DEVANAGARI CHARACTER RECOGNITION USING NEURAL NETWORKS

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Tavishi Sharma(161310)

Under the supervision of

Prof. Dr. Mrityunjay Singh

to

Department of Computer Science & Engineering and Information Technology

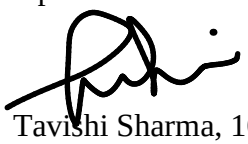
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"**Devanagari Character Recognition using Neural Networks**"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from January 2020 to May 2020 under the supervision of **Dr. Prof. Mrityunjay Singh.**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.



Tavishi Sharma, 161310

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

(Supervisor Signature)

Dr. Mrityunjay Singh

Dated:

ACKNOWLEDGEMENT

My special gratitude to my project guide **Prof. Dr. Mrityunjay Singh** for his inspiration, adroit guidance and constant supervision throughout the project.

I am very grateful to all the professors and lecturers of our department for their cooperation and keen interest throughout this project.

My sincere thanks to all my friends who helped me during this project.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

An Ancient Script- Devanagari is used for over 120 spoken Indo-Aryan languages, including Hindi, Nepali, Maithili, Marathi, Bhojpuri, Awadhi and Newari. Millions of people in India use this script for writing documents in Marathi and Hindi. This script has also been used for writing the Indian mythology. Due to so much importance of this script, handwritten devanagari character recognition has gained popularity over years.

This project is also an attempt to study the algorithms and then implement them to create a model for efficient prediction and recognition of handwritten devanagari characters.

CHAPTER 1

INTRODUCTION

1.1 Project background

The technique of recognition of handwritten characters has been amongst the most captivating and challenging research regions of image processing and pattern recognition in the ongoing years. HCR can be defined as the classification of data based on knowledge already gained or information obtained from different handwritten digits or characters.

It is the potential of a computer to obtain and then understand handwritten characters originating from photographs, paper documents etc. The written text which is in the form of an image can be inputted off line by using the technique of optical scanning (optical character recognition)from a piece of paper or smart recognition of words. The other procedure is to sense the moving of pen tip on line, for example by using a hardware device which is pen based. The main goal of a handwriting recognition system is to predict the handwritten characters correctly no matter how much variations in handwriting. . In HCR computer interprets handwritten input for example bank signatures, images of various handwritings etc. HCR involves various phases:

1. Preprocessing of data
2. Feature extraction
3. Classification.

Handwritten Character recognition can be broadly classified into these two subtypes:

1. On-line character recognition
2. Off-line character recognition.

The image of any kind of handwritten text can be sensed using “Off-line character recognition”. It also performs the automatic conversion of text into letter codes which can be understood by the computer system.

On-line recognition of character includes the automatic conversion of text data as soon as it is written on a particular digital device, in which a sensor senses the pen-tip movements. This type of data is known as digital ink and can be perceived as a digital outline of handwriting which can be sensed line by line. The obtained signal is transformed into letter codes which can be understood by the computer system.

The focus of this project is to identify various handwritten devanagari characters. It is a type of offline character recognition.

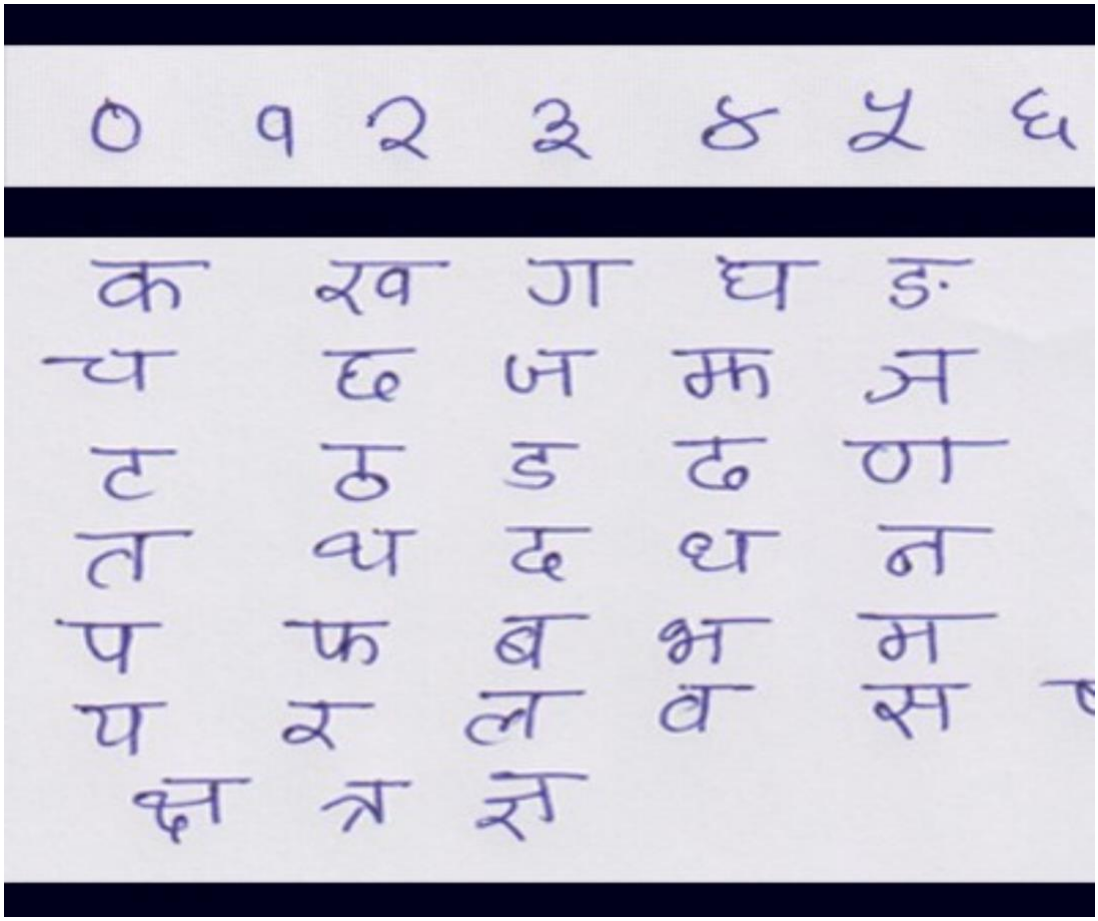


Figure 1.1 Devanagari handwritten characters

1.2 Problem Statement

Recognition of handwritten devanagari characters using convolutional neural networks for feature extraction and using random forest classifier and multilayer perceptron

classifier for classification of characters.

But, to implement this task there are many challenges, some of them being:

1. Challenging because there is a variation of same character due to the change of fonts and sizes. As different people have different handwritings.
2. Devanagari script characters have complicated shapes and characters are similar to one another which makes the recognition a challenging task.

Due to this the results in the recognition of character process are not very accurate. There may be chances of significant errors.

1.3 Objectives

The objectives of the project are:

1. To improve the techniques for devanagari handwritten character recognition.
2. To study and improve the algorithm in order to minimize the error.

1.4 Scope

The scope of this project consists of exploring the handwritten devanagari character recognition algorithm.

This project helps to understand the concept of handwriting recognition and to identify the best technique for improving the accuracy of the system. The example of application of handwriting recognition can be seen in bank cheques checking process, document reading, and postal code or address recognition as well as have potential in reading aid for the disabled ones. The applications of this system are very wide, hence ther is an urgent need of improving the efficiency and performance of the models.

1.5 Introduction to Neural Network

- Neural network is a technique based on biological neural networks that are a part of animal brain.
- It is composed of a group of nodes which are connected to each other, similiar to human neurological system.

Symbolical Meanings :-

- Circular nodes represent artificial neuron.
- Arrows represent the connection between output of one artificial neuron to input of another artificial neuron. These connections can transport signals between various neurons. This is analogous to synapses in biological brain.

