HANDWRITING RECOGNITION USING MACHINE LEARNING

Project report submitted in partial fulfilment of the requirement of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

Hitesh Thakur (171028)

UNDER THE GUIDANCE OF

Mr. Munish Sood

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

May 2021

TABLE OF CONTENTS

DECLARATION

I hereby declare that the work revealed in the B.Tech Project Report named "**Handwriting Recognition using Machine Learning**" submitted at **Jaypee University of Information Technology,Waknaghat, India** is a genuine record of my work done under the management of Mr. Munish Sood. I have not presented this turn out somewhere else for some other degree or recognition.

[illegible]

Hitesh Thakur

171028

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

[illegible]

Mr. Munish Sood

Date: 29/06/2021

Head of the Department/Project Coordinator

ACKNOWLEDGEMENT

I would like to thank God for guiding us throughout our academic journey and to acknowledge our project supervisor Mr. Munish Sood, for his undying support, priceless motivation and guidance throughout the project duration. Moreover, I extend my sincere gratitude to all the faculties and non-teaching staff of the Department of Electronics and Communication Engineering for their contribution towards the success of this work.

My friends have also helped and motivated us at every step of this project. Without such immense support, making of this project would have been very challenging.

LIST OF ACRONYMS AND ABBREVIATIONS

ML	Machine Learning
CNN	Convolutional Neural Network
MNIST	Modified National Institute of Standards and Technology
AI	Artificial Intelligence
OCR	Optical character recognition
KNN	K-Nearest Neighbour
RNN	Recurrent Neural Network
VM	Virtual Machine
LSTM	Long short term memory
CTC	Connectionist temporal classification

LIST OF FIGURES

ABSTRACT

This project is very helpful in classifying handwritten words and characters and converting them into digital format. I have used a concept of machine learning known as recurrent convolutional neural networks to recognize and classify words and characters.

This model is going to be very helpful in doing basic tasks like data entry, keeping receipts, maintaining medical records etc. I will be training this model using a large amount of data so that it can learn easily and efficiently.

CHAPTER – 1

INTRODUCTION

1.1 Introduction

Machine Learning is a part of Artificial Intelligence (AI). It uses various techniques to provide machines the ability to learn with the help of large amounts of data. While Machine Learning has reduced the burden on programmers to explicitly program the computers, it has touched the areas which have never been explored before. This has led to various technological advances. There are two types of Machine Learning algorithms :

- Supervised Learning Algorithms
- Unsupervised Learning Algorithms
- Reinforcement Learning Algorithms

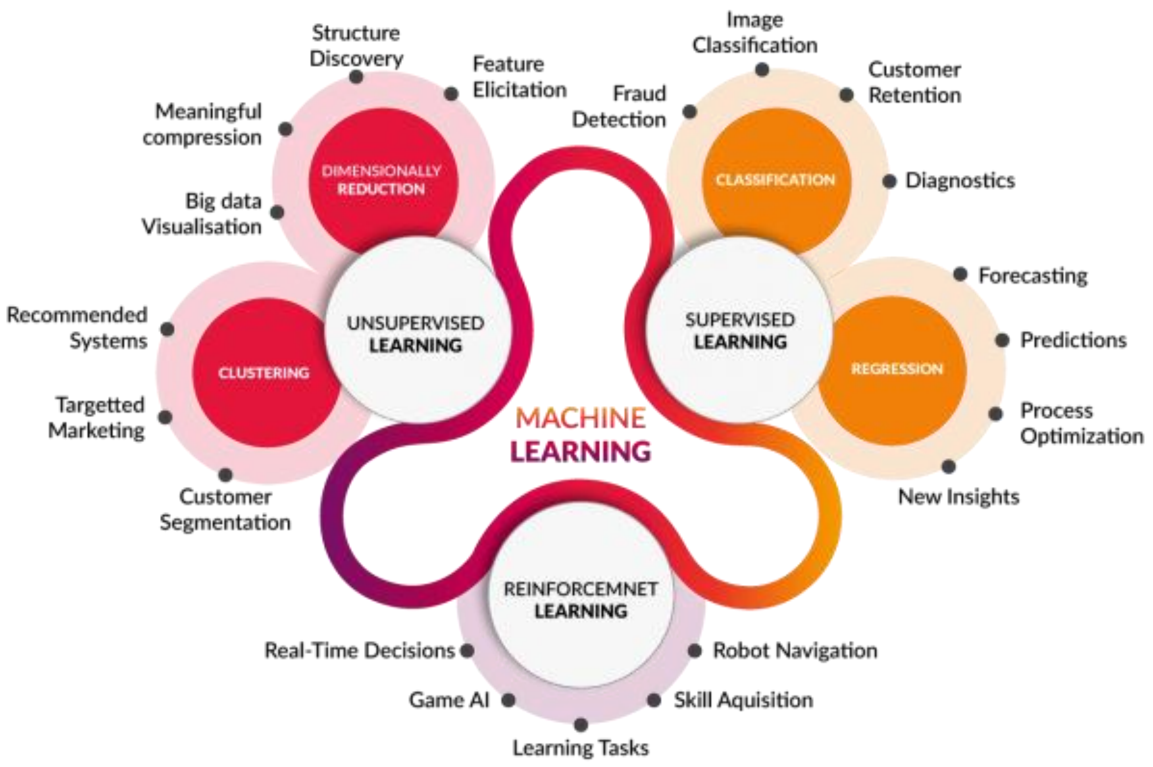


Fig 1.1: Types of ML

In Supervised learning, you train the machine using data which is well "checked." It suggests some data is starting to be marked with the correct answer. It might be stood out from acknowledging which occurs inside seeing a chief or an instructor.

A regulated taking in computation gains from named getting ready data, urges you to foresee results for unforeseen data. Adequately building, scaling, and passing on exact oversaw AI Data science models requires some genuine energy and particular capacity from a gathering of astoundingly skilled data scientists. Furthermore, Data scientists must revamp models to guarantee the encounters given remain substantial until its data changes.

Independent learning is an AI technique, where you don't need to coordinate the model. Taking everything into account, you need to allow the model to manage its own to discover information. It essentially deals with the unlabelled data.

Independent learning estimations license you to perform all the more confounding taking care of tasks diverged from coordinated learning. But, independent learning can be more eccentric differentiated and other typical learning significant learning and stronghold learning methods.

Importance of Machine Learning:

Through cutting edge processing advancements, AI isn't what it resembled previously. It was conceived from design acknowledgment and the hypothesis that PCs can learn without being modified to play out specific assignments. Researchers intrigued by AI needed to check whether PCs could gain from information. The iterative part of AI is significant on the grounds that as models are presented to new information, they can autonomously adjust. They gain from past calculations to deliver dependable, repeatable choices and results.

Simple Programming Model: Machine Learning models can be programmed using Python which is one of the easiest yet the most productive programming languages. Also Python has one of the biggest developer communities who have provided us with huge libraries which make machine learning algorithms easy to implement.

Cost Effective: Machine Learning with Python guarantees that it doesn't consume a gap in your pocket with regards to overseeing humongous measures of data. This has been an issue with ancestor programming which has been cost restrictive. Numerous organizations have needed to erase and downsize data with the end goal to diminish their expenses.

1.2 Handwriting Recognition:

It's the ability of the computer to interpret and recognise handwritten input. It’s sometimes known as HTR(handwritten text recognition). This could be a scanned handwritten document or a photo of a handwritten note, for instance. The growth and proliferation of touch screens add another way to input handwriting.
The goal of handwriting recognition has been around since the 80s — and has suffered from accuracy issues from the beginning. There are two types of handwriting recognition. First is the older of the two, known as offline handwriting recognition. This is where the handwritten input is scanned or photographed and given to the computer.
The second is online, which is where the writing is input through a stylus/touchscreen. This offers the computer more clues about what’s being written. (For instance, stroke direction and pen weight)

1.3 Problem Statement:

There is an abundance of Handwritten Data which is easily available on the internet. This data can be used for training Machine Learning models which can convert handwritten documents into digital form. This could be advantageous over the following domains:

Healthcare:
Handwriting Recognition can be very beneficial in maintaining the Patient records which are handwritten. This will be a gamechanger for the healthcare industry and make it very easy for patients to access their medical records. It will also decrease the chances of people misreading prescriptions and getting wrong medicines.

Building Databases:
As we all know that paper documents can be destroyed by different means such as floods, fire breakouts and termites. So handwriting recognition will help us overcome all these problems by digitally storing the data over the cloud.

Reducing the cost of storing the Data:

Saving a large amount of data in Physical form can be very costly as it requires huge storage spaces and accessing this data can also be very challenging. Converting the same data to digital format is very useful and cost effective. It will also decrease the use of paper for copying and storing this data will be very beneficial for the environment also.

CHAPTER-2
LITERATURE SURVEY

2.1 The Recognition for Handwritten English Letters

A Review Character affirmation is one of the most captivating and testing research areas in the field of Image taking care of. English character affirmation has been extensively amassed over the most recent 50 years. Nowadays different approaches are in use for character affirmation. File affirmation, progressed library, scrutinizing bank store slips, examining postal addresses, removing information from checks, data area, applications for charge cards, clinical inclusion, credits, charge records, etc are application locales of electronic report getting ready. This paper gives a survey of assessment turn out finished for affirmation of physically composed English letters. In Hand formed substance there is no necessity in the creating style. Interpreted letters are difficult to see as a result of various human handwriting styles, assortment in point, size and condition of letters. Various strategies of physically composed character affirmation are analyzed here close by their introduction.[1]

2.2 Feature Extraction For Handwritten Alphabets Recognition System which is diagonal based Using Neural Network:

A detached interpretation in successive request character affirmation systems using a multilayer feed forward neural association is depicted in the paper. Another strategy, called, corner to corner based component extraction is introduced for eliminating the features of the physically composed letters all together. Fifty instructive records, each containing 26 letter sets created by various people, are used for setting up the neural association and 570 particular translated all together characters are used for testing. The proposed affirmation system performs well indeed, yielding more raised degrees of affirmation 6 accuracy diverged from the structures using the conventional level and vertical procedures for incorporating extraction. This system will be sensible for changing over translated reports into essential substance structure and seeing physically composed names.[2]

2.3 Using Neural Network for optical character recognition by Image Preprocessing:

Essential errand of this postulation is to make a hypothetical and pragmatic premise of preprocessing of printed text for optical character acknowledgment utilizing forward-feed neural organizations. Show application was made and its boundaries were set by aftereffects of acknowledged investigations.[3]

2.4 Handwriting Recognition of Historical Documents:

The majority of current offline handwritten text recognition (HTR) algorithms operate at the line level, converting the text-line picture into a series of feature vectors.These characteristics are supplied into an optical model (for example, a recurrent model).In order to distinguish handwritten characters, a neural network was used. Recent work on document-level text identification and localization and combined line segmentation and identification at the paragraph level has yielded encouraging results.The end outcome However, the finest outcomes in terms of recognition are still to be found. Systems that work at the line level are able to do this.

In this model thirteen stack convolutional layers and three bidirectional layers are used having 256 units in each layer. ReLU is also used to have non- linearity after each layer of CNN. Bidirectional LSTM is used with CTC[5] loss function for making the model end to end trainable. This model was found to be very accurate in predicting on the READ dataset and this model also won second place in ICDAR2017 competition.

