RV Educational Institutions ®
RV College of Engineering ®

Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE, New Delhi, Accredited By NAAC, Bengaluru And NBA, New Delhi

Go, change the world

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Handwritten Character Recognition using Neural Networks

MINOR PROJECT REPORT

Submitted by,

Anish Kuila **USN – 1RV18CS025**

Aman Singh **USN – 1RV18CS017**

Under the guidance of

Dr. Deepamala N
Associate Professor
Dept of CSE
RV College of Engineering

In partial fulfilment for the award of degree
of
Bachelor of Engineering
in
Computer Science and Engineering
2020-2021

RV COLLEGE OF ENGINEERING®, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the minor project work titled ***'Handwritten Character Recognition using Neural Networks'*** is carried out by **Anish Kuila (1RV18CS025), Aman Singh (1RV18CS025),** who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfilment for the award of degree of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2020-2021. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the minor project report deposited in the departmental library. The Minor Project report has been approved as it satisfies the academic requirements in respect of minor project work prescribed by the institution for the said degree.

Signature of Guide	**Signature of Head of the Department** **Dr. Ramakanth Kumar P**	**Signature of Principal** **Dr.K.N.Subramanya**

External Viva

Name of Examiners	**Signature with Date**
1	
2	

RV COLLEGE OF ENGINEERING®, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Anish Kuila and Aman Singh students of sixth semester B.E., department of CSE, RV College of Engineering, Bengaluru, hereby declare that the minor project titled **'Handwritten Character Recognition using Neural Networks'** has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering** in **Computer Science and Engineering** during the year 2020-21.

Further we declare that the content of the report has not been submitted previously by anybody for the award of any degree or diploma to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date: 23/7/2021

Name **Signature**

1. Anish Kuila (1RV18CS025)

2. Aman Singh (1RV18CS017)

ACKNOWLEDGEMENT

We are indebted to our guide**, Dr Deepamala N,** Associate Professor**, Dept of CSE** for her wholehearted support, suggestions and invaluable advice throughout our project work and also helped in the preparation of this thesis.

We also express gratitude to our Minor Project lab faculty **Mrs. Azra Nasreen, Assistant Professor** and **Mrs. Sandhya S, Assistant Professor,** Department of Computer Science and Engineering for their valuable comments and suggestions.

Our sincere thanks to **Dr. Ramakanth Kumar P.**, Professor and Head, Department of Computer Science and Engineering, RVCE for his support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for his appreciation towards this project work.

We thank all the **teaching staff and technical staff** of Computer Science and Engineering department, RVCE for their help.

Lastly, we take this opportunity to thank our **family** members and **friends** who provided all the backup support throughout the project work.

ABSTRACT

Handwriting recognition is the ability of a machine to receive and interpret handwritten input from multiple sources like paper documents, photographs, touch screen devices etc. It is used very much in digital libraries and reading postal addresses, bank check accounts and forms in banks. The key objective of this project is to design handwritten character recognition for English alphabets using neural networks.

There are six phases to do handwritten character recognition which are image acquisition, pre-preprocessing, segmentation, feature extraction, classification and post-processing. This project is done using python programming language in Jupyter platform with many machine learning libraries. This project will help the user to interpret a handwritten English alphabet on devices.

The result obtained at the end was with 97% accuracy which is high compared to the existing results from the literature. The validation and training accuracy were higher than 95% and their losses were significantly low.

Thus the predicted character from the image character is highly accurate and the system is also quite reliable. The Future work of this paper was to implement a better technique that not only provides better and more accuracy and less error rate but also faster.

TABLE OF CONTENTS

List of Figures **Page No**

CHAPTER 1

INTRODUCTION

The purpose of this project is to take handwritten English characters as input, process the character, train the neural network algorithm to recognize the pattern and modify the character into a beautified version of the input. Handwriting recognition is the ability of a machine to receive and interpret handwritten input from multiple sources like paper documents, photographs, touch screen devices etc. Recognition of handwritten and machine characters is an emerging area of research and finds extensive applications in banks, offices and industries.

Neural computing is a comparatively new field, and design components are therefore less well specified than those of other architectures. Neural computers implement data parallelism. Neural computer are operated in way which is completely different from the operation of normal computers. Neural computer are trained (not Programmed) so that given a certain starting state (data input); they either classify the input data into one of the number of classes or cause the original data to evolve in such a way that a certain desirable property is optimized. Pattern recognition is perhaps the most common use of neural networks. The neural network is presented with a target vector and also a vector which contains the pattern information; this could be an image or a handwritten data.

1.1 STATE OF ART DEVELOPMENTS

Handwritten Character Recognition works in stages as preprocessing, segmentation, feature extraction and recognition using neural network. Preprocessing includes series of operations to be carried out on document image to make it ready for segmentation. During segmentation the document image is segmented into individual character or numeric image then feature extraction technique is applied on character image. Finally feature vector is presented to the selected algorithm for recognition. Here these extracted features are provided to Neural

Network in our project's case the convolutional neural network (CNN) for recognition of character.

