Handwritten Character Recognition with Neural Network

A Project Report

Submitted in the partial fulfilment of the requirements for

the award of the degree of

Bachelor of Technology

in

Department of Electronics and Communication Engineering

by

180040132	**N.Satish**
180040135	**Talasila Venkata Teja**
180040153	**Toviti Narendra Kumar**

under the supervision of

Raghuveera sir

Department of Electronics and Communication Engineering

Koneru Lakshmaiah Education Foundation

(Deemed to be University estd., u/s 3 of UGC Act 1956)

Greenfields, Vaddeswaram, Guntur (Dist.), Andhra Pradesh - 522502

November, 2019

DECLARATION

The Project Report entitled "**Handwritten Character Recognition with Neural Network** " is a record of bonafide work of 180040132 - N. SATISH , 180040135 - TALASILA VENKATA TEJA , 180040153 - Toviti Narendra Kumar submitted in partial fulfillment for the award of B.Tech in Electronics and Communication Engineering to the K L University. The results embodied in this report have not been copied from any other departments/University/Institute

180040132	**N. SATISH**
180040135	**TALASILA VENKATA TEJA**
180040153	**TOVITI NARENDRA KUMAR**

(DEEMED TO BE UNIVERSITY)

CERTIFICATE

This is to certify that the Project Report entitled "**Handwritten Character Recognition with Neural Network**" is being submitted by 180040132 - N. Satish , 180040135 - T.Vankata teja , 180040153 - TOVITI NARENDRA KUMAR submitted in partial fulfilment for the award of B.Tech in Electronics and Communication Engineering to the K L University is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

Signature of the Guide	**Signature of the HOD**
_______________	Dr. SUMAN MALOJI

ACKNOWLEDGEMENT

The success in this project would not have been possible but for the timely help and guidance rendered by many people. Our wish to express my sincere thanks to all those who has assisted us in one way or the other for the completion of my project.

Our greatest appreciation to our guide nagendram mam , Professor, Department of Electronics and Communication Engineering which cannot be expressed in words for his tremendous support, encouragement, and guidance for this project.

We express our gratitude to **Dr.SUMAN MALOJI,** Head of the Department for Electronics and Communication Engineering for providing us with adequate facilities, ways and means by which we can complete this project-based Lab.

We thank all the members of teaching and non-teaching staff members, and who have assisted me directly or indirectly for successful completion of this project.

Finally, we sincerely thank my parents, friends and classmates for their kind help and co-operation during my work.

180040132	**N. SATISH**
180040135	**TALASILA VENKATA TEJA**
180040153	**TOVITI NARENDRA KUMAR**

TABLE OF CONTENTS

ABSTRACT

A human learns to perform a task by practicing and repeating it again and again so that it memorizes how to perform the tasks.

Then the neurons in his brain automatically trigger and they can quickly perform the task they have learned. Deep learning is also very similar to this. It uses different types of neural network architectures for different types of problems. For example

– object recognition, image and sound classification, object detection, image segmentation, etc.

The handwritten character recognition is the ability of computers to recognize human handwritten character . It is a hard task for the machine because handwritten character are not perfect and can be made with many different flavors.

The handwritten character recognition is the solution to this problem which uses the image of a character and recognizes the character present in the image.

In this machine learning project, we will recognize handwritten characters, i.e, English alphabets from A-Z. This we are going to achieve by modeling a neural network that will have to be trained over a dataset containing images of alphabets.

1

Introduction :-

Handwriting recognition (HWR), also known as **Handwritten** Text **Recognition** (HTR), is the ability of a computer to receive and interpret intelligible **handwritten** input from sources such as paper documents, photographs, touch-screens and other devices.

MNIST is a widely used dataset for the **hand-written digit classification** task. It consists of 70,000 labeled 28x28 pixel grayscale images of **hand-written digits**. The dataset is split into 60,000 training images and 10,000 test images.

Optical character **recognition** (OCR) is the most mainstream technique used for **handwriting recognition**

Recognition of **handwriting** using **OCR** is based on a technology known as ICR. ICR or Intelligent Character **Recognition can recognize** individual, printed hand-written characters.

The **MNIST database** (Modified National Institute of Standards and Technology **database**) is a large **database** of handwritten digits that is commonly used for training various image processing systems. It was **created** by "re-mixing" the samples from NIST's original **datasets**

The printed or **handwritten** document is first scanned, or written onto a touchscreen mobile or tablet device..... The HWR app then separates each character and – using a pre-programmed bank of algorithms – matches it to what it thinks is the most likely letter on a database.