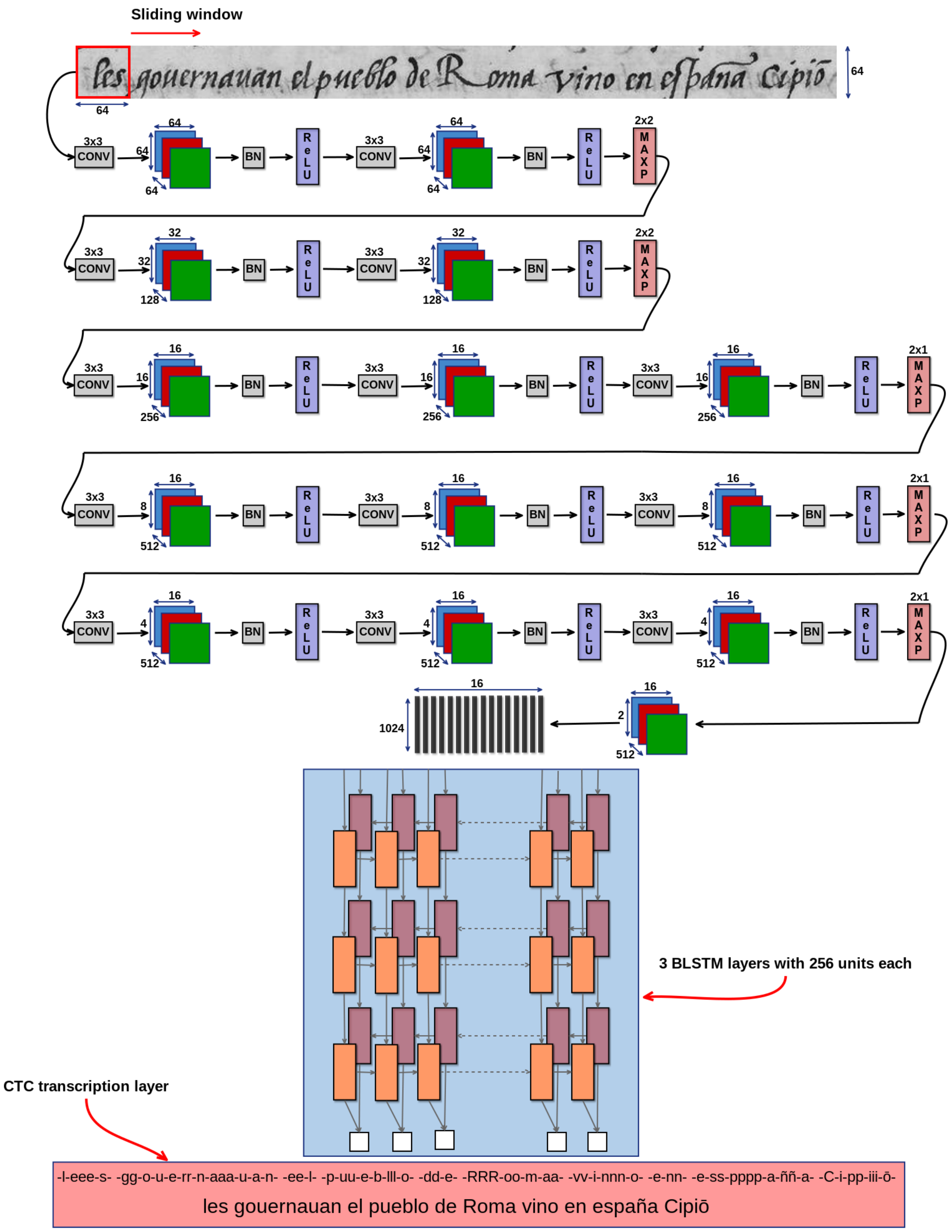


Fig 2.1: RNN model

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 Software Used:

- Jupyter Notebook
- Google Colab
- Kaggle

3.1.1 Requirements of System:

Windows 10, Windows Vista, Linux, Ubuntu.

Jupyter Notebook:

The Jupyter Notebook is an open-source web application that licenses you to make and share reports that contain live code, conditions, observations and record text. Uses include: data cleaning and change, numerical reenactment, quantifiable showing, data recognition, AI, and impressively more. Jupyter Notebooks are a fantastic technique to make and rehash on your Python code for data assessment.

Why a Jupyter NoteBook?

They turn into latex reports really easy. They also turn into slideshows really easy. They also let you run blocks of code really easily.

Some other alternatives: PyCharm, RStudio

Google Collab:

It is made by Google’s Research Department. It's used to execute and write Python code online in a browser. It is very useful and efficient for people who are trying to learn Machine Learning.

3.1.2 Minimum Hardware Requirement:

2GB RAM

Pentium Dual Core 4

Memory Required 20 MB

3.2 Modelling

We are using an incremental model in this project.

What is an Incremental Model?

Incremental Model is a system of programming improvement where prerequisites are broken into various autonomous modules of programming advancement cycle. Steady advancement is done in endeavors from investigation plan, execution, testing/confirmation, support. Each accentuation will encounter the necessities, structure, coding and testing stages. Each resulting appearance of the system adds ability to the last release until the point that all arranged value has been completed .

The structure is placed into age when the essential expansion is passed on. The primary growth is much of the time a middle thing where the crucial necessities are tended to, and worthwhile features are remembered for going with increments. At the point when the middle thing is br0ke somewhere around the cust0mer, there is other plan improvement for the accompanying enlargement.

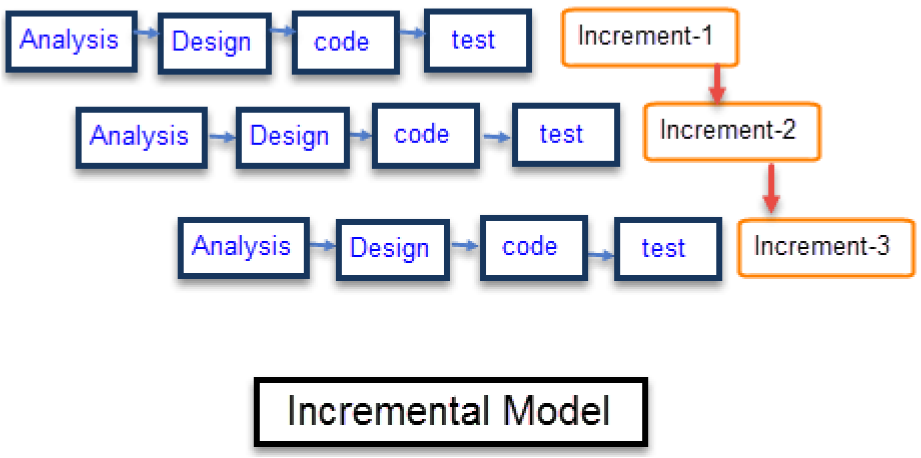


Fig 3.1: Pictorial Representation of Incremental Model

3.3 Designing

Volume of information is extending every day that we can manage business trades, sensible data, pictures, chronicles and various others. In this way, we need a system that will be good for isolating the information open and that can normally make reports, viewpoints or overview of data for better use.
There are three phases in designing a model:

Training Data: It is the data on which the machine learning model learns and trains itself. Usually it is large in comparison to test data.
Validation Dataset: Hyper-parameters of a classifier. It is sometimes also called the development set.

Test Data: It is the data on which testing is done and the model is evaluated on the basis of results obtained from this dataset. Usually it is small in comparison to training data.

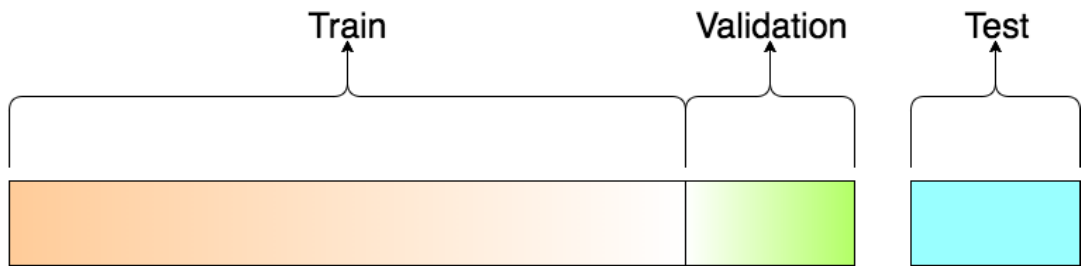


Fig 3.2: Three phases of training data

The 7 Steps of approaching a framework in Machine Learning:

Stage-1: Data Collection.

Stage-2: Data Preparation.

Stage-3: Choose a Model.

Stage-4: Train the Model.

Stage-5: Evaluate the Model.

Stage-6: Parameter Tuning.

Stage-7: Make Prediction

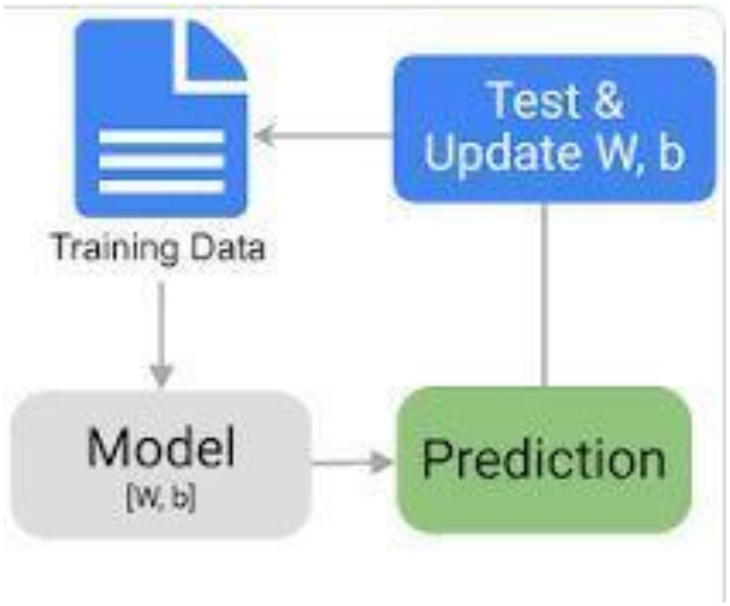


Fig 3.3: Frameworks for Approaching the Machine Learning Process

CHAPTER-4

ALGORITHMS

4.1 Algorithms for Classification Techniques

4.1.1 Logistic Regression Algorithm:

Logistic Regression is an algorithm that is utilized for binary classification and is the most basic algorithm for classification techniques.It uses sigmoid function for predicting the output. The algorithm makes use of the decision boundary to predict the output.

Tools: Python, Jupyter-Notebook.

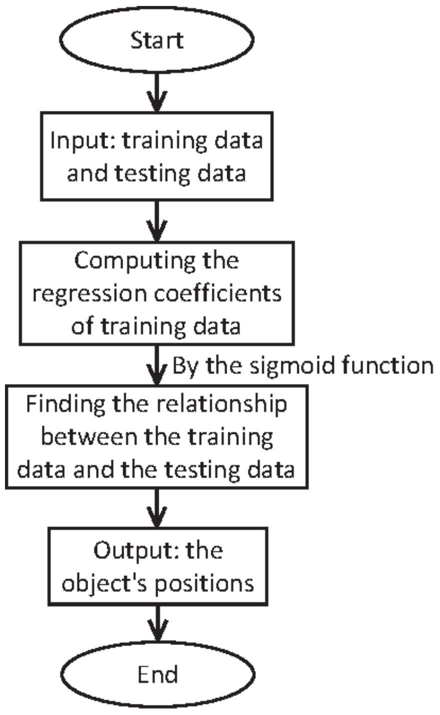

Start
Input: training data and testing data
Computing the regression coefficients of training data
By the sigmoid function
Finding the relationship between the training data and the testing data
Output: the object's positions
End

Fig 4.1: Logistic Regression Algorithm

4.1.2 Artificial Neural Networks :

In ANN (Artificial Neural Networks) we manufacture a type of transient states, which allows the machine to learn in a more refined manner. The objective of this article is to draw out the arrangement of ANN figuring in relating to the value of the psyche. It is truly said that the working of ANN takes its hidden establishments from the neural association living in the human brain. ANN chips away at something suggested as Hidden State. These covered states resemble neurons. All of these covered states is a transient structure which has a probabilistic lead. A framework of such hid state goes probably as a platform between the data and the yield.