1.2 MOTIVATION

The idea of Neural Network in HCR will brings us the reading of various combined style of writing a character. In forensic application HCR will be an effective method for evidence collection. It will also help to reduce noise from the original character. Our method develop accuracy in recognizing character in divert font and size. More set of sample invites more accuracy rate because of heavy training and testing session.

1.3 PROBLEM STATEMENT

To develop a system that recognizes handwritten English alphabetic characters using Neural Networks i.e in other words if we can write the character "A" the system predict the character that it is truly "A" or the input character is nearer to "A" or something else.

1.4 OBJECTIVES

- To provide a highly accurate recognition and minimum training time of handwritten English characters using neural network.

- To provide an easy and more reliable user interface to input the object image and eliminate unwanted patterns to provide a clearer image.

1.5 METHODOLOGY

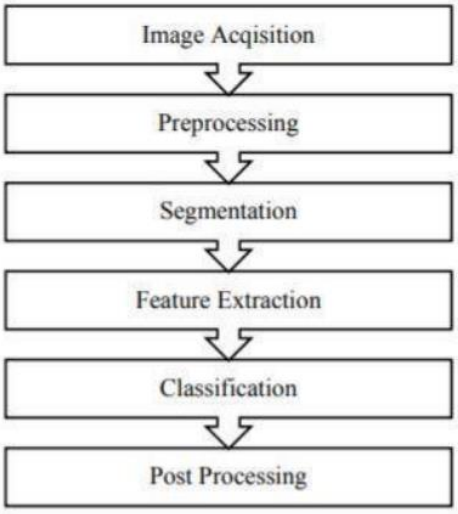


FIGURE: BLOCK DIAGRAM FOR HCR

HCR works in six stages as image acquisition, pre-preprocessing, segmentation, feature extraction, classification and post-processing. In **Image acquisition** stage, the input image is provided to the recognition system. The input can be either in an image format such as JPEG, PNG, etc. or scanned image, digital camera or any other suitable digital input device or one can draw on the canvas provided on the user interface. The second method called as **Pre-processing** is the entry method for recognition of character and very important in deciding the recognition rate. Preprocessing works to normalize the strokes and also remove variations that can reduce the rate of accuracy. Preprocessing mainly works on the various distortions like the irregular text size, points missed during the pen movement, uneven spaces, etc. Then we have the **Segmentation** stage which is used to convert input image consisting of many characters into the individual characters. The techniques used are word, line and character segmentation. It is generally performed by dividing single characters from the word picture. The fourth stage **Feature Extraction** aims to allow the extraction of pattern which is most important for classification. Some of the feature extraction techniques like histogram, chain code, zoning and gradient based features can be applied to extract the features of individual characters. The decision making is done in the **Classification** phase. For recognizing the characters, the extracted features are used. Different classifiers like SVM and Neural Networks are used. The classifiers sorts the given input feature with

reserved pattern and find the best matching class for input, for which Soft Max Regression is used. The final and last phase of the character recognition is the **Post-processing**. It is the procedure for correcting the misclassified output by using natural language. It processes output by getting it after the shape has been recognized. If the shape is recognized purely then the accuracy can be improved in accordance with the knowledge of language.

1.6 SUMMARY

Neural Networks are recently being used in various kind of pattern recognition. Handwritings of different person are different; thus it is very difficult to recognize the handwritten characters. Handwritten Character recognition is an area of pattern recognition that has become the subject of research during the last couple of decades. It can effectively recognize a particular character of type format using the neural network approach.

Handwritten character recognition can reduce man-power to convert old literature into digitized form manually. It could also make the digitized library rich with any language. Handwritten character recognition works in six stages as image acquisition, pre-preprocessing, segmentation, feature extraction, classification and post-processing. Our model focuses on a highly accurate recognition and minimum training time of handwritten English characters using convolution neural network (CNN). It also provides an easy and more reliable user interface to input the object image and eliminate the unwanted patterns to provide a clearer image.

CHAPTER 2

LITERATURE SURVEY

In the topics below you will be able to get an idea about different works provided by different researchers in the area of handwritten character recognition.

2.1 RELATED WORK

"Handwritten Character Recognition using Neural Network and Tensor Flow" was published by Megha Agarwal, Shalika, Vinam Tomar, Priyanka Gupta in International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, Issue- 6S4, April 2019. Their objective was to develop a handwritten character recognition software with a very high accuracy rate and with minimal time and space complexity and also optimal. They were able to find out that the Feature extraction methods like diagonal and direction techniques are way better in generating high accuracy results compared to many of the traditional vertical and horizontal methods. Neural network with best tried layers gives the plus feature of having a higher tolerance to noise thus giving accurate results. The use of normalization along with feature extraction yielded the better and higher accuracy results in character recognition. Also the bigger the training data set and better the neural network design, the better accurate is the result. But there was one limitation to their paper and that is the use of spell checking to correct handwritten character text will not typically permit the case of the letters to be considered **[1]**.