Neural networks incorporate the following two fundamentals from human body:

- Neurons in animal brain areknown as nodes in case of Neural Networks.
- Synapses in animal brain are known as weights in Neural Networks.

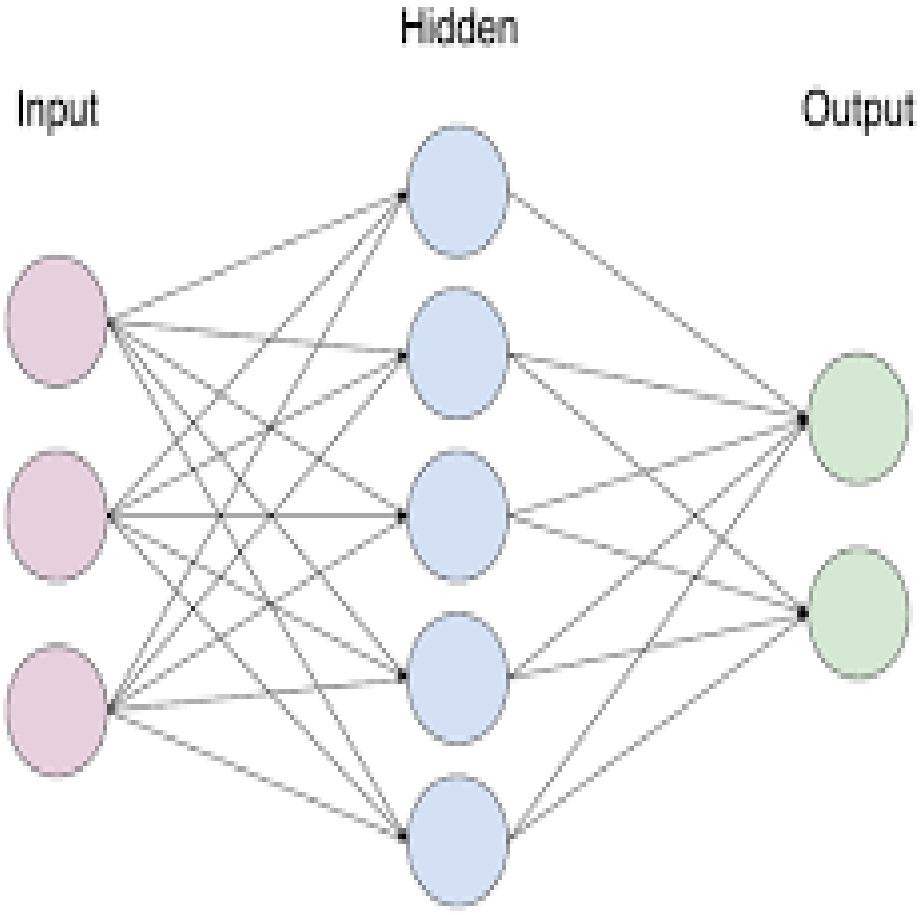


Figure 1.2 Neural Network

Biological neural networks in human brain is the inspiration behind this technique. The artificial neural network works in the same fashion as that of animal brain. Multiple signal are fed to a node and it has a capacity to make decisions about it and the outputs can be more than one. The output is inspired by the preceding logic and weight that is allocated to a particular node. It is always possible to assign weight to the nodes. Calculations can be executed at the nodes. So in this manner by putting calculations to use a logic constructed for recognition and prediction of characters.

Neural network works in three main domains :-

1. Supervised Learning :- In case of supervised learning a machine learns a function that maps an input to an output on the basis of previous input output pairs. Every example has an input object and the corresponding output. In this type of learning, the cost function is computed on the basis of input data. Difference between the data which is inputted and which is outputted is referred to as cost function. Ther is a posiibility of computation of cost function on the basis of data inputted.

2. Un-supervised Learning :- This type of learning is to train the artificial intelligence algorithm using data that is neither segregated nor labelled and training the algorithm to act on that data without any supervision. Here the cost function can or can not be computed on the basis of the input data.

3. Reinforcement Learning :- In this type of learning we have an idea that how our input is going to interact with environment. But we have no clue about our input. Only thing we have an idea about is its behavior within a particular environment, its results and effects.

1.6 Basic Neural Network

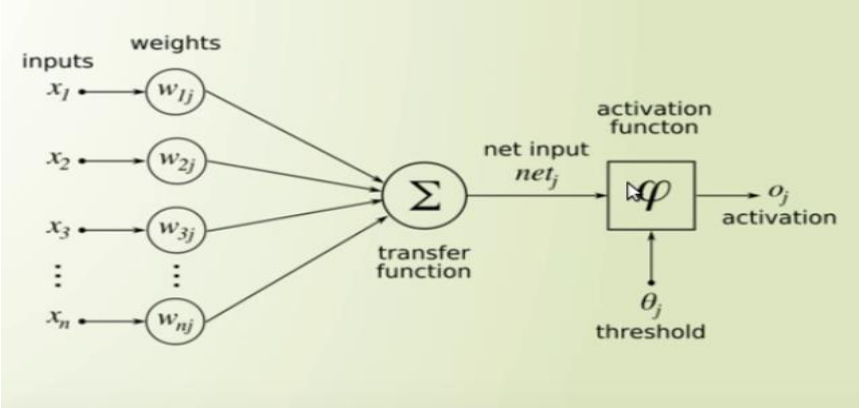


Figure 1.3 Basic Neural Network

Here, x1, x2…….xn are inputs.

Allocated weights for each input are W1j, W2j……,Wnj, there is also a transfer function which performs the task of combining every input and produces an appropriate result for the data inputted.

Transfer function performs the task of converting the input function to output function.

The result produced by a node depends upon the activation function used.

1.7 Multi Layer Perceptron (MLP)

Neural Networks are composed of different layers. This arrangement is called as Multi Layer Perceptron(MLP). A different function is performed by each layer on the received data.

These layers are:

- One input layer
- One output layer
- One or more hidden layers

Input layer receives the basic data. Extraction of data from one set of neurons and providing this output to another set of neurons is performed by hidden layers.

Capturing of all the minute details is done by hidden layers, resulting in exploring various relationships between different inputs. A result which is simple and easy to understand is produced by output layer.

1.8 Neural Network as Classifier Systems

One foremost application of artificial neural networks is that they can be very effective classifiers. NN consists of three layers, this arrangement is called as Multi-Layer Perceptron (MLP). For example, an ANN can be trained particularly for the classification of dogs vs cats. In this example, the two classes are cats and dogs.