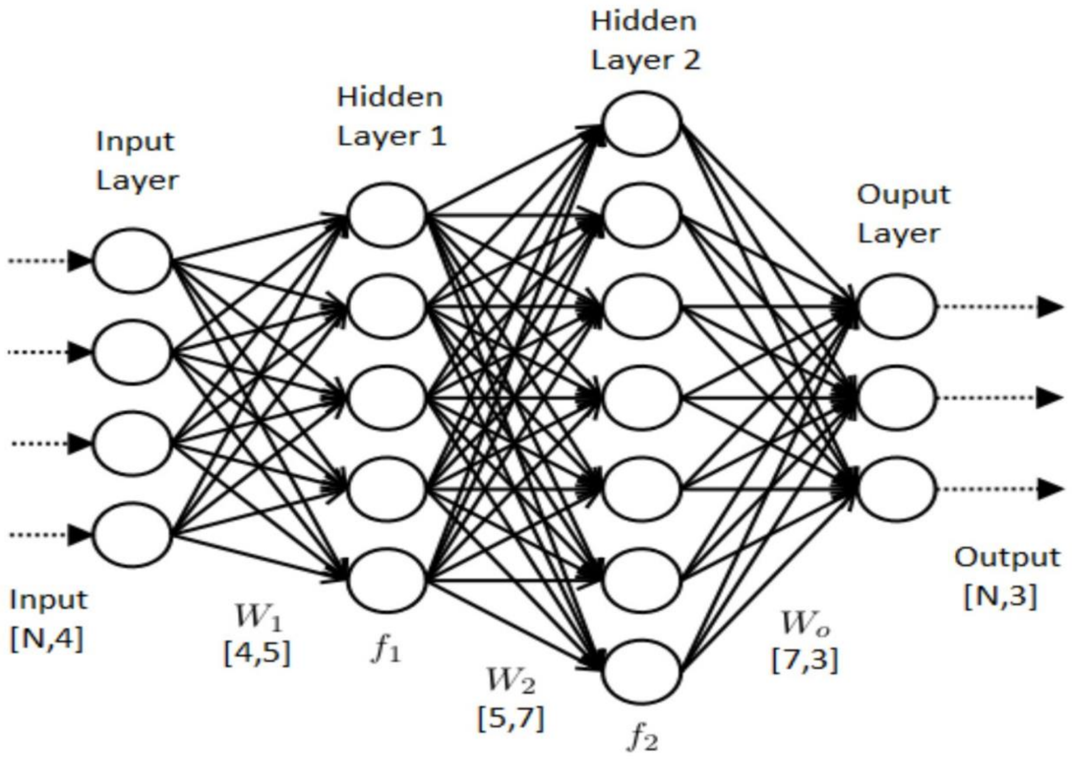


Fig 4.2: Different Layers of Neural Network

4.1.3 Random Forest Algorithm:

The estimation is incredibly standard in various competitions. The end yield of the model takes after a black box and from this time forward should be used wisely. Subjective woods looks like a bootstrapping computation with a Decision tree model. In Random Forest, we create different trees rather than a lone tree in the CART model. To describe another article subject to attributes, each tree gives a portrayal and we express the tree "votes" for that class. The forest area picks the course of action having the most votes (over all the trees in the forested areas) and if there ought to emerge an event of backslide, it takes the ordinary of yields by different trees.[4]

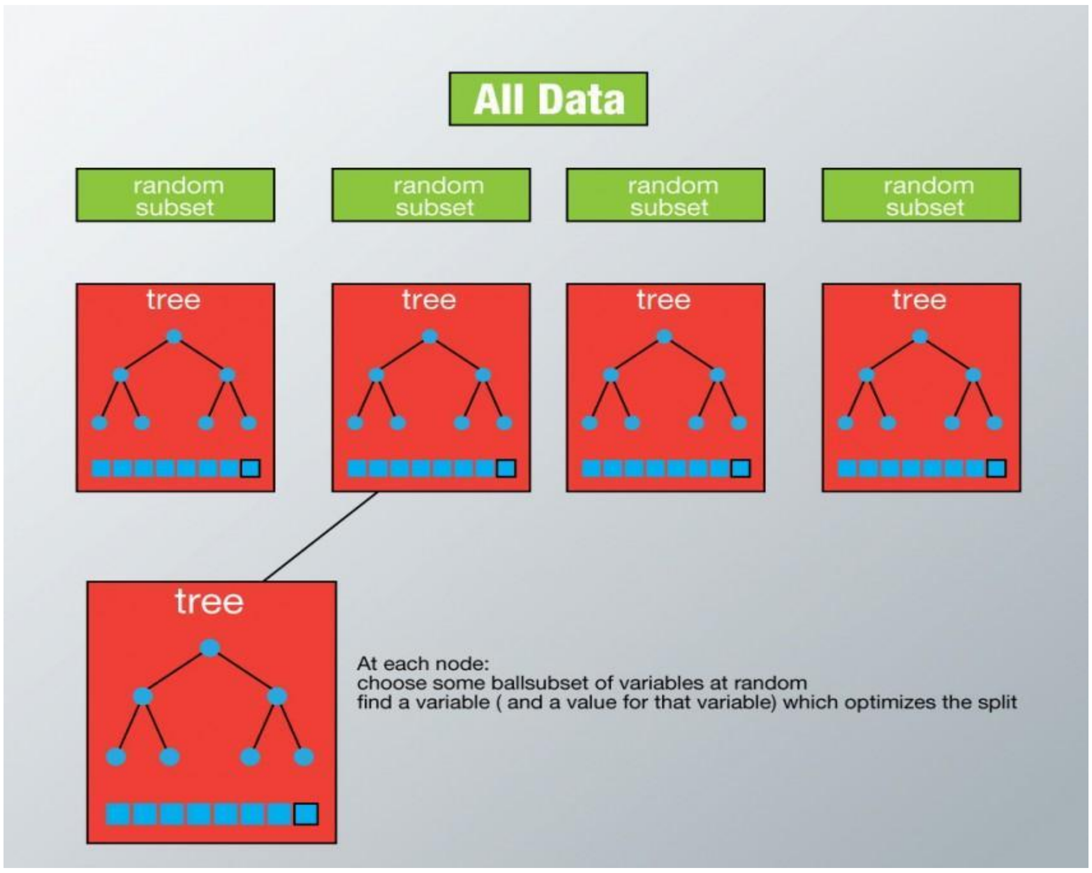

All Data
random subset
random subset
random subset
random subset
tree
tree
tree
tree
tree
At each node:
choose some ballsubset of variables at random
find a variable (and a value for that variable) which optimizes the split

Fig 4.3: Tree Structure

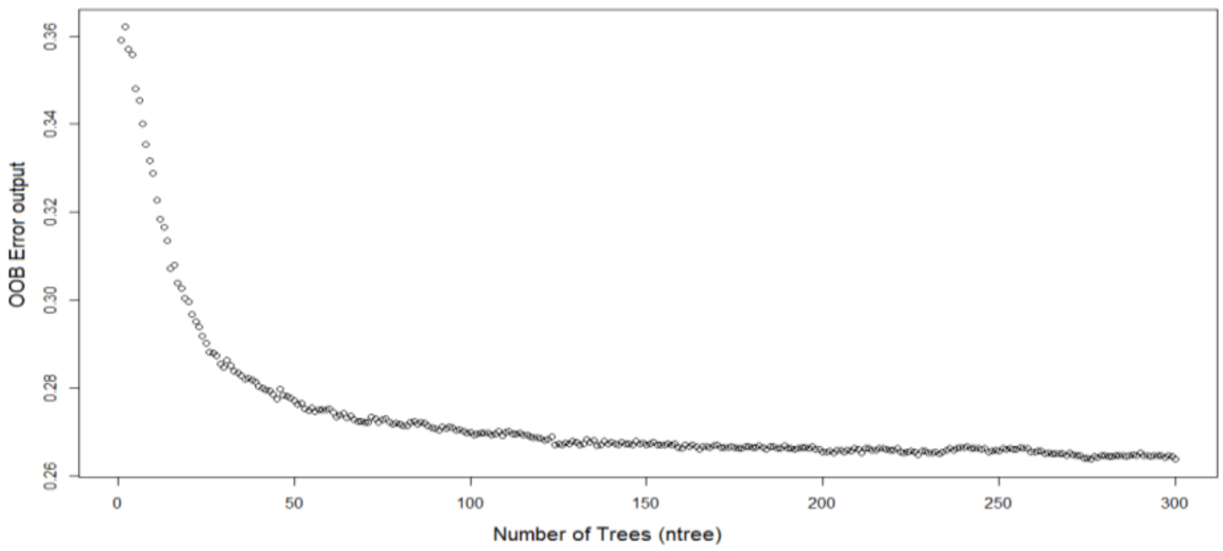


Fig 4.4: Number of trees vs OOB Error rate

Two stages of Random Forest Algorithm:

Making of random forest code.

Code for prediction using random forest.

4.2 Comparing Different Algorithms:

Algorithm	**Technique**	**Scalability**	**Faster**
Logistic Regression	Apply sigmoid function and predict.	Depends on how large the dataset is.	It is very fast for binary classification.
Artificial Neural Networks	Activate the nodes of the next layer and then apply backpropagation.	Can take huge number of features and gets refined as no. of nodes increases	Fast for multiclass classification
Random Forest	Each tree votes for a class and the one with the most vote wins.	Highly scalable and is very efficient for multiclass classification.	Fast for multiclass classification

Fig 4.5: Comparison Table of Algorithm

4.3 Neural Network:

A neural association is a movement of estimations that attempts to see concealed associations in a lot of data through a cycle that duplicates the way where the human brain works. Neural associations can conform to changing data so the association makes the best result without hoping to redesign the yield models.

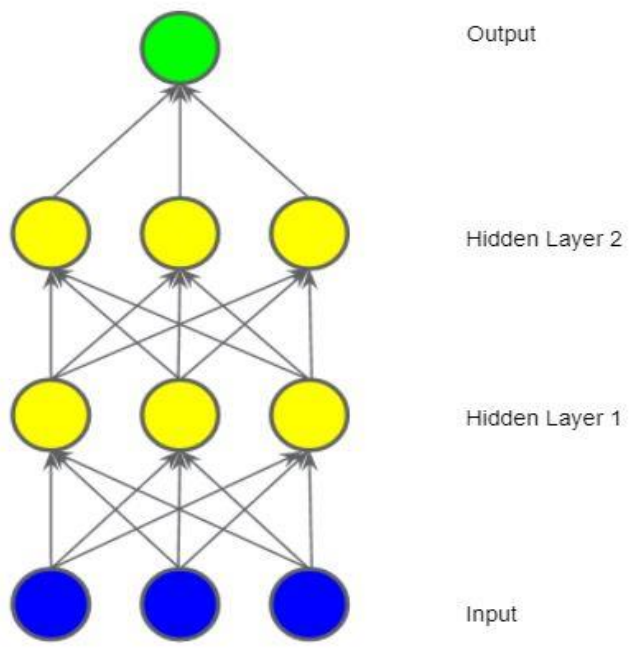


Fig 4.6: Neural Network with two Hidden Layers

4.4 Convolutional Neural Network:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning estimation which can take in a data picture, consign centrality (learnable burdens and tendencies) to various points/objects in the image and have the choice to isolate one from the other. The pre-taking care required in a ConvNet is a ton lower when diverged from other portrayal figurines. While in unrefined methodologies channels are hand-planned, with enough getting ready, ConvNets can get comfortable with these channels/characteristics.

The designing of a ConvNet is like that of the accessibility illustration of Neurons in the Human

Brain and was excited by the relationship of the Visual Cortex. Solitary neurons respond to upgrades simply in a kept territory of the visual field known as the Receptive Field. A collection of such fields cover the entire visual region.