"Pattern Recognition - Recognition of Handwritten Document Using Convolutional Neural Networks" was published by M.Rajalakshmi, P.Saranya, P.Shanmugavavidu in IEEE International conference on Intelligent techniques and control, 2019. Their objective was to develop a handwritten character recognition using Convolutional neural network (CNN). CNN is a type of neural network designed to process pixel data using several layers

of filters for feature extraction. This paper helps the researcher get new ideas to develop a new technique with good environment and architecture to propose more accuracy and less error rate. The major bottlenecks in this system are the issues of recognizing unconstrained handwritings like cursive, block, and tilt that cause huge variation in writing styles, the overlapping and the interconnections between characters. The CNN adds significant improvements to the approach of Handwritten Character Recognition. But there is a limitation to their paper which is CNN is slower due to an operation such as maxpool layer. Also the training process takes a lot of time if the computer doesn't consist of a good GPU **[2]**.

"Enhancement of segmentation and zoning to improve the accuracy of handwritten character recognition" was published by Megha S.E. Benita Galaxy, S. Selvin Ebenezer in International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) - 2016. Their objective was to improve the accuracy of handwritten character recognition by enhancing segmentation and zoning. In this paper they say that segmentation is a difficult task because segmenting the cursive characters isn't quite easy. So in order to achieve a better and efficient segmentation, they needed to enhance the concept of zoning by introducing a method called contour tracing. Now this method identifies the boundary pixels of the digital region and also determines the width of the character to be segmented. This method was able to improve the segmentation process and thus the accuracy of the characters **[3]**.

"A Combined Approach of Feature Selection and Machine Learning Technique for Handwritten Character Recognition" was published by Saptadeepa Kalita, Diwakar Gautam, Ashok Kumar Sahoo, Rajiv Kumar in Proceedings of 4th International Conference on Cyber Security (ICCS) 2018. Their objective was to recognize handwritten English alphabets and also reduce the redundancy of feature set using feature selection method. In this paper, the accuracy seems to be better when using feature selection algorithm. The features of Handwritten characters are extracted using gradient-based measures. The redundancy of feature set is reduced using feature selection method, and machine learning

approach is deployed to generalize the decision boundary. Observation is made on the accuracy that is obtained from three different feature selection techniques Minimum Redundancy and Maximum Relevance (mRMR), Joint Mutual Information Feature Selection (JMI), Relief Feature Selection. It selects the best features from the feature vector obtained from the feature extraction technique. Limitation of this paper is that the filter method ignores the interaction with the classifier and each feature is considered independently thus ignoring feature dependencies **[4]**.

"Performance Optimization and Comparative Analysis of Neural Networks for Handwritten Devanagari Character Recognition" was published by Sushama Shelke, Shaila Apte in (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7, 2016. Their objective is to develop handwritten character recognition for Devanagiri characters. They were able to find out that increasing the number of structural classes for complex characters helps in improving the performance of the system with respect to both accuracy and time. Also adding shape details of the characters during feature extraction enhances the recognition rate. And out of the 3 neural network approaches: Feed-forward back-propagation, Cascade-forward back-propagation and Elman back-propagation, Elman back propagation has given the highest recognition rate. This paper also provides few techniques for optimizing the recognition accuracy at pre-classification stage, feature extraction and recognition stage. But the issue is that the actual performance of back-propagation on a specific problem is dependent on the input data. Back-propagation can also be quite sensitive to noisy data **[5]**.

2.2 SUMMARY

There are many papers published by different journals. Convolutional neural network (CNN) is a type of neural network designed to process pixel data using several layers of filters for feature extraction. Even though the use of CNN in handwritten character recognition is slower due to maxpool layers it is still faster than machine learning and gives better accuracy. Their idea is to develop a new technique with good environment and architecture to propose

more accuracy and less error rate. Their idea is also to improve the six phases -image acquisition, pre-preprocessing, segmentation, feature extraction, classification and post-processing in their areas to make it more effective.

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATION OF HCR

3.1 FUNCTIONAL REQUIREMENTS

The functional requirements for a system describe what the system should do.

- The developed system should recognize handwritten English character present in the image.
- System shall show the error message to the user when given input is not in the required format.
- System must provide the quality of service to user.
- System must provide good and high accuracy for character recognition.
- The system should recognize characters of an image in any format like jpg and png.

3.2 NON-FUNCTIONAL REQUIREMENTS

As the name suggests these are the requirements that are not directly interacted with specific functions delivered by the system.

Performance: Handwritten characters in the input image will be recognized with an accuracy of above 90.

Functionality: This software will deliver on the functional requirements.

Availability: This system will retrieve the handwritten character regions only if the image contains characters in it.

Flexibility: It provides the users to load the image easily.

Learn ability: The software is very easy to use and reduces the learning work.

3.3 HARDWARE REQUIREMENTS

- Processor : Intel Pentium 4 or above
- System bus : 32 bit/64bit
- RAM: 4GB
- Operating System: Windows 8
- HDD: 10GB or more

3.4 SOFTWARE REQUIREMENTS

- Programming language : Python
- IDE : Jupyter
- Numpy
- Opencv2
- Keras
- Pandas
- Matplotlib
- Tensorflow (Keras uses TensorFlow in backend and for some image preprocessing)

3.5 SUMMARY

The key functionality is to recognize the handwritten English alphabetic characters present in the image. Python programming language is used to develop this tool due to suitable libraries that are required.