Literature review :-

A human learns to perform a task by practicing and repeating it again and again so that it memorizes how to perform the tasks.

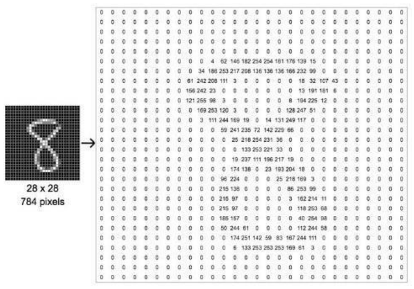
Then the neurons in his brain automatically trigger and they can quickly perform the task they have learned. Deep learning is also very similar to this. It uses different types of neural network architectures for different types of problems. For example

– object recognition, image and sound classification, object detection, image segmentation, etc.

The handwritten character recognition is the ability of computers to recognize human handwritten character . It is a hard task for the machine because handwritten character are not perfect and can be made with many different flavors.

The handwritten character recognition is the solution to this problem which uses the image of a character and recognizes the character present in the image.

In this machine learning project, we will recognize handwritten characters, i.e, English alphabets from A-Z. This we are going to achieve by modeling a neural network that will have to be trained over a dataset containing images of alphabets.



3

Design methodology:-

Recognition of **handwriting** using **OCR** is based on a technology known as ICR. ICR or Intelligent Character **Recognition can recognize** individual, printed hand-written characters. The **tesseract algorithm is available on Google Code, and is** one of the best open source OCR out there.

It has already been stated that the primary goal of our project—an Android application—is able to recognize handwritten characters based on user's touch input and image/camera input in an offline manner. Also, the application provides means of adding and learning a new character and learning interactively from user's feedback.

Android has been selected as the most feasible platform, because many Android devices feature capacitive touch screens, which are very friendly to finger touch input, and cameras, which are necessary to satisfy the camera recognition requirement. Although developing a high quality Android application according to Android development guidelines has not been our primary focus, these have nonetheless not been ignored.

Feature-wise, the application distinguishes two cases: learning from touch input, taking into account both the character bitmap and the individual strokes the user writes, as opposed to learning from a picture bitmap, where we only recognize the character bitmap. Formally, both cases fall into the offline approach to handwriting recognition. In this approach, the complete character image is the only information available.

Online recognition, on the other hand, describes storing the two-dimensional coordinates of points of the writing as a function of time, hence individual strokes are ordered. Therefore, when recognizing user touch input, the use of stroke analysis might resemble the online approach as well because of the additional data provided, even though the strokes are not ordered in time.

As such, the system is able to precisely separate certain characters of high bitmap similarity, as in the case of the '8' and the 'B' characters, while being invariant to the order of the strokes the user writes.

4

Data set :-

This dataset consists of more than four hundred thousand handwritten names collected through charity projects to support disadvantaged children around the world.

Optical Character Recognition (OCR) utilizes image processing technologies to convert characters on scanned documents into digital forms. It typically performs well in machine printed fonts. However, it still poses a difficult challenges for machines to recognize handwritten characters, because of the huge variation in individual writing styles.

There are 206,799 first names and 207,024 surnames in total. The data was divided into a training set (331,059), testing set (41,382), and validation set (41,382) respectively.

Labels of all images created via human-in-the-loop anotation on the Appen platform are also provided, enabling you to extend the data set with your own data.

The input data in this job is a hundreds of thousands of images of handwritten names. In the “Data” tab above, you’ll find the transcribed images broken up into test, training, and validation sets.

Code for notebook:-

```
from keras.datasets import mnist

import matplotlib.pyplot as plt

import cv2

import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout

from keras.optimizers import SGD, Adam

from keras.callbacks import ReduceLROnPlateau, EarlyStopping

from keras.utils import to_categorical

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from keras.utils import np_utils

import matplotlib.pyplot as plt

from tqdm import tqdm_notebook

from sklearn.utils import shuffle


# Read the data...

data = pd.read_csv(r"D:\a-z alphabets\A_Z Handwritten Data.csv").astype('float32')
```

```
# Split data the X - Our data , and y - the prdict label

X = data.drop('0',axis = 1)

y = data['0']

# Reshaping the data in csv file so that it can be displayed as an image...

train_x, test_x, train_y, test_y = train_test_split(X, y, test_size = 0.2)

train_x = np.reshape(train_x.values, (train_x.shape[0], 28,28))

test_x = np.reshape(test_x.values, (test_x.shape[0], 28,28))

print("Train data shape: ", train_x.shape)

print("Test data shape: ", test_x.shape)

# Dictionary for getting characters from index values...

word_dict =
{0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O',15:'P',16:'
Q',17:'R',18:'S',19:'T',20:'U',21:'V',22:'W',23:'X', 24:'Y',25:'Z'}

# Plotting the number of alphabets in the dataset...

train_yint = np.int0(y)

count = np.zeros(26, dtype='int')

for i in train_yint:

    count[i] +=1

alphabets = []

for i in word_dict.values():

    alphabets.append(i)

fig, ax = plt.subplots(1,1, figsize=(10,10))

ax.barh(alphabets, count)
```