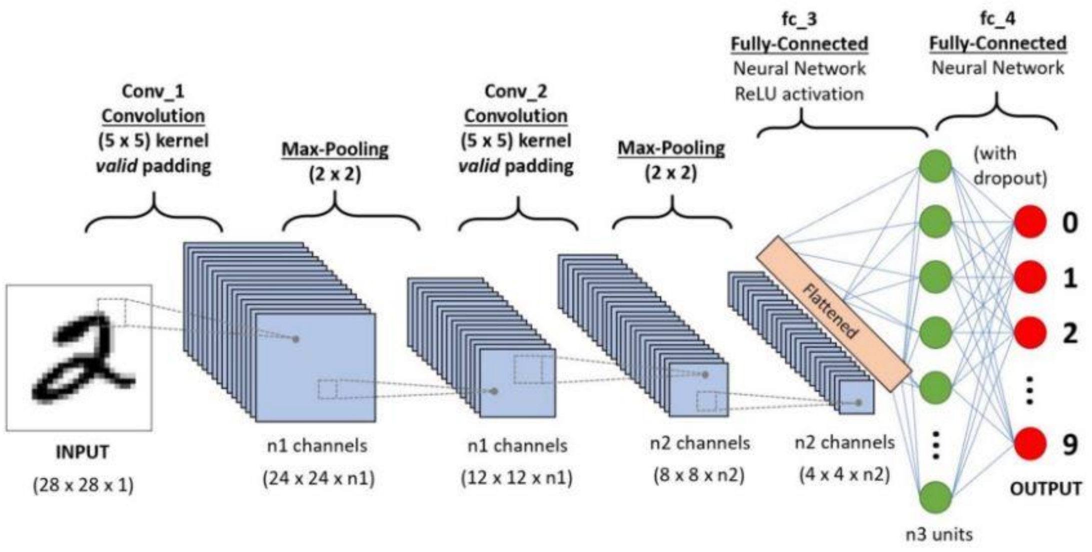


Fig 4.7: Digit Recognition with CNN

4.5 Recurrent Neural Network:

A recurrent neural network is a type of neural network having feedback , the output from the previous step is fed into the current step as an input. Unlike CNNs and ANNs they have a memory element where they can store the sequence of outputs. This feature makes them suitable for applications like speech recognition, handwriting recognition, and also for making predictions.

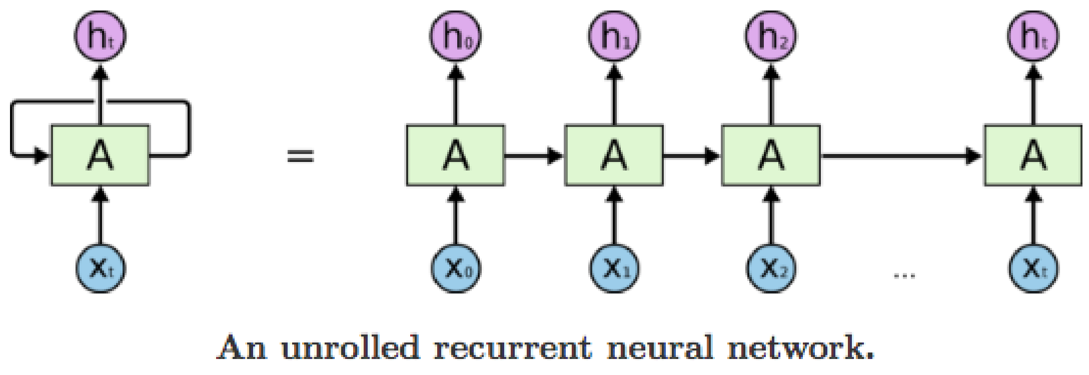


Fig 4.8: RNN

But there are some limitations of RNNs and the 2 major limitations are:

Exploding gradients

Vanishing gradients

4.4.1 Exploding gradients:

In a neural network weights are updated constantly but if an error gradient is being used by the model to update the network. If an error gradient which is assigned a very large value is used to update the weights of the model, this could accumulate and become very huge with every iteration. The problem of exploding gradients makes the network unstable and the loss of the network becomes very high. This problem can easily be solved by gradient clipping and squishing the gradients.

4.4.2 Vanishing gradients:

This is exactly the opposite of exploding gradients, here the values of gradients are very small. So the model stops learning then and skips ahead without learning them. This is very hard to solve as compared to exploding gradients. But this problem can be solved by the use of LSTMs.

4.5 LSTM:

Long short term memory(LSTM) is a special kind of recurrent neural network which is capable of learning long term dependencies. LSTMs make it possible for RNNs to remember inputs for a longer period of time because of the presence of a memory element in LSTMs.

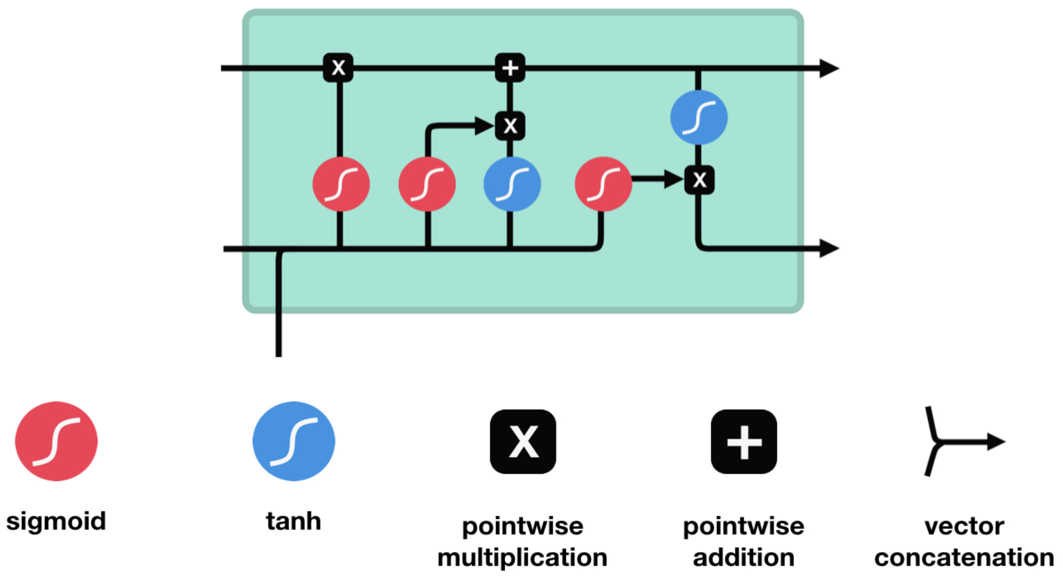


Fig 4.9: LSTM

There are different operations happening inside the LSTM cell, these enable LSTM to remember the useful information and forget the information which will not help the model to improve further.

4.6 CTC loss:

Connectionist Temporal Classification Loss, or CTC Loss is very helpful in tasks like speech and handwriting recognition. It is help full where there are pauses or repeating words or alphabets. Between a continuous (unsegmented) time series and a target sequence, it estimates a loss. It does this by accumulating the probabilities of alternative input-target alignments, yielding a loss value that is differentiable with respect to each input node. The input to target alignment is expected to be many-to-one.

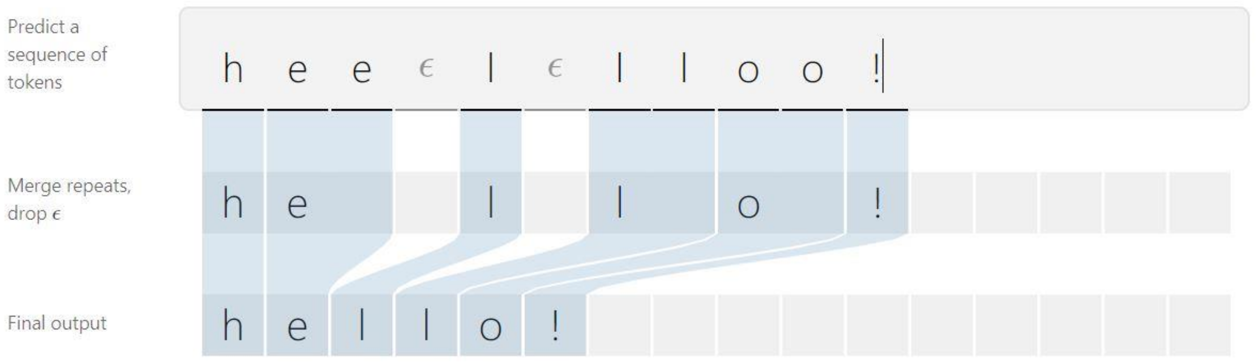


Fig 4.10: CTC functioning

CHAPTER-5

IMPLEMENTATION

5.1 Tools and Techniques:

We have used Python for implementing our project.

Why Python?

Python is very easy to understand and is a beginner friendly high level language. Python's simplicity allows us to write reliable systems. Python is more mechanical and models are quickly trained for machine learning.

Datasets Used:

The MNIST database of physically composed digits, open from this page, has an arrangement set of 60,000 models, and a test set of 10,000 models. It is a subset of a greater set open from NIST. The digits have been size-normalized and centered in a fixed-size picture.

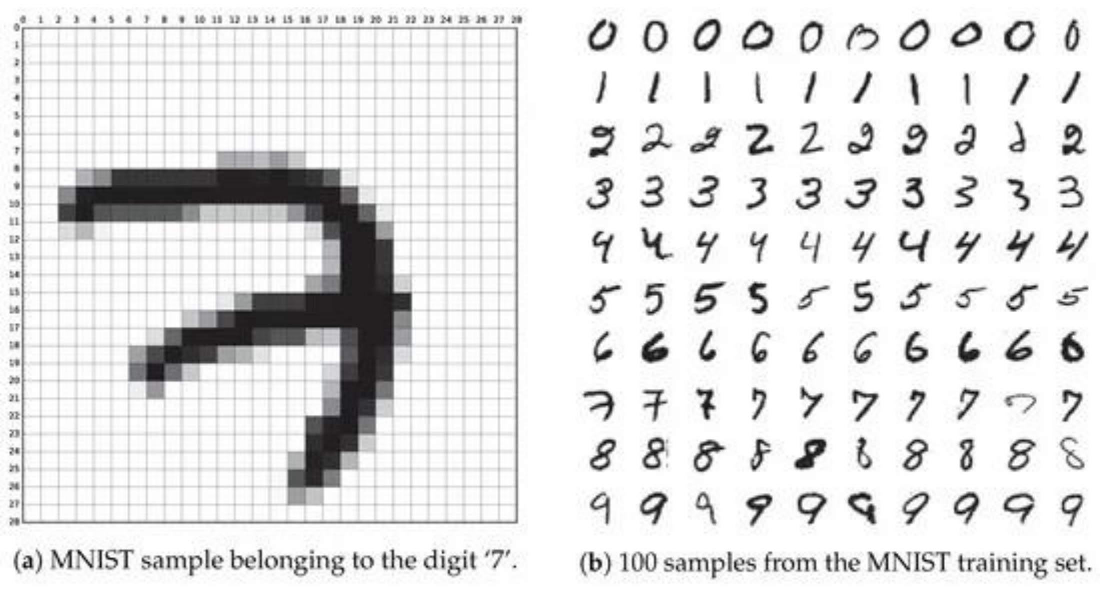

(a) MNIST sample belonging to the digit '7'.
(b) 100 samples from the MNIST training set.

Fig 5.1: MNIST Dataset

Another dataset used in this project is “Handwriting Recognition” dataset. This dataset is a compilation of about 4 Lakh handwritten names. There are 2.06 Lakh first names and 2.07 Lakh surnames in total in this dataset. The data is further divided into a training set of 3.36 Lakh , testing set of 41.83 Thousand and validation set of 41.38 Thousand respectively.