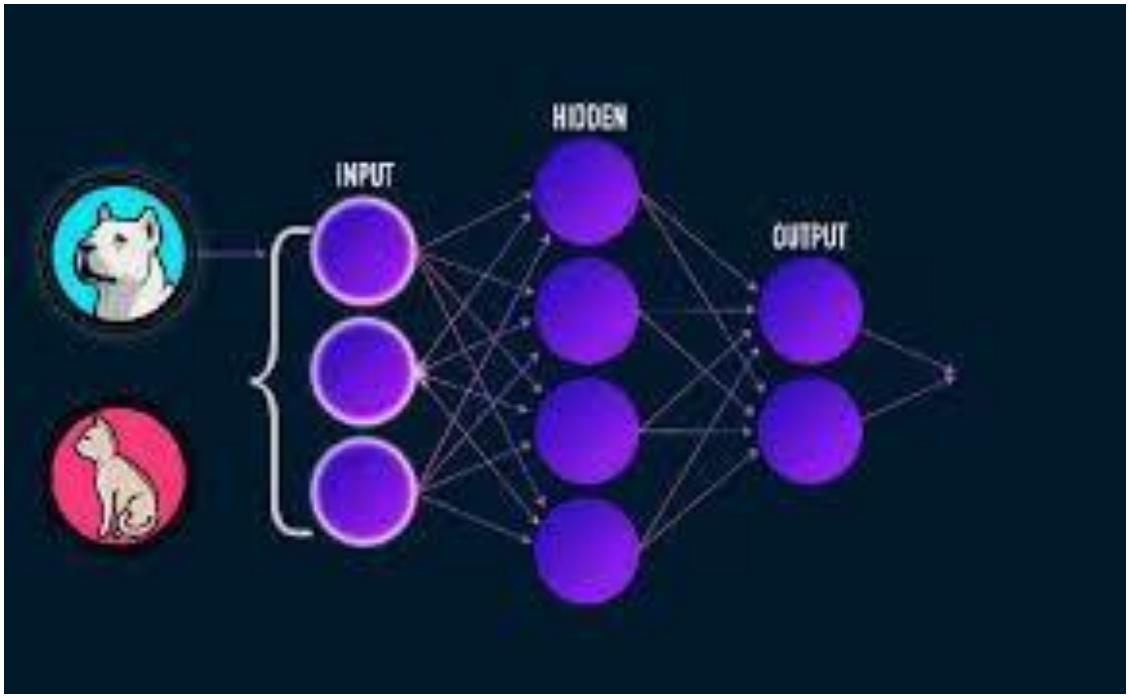


Figure 1.4 Neural network as a classifier system example

1.9 Introduction to Convolutional Neural Network

- Convolutional neural network is a subtype of deep neural networks. Their foremost application is analysis of visual imagery.
- In Neural Networks, CNN is the most widely applied for recognition of images,

classification of images and face detection and recognition techniques.

- It is a subtype of Neural Networks. Data is treated as spatial in this subtype of neural network. The nodes are not connected every node in the preceding layer rather nodes are connected to those nodes which are close to them and all have the same weight.
- Complex images get simplified and become easier to understand after feature extraction using cnn.

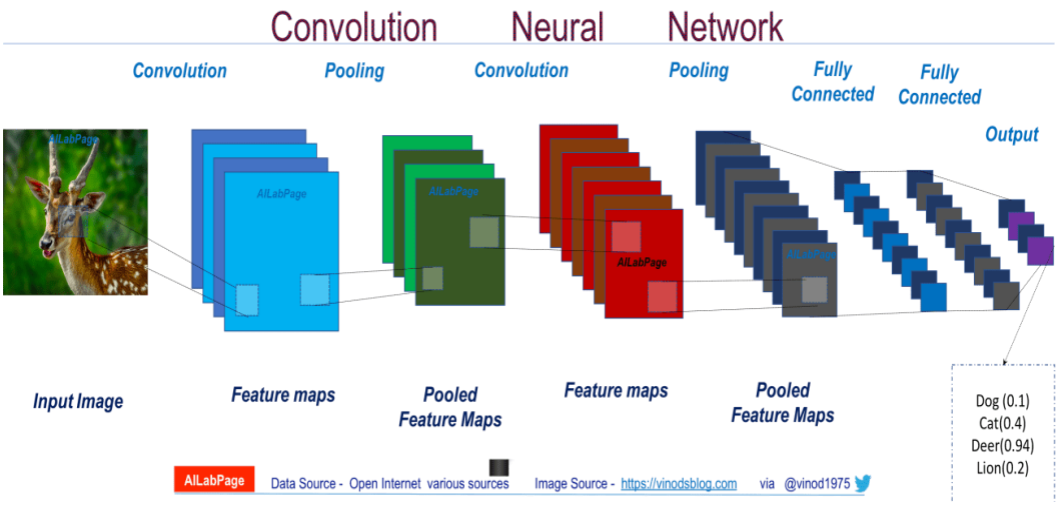


Fig 1.5 Convolutional Neural Network Layers

1.10 Why use Neural Networks?

- Neural network have the potential to learn on their own and generate the output that is just not confined to the input provided to them.
- Rather than storing the input in a database it uses its own network to store the input. Hence, the data loss doesn't affect its working.
- It can also be used for the extraction of patterns that are too complex and hard for human to understand or recognize.
- A trained neural network can be considered as an expert in the field of information it has been given. This trained neural network can be used to provide solutions given to new situations of interest and answer what if questions. Other Advantages Include:

1. Adaptive Learning :- It is an ability to learn how to perform tasks based on the data provided for experience in the beginning or during the training of data.

2. Real Time Operation :- ANN calculations can be performed alongside. Dedicated hardware devices are also being produced which have these capabilities.

3. Self-Organization :- An ANN has the capability of creating its own representation or organization of the information which is received by it during the time of training.

1.11 Introduction to Random Forest Classifier

- A random forest is actually a meta estimator. Its main purpose is to fit various decision tree classifiers on a number of subsamples of a particular dataset.
- This classifier uses averaging for the pupose of improving efficiency of prediction and also for the purpose of controlling overfitting.
- A random forest is basically a name given to a set of decision trees from subset which is selected from training set.
- Random Forest Tree Algorithm is a type of Ensemble algorithm. An ensemble algorithm is one in which one or more algorithms of of same or different type which are particularly used for classification are combined.

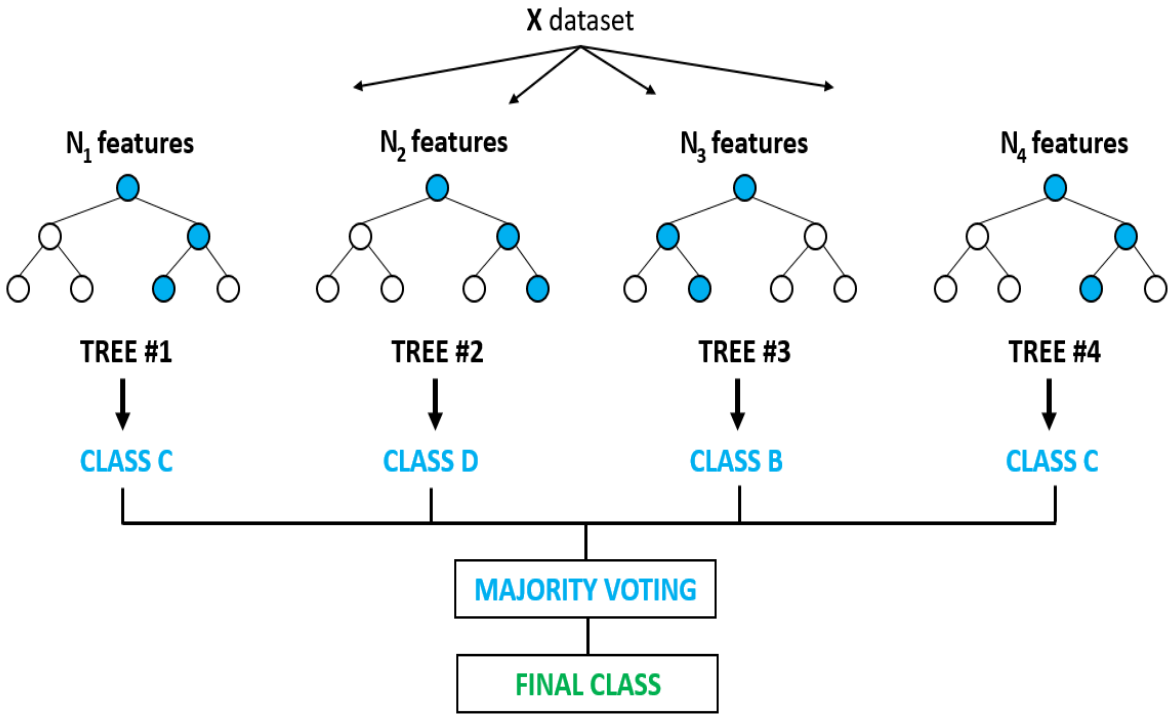


Fig 1.6 Random forest Classifier structure

1.11 Methodology

In this model, CNN has been used for feature extraction and Multi Layer Perceptron classifier and Random Forest Classifier have been used for classification purposes. The accuracy of these two classifiers has been compared in this model.

