   0   1   1   0   1   1   0   0   0   0   1   1   1
0   0   0   0   0   0   1   1   0   0   0   0   0   0   1   1   0   0   0   0   0
0   1   1   0   0   0   0   0   1   1   1   0   0   0   1   1   1   0   1   1   1
1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
```



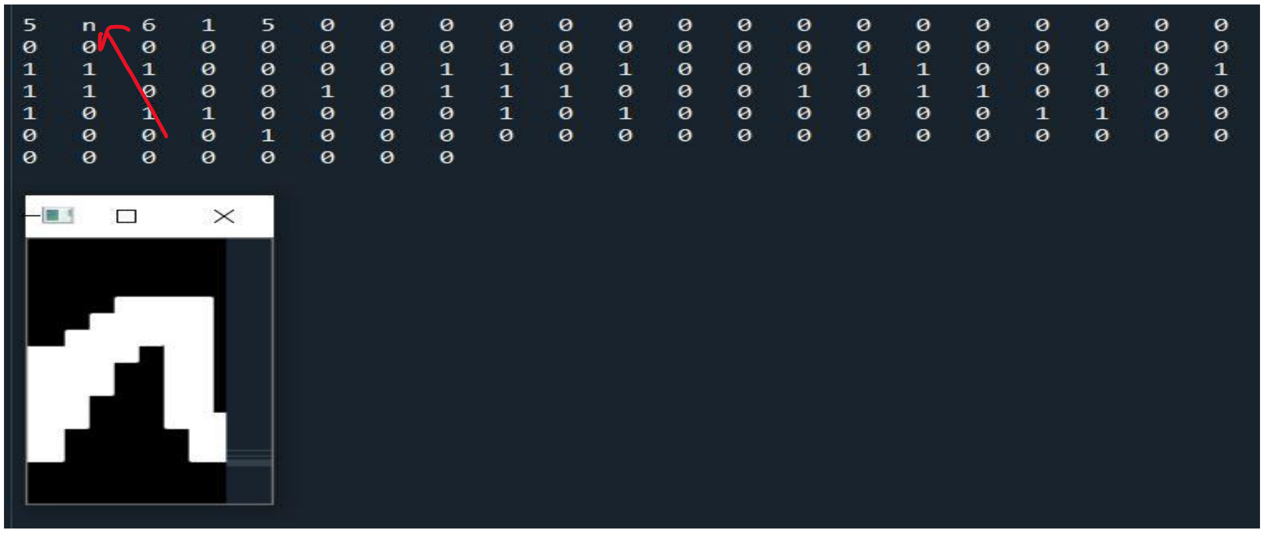
```
2   m   3   1   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   1   1   1   0   1
1   0   1   1   0   1   1   1   0   1   1   0   0   0   1   1   0   1   1   0   0
0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
```



```
3    m    4    1    3    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    1    1    1    0    0    1    1    0    1
0    1    0    1    1    1    0    1    0    1    1    1    0    1    0    1    0    0    1    0    0
1    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0
```



```
4    a    5    1    4    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    1    1    0    0    0    0    0    1    1    0    0    0    0    0    1
0    0    0    1    1    0    0    1    0    0    1    0    1    1    0    1    1    1    0    0    0
1    0    0    0    0    0    0    0    1    1    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0
```



```
5    n    6    1    5    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
1    1    1    0    0    0    0    1    1    0    1    0    0    0    1    1    0    0    1    0    1
1    1    0    0    0    1    0    1    1    1    0    0    0    1    0    1    1    0    0    0    0
1    0    1    1    0    0    0    0    1    0    1    0    0    0    0    0    0    1    1    0    0
0    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0
```

Part II: *Building and OCR engine*

Create a new python file and import these packages:

```
import numpy as np
import neurolab as nl
```

Define the input file:

```
# Define the input file
input_f = 'letter.data'
```

Define the number of datapoints that will be loaded:

```
#Define the number of datapoints to be loaded from the input file
num_data = 50
```

Define the string containing all the distinct characters:

```
#String containing all the distinct characters
orig_labels = 'omandig'
```

Extract the number of distinct classes:

```
#Compute the number of distinct characters
num_orig_labels = len(orig_labels)
```

Define the train and test spilt. We will use 70% for training and 30% for testing:

```
#Check
num_train = int(0.7*num_data)
num_test = num_data - num_train
```

Define the dataset extraction parameters:

```
#Define dataset extraction parameters
start = 6
end = -1
```

Create the dataset

```
#Creating the dataset
data = []
labels = []
with open(input_f, 'r') as f:
    for line in f.readlines():
        list_vals = line.split('\t')
```

If the label is not in our set of labels, we should skip it