CHAPTER 4

DESIGN OF HCR

4.1 HIGH LEVEL DESIGN

4.1.1 SYSTEM ARCHITECTURE

A deployment diagram shows the allocation of processes to processors in the physical design of a system. A deployment diagram may represent all or part of the process architecture of a system.

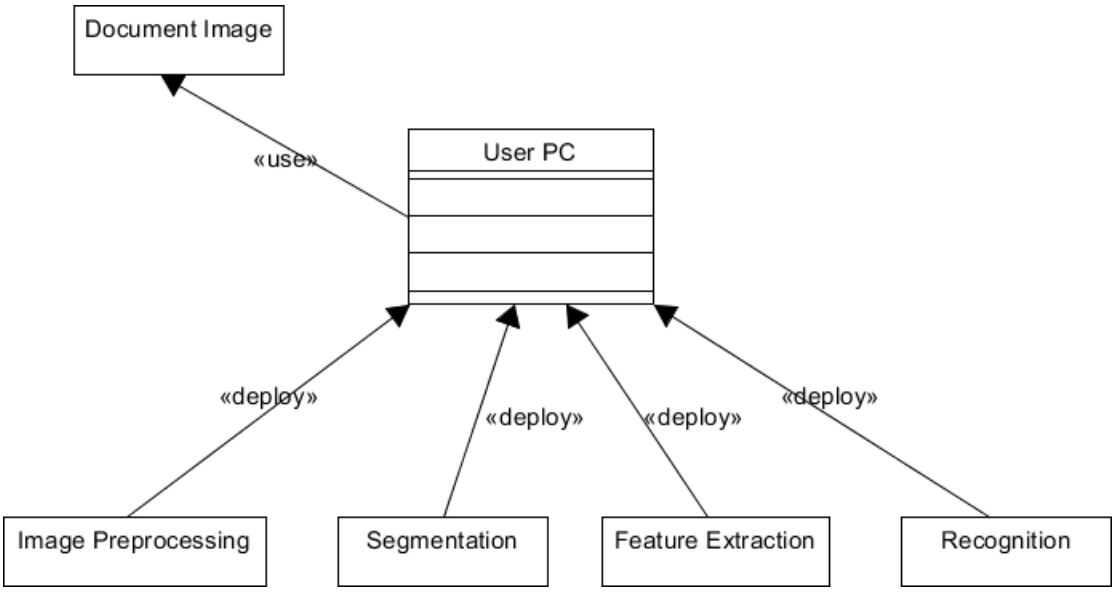
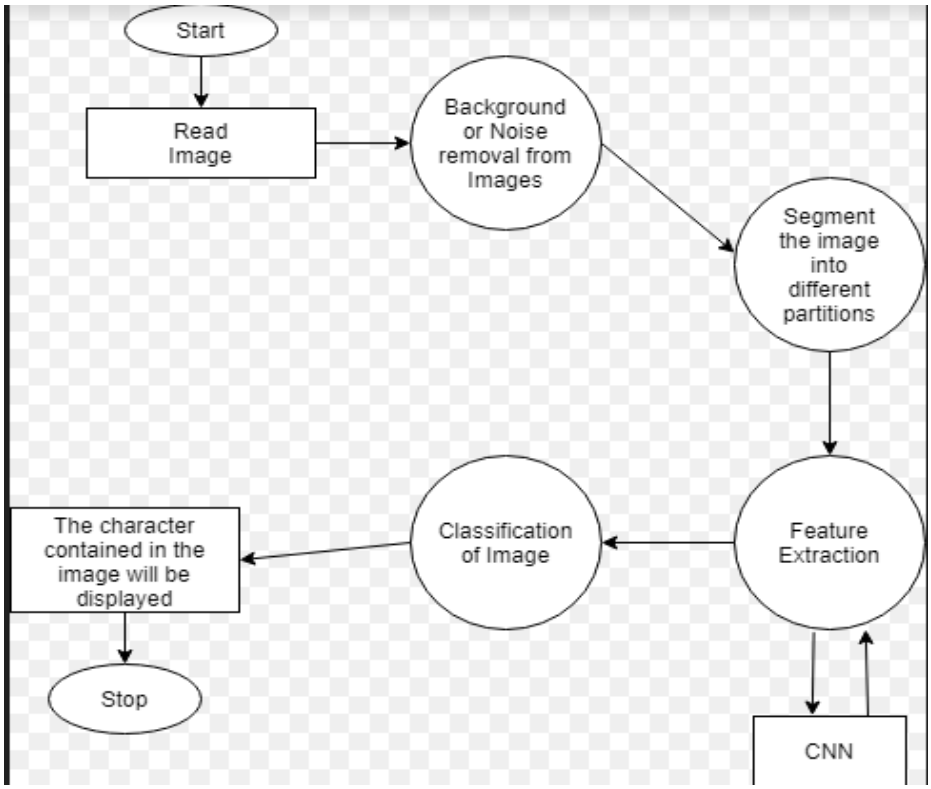


Figure: Deployment Diagram

4.2 DETAILED DESIGN

4.2.1 STRUCTURE CHART



DATA FLOW DIAGRAM –

Data flow diagram (DFD) is also called as Bubble Chart is a graphical technique, which is used to represent information flow, and transformers those are applied when data moves from input to output. DFD represents system requirements clearly and identify transformers those becomes programs in design. DFD may further partitioned into different levels to show detailed information flow e.g. level 0, level 1 etc.