The approach used has been described as follows:

1. Configure and build a CNN network to extract features.
2. Extraction of features using Convolutional Neural Network. The output obtained here was further splitted into training data and validation data for the purpose of evaluating classifier performance.
3. Classification of 36 unique devanagari characters done using following classifiers:

- Multi-layer perceptron classifier
- Random Forest Classifier

4. Training of model on 36 unique devanagari characters with 1700 images of each character. Training was done using the features which were extracted in step 2.
5. Prediction of character. Test data created in step 2 was used for evaluating the performance of the classifiers.
6. Calculation of accuracy of the two classification models which have been used.

CHAPTER 2

LITERATURE SURVEY

Following research papers have been referred to in order to bring this project to a successful completion:

1. Ashwin S Ramtake, Millind E Rane **"Survey on Offline Recognition of Handwritten Devanagari Script"** In this paper HMM is used for feature extraction. This is followed by segmentation of image where the text is extracted from the image for the purpose of reading individual characters. Handwritten images have been classified using SVM and MLP. Hence it aims at achieveing automatic recognition of handwritten Devanagari characters.

2. Aradhana Malankara 1, Prof. Mitul M Patel 2 **"Handwritten Devanagari Script Recognition: A Survey"** This paper surveys the various research work done in this field. This paper describes the various methods for recognition of characters and the different applications of handwritten Devanagari character recognition systems.

3. Kapil Mehrotra, Saumya Jetley, Akash deshmukh, Swapnil Belhe **"Unconstrained Handwritten Devanagari Character Recognition using Convolutional Neural Networks"** This paper deals with offline recognition of handwritten Devanagari characters. CNN has been used for feature extraction and as a classifier. Training of CNN has been described in this paper.

4. **Arjun Singh , Kansham Angphun Maring[1]** In machine learning, one of the most important task is preprocessing of data. This paper involves various preprocessing techniques, such as, binarization, normalization, feature extraction, feature scaling that helps in reducing the inference time, that our classifier takes for building up the model. Many insights regarding Feature Extraction are also shared in this paper.

5. **G. E. Hinton and R. Salakhutdinov [4]** This paper provides a gist of how to deal with datasets with high dimentionality. It is common that, models with high dimentionality will lead to overfitting. Hence, we need to keep the features that are relevant. Relevancy can be decided on the type and the motive for which the model is trained. It is to be decided by one's instinct. In the crux, we need to keep the features in such a way that, all the relevancies are kept, as well as no overfitting or similar problem persists.

CHAPTER 3

SYSTEM DEVELOPMENT

Recognition of handwritten devanagari characters has been performed using Neural Networks. CNN has been used for extracting relevant features from the dataset. The features which are extracted using CNN are then fed to classifiers- Random Forest Classifier and Multi Layer Perceptron Classifier. Efficiencies of these two classifiers have been analyzed.

The following basic approach has been followed:

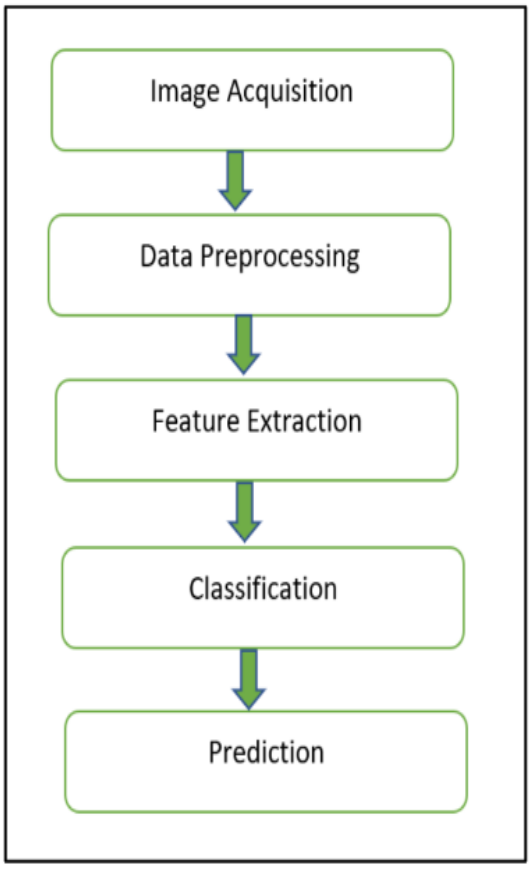


Figure 3.1 Character recognition system approach

3.1 UCI dataset

The dataset has been taken from UCI. It comprises training and testing data. Each of

these contains thirty six unique devanagari characters. All the images of one character have been put in a folder and that folder is named as the name of that particular alphabet in english. Every such folder consists of 1700 images of that particular character. Hence, in total there are 1700*36 which equals 61200 images in the dataset.

The dataset is in the form of 32*32 pixels grayscale images. For the purpose of training the model, these images have to be transformed into an array and then to be moved into an image matrix.

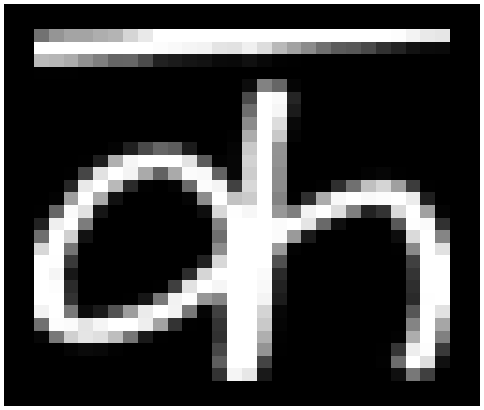


Fig 3.2 Various characters in the UCI dataset

3.2 Preprocessing Techniques

Data preprocessing is technique in data mining which is employed for transformation of raw data into a useful format.

3.2.1 Steps Involved in Data Preprocessing:

1. Data Cleaning:

   The dataset used for training and testing the model has many unnecessary information. To remove this not required information data cleaning is performed. It basically deals with the removal of missing data, noisy data etc. Following are the subparts of data cleaning:

- Missing Data:

  This situation happens when all the data is not present. It can be dealt with in a no of ways. Some of them are:

  - ✓ Ignoring the tuples:

    This method can be applied only when the dataset which is available with us is very large and within a tuple many values have gone missing.

  - ✓ Missing values filled up:

    In order to perform this task a no of procedures are available. Most probable value can be used to make up for the missing value.

- Noisy Data:

  Any data which has no meaning and which the machine is unable to understand and comprehend is referred to as noisy data. Data entry errors, faulty collection of data are the main causes for generation of noisy data. It has to be dealt with by applying any one :

  - ✓ Binning Method:

    This procedure is applied on data which is sorted and needs smoothening. Segmentation of data is done and then different methods are employed to finish the job. Every segment of data is dealt with differently. In order to complete the task, boundary values can be employed.

  - ✓ Regression:

    Under this, data smoothening is performed by fitting the data to a regression function. The regression technique employed may be linear (having only one independent variable) or multiple (having more than one independent variables)

  - ✓ Clustering:

    Grouping of similar kind of data in a cluster is referred to as clustering.

2. Data Transformation:

Mentioned procedure is performed for the purpose of transforming the data in proper forms which is relevant for further working on that data. It is performed in following ways:

- Normalization:
  This process is performed fpor the purpose of scaling the values of data within a specified range.
- Attribute Selection:
  Construction of new attributes from the existing ones, is the goal of performing this process.
- Discretization:
  The values of numerical attributes which are raw and not refined are replaced by intermediate levels in this process.
- Concept Hierarchy Generation:
  Conversion of attributes from lower to higher level in hierarchy is performed here.