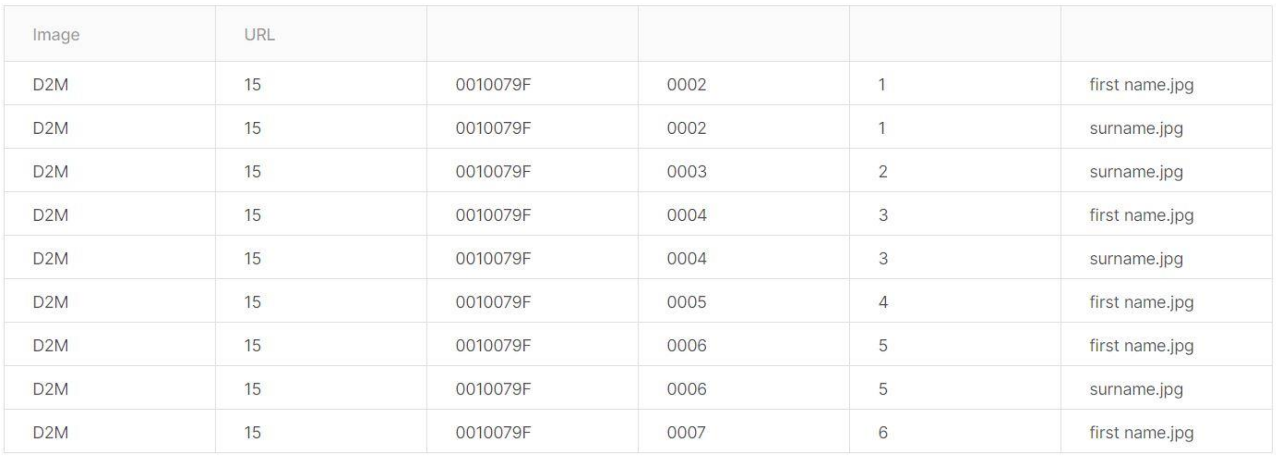


Image	URL				
D2M	15	0010079F	0002	1	first name.jpg
D2M	15	0010079F	0002	1	surname.jpg
D2M	15	0010079F	0003	2	surname.jpg
D2M	15	0010079F	0004	3	first name.jpg
D2M	15	0010079F	0004	3	surname.jpg
D2M	15	0010079F	0005	4	first name.jpg
D2M	15	0010079F	0006	5	first name.jpg
D2M	15	0010079F	0006	5	surname.jpg
D2M	15	0010079F	0007	6	first name.jpg

5.2 Block Diagram

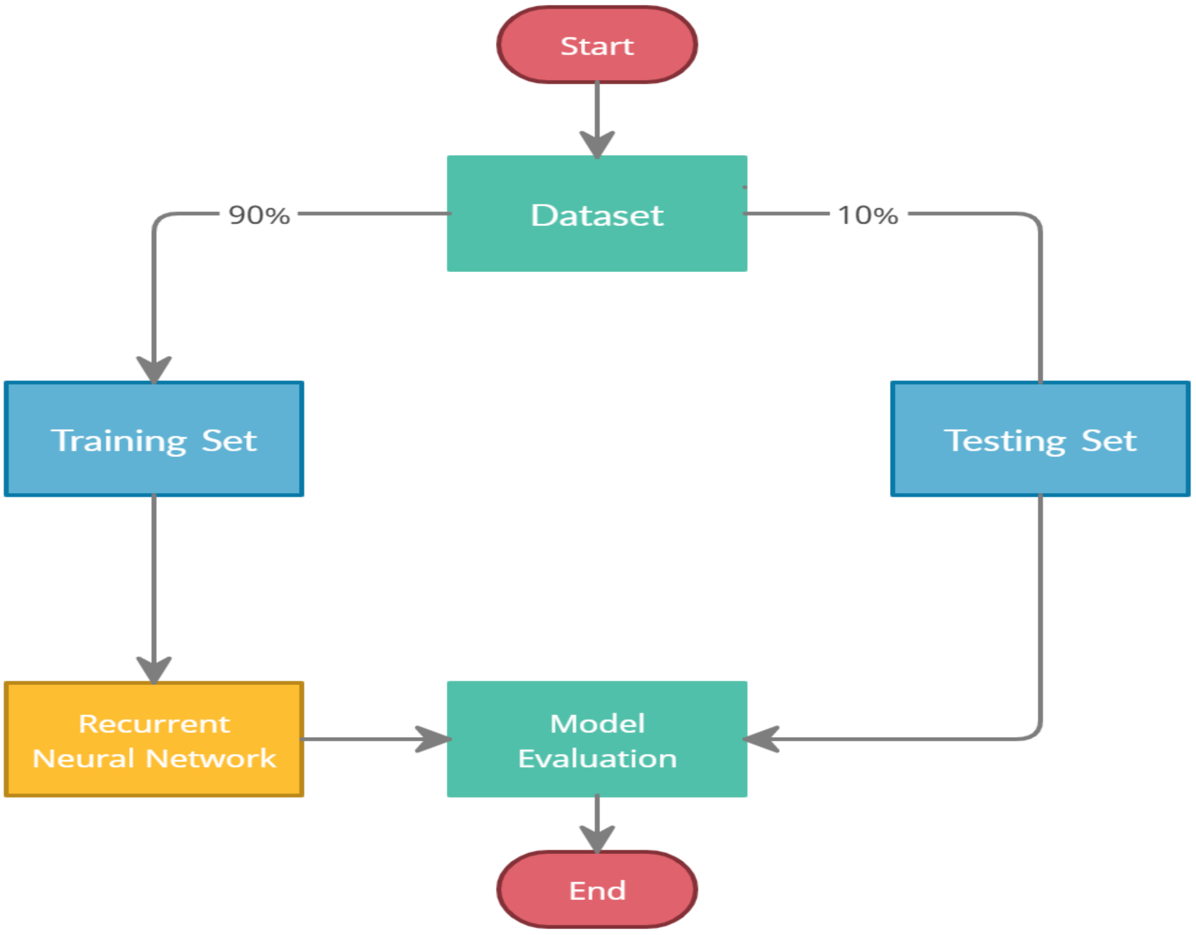


Fig 5.2: Block Diagram

The data from the dataset is divided into training and validation sets which are in the ratio of 10:1.

Training set contains 38000 images whereas the testing set contains 3800 images.

The RNN is trained on the images from the training set and then its accuracy is checked by using the model on the validation set.

5.3 Libraries Used:

We have used following libraries in the implementation code:

- Pandas
- Numpy

- Matplotlib
- Tensor-Flow
- Keras

5.4 Specification of virtual machine used:

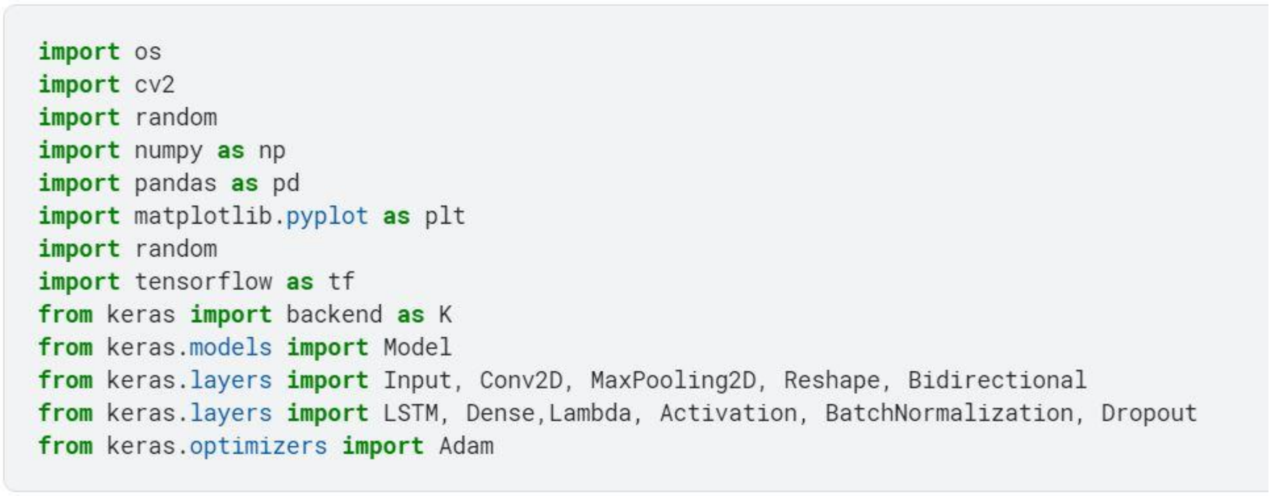
CPU : Intel(R) Xeon(R) CPU @ 2.20GHz

GPU : Tesla P100 with 16GB VRAM

RAM : 14 GB DDR4

HDD : 73 GB

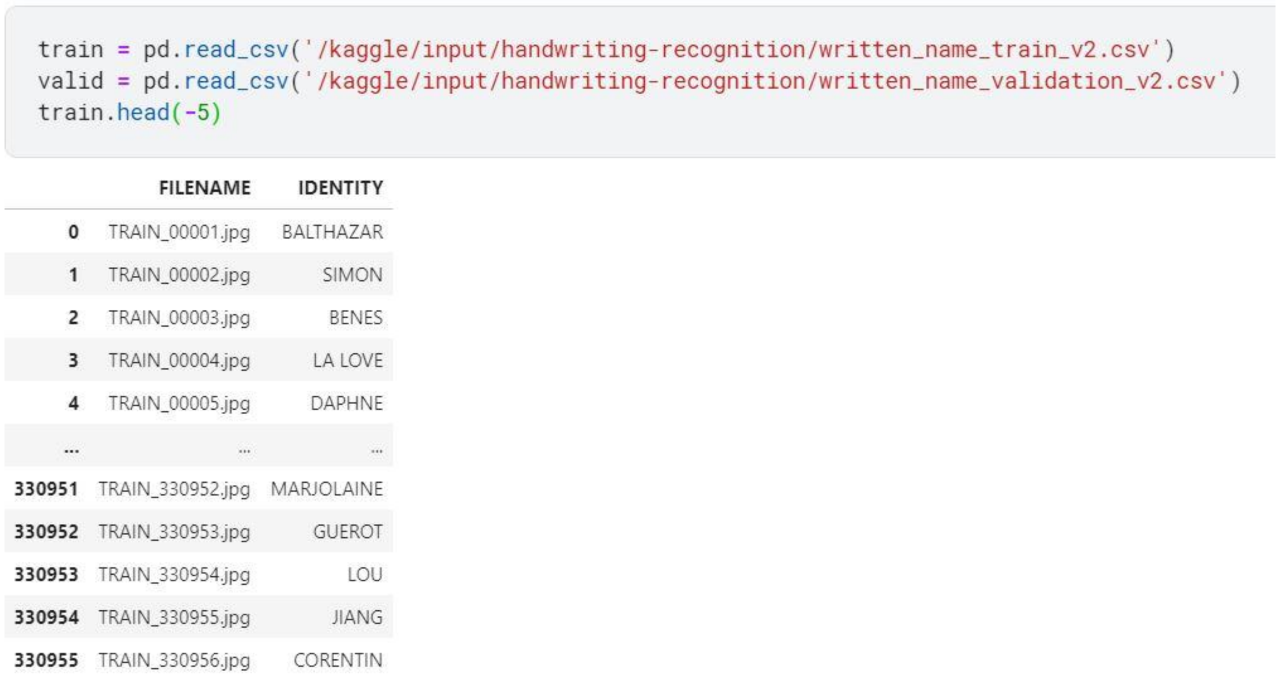
5.5 Code:



```
import os
import cv2
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import tensorflow as tf
from keras import backend as K
from keras.models import Model
from keras.layers import Input, Conv2D, MaxPooling2D, Reshape, Bidirectional
from keras.layers import LSTM, Dense,Lambda, Activation, BatchNormalization, Dropout
from keras.optimizers import Adam
```

Fig 5.3: Importing libraries

Firstly we import all the libraries that are necessary for this model like pandas, keras, matplotlib, numpy, etc. These libraries are very important in this project.