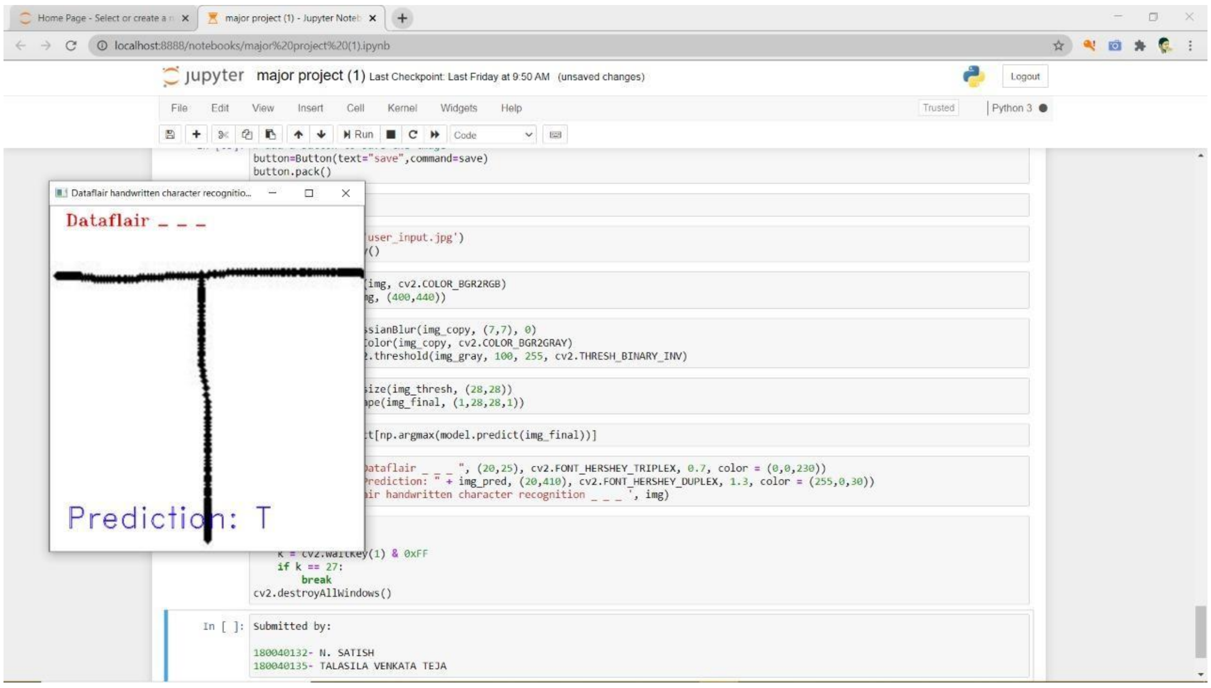
```
plt.xlabel("Number of elements ")

plt.ylabel("Alphabets")

plt.grid()

plt.show()

#Shuffling the data ...

shuff = shuffle(train_x[:100])

fig, ax = plt.subplots(3,3, figsize = (10,10))

axes = ax.flatten()

for i in range(9):

    axes[i].imshow(np.reshape(shuff[i], (28,28)), cmap="Greys")

plt.show()

#Reshaping the training & test dataset so that it can be put in the model...

train_X = train_x.reshape(train_x.shape[0],train_x.shape[1],train_x.shape[2],1)

print("New shape of train data: ", train_X.shape)

test_X = test_x.reshape(test_x.shape[0], test_x.shape[1], test_x.shape[2],1)

print("New shape of train data: ", test_X.shape)

# Converting the labels to categorical values...

train_yOHE = to_categorical(train_y, num_classes = 26, dtype='int')

print("New shape of train labels: ", train_yOHE.shape)

test_yOHE = to_categorical(test_y, num_classes = 26, dtype='int')

print("New shape of test labels: ", test_yOHE.shape)
```

```
# CNN model...

model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)))

model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'))

model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'valid'))

model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Flatten())

model.add(Dense(64,activation ="relu"))

model.add(Dense(128,activation ="relu"))

model.add(Dense(26,activation ="softmax"))

model.compile(optimizer = Adam(learning_rate=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0001)

early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0,
mode='auto')

history = model.fit(train_X, train_yOHE, epochs=1, callbacks=[reduce_lr, early_stop],
validation_data = (test_X,test_yOHE))
```

```
model.summary()

model.save(r'model_hand.h5')

# Displaying the accuracies & losses for train & validation set...

print("The validation accuracy is :", history.history['val_accuracy'])

print("The training accuracy is :", history.history['accuracy'])

print("The validation loss is :", history.history['val_loss'])

print("The training loss is :", history.history['loss'])

#Making model predictions...

pred = model.predict(test_X[:9])

print(test_X.shape)

# Displaying some of the test images & their predicted labels...

fig, axes = plt.subplots(3,3, figsize=(8,9))

axes = axes.flatten()

for i,ax in enumerate(axes):

    img = np.reshape(test_X[i], (28,28))

    ax.imshow(img, cmap="Greys")

    pred = word_dict[np.argmax(test_yOHE[i])]

    ax.set_title("Prediction: "+pred)

    ax.grid()

# Prediction on external image...

img = cv2.imread(r'C:\Users\abhij\Downloads\img_b.jpg')

img_copy = img.copy()
```