3. Data Reduction:

In implementing any machine learning model we have to deal with huge amount of data. Analysis becomes a challenging task in such cases. As a solution to this problem we use data reduction techniques. Increasing the storage efficiency, reducing the data storage are the main aims of this step.

The various steps for this purpose are:

- Data Cube Aggregation:
  It leads to the construction of a data cube.
- Attribute Subset Selection:
  It is based on the principle that the attributes which are relevant should be retained and the rest attributes should all be discarded. Attribute selection can be performed in a manner that the attribute which has a p-value greater than significant level should be discarded.

- Numerosity Reduction
- Dimensionality Reduction

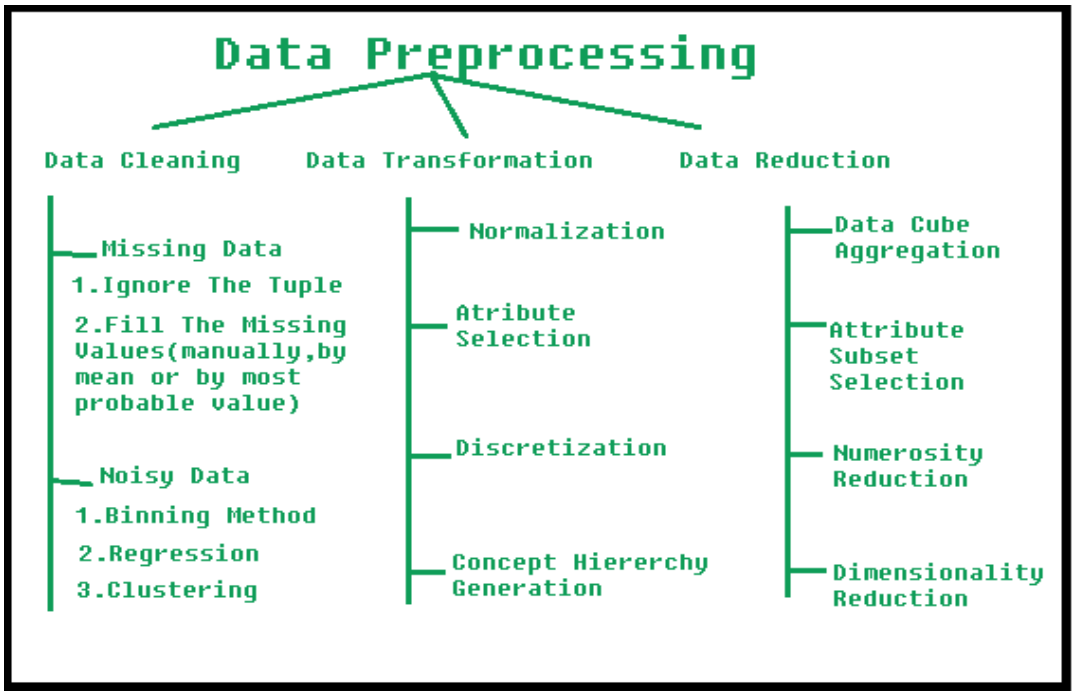


Fig 3.3 Data Preprocessing Techniques

There can be various techniques that can be applied on the given dataset so as to convert it into a suitable format for feeding it into the classifier. The foremost and most necessary technique is the removal of noise. It can be any of the multiple noise removal techniques. The noise can occur in any form, and at any place in the dataset. For ex, the blurred part in a picture can be marked as noise.

(a) **(b**

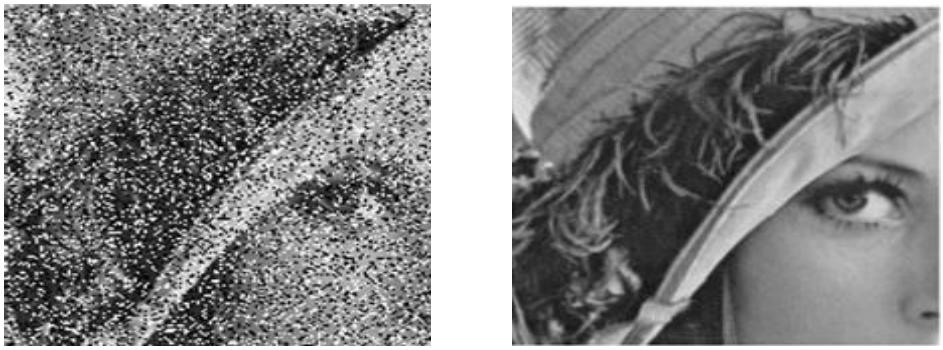


Fig 3.4 (a) normal image (b) preprocessed image

- There are various techniques that can be used to eliminate this problem. Median Filtering is one such technique. This is a widely used technique for getting rid of this type of noise.
- As the name suggests, it operates by choosing a bunch of pixels and later finding it's median. This value of median obtained, is then assigned to it's neighboring pixels.

For example :-

In this model, data is preprocessed by fetching the name of the character from the destination folder and then storing that name into a label array. This label array is further flattened and put into an image matrix for the purpose of training the model.

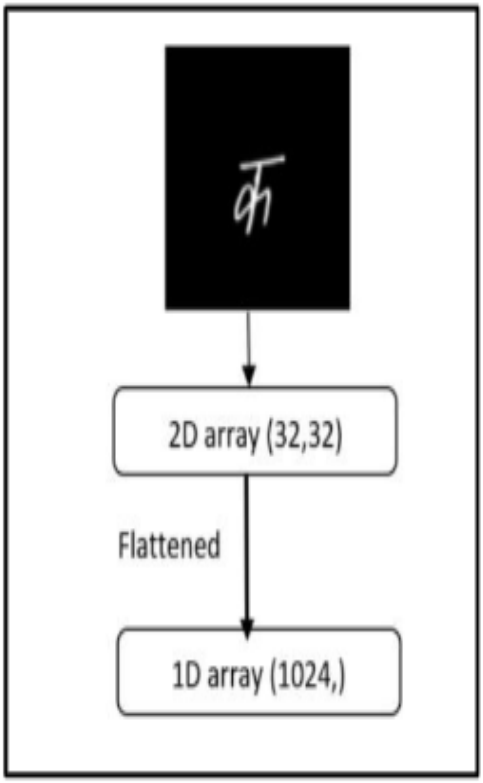


Figure 3.5 Image Preprocessing

- Data preprocessing is an important step in the model development process. It is a process used to remove outliers and standardize the data so that data can be

transformed to a form which is more easy for system to comprehend.

- It is a very important step as it leads to cleaner and more manageable data sets. The most relevant attributes must be used and the rest must be discarded.

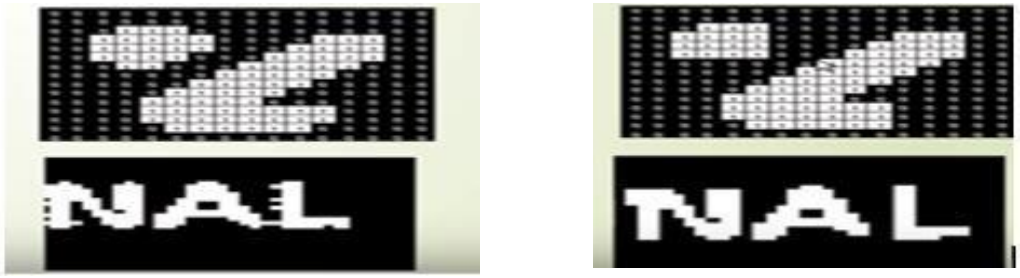
- Minimisation and eventual removal of conflicting, redundant and non useful data is the sole purpose of data preprocessing.

- It makes training a model easier and also leads to efficient storage utilization. It is an essential step in data mining. Without this process absurd results can be produced by the model.