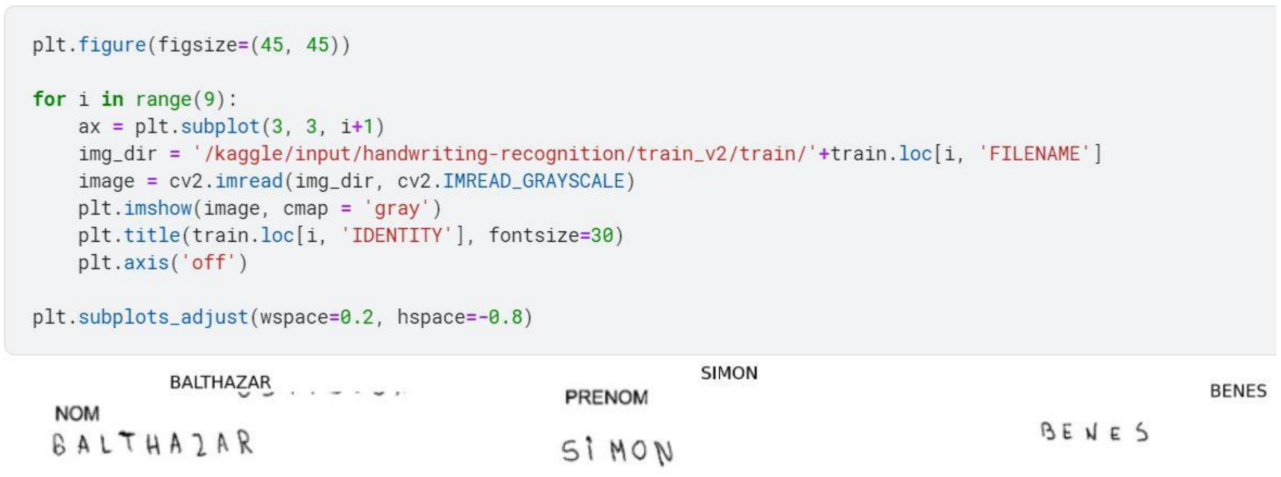


```
train = pd.read_csv('/kaggle/input/handwriting-recognition/written_name_train_v2.csv')
valid = pd.read_csv('/kaggle/input/handwriting-recognition/written_name_validation_v2.csv')
train.head(-5)
```

	FILENAME	IDENTITY
0	TRAIN_00001.jpg	BALTHAZAR
1	TRAIN_00002.jpg	SIMON
2	TRAIN_00003.jpg	BENES
3	TRAIN_00004.jpg	LA LOVE
4	TRAIN_00005.jpg	DAPHNE
...	...	...
330951	TRAIN_330952.jpg	MARJOLAINE
330952	TRAIN_330953.jpg	GUEROT
330953	TRAIN_330954.jpg	LOU
330954	TRAIN_330955.jpg	JIANG
330955	TRAIN_330956.jpg	CORENTIN

Fig 5.4: Loading dataset

In this step we are loading the data from our selected dataset. Train and valid are loading data from training and validation files of the dataset. Pandas method called head is used to view first and last 5 rows from the loaded dataset.



```
plt.figure(figsize=(45, 45))

for i in range(9):
    ax = plt.subplot(3, 3, i+1)
    img_dir = '/kaggle/input/handwriting-recognition/train_v2/train/'+train.loc[i, 'FILENAME']
    image = cv2.imread(img_dir, cv2.IMREAD_GRAYSCALE)
    plt.imshow(image, cmap = 'gray')
    plt.title(train.loc[i, 'IDENTITY'], fontsize=30)
    plt.axis('off')

plt.subplots_adjust(wspace=0.2, hspace=-0.8)
```

Fig 5.5: Viewing data

Here we are using matplotlib to view the images and the labels of theimages. This is done to check if the images and labels match or not. As we can see the labels and words in the images are correct so we can now move forward.

```
print("Number of NaNs in train set      : ", train['IDENTITY'].isnull().sum())
print("Number of NaNs in validation set : ", valid['IDENTITY'].isnull().sum())
```

```
Number of NaNs in train set      :  565
Number of NaNs in validation set :  78
```

```
train.dropna(axis=0, inplace=True)
valid.dropna(axis=0, inplace=True)
print("Number of NaNs in train set      : ", train['IDENTITY'].isnull().sum())
print("Number of NaNs in validation set : ", valid['IDENTITY'].isnull().sum())
```

```
Number of NaNs in train set      :  0
Number of NaNs in validation set :  0
```

Fig 5.6: Cleaning data

In this step we are checking that if there are any cells that are having missing data, these cells are filled with NaN. Then we are using dropna to remove these rows from our training and validation sets.

```
unreadable = train[train['IDENTITY'] == 'UNREADABLE']
unreadable.reset_index(inplace = True, drop=True)

plt.figure(figsize=(15, 10))

for i in range(6):
    ax = plt.subplot(2, 3, i+1)
    img_dir = '/kaggle/input/handwriting-recognition/train_v2/train/'+unreadable.loc[i, 'FILENAME']
    image = cv2.imread(img_dir, cv2.IMREAD_GRAYSCALE)
    plt.imshow(image, cmap = 'gray')
    plt.title(unreadable.loc[i, 'IDENTITY'], fontsize=12)
    plt.axis('off')

plt.subplots_adjust(wspace=0.2, hspace=-0.8)
```

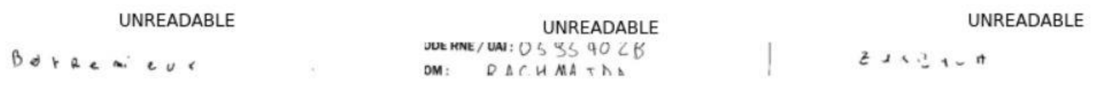
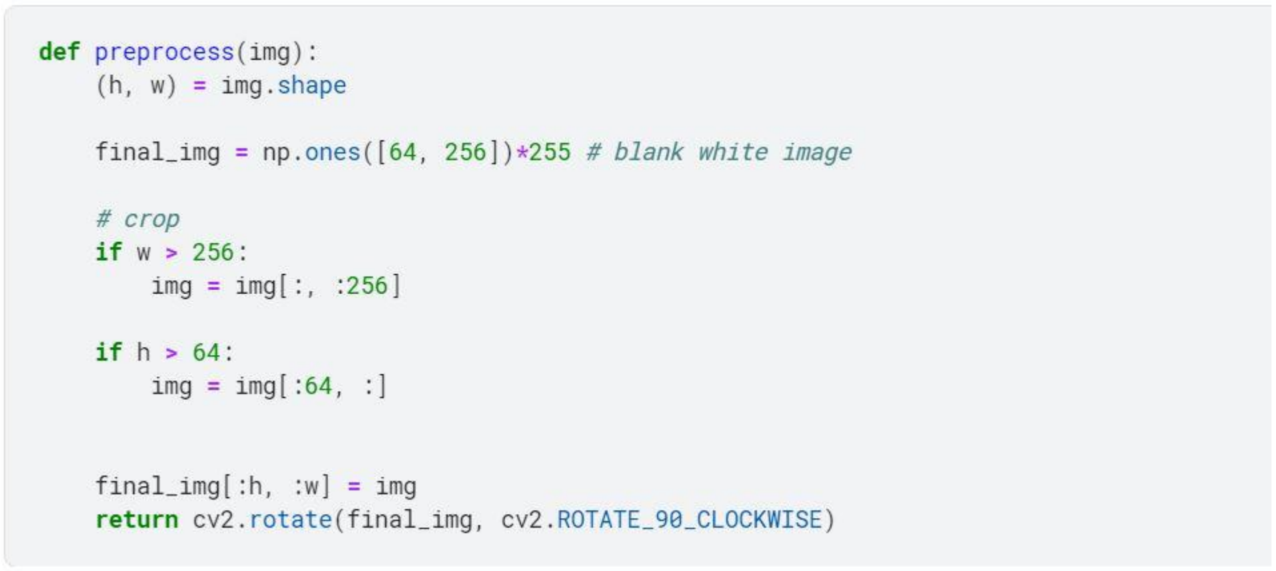


Fig 5.7: Removing unreadable images

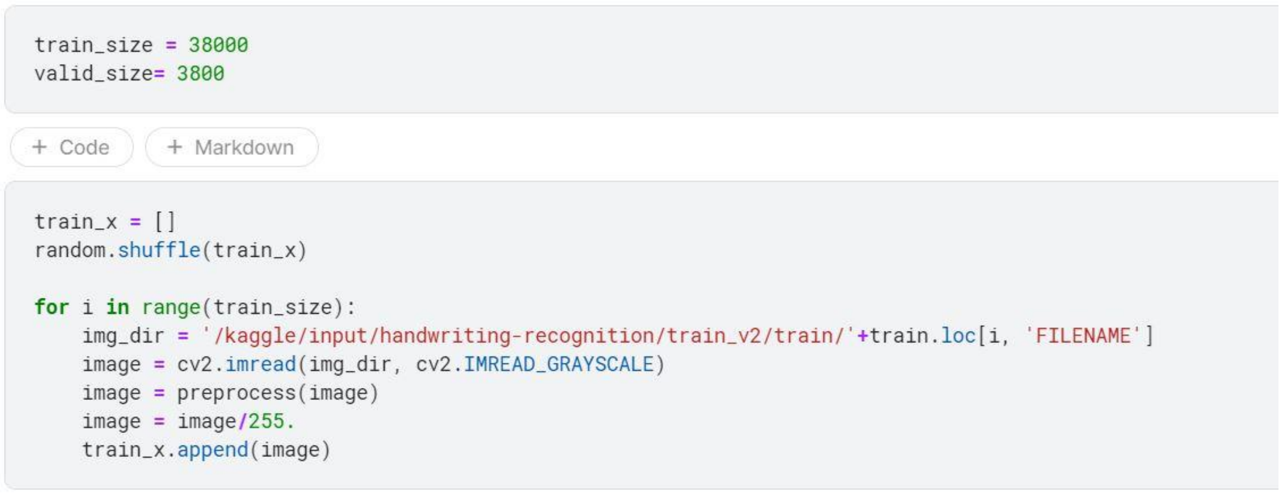
Some images in this dataset are also labeled as “UNREADABLE”, so we are finding these images and are removinging them from our training data and validation data as this can hinder the performance of our model.



```python
def preprocess(img):
    (h, w) = img.shape

    final_img = np.ones([64, 256])*255 # blank white image

    # crop
    if w > 256:
        img = img[:, :256]

    if h > 64:
        img = img[:64, :]


    final_img[:h, :w] = img
    return cv2.rotate(final_img, cv2.ROTATE_90_CLOCKWISE)
```

Fig 5.8: Preprocessing the images

All the images of this data set are not of uniform dimensions so in this step we are trying to make the size of all the images the same. We are doing this by cropping the image if the image is larger than the required dimensions and if the image is smaller then we are padding it to make all the images of uniform size.



```python
train_size = 38000
valid_size= 3800
```

```python
train_x = []
random.shuffle(train_x)

for i in range(train_size):
    img_dir = '/kaggle/input/handwriting-recognition/train_v2/train/'+train.loc[i, 'FILENAME']
    image = cv2.imread(img_dir, cv2.IMREAD_GRAYSCALE)
    image = preprocess(image)
    image = image/255.
    train_x.append(image)
```

Fig 5.9: Loading training images

In this step we are defining the size of the training and the validation data. We are also loading all the images from the data into memory and using the preprocess function built in the last step on the loaded images.

```python
valid_x = []
random.shuffle(valid_x)


for i in range(valid_size):
    img_dir = '/kaggle/input/handwriting-recognition/validation_v2/validation/'+valid.loc[i, 'FILENAME']
    image = cv2.imread(img_dir, cv2.IMREAD_GRAYSCALE)
    image = preprocess(image)
    image = image/255.
    valid_x.append(image)
```

```python
train_x = np.array(train_x).reshape(-1, 256, 64, 1)
valid_x = np.array(valid_x).reshape(-1, 256, 64, 1)
```

Fig 5.10: Loading validation images

Here we are loading validation images in the same fashion. We are also reshaping all the images of the train and validation set and normalizing them in the range of 0-1.