```
#Check if the label is in our ground truth labels. If not, we should skip it.
if list_vals[1] not in orig_labels:
            continue
```

Extract the current label and append it to the main list:

```
# Extract the current label and append it to the main list
label = np.zeros((num_orig_labels,1))
        label[orig_labels.index(list_vals[1])]=1
        labels.append(label)
```

Extract the Character vector and append it to the main list:

```
# Extract the Character Vector and append it to the main list
cur_char = np.array([float(x) for x in list_vals[start:end]])
        data.append(cur_char)
```

Exit the loop once we have created the dataset:

```
# Exit the loop once we have created the dataset
if len(data) >= num_data:
            break
```

Convert the list into numpy arrays:

```
#Convert the list into numpy arrays
data = np.asfarray(data)
labels = np.array(labels).reshape(num_data, num_orig_labels)
```

Extract the number of dimensions:

```
#Extract the number of dimensions
num_dims = len(data[0])
```

Create a feedforward neural network and set the training algorithm to gradient descent:

```
#Create a feedforward neural network
nn = nl.net.newff([[0,1] for _ in range(len(data[0]))], [128,16,num_orig_labels])
#set the training algorithm to gradient descent
nn.trainf = nl.train.train_gd
```

Train the neural network:

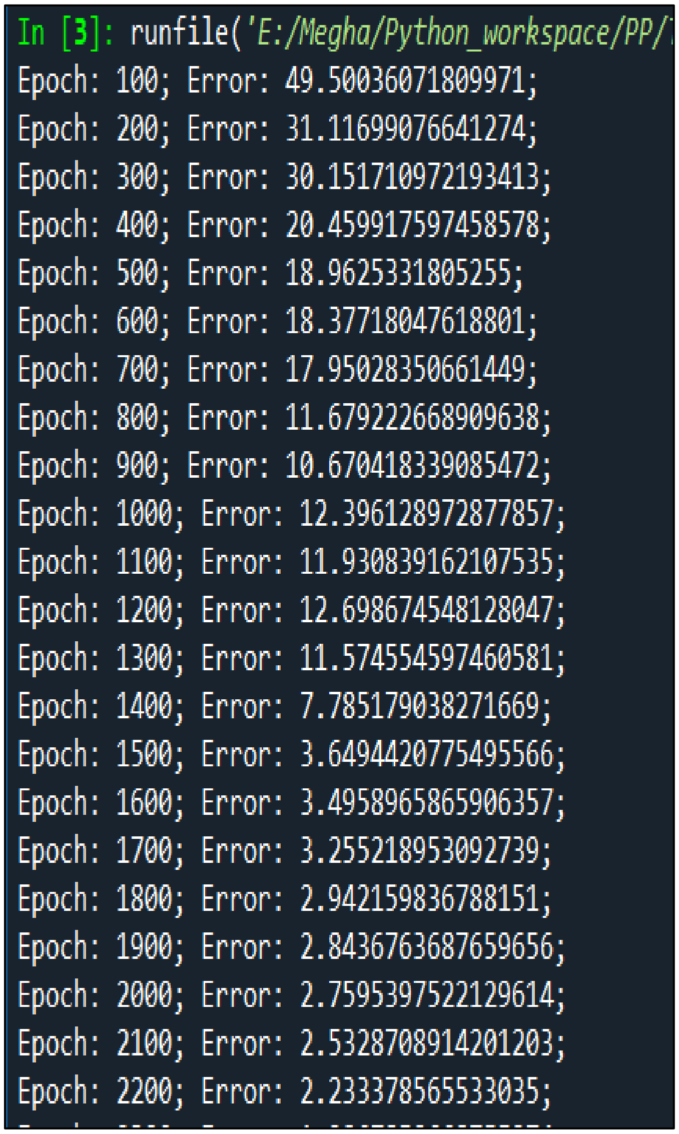
```
#train the neural network
error_progress = nn.train(data[:num_train,:], labels[:num_train,:],
epochs=10000,show=100, goal=0.01)
```

Predict the output for test data:

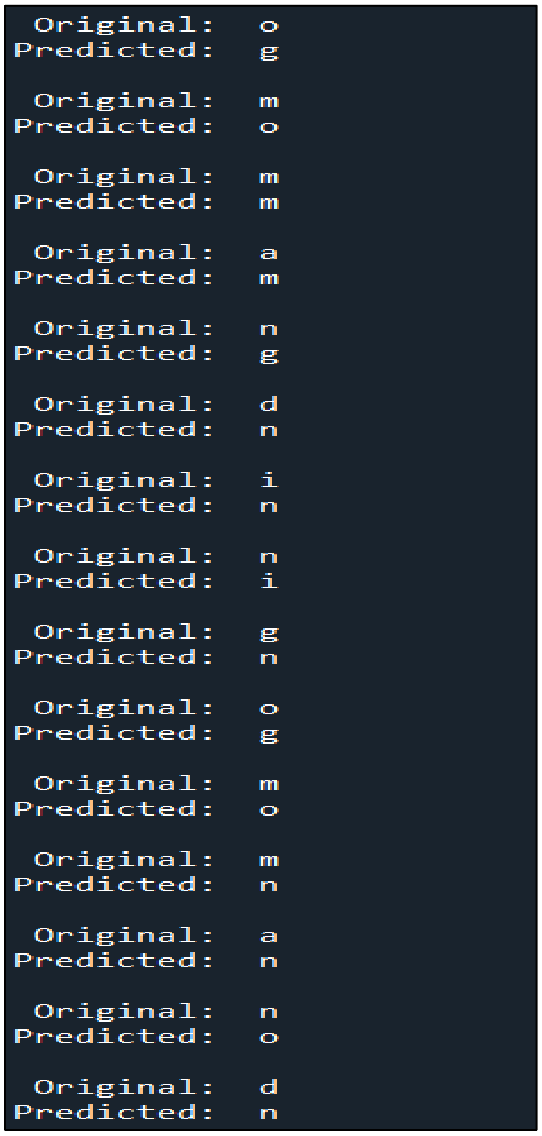
```
#Predict the output for the test data
pred_test = nn.sim(data[num_train:,:])
for i in range(num_test):
    print('\n Original: ', orig_labels[np.argmax(labels[i])])
    print('Predicted: ', orig_labels[np.argmax(pred_test[i])])
```

If you run the code you will see the folowing on your console. It will keep going till 10000 expochs, once done we will see the original and predicted charcters. If we use a bigger dataset and train longer then we will get higher accuracy.

OUTPUT:



```
In [3]: runfile('E:/Megha/Python_workspace/PP/
Epoch: 100; Error: 49.50036071809971;
Epoch: 200; Error: 31.11699076641274;
Epoch: 300; Error: 30.151710972193413;
Epoch: 400; Error: 20.459917597458578;
Epoch: 500; Error: 18.9625331805255;
Epoch: 600; Error: 18.37718047618801;
Epoch: 700; Error: 17.95028350661449;
Epoch: 800; Error: 11.679222668909638;
Epoch: 900; Error: 10.670418339085472;
Epoch: 1000; Error: 12.396128972877857;
Epoch: 1100; Error: 11.930839162107535;
Epoch: 1200; Error: 12.698674548128047;
Epoch: 1300; Error: 11.574554597460581;
Epoch: 1400; Error: 7.785179038271669;
Epoch: 1500; Error: 3.6494420775495566;
Epoch: 1600; Error: 3.4958965865906357;
Epoch: 1700; Error: 3.255218953092739;
Epoch: 1800; Error: 2.942159836788151;
Epoch: 1900; Error: 2.8436763687659656;
Epoch: 2000; Error: 2.7595397522129614;
Epoch: 2100; Error: 2.5328708914201203;
Epoch: 2200; Error: 2.233378565533035;
```



```
 Original:   o
Predicted:   g

 Original:   m
Predicted:   o

 Original:   m
Predicted:   m

 Original:   a
Predicted:   m

 Original:   n
Predicted:   g

 Original:   d
Predicted:   n

 Original:   i
Predicted:   n

 Original:   n
Predicted:   i

 Original:   g
Predicted:   n

 Original:   o
Predicted:   g

 Original:   m
Predicted:   o

 Original:   m
Predicted:   n

 Original:   a
Predicted:   n

 Original:   n
Predicted:   o

 Original:   d
Predicted:   n
```

CONCLUSION

Every OCR step is very important. The whole OCR process will fail if any step cannot handle the given image correctly. Every algorithm must work correctly on the highest range of images, that is why there are only a few good universal OCR systems in the market. On the other hand, if some features of given images are known, the task becomes much easier. You can get better recognition quality if only one kind of images must be processed. To achieve the best results, if some features of the images are known, a good OCR system must be able to adjust the most important parameters of every algorithm. Sometimes that's the only way to improve recognition quality. Unfortunately, even now there are no OCR systems that are as good as humans, and it looks like such systems will not be created in the near future.

REFERNECES

1. Optical Character Recognition – SDK tutorials – https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/

2. What's OCR – Data Identification - http://www.dataid.com/aboutocr.htm

3. Optical Character Recognition – Marina Samuel – http://www.aosabook.org/en/500L/optical-character-recognition-ocr.html

4. Deep learning and neural networks with python –Frank Kane – http://www.aosabook.org/en/500L/optical-character-recognition-ocr.html