Figure: DFD(level 0) for HCR System

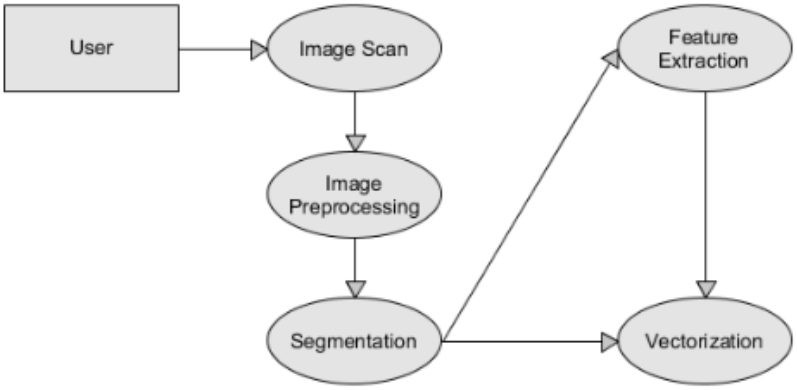


Figure: DFD(level 1) for Image Reader

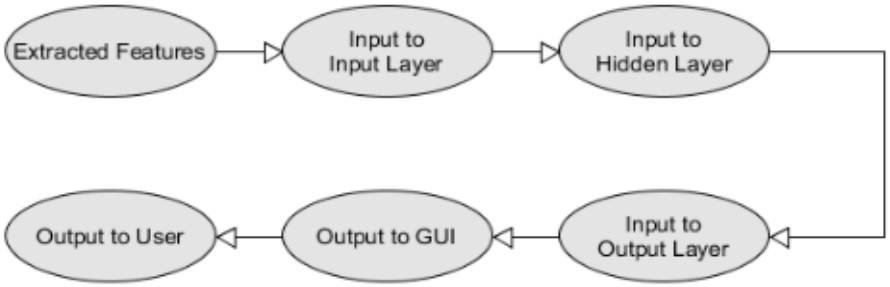


Figure: DFD(level 1) for Recognition Unit

4.2.2 FUNCTIONAL DESCRIPTION OF THE MODULES

The proposed work is divided into four modules:

1. Image Preprocessing
2. Segmentation
3. Feature Extraction
4. Classification

Image Preprocessing -

The image is preprocessed using different image processing algorithms like Inverting image, Gray Scale Conversion and image thinning.

1. Noise Removing: To eliminate the unwanted or undesired patterns, a technique called noise removing is used. There's a technique like Uniform and non uniform filtering that is to be used.
2. Binarization: In this all typed characters are translated into grey-scale picture. Each and every image of character is to be catched vertically after translating the gray scale image into the binary matrix.

3. Normalization: It is the process of translating a picture data into the standard required form. Mainly sizing along with skewing normalizations are performed. Size modifies the picture image in to the predefined fixed size. While skew is used during the time of scanning when the text is deviated from the base line and for this skewing and detections and their back propagation results are required.

Segmentation -

After preprocessing of the image segmentation is done. This is done with the help of following steps:

1. Remove the borders
2. Divide the text into rows
3. Divide the rows(lines) into words
4. Divide the words into letters

Feature Extraction -

Once the character is segmented we generate the binary glyphs and calculate the summation of each rows and columns values as a features. Some of the Feature extraction techniques like Principle Component Analysis (PCA), Scale Invariant Feature Extraction (SIFT), Linear Discriminant Analysis (LDA), Histogram, Chain Code (CC), zoning and Gradient based features can be applied to extract the features

of individual characters. All of these features are used to train the given system. Each of the segmented image is taken of some pixel of dimension 28 * 28.

Classification -

In this phase, we are going to train and test the Neural Network. For recognizing the characters, the extracted features are used. Different classifiers like SVM and Neural Networks are used. The classifiers sorts the given input feature with reserved pattern and find the best matching class for input, for which Soft Max Regression is used. Soft Max regression assigns the probability to each result thus classification becomes easy.

4.3 SUMMARY

The handwritten characters are recognized after it goes through four phases which are image processing, segmentation, feature extraction and classification. All these modern techniques are used for this phases.

CHAPTER 5

IMPLEMENTATION OF HCR

5.1 PROGRAMMING LANGUAGE SELECTION

Python – It's an interpreted high-level general-purpose programming language.