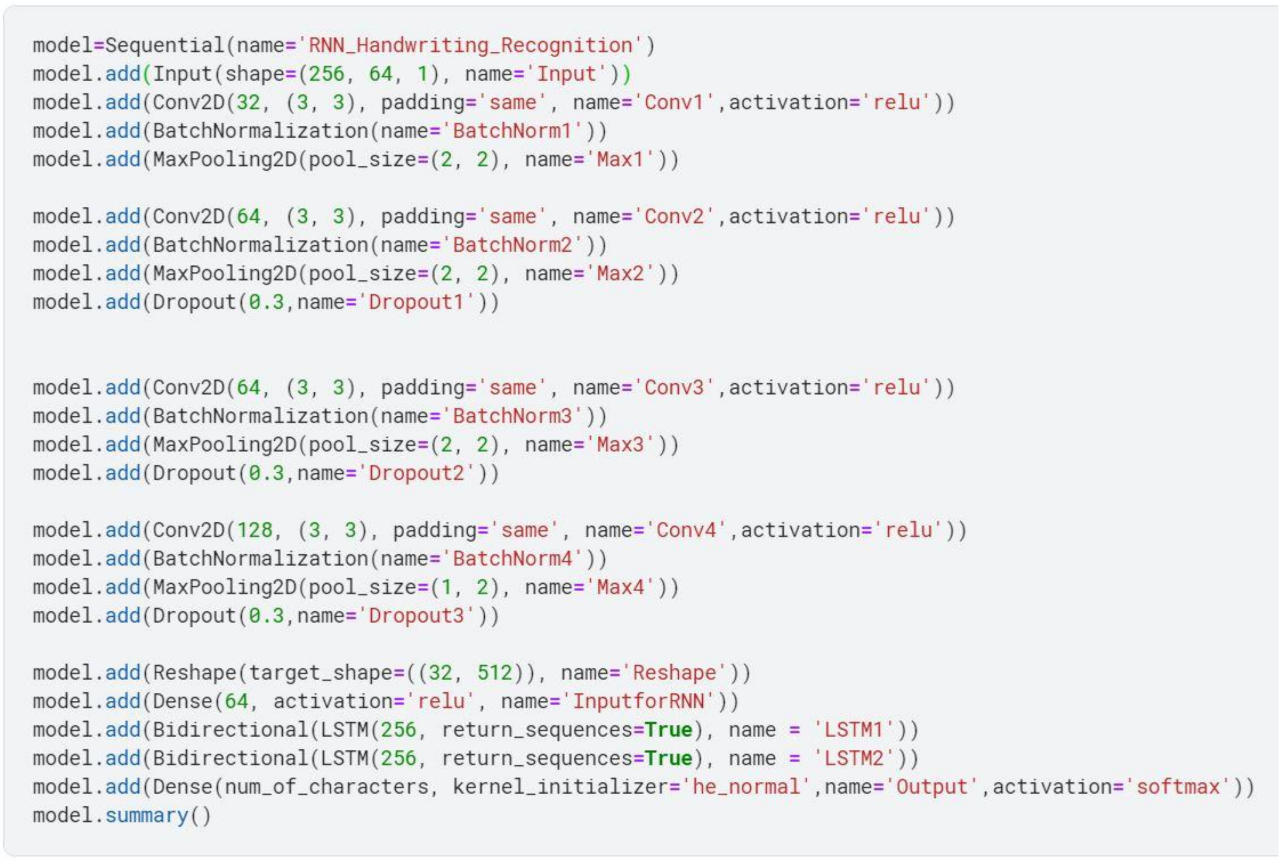
```python
alphabets = u"ABCDEFGHIJKLMNOPQRSTUVWXYZ-' "
max_str_len = 24 # max length of input labels
num_of_characters = len(alphabets) + 1 # +1 for ctc pseudo blank
num_of_timestamps = 64 # max length of predicted labels

def label_to_num(label):
    label_num = []
    for ch in label:
        label_num.append(alphabets.find(ch))
    return np.array(label_num)

def num_to_label(num):
    ret = ""
    for ch in num:
        if ch == -1:  # CTC Blank
            break
        else:
            ret+=alphabets[ch]
    return ret
```

```python
name = 'HITESH'
print(name, '\n',label_to_num(name))
```

```
HITESH
 [ 7  8 19  4 18  7]
```

Fig 5.11: Preparing labels

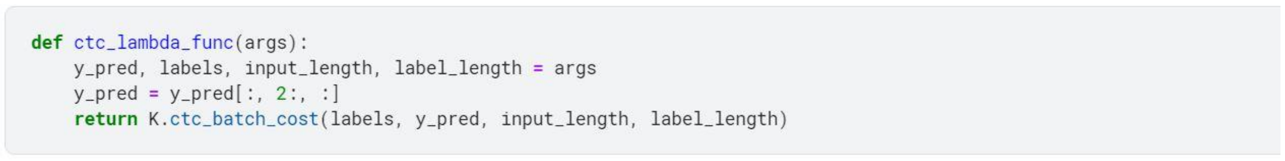
Here we are making our labels suitable for CTC function, all the labels are converted to numbers.In the example alphabets of the name “HITESH” are converted to [7 8 19 4 18 7] , H corresponding to 7, I to 8 and so on.



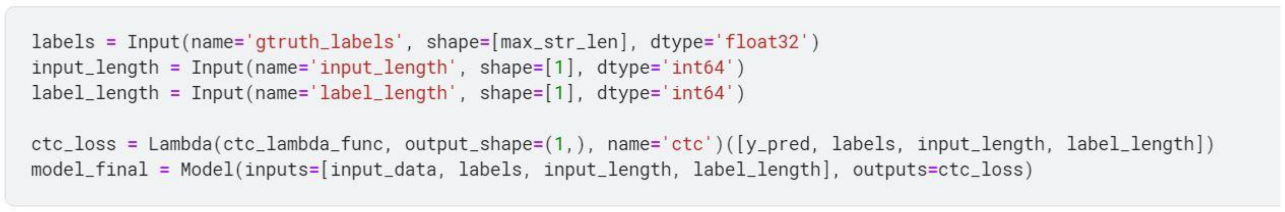
```
model=Sequential(name='RNN_Handwriting_Recognition')
model.add(Input(shape=(256, 64, 1), name='Input'))
model.add(Conv2D(32, (3, 3), padding='same', name='Conv1',activation='relu'))
model.add(BatchNormalization(name='BatchNorm1'))
model.add(MaxPooling2D(pool_size=(2, 2), name='Max1'))

model.add(Conv2D(64, (3, 3), padding='same', name='Conv2',activation='relu'))
model.add(BatchNormalization(name='BatchNorm2'))
model.add(MaxPooling2D(pool_size=(2, 2), name='Max2'))
model.add(Dropout(0.3,name='Dropout1'))


model.add(Conv2D(64, (3, 3), padding='same', name='Conv3',activation='relu'))
model.add(BatchNormalization(name='BatchNorm3'))
model.add(MaxPooling2D(pool_size=(2, 2), name='Max3'))
model.add(Dropout(0.3,name='Dropout2'))

model.add(Conv2D(128, (3, 3), padding='same', name='Conv4',activation='relu'))
model.add(BatchNormalization(name='BatchNorm4'))
model.add(MaxPooling2D(pool_size=(1, 2), name='Max4'))
model.add(Dropout(0.3,name='Dropout3'))

model.add(Reshape(target_shape=((32, 512)), name='Reshape'))
model.add(Dense(64, activation='relu', name='InputforRNN'))
model.add(Bidirectional(LSTM(256, return_sequences=True), name = 'LSTM1'))
model.add(Bidirectional(LSTM(256, return_sequences=True), name = 'LSTM2'))
model.add(Dense(num_of_characters, kernel_initializer='he_normal',name='Output',activation='softmax'))
model.summary()
```

Fig 5.12: RNN Model

In this step the whole RNN model has been built. I have used 4 Covnets with max pooling and batch normalization so that our model can learn the features. The output is reshaped so that it can be fed into the LSTM RNN which will be learning the words. At last classification is done in the last dense layer.



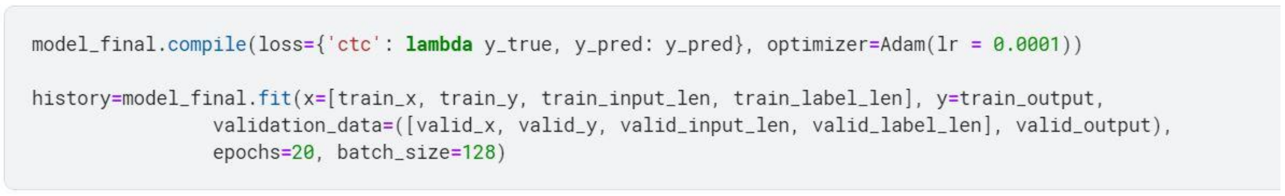
```python
def ctc_lambda_func(args):
    y_pred, labels, input_length, label_length = args
    y_pred = y_pred[:, 2:, :]
    return K.ctc_batch_cost(labels, y_pred, input_length, label_length)
```



```python
labels = Input(name='gtruth_labels', shape=[max_str_len], dtype='float32')
input_length = Input(name='input_length', shape=[1], dtype='int64')
label_length = Input(name='label_length', shape=[1], dtype='int64')

ctc_loss = Lambda(ctc_lambda_func, output_shape=(1,), name='ctc')([y_pred, labels, input_length, label_length])
model_final = Model(inputs=[input_data, labels, input_length, label_length], outputs=ctc_loss)
```

Fig 5.13: CTC loss

Here we have built the CTC loss function so that we can train our model using CTC loss.CTC is used when there is a use of LSTM layer. CTC does a very good job in applications like speech recognition and handwriting recognition.



```python
model_final.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer=Adam(lr = 0.0001))

history=model_final.fit(x=[train_x, train_y, train_input_len, train_label_len], y=train_output,
                validation_data=([valid_x, valid_y, valid_input_len, valid_label_len], valid_output),
                epochs=20, batch_size=128)
```

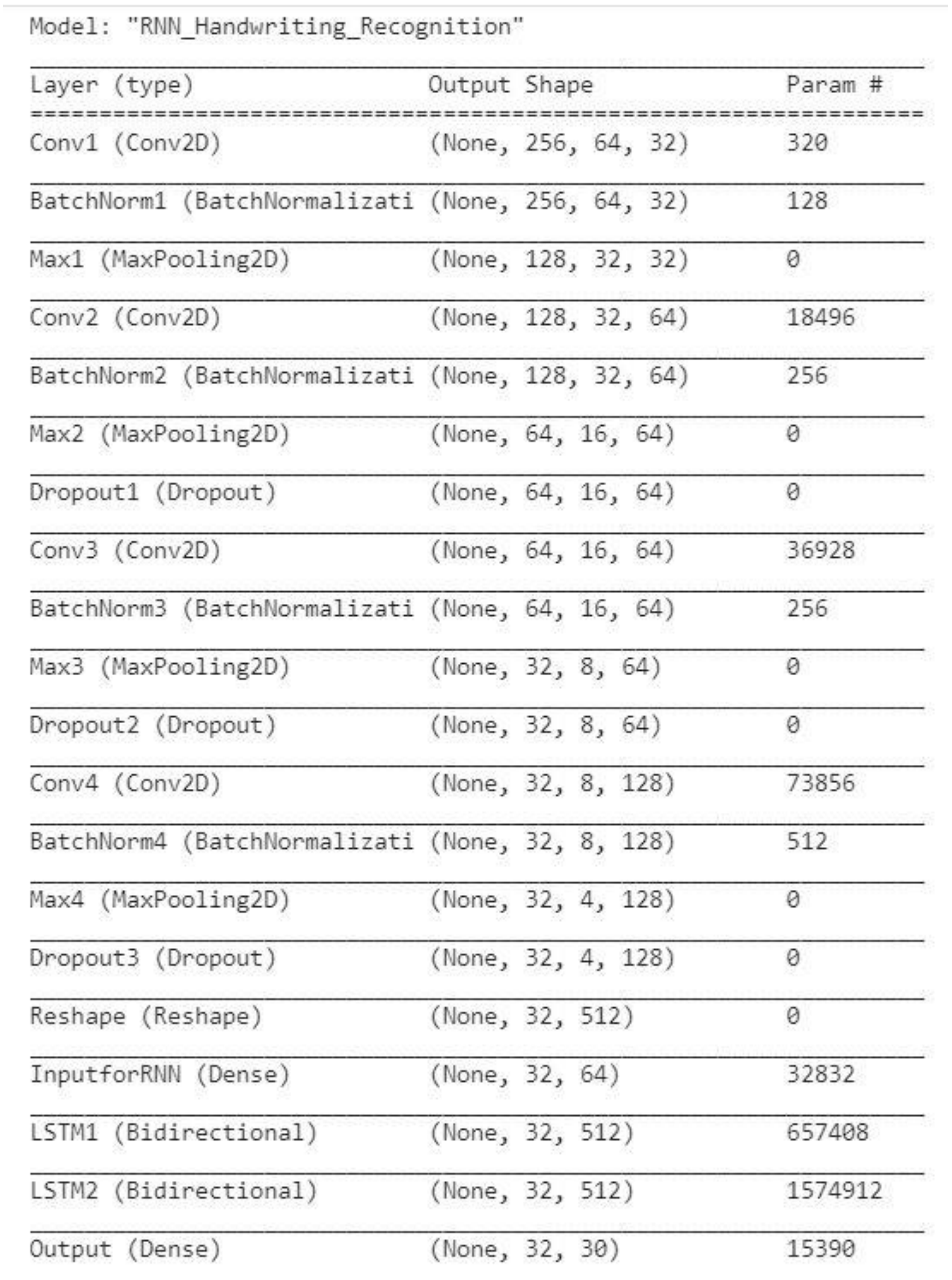
Fig 5.14: Training the model

And finally we are training the model so that we can find out how well this model is fitting the data. We have used the lambda function to call the CTC function we built in the last step.