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

img = cv2.resize(img, (400,440))

img_copy = cv2.GaussianBlur(img_copy, (7,7), 0)

img_gray = cv2.cvtColor(img_copy, cv2.COLOR_BGR2GRAY)

_, img_thresh = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)

img_final = cv2.resize(img_thresh, (28,28))

img_final =np.reshape(img_final, (1,28,28,1))

img_pred = word_dict[np.argmax(model.predict(img_final))]

cv2.putText(img, "Dataflair _ _ _ ", (20,25), cv2.FONT_HERSHEY_TRIPLEX, 0.7, color =
(0,0,230))

cv2.putText(img, "Prediction: " + img_pred, (20,410), cv2.FONT_HERSHEY_DUPLEX,
1.3, color = (255,0,30))

cv2.imshow('Dataflair handwritten character recognition _ _ _ ', img)


while (1):

   k = cv2.waitKey(1) & 0xFF

   if k == 27:

      break

cv2.destroyAllWindows()
```

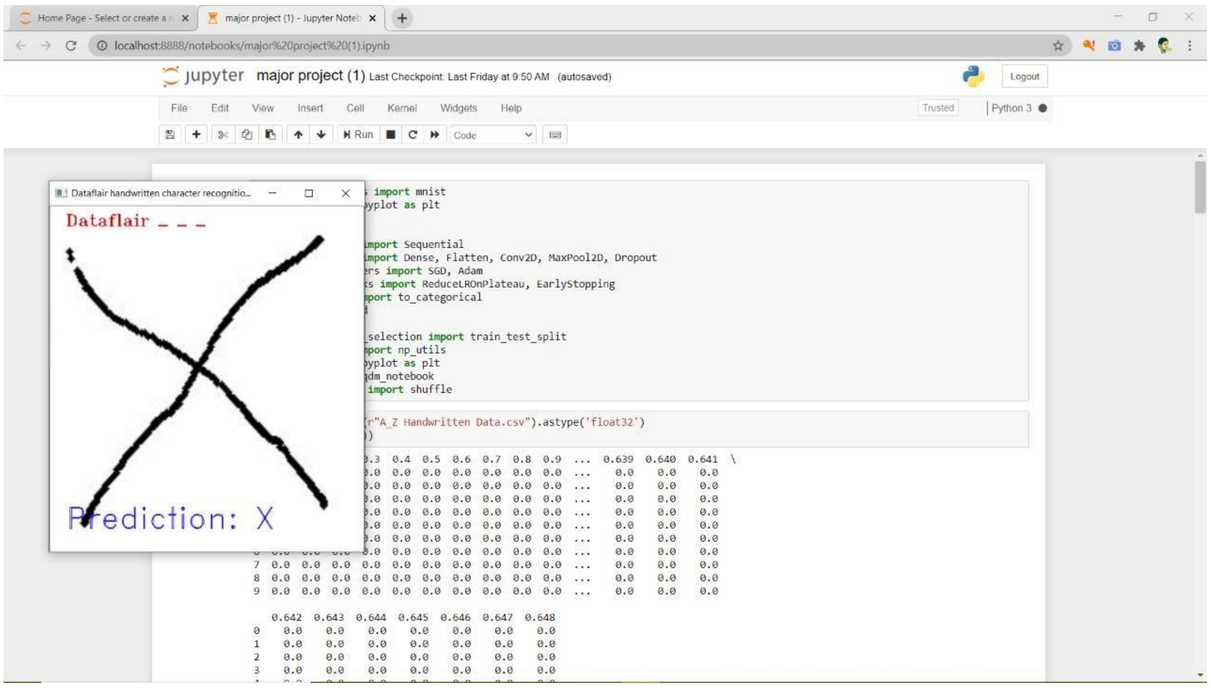
Outputs:-



major project (1) Last Checkpoint: Last Friday at 9:50 AM (unsaved changes)
Dataflair handwritten character recognitio...
Dataflair _ _ _
Prediction: T