5.2 PLATFORM SELECTION

IDE – Jupyter (For Build and Testing)

Command prompt (For Execution)

5.3 CODE CONVENTIONS

```
# Import libraries and packages

import matplotlib.pyplot as plt
import cv2
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.utils import to_categorical
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle

# Read the data
```

```
data = pd.read_csv("C:\HCR\HCRnew\handwritten_chars_recog1.csv").astype('float32')
print(data.head(10))

# Split data, X - Our data, and y - the predict 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

y_int = np.int0(y)
count = np.zeros(26, dtype='int')
for i in y_int:
    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):
    _,shu = cv2.threshold(shuff[i], 30, 200, cv2.THRESH_BINARY)
    axes[i].imshow(np.reshape(shuff[i],(28,28)),cmap="Greys")

#Displays 9 plots of random thresholded images

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 test 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"))


#Compiling and Fitting Model

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('C:\HCR\HCRnew\img_d.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, "Prediction: " + img_pred, (20,410), cv2.FONT_HERSHEY_DUPLEX,
1.3, color = (255,0,30))
cv2.imshow('Handwritten character recognition', img)
```

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

5.4 SUMMARY

Python has been chosen as the programming language due to its reach to a set of libraries like pandas, matplotlib, tensorflow, etc. It uses Jupyter as the platform to build and test but command prompt for execution.

CHAPTER 6

EXPERIMENTAL RESULTS AND TESTING OF HCR

6.1 EVALUATION METRICS

- The HCR system was tested on several different scanned images containing handwritten characters with different styles and the results were highly encouraging.
- The proposed method performs preprocessing on the image for removing the noise and further uses feature extraction using gradient technique.
- The scanned image is successfully recognized in different file formats like png and jpg.
- The proposed methodology has produced good results for images containing handwritten characters written in different styles, different size and alignment with varying background. It classifies most of the handwritten characters correctly if the image contains less noise in the characters and also in the background. Characters written with legible handwriting are classified more accurately.

6.2 EXPERIMENTAL DATASET

The dataset for this project contains 372450 images of alphabets of 28×2, all present in the form of a CSV file. Since the csv data file is large number of rows and columns, only sample csv file format is presented below:

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	...	0.640	0.641	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	0.0	0.0	...	0.0	0.0	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	0.0	0.0	...	0.0	0.0	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	0.0	0.0	...	0.0	0.0	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	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
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
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
6	0.0	0.0	0.0	0.0	0.0	0.0	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	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	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	0.0	0.0	0.0	0.0	0.0

[10 rows x 785 columns]

6.3 PERFORMANCE ANALYSIS

A handwritten character in the input image is recognized with an accuracy of above 90. We print out the training & validation accuracies along with the training & validation losses for the character recognition. The accuracy has been more than 97% while the losses has been significantly less.

6.4 UNIT TESTING

In this each module is tested individually. Criteria selected for identifying unit test module is to identify module that has core functionality implementation. Module could be an individual or procedure.

The following is a list of functions for unit testing that will tested:

- Select the scanned input image of handwritten document.
- Apply Preprocessing.
- Apply Segmentation.
- Apply Feature Extraction.
- Extract Digital character.

6.5 INTEGRATION TESTING

Integration testing integrates individual modules and tested as a group. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system for testing.

6.6 SYSTEM TESTING

Validation testing -

The process of evaluating software during or at the end of the development process in to determine whether it satisfies specified requirements.

GUI Testing -

GUI testing is the process of testing a product's graphical user interface to ensure it meets its specification, ensuring the navigation between icons/buttons with source code.

6.7 SUMMARY

This program has been tested at different phases of testing life-cycles. Multiple handwritten characters were scanned and tested. The hand written characters are predicted correctly.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSIONS AND FUTURE ENHANCEMENTS

Many regional languages throughout world have different writing styles which can be recognized with HCR systems using proper algorithm and strategies. We have learning for recognition of English characters. It has been found that recognition of handwritten character becomes difficult due to presence of odd characters or similarity in shapes for multiple characters and the difficulty in the process of segmentation. But however using the effective and efficient ways the result which was got was correct up to more than 90% of the cases. Scanned image is pre-processed to get a cleaned image and the characters are isolated into individual characters.

Preprocessing work is done in which normalization, filtration is performed using processing steps which produce noise free and clean output. Managing our evolution algorithm with proper training, evaluation other step wise process will lead to successful output of system with better efficiency. Use of some feature selection method through neural network will provide better recognition result of English characters. This work will also be helpful to the researchers for the work towards other script.

This work can be further extended to the character recognition for other languages. It can be used to convert the fax and news papers into text format. It can be used in post office for reading postal address. With new advanced technologies in future we could find more ways to make the handwritten character recognition faster using neural networks.

7.2 LIMITATIONS

- In the area of handwritten character recognition, the original image scanned should be a single character and not whole text.
- The use of CNN makes the process slower because of the maxpool layer.
- Many languages have special characters, and unless the correct HCR software is loaded, those characters can be lost or incorrectly recognized.

7.3 SUMMARY

We have successfully developed Handwritten character recognition (Text Recognition) with Python, Tensorflow, and Machine Learning libraries.

The Handwritten characters have been recognized with more than 97% test accuracy. This can be also further extended to identifying the handwritten characters of other languages too.

REFERENCES

[1] S S Sayyad, Abhay Jadhav, Manoj Jadhav, Smita Miraje, Pradip Bele, Avinash Pandhare, *"Devnagiri Character Recognition Using Neural Networks"* ,International Journal of Engineering and Innovative Technology (IJEIT)Volume 3, Issue 1, July 2013.