3.3 Morphological Operations

It is another type of preprocessing technique. The crux operation happening here is, this technique is employed to eliminate, or helps in reducing little disturbances present in the image of the character, and also makes differentiating characters from other characters easier. The main aim of this technique is to reduce the noise.

.



(a) **(b)**

Fig3.6 (a) Original Image (b) Resulting Image

3.4 Feature Extraction

Feature Extraction is a process of describing the the shape information which is relevant and contained in a pattern. It is done in order to make the procedure of classifying easy. Manageable groups of data are formed from the initial set of unrefined data so as to make processing easy.

Feature extraction is basically a process of minimizing the actual number of resources neede to define a dataset which is quite large. During analysis performance of data which is complex the greatest problems appears from how many number of variables are involved. When greater amount of variables are involved it generallyleads to the requirement of large memory and also more computation power, also there is a possibility that of overfitting of training samples. It is a general assumption that feature extraction which is properly optimized leads to the construction of more effective models.

By using constructed sets of application-dependent features, results can be improved to a large extent.

The features of the image need to be selected in such a manner that the intra-class variability is decreased and the inter-class discriminability is increased in the feature space. CNN has by far been the best neural network for feature extraction.
The activation function “RELU” has been employed for input layers and the hidden layers and “sigmoid” activation function has been employed in the output layer.

3.4.1 Convolutional Neural Network:

It is a subtype of Neural Networks. Data is treated as spatial in this subtype of neural network. The nodes are not connected every node in the preceding layer rather nodes are connected to those nodes which are close to them and all have the same weight. Complex images get simplified and become easier to understand after feature extraction using cnn.
A CNN consists of one or many convolutional layers. This is followed by one or many fully connected layers. One of the benefits of CNN is that they can be easily trained and have really less parameters than other fully connected networks.

3.4.2 Architecture of CNN:

- A CNN comprises of a number of convolutional and other sub sampling layers. These are followed layers which are fully connected.
- A m*m*r image is an input to the convolutional layer where m is the height and width of the image and the no. of channels is r.
- k filters are present in a convolutuional layer. The size of each filter is n*n*q where n has dimensions smaller than that of the image and q may or may not be same as r. r varies from kernel to kernel.
- The filters produce a connected structure which are convolved with the image and k feature maps are produced of size m-n+1.
- Subsampling of each map is then performed. Before or after the subsampling is done sigmoidal non linearity and an additive bias is applied to every feature map.

A large no of fully connected layers are present after the convolutional layer. The layers which are densely connected are same as the layers in a standard multilayered neural network.

3.4.3 How CNN performs feature extraction?

- Just like ANN, CNN consists of multiple layers. But there are two layers which make it unique- Convolutional Layer and the Pooling layer.
- Just like other neural networks it also has the rectified linear unit or the RELU layer and a fully connected layer.
- Convolutional Neural Network works by placing a filter over an array of image pixels. This leads to the creation of a convolved feature map. The task of pooling layer is to reduce the sample size of a feature map. It results in a faster processing speed as the no of parameters which need to be processed are reduced. Output of pooling layer is a pooled feature map.
- These steps lead to feature extraction where in the system creates its own image of data according to the various mathematical rules

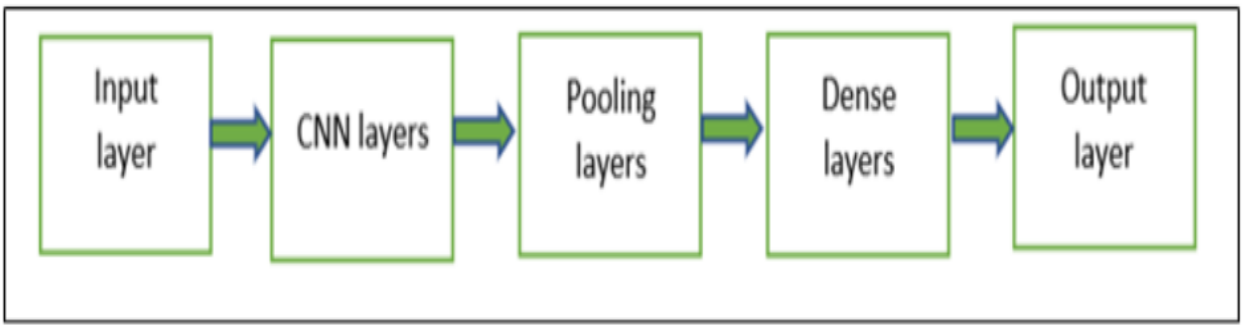


Fig 3.7 Neural Network Architecture implemented in our model

3.4.4 RELU function:

RELU stands for Rectified Linear Unit. RELU function is an activation function defined as the positive part of its argument: where input to neuron is x. It is also known as a ramp function. This activation function has been used for input and hidden layers in this model.

RELU function is the most commonly used activation function in machine learning and deep learning models. If it receives any negative input it returns zero, but if it recieves any positive value it returns that value back.

RELU function works perfectly in most models, and hence is used extensively.

RELU function solves two purposes:

- It helps a model to account for interaction effects: An interaction effect occurs when a variable affects a prediction in a different manner depending on the value another variable.
- It helps a model to account for non linear effects.

3.4.5 Sigmoid Function (Activation):

It is also a type of activation function. The main task of this function is to limit the output between zero and one. Hence this function is very helpful for predicting probabilities. This function is time and again used in machine learning models, particularly for testing the artificial neural networks, for understanding what output a node will produce. In this model, “sigmoid function” has been used in the output layer.

Sigmoid Functions are used widely in neural networks. What distinguishes the perceptron from sigmoid neuron or logistic neuron is the presence of the sigmoid function or the logistic function in the sigmoid neuron.

On one hand, the perceptron outputs discrete 0 or 1 value, a sigmoid neuron outputs a more smooth or continous range of values between 0 and 1.

It is required in Neural Networks that the output changes very slowly with input. Deep neural networks are what is in use presently and observing how a small change in the bias value or the weights associated with the artificial neurons affects the overall value of the output of the

the neuron is very important. A perceptron may have the outputs flipped suddenly with a small change in the input value, thus leaving the machine learning engineer baffled. To observe the tiny changes in the output to come at the correct value of input, we need a function to be applied on the dot product of weights and bias value so that overall output is smooth. But now, the function could have been any function **f()** that is smooth in nature like quadratic function,cubic function. The reason sigmoid function is chosen is that, exponential functions generally are similar to handle mathematically and since learning algorithms involve lots of differentiation, thus chosing a function that is computationally cheaper to handle is quite good.