```
button=Button(text="save",command=save)
button.pack()

user_input.jpg')
()

(img, cv2.COLOR_BGR2RGB)
g, (400,440))

sianBlur(img_copy, (7,7), 0)
olor(img_copy, cv2.COLOR_BGR2GRAY)
.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)

ize(img_thresh, (28,28))
pe(img_final, (1,28,28,1))

t[np.argmax(model.predict(img_final))]

ataflair _ _ _ ", (20,25), cv2.FONT_HERSHEY_TRIPLEX, 0.7, color = (0,0,230))
rediction: " + img_pred, (20,410), cv2.FONT_HERSHEY_DUPLEX, 1.3, color = (255,0,30))
ir handwritten character recognition _ _ _ ', img)

    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows()
```

```
In [ ]: Submitted by:

        180040132- N. SATISH
        180040135- TALASILA VENKATA TEJA
```



major project (1) Last Checkpoint: Last Friday at 9:50 AM (autosaved)
Dataflair handwritten character recognitio...
Dataflair _ _ _
Prediction: X

```
 import mnist
yplot as plt

mport Sequential
mport Dense, Flatten, Conv2D, MaxPool2D, Dropout
rs import SGD, Adam
s import ReduceLROnPlateau, EarlyStopping
port to_categorical

selection import train_test_split
port np_utils
yplot as plt
dm_notebook
 import shuffle

r"A_Z Handwritten Data.csv").astype('float32')
)
```

```
.3  0.4  0.5  0.6  0.7  0.8  0.9  ...  0.639  0.640  0.641  \
.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  0.0  ...    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  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
7  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
8  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
9  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

   0.642  0.643  0.644  0.645  0.646  0.647  0.648
0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
1    0.0    0.0    0.0    0.0    0.0    0.0    0.0
2    0.0    0.0    0.0    0.0    0.0    0.0    0.0
3    0.0    0.0    0.0    0.0    0.0    0.0    0.0
```

Conclusion:-

This work has mostly been focused on the machine learning methods used in the project. At first, we reviewed the approaches that are nowadays used in similar applications. After that, we delved into the inner workings of a multilayer perceptron, focusing on backpropagation and resilient back, which has been implemented an the Android application.

With the knowledge we had described, we specified the requirements of the project and planned the solution. During the development of the application, we ran into a few problems, which, along with the application structure and details, have been described in the implementation chapter.

Finally, the results of the implementation of the learning algorithms have been compared. The Android application performs character recognition based on touch, image, and camera input. We have developed a Java package containing classes that implement the multilayer perceptron learning model, which can also be used in other applications due to its modular design that supports the loose coupling principle.

The application itself uses this package in such way. Several improvements for the application or the learning model used within can be suggested. For example, the feature extraction performed by the neural network could be constrained to operate on more strictly preprocessed data. Also, several classifiers learning on different features could be combined to make the system more robust.

Additionally, an unsupervised clustering learning model, such as an ART network , could be used on the raw input, whose output would be connected to a multilayer perceptron.

REFERENCES :-

1> http://davinci.fmph.uniba.sk/~uhliarik4/recognition/files/thesis.pdf

2> https://link.springer.com/article/10.1007/s11036-019-01243-5

3> https://github.com/githubharald/SimpleHTR