[2] Gunjan Singh,Sushma Lehri, *" Recognition of Handwritten Hindi Characters using Back propagation Neural Network*, International Journal of Computer Science and Information Technologies ISSN 0975-9646, Vol. 3 (4) , 2012,4892-4895.

[3] Miroslav NOHAJ, Rudolf JAKA, *"Image preprocessing for optical character recognition using neural networks"* Journal of Patter Recognition Research, 2011.

[4] Nisha Sharma et al, *"Recognition for handwritten English letters: A Review"*International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 7, January 2013.

[5] 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, Feb 2011.

[6] Megha Agarwal, Shalika, Vinam Tomar, Priyanka Gupta, *"Handwritten Character Recogition using Neural Network and Tensor Flow"* International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, Issue- 6S4, April 2019.

[7] M.Rajalakshmi, P.Saranya, P.Shanmugavavidu, *"Pattern Recognition - Recognition of Handwritten Document Using Convolutional Neural Networks"* IEEE International conference on Intelligent techniques and control, 2019.

[8] S.E. Benita Galaxy , S. Selvin Ebenezer, *"Enhancement of segmentation and zoning to improve the accuracy of handwritten character recognition"* International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016.

[9] Saptadeepa Kalita, Diwakar Gautam, Ashok Kumar Sahoo, Rajiv Kumar, *"A Combined Approach of Feature Selection and Machine Learning Technique for Handwritten Character Recognition"* Proceedings of 4th International Conference on Cyber Security (ICCS) 2018.

[10] Sushama Shelke, Shaila Apte, *"Performance Optimization and Comparative Analysis of Neural Networks for Handwritten Devanagari Character Recognition"* (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7, 2016.

[11] Singh, Sameer, Mark Hewitt, *"Cursive Digit And Character Recognition on Cedar Database"*, Pattern Recognition, 2000. Proceedings. 15th international conference on. Vol. 2. IEEE 2000.

[12] Salvador España-Boquera, Maria J. C. B., Jorge G. M. and Francisco Z. M., *"Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models",*

IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 4, April 2011.

[13] Sandhya Arora, *"Combining Multiple Feature Extraction Techniques for Handwritten Devnagari Character Recognition",* IEEE Region 10 Colloquium and the Third ICIIS, Kharagpur, INDIA, December 2008.

[14] Nafiz Arica, and Fatos T. Yarman-Vural, *"Optical Character Recognition for Cursive Handwriting",* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.24, no.6, pp. 801-113, June 2002.

[15] Bluche, T., Ney, H., & Kermorvant, C. (2013, August). *"Feature extraction with convolutional neural networks for handwritten word recognition"* In Document Analysis and Recognition (ICDAR), 2013 12th International Conference on (pp. 285-289). IEEE.

[16] Wu, C., Fan, W., He, Y., Sun, J., & Naoi, S. (2014, September). *"Handwritten character recognition by alternately trained relaxation convolutional neural network"* In Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on (pp. 291-296). IEEE.

[17] Szirányi, T., & Csicsvári, J. (1993), *"High-speed character recognition using a dual cellular neural network architecture (CNND)"* IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 40(3), 223-231.

[18] Tripathi R.C,Kumar.V. ,"*Character Recognition : A neural Network Approach"* Proceedings published in international journal of computer Applications, pp.17-20, 2012.

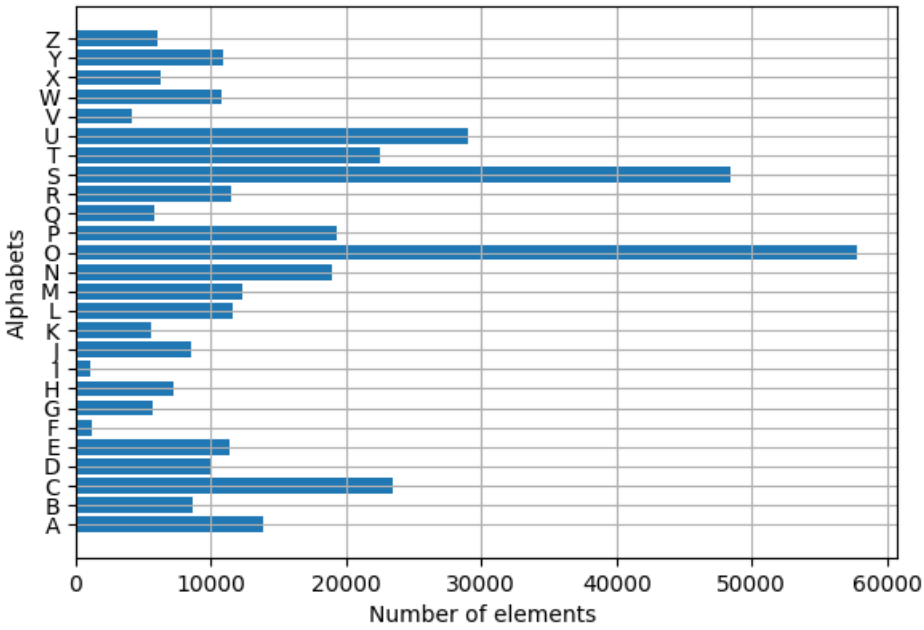
[19] Jibu Mathew C, Ravi C. Shinde, Prof. C. Y. Patil, "Segmentation Techniques for Handwritten script Recognition System" IEEE, International Conference on Circuit, Power and Computing Technologies [ICCPCT], 2015.