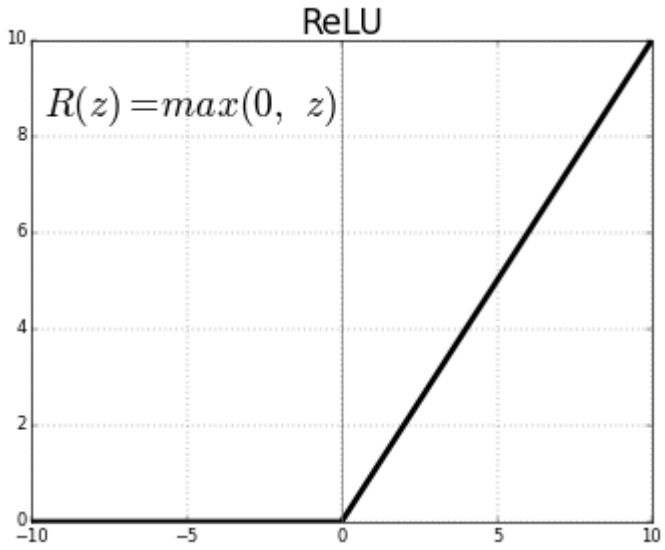


Fig 3.8 RELU activation function

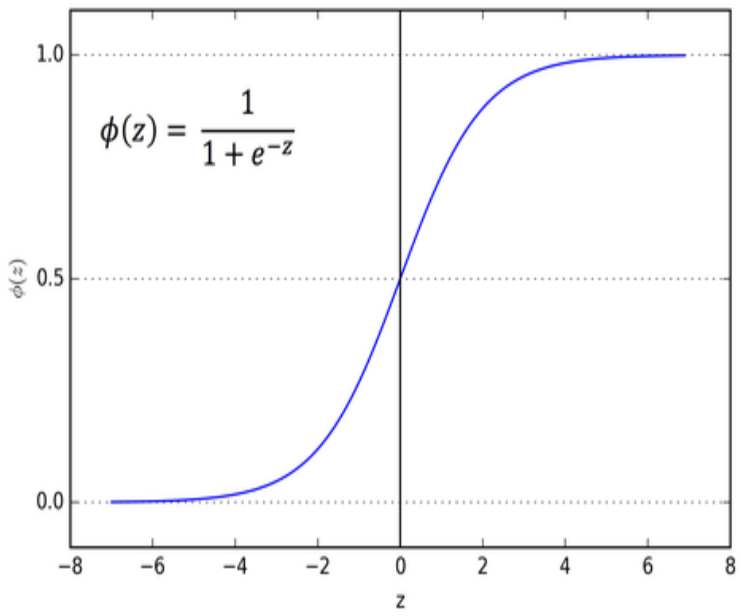


Fig 3.9 Sigmoid activation function

3.5 Classification

The features which are extracted from the Neural Network are fed to the classifiers.
Then the classifiers can be effectively used for the prediction of the target labels on the basis of extracted features.
Since there are 36 different target labels this problem is a multiclass classification problem.
Following two classifiers have been employed for the purpose of classification:

- Multi Layer Perceptron Classifier
- Random Forest Classifier

3.5.1 Multi Layer Perceptron Classifier:

- Multi Layer Perceptron is a type of supervised learning algorithm. The layer on the leftmost side which contains a set of neurons represents input features. The hidden layer also is made up of a no of neurons. The neurons which form a part of hidden layer

change the values from the preceeding layer on the basis of a weighted linear summation.

- Multi-Layer perceptron networks are sometimes called as “vanilla” neural networks, when there is single layer which is hidden.
- At least three layers of nodes are present in an MLP. Every node uses non linear activation function, except the input nodes.
- MLP is trained using a supervised learning technique called Backpropagation. The multiple layers present in it and the use of non linear activation function differentiate MLP from a simple neural network.
- They can be effectively used for distinguishing data which can not be separated in a linear fashion.
- MLPs are connected completely. Every node present in a particular layer is connected to every node in the succeeding layer with a weight wij.
- Learning happens in the perceptron by transforming weights after data processing at every stage takes place, based on the comparison of output with the correct prediction. This is a type of supervised learning, which happens through the technique of backpropagation.
- Multilayer perceptron does not mean a single perceptron which consists of multiple layers. Instead, it comprises a number of them which are arranged into subsequent layers. Moreover, MLP "perceptrons" can not be said to be perceptrons in the strictest sense possible. A true perceptron is actually one which does binary classification, however on the other hand an MLP neuron is unrestrained to either perform regression or classification.
- It can be effectively used for distinguishing data which can not be separated in a linear fashion.

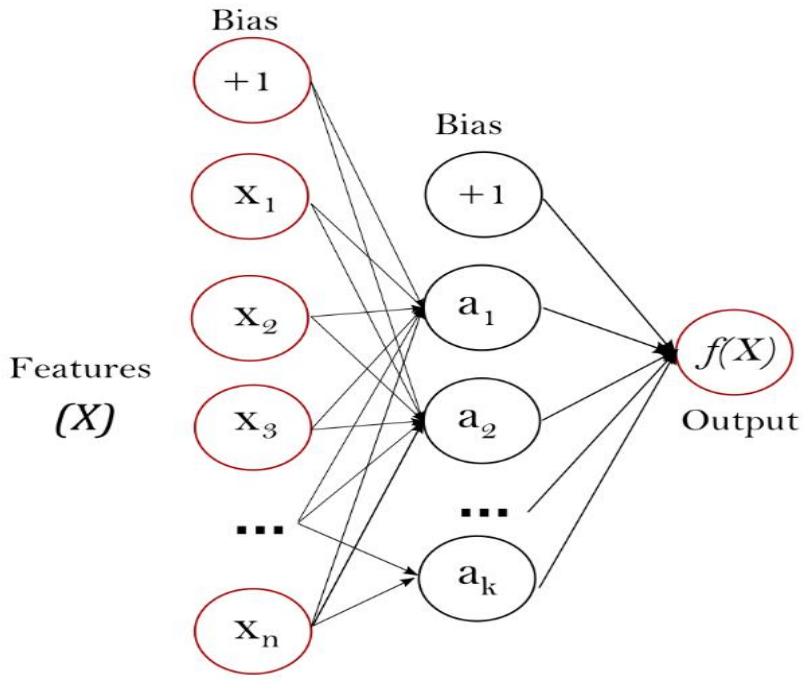


Fig 3.10 One hidden layer of MLP

3.5.2 Classification using MLP:

Class MLPClassifier is used for implementing a multi layer perceptron algorithm. It is trained using Backpropagation.

MLP is trained on two arrays:

- array X whose size is (n_samples, n_features). This array contains the training samples. These training samples are represented as floating point feature vectors.
- Array Y which is of size (n_samples). This array contains the target values for the training samples.

```
>>> from sklearn.neural_network import MLPClassifier
>>> X = [[0., 0.], [1., 1.]]
>>> y = [0, 1]
>>> clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
...                     hidden_layer_sizes=(5, 2), random_state=1)
...
>>> clf.fit(X, y)
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(5, 2), random_state=1,
              solver='lbfgs')
```

Fig 3.11 Training MLP

The advantages of Multi Layer Perceptron are:

- It has the potential to learn and implement non linear models
- It has the potential to learn in real time using partial_fit.

3.5.3 Random Forest Classifier:

- A random forest is actually a meta estimator. Its main purpose is to fit various decision tree classifiers on a number of subsamples of a particular dataset.
- This classifier uses averaging for the purpose of improving accuracy of prediction and also for the purpose of controlling overfitting.
- A random forest is basically a name given to a set of decision trees from subset which is selected from training set.
- Random Forest Tree Algorithm is a type of Ensemble algorithm. An ensemble algorithm is one in which one or more algorithms of of same or different type which are particularly used for classification are combined.
- Random forest, as its literal meaning implies, is made up of a huge no. of discrete decision trees functioning as an ensemble. Every discrete tree which is the part of a random forest puts forward an identification result and the class which has the maximum votes is the final prediction of the model.
- There is a very simple and powerful fundamental cocept behind random forest-the wisdom of crowds.

- It is based on the principle that a large no of models (trees) which are relatively uncorrelated and functioning as a single model are most likely to perform better than individual constituent models.
- The major reason behind good performance of a random forest algorithm is the low correlation between the individual models. This results in trees protecting the other trees from their individual errors.
- While there may be some trees whose decision is wrong and others maybe there whose decision is correct. So as a collective entity the trees are able to move forward in the correct direction.
- The prerequisite conditions for good performance of Random Forest Classifier are:
    - ✓ Low correlations between the trees and the errors committed by them.
- Random forest classifier can be employed for both regression as well as classification.
- Applications of random forests include image classification, recommendation engines and feature selection.
- Working of Random Forest Classifier algorithm can be put forth in four simple steps:
    - ✓ Selection of random samples from a given dataset.
    - ✓ Construction of a decision tree for every sample and obtain the result from every decision tree
    - ✓ Performing a vote for each result which is predicted.
    - ✓ The predicted result with maximum number of votes is chosen as the final prediction.