The batch size is set to 128 which is chosen arbitrarily and the model will be trained for 20 epochs before giving any result.

Adam optimizer is used in this model with a learning rate of 0.0001.

5.6 RNN Model Implemented:



```
Model: "RNN_Handwriting_Recognition"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Conv1 (Conv2D)               (None, 256, 64, 32)       320
_________________________________________________________________
BatchNorm1 (BatchNormalizati (None, 256, 64, 32)       128
_________________________________________________________________
Max1 (MaxPooling2D)          (None, 128, 32, 32)       0
_________________________________________________________________
Conv2 (Conv2D)               (None, 128, 32, 64)       18496
_________________________________________________________________
BatchNorm2 (BatchNormalizati (None, 128, 32, 64)       256
_________________________________________________________________
Max2 (MaxPooling2D)          (None, 64, 16, 64)        0
_________________________________________________________________
Dropout1 (Dropout)           (None, 64, 16, 64)        0
_________________________________________________________________
Conv3 (Conv2D)               (None, 64, 16, 64)        36928
_________________________________________________________________
BatchNorm3 (BatchNormalizati (None, 64, 16, 64)        256
_________________________________________________________________
Max3 (MaxPooling2D)          (None, 32, 8, 64)         0
_________________________________________________________________
Dropout2 (Dropout)           (None, 32, 8, 64)         0
_________________________________________________________________
Conv4 (Conv2D)               (None, 32, 8, 128)        73856
_________________________________________________________________
BatchNorm4 (BatchNormalizati (None, 32, 8, 128)        512
_________________________________________________________________
Max4 (MaxPooling2D)          (None, 32, 4, 128)        0
_________________________________________________________________
Dropout3 (Dropout)           (None, 32, 4, 128)        0
_________________________________________________________________
Reshape (Reshape)            (None, 32, 512)           0
_________________________________________________________________
InputforRNN (Dense)          (None, 32, 64)            32832
_________________________________________________________________
LSTM1 (Bidirectional)        (None, 32, 512)           657408
_________________________________________________________________
LSTM2 (Bidirectional)        (None, 32, 512)           1574912
_________________________________________________________________
Output (Dense)               (None, 32, 30)            15390
```

Fig 5.15: RNN Model

In this model there are 4 Convolution layers which are used for feature extraction from the image.

There are 4 Max Pooling layers to get only prominent features.

3 dropout layers are also there so that the model does generalize and overfit the data.

The reshape layer is used to reshape the data from the Covnets and make it suitable to feed it into the RNN.

The RNN layer uses bidirectional LSTM so that words can be predicted accurately.

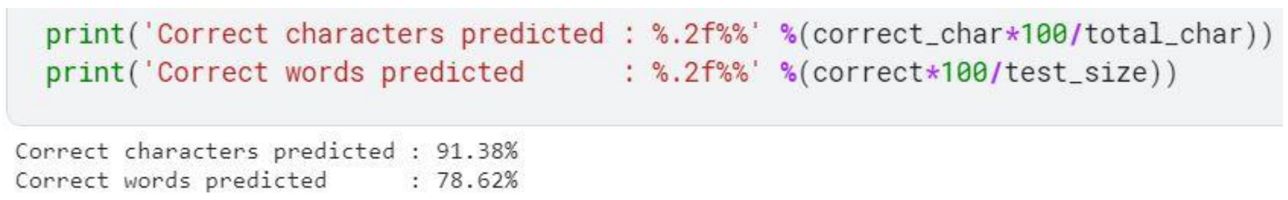
At the end we have used a dense layer with then outputs to classify the words.

CHAPTER-6

RESULT AND CONCLUSION

6.1 Results:

I have successfully trained our RNN model on the selected dataset and tested it. This model is able to recognise the handwritten words with high accuracy.



```
print('Correct characters predicted : %.2f%%' %(correct_char*100/total_char))
print('Correct words predicted      : %.2f%%' %(correct*100/test_size))
```

```
Correct characters predicted : 91.38%
Correct words predicted      : 78.62%
```

Fig 6.1: Accuracy of model

The model has been able to predict different characters with an accuracy of more than 90%.

And words are predicted correctly with an accuracy of 78%.

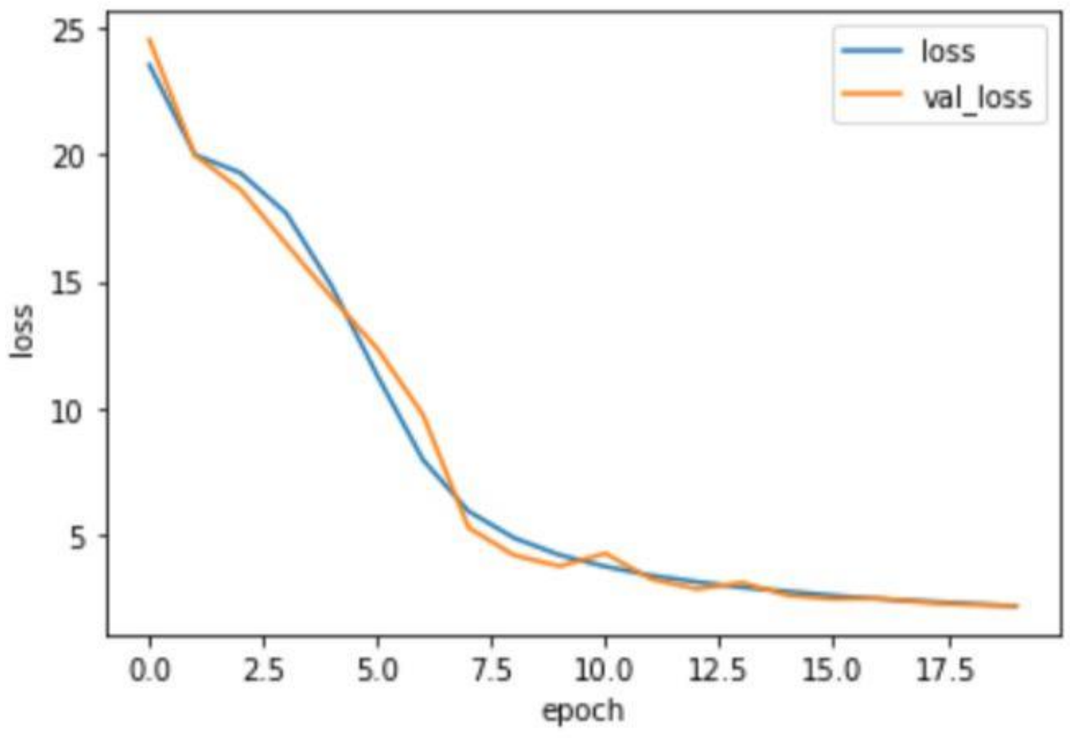


Fig 6.2: Validation loss and loss graph

In figure 6.2 we can see that the loss and validation loss are decreasing very fast for the first 10 epochs and after that it lowers at a very slow pace. In 20 the epoch loss has become very small.

Further training this model for more epochs would have been a waste of resources as after this there is very less room to improve.

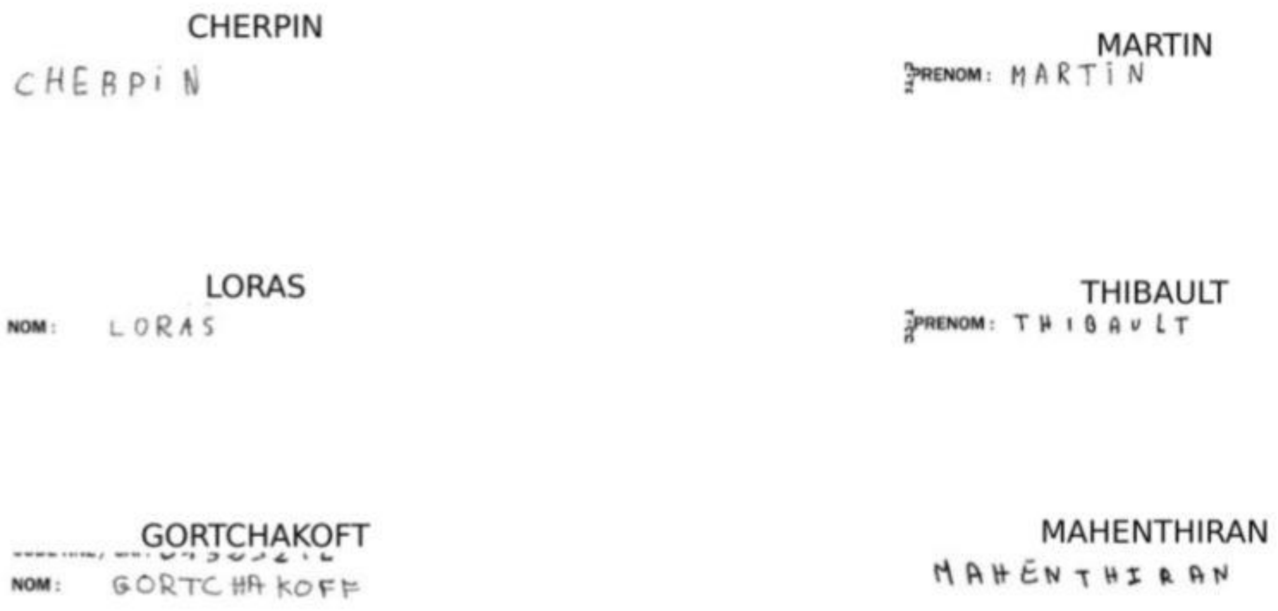


Fig 6.2: Live Prediction

The model was able to correctly predict 5 out of 6 images. It was not able to predict the word "GORTCHAKOFT" are it is not written properly.

6.2 Conclusion:

This project will be helpful in converting the handwritten text to a digital format which will be very helpful in converting old handwritten documents and notes to text files. These files are easy to store, edit, share and read easily. This project will also help in preserving old important documents which are difficult to store physically.

6.3 Future Scope:

The accuracy of this model can be further improved by using more training data which will help this model to learn and generalize better.

Some of the images in the dataset are not of very good quality and the annotations of some images are also wrong.

Removing such images will also help in model’s learning,

REFERENCES

[1] Nisha Sharma et al, “Recognition for handwritten English letters: A Review”International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 7

[2] J.Pradeep et al,, “Diagonal based feature extraction for handwritten alphabets recognition System using neural network”International Journal of Computer Science and Information Technology (IJCSIT), Vol 3, No 1

[3] Miroslav NOHAJ, Rudolf JAKA, “Image preprocessing for optical character recognition using neural networks”Journal of Pattern Recognition Research

[4] Jehad Ali, Rehanullah Khan, Nasir Ahmad, Imran Maqsood " Random Forests and Decision Trees ".

https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

[5] A. Graves, S. Fern´andez, F. Gomez, and J. Schmidhuber,“Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 369–376.