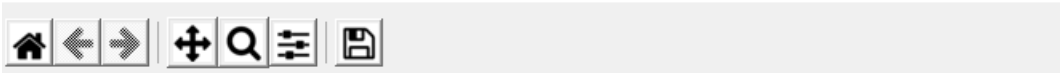
[20] Dellepiane, S. "Image segmentation: errors, sensitivity, and uncertainty." In Engineering in Medicine and Biology Society, 1991. Vol. 13: 1991, Proceedings of the Annual International Conference of the IEEE, pp. 253-254. IEEE, 1991.

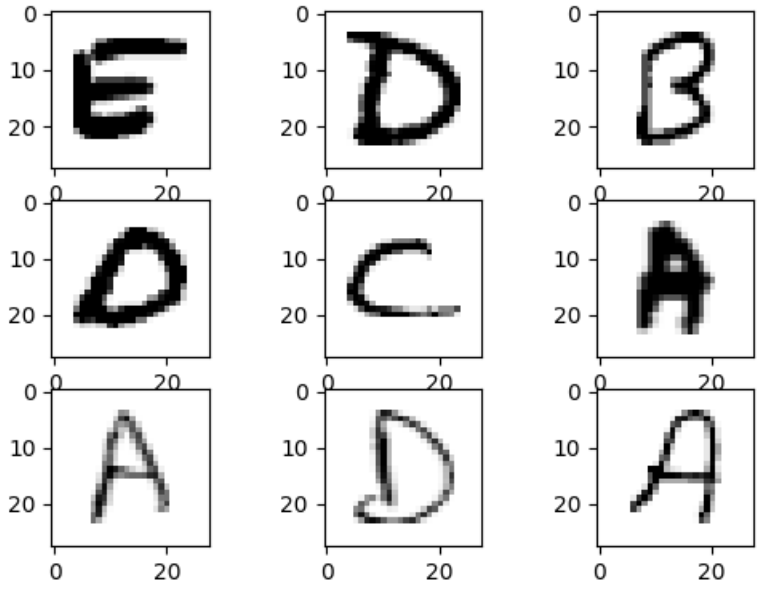
APPENDICES

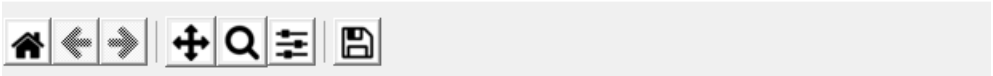
APPENDIX 1: SCREENSHOTS

```
C:\HCR\HCRnew>python hcrnew.py
2021-07-23 13:25:18.874343: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dyna
mic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2021-07-23 13:25:18.874732: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if
 you do not have a GPU set up on your machine.
     0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  ...  0.640  0.641  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  0.0  0.0  ...    0.0    0.0    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  0.0  0.0  ...    0.0    0.0    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  0.0  0.0  ...    0.0    0.0    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  0.0  0.0  ...    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
0.0
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
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
6  0.0  0.0  0.0  0.0  0.0  0.0  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    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    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    0.0    0.0    0.0    0.0
0.0
```


Alphabets
Z
Y
X
W
V
U
T
S
R
Q
P
O
N
M
L
K
J
I
H
G
F
E
D
C
B
A
0
10000
20000
30000
40000
50000
60000
Number of elements







```
[10 rows x 785 columns]
Train data shape:  (47999, 28, 28)
Test data shape:  (12000, 28, 28)
New shape of train data:  (47999, 28, 28, 1)
New shape of test data:  (12000, 28, 28, 1)
New shape of train labels:  (47999, 26)
New shape of test labels:  (12000, 26)
2021-07-23 13:32:46.318582: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not
load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2021-07-23 13:32:46.319008: W tensorflow/stream_executor/cuda/cuda_driver.cc:326] failed call to cuIni
t: UNKNOWN ERROR (303)
2021-07-23 13:32:46.441232: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA
 diagnostic information for host: kalyani-pc
2021-07-23 13:32:46.442603: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: kalya
ni-pc
2021-07-23 13:32:46.706549: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binar
y is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions
in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-07-23 13:32:50.217118: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of th
e MLIR Optimization Passes are enabled (registered 2)
1500/1500 [==============================] - ETA: 0s - loss: 0.0982 - accuracy: 0.9760
```

```
s: 0.0317 - val_accuracy: 0.9912
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 64)                32832
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 26)                3354
=================================================================
```

```
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 64)                32832
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 26)                3354
=================================================================
Total params: 137,178
Trainable params: 137,178
Non-trainable params: 0
_________________________________________________________________
The validation accuracy is : [0.9911666512489319]
The training accuracy is : [0.9760411381721497]
The validation loss is : [0.03173509240150452]
The training loss is : [0.0982300341129303]
(12000, 28, 28, 1)
```