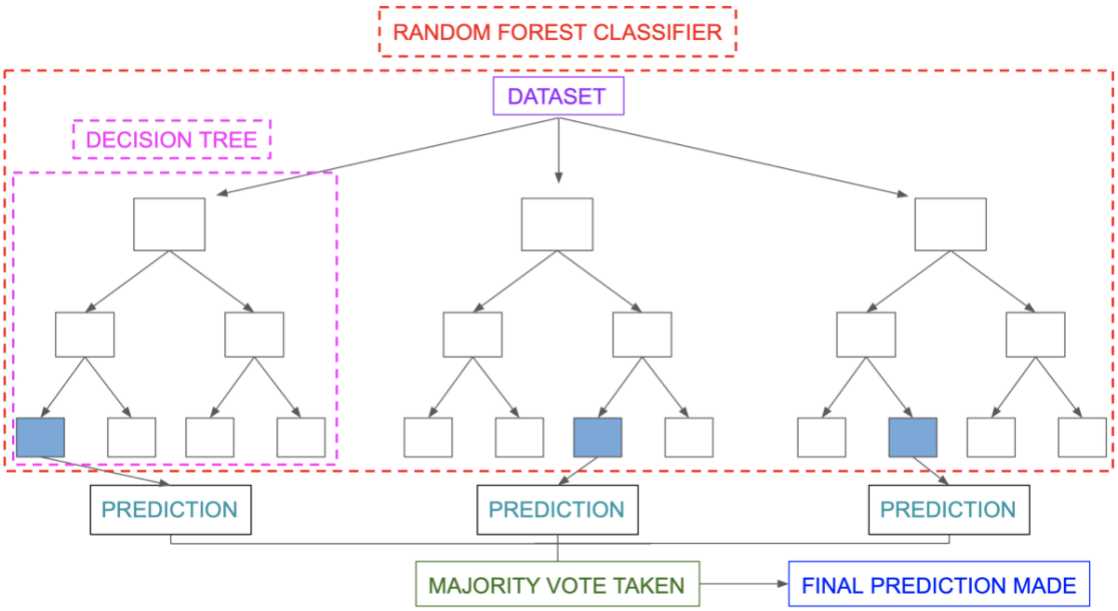


Fig 3.12 Random Forest Classifier Structure

Class RandomForestClassifier is used for implementing Random Forest Algorithm. It has various parameter values whose sole purpose is to control the tree size (eg. max_depth, in_samples_leaf etc). Controlling the size of the tree by specifying values for these parameters helps in reducing and monitoring memory consumption. Random permutation of features at each split takes place.

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.datasets import make_classification
>>> X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
>>> clf = RandomForestClassifier(max_depth=2, random_state=0)
>>> clf.fit(X, y)
RandomForestClassifier(...)
>>> print(clf.predict([[0, 0, 0, 0]]))
```

Fig 3.13 Implementing Random Forest Classifier

3.6 Prediction

The trained model is then used and tested. Test data which was separated from the dataset

in initial steps is used for testing the model. This test data was employed for the purpose of assessing the performance of the classifiers. The model is trained on the images of all the devanagari characters. The model is then employed for the prediction of characters with high accuracy.

3.7 Evaluation of classifiers

After training the classifiers, using features which were extracted using cnn, testing data was employed to evaluate the performance of the classification algorithms. Their accuracies of prediction were calculated and compared.

CHAPTER 4

PERFORMANCE ANALYSIS

- The performance of the model has been modest.
- This model was an attempt to modify the approach of handwriting recognition by merging the multi class classifiers and neural networks. In this model, Convolutional Neural Network has been configured for the purpose of extracting features from the images. The features which were read were further fed to multi class classifiers.
- The classifiers have been made to learn using these extracted features and the labels against them from training data.
- Testing data which was formed is used for assessing the performance of the classifiers.
- In order to better understand the variations, two different classifiers Random Forest Classifier and Multi Layer Perceptron Classifier have been used and evaluated.
- Performance of each of the classifier had considerable variations.

3.1 Key Findings

- There were small variations observed in the performance accuracies of the two classifiers which had taken the extracted features from CNN as input and had performed the prediction of target labels on test data.
- The accuracy of CNN model if employed for classification came out to be in a range between 70 to 80% for 10 epochs..
- On the other hand if the features extracted by CNN were fed to the classifiers, then their classification accuracy was found to be in the range between 72 and 80%.
- Hence using the classifiers along with CNN led to a little hike in accuracy.

4.2 Performance of the model at various steps is as follows:

1. Import necessary libraries.

```
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.image as mpimage
from PIL import Image
import numpy as np
import os
import random
from sklearn import ensemble,preprocessing
import keras.utils.np_utils
```

2. Import dataset which is in a folder and create an image matrix. Size of the image matrix is made to print.

```
Size of the image matrix =  62668800
```

3. Convert matrix into array

4. Separate the labels and features of the images

```
Features size =  (61200, 1024)
Labels size =  (61200,)
['yna' 'yna' 'yna' ..., 'jha' 'jha' 'jha']
```

5. Transformation of categorical variables using Onehot encoding and printing the mapping of one hot encoding

```
{'adna': 0, 'ba': 1, 'bha': 2, 'cha': 3, 'chha': 4, 'chhya': 5, 'da': 6, 'daa': 7, 'dha': 8, 'dhaa': 9, 'ga': 10, 'gha': 11, 'g
ya': 12, 'ha': 13, 'ja': 14, 'jha': 15, 'ka': 16, 'kha': 17, 'kna': 18, 'la': 19, 'ma': 20, 'motosaw': 21, 'na': 22, 'pa': 23,
'patalosaw': 24, 'petchiryakha': 25, 'pha': 26, 'ra': 27, 'taamatar': 28, 'tabala': 29, 'tha': 30, 'thaa': 31, 'tra': 32, 'wa
w': 33, 'yaw': 34, 'yna': 35}
```

6. Splitting the data into training and testing data. Size of the both is made to print.

```
Train size =  (42840, 1024)   (42840, 36)
```

7. Creating a CNN model for the purpose of feature extraction. The model summary is made to print.

Layer (type)	Output Shape	Param #
conv2d_120 (Conv2D)	(None, 30, 30, 4)	40
conv2d_121 (Conv2D)	(None, 28, 28, 4)	148
max_pooling2d_60 (MaxPooling	(None, 14, 14, 4)	0
conv2d_122 (Conv2D)	(None, 12, 12, 4)	148
conv2d_123 (Conv2D)	(None, 10, 10, 4)	148
max_pooling2d_61 (MaxPooling	(None, 5, 5, 4)	0
flatten_60 (Flatten)	(None, 100)	0
dense_59 (Dense)	(None, 20)	2020

8. Fitted the model on training data and tested on testing data

```
Train on 42840 samples, validate on 18360 samples
Epoch 1/10
42840/42840 [==============================] - 27s 636us/step - loss: 2.4581 - acc: 0.3239 - val_loss: 1.9653 - val_acc: 0.4409
Epoch 2/10
42840/42840 [==============================] - 25s 588us/step - loss: 1.7785 - acc: 0.4982 - val_loss: 1.6624 - val_acc: 0.5247
Epoch 3/10
42840/42840 [==============================] - 29s 678us/step - loss: 1.5558 - acc: 0.5585 - val_loss: 1.4855 - val_acc: 0.5782
Epoch 4/10
42840/42840 [==============================] - 28s 660us/step - loss: 1.4145 - acc: 0.5949 - val_loss: 1.3734 - val_acc: 0.6127
Epoch 5/10
42840/42840 [==============================] - 29s 688us/step - loss: 1.3105 - acc: 0.6266 - val_loss: 1.2918 - val_acc: 0.6333
Epoch 6/10
42840/42840 [==============================] - 28s 662us/step - loss: 1.2304 - acc: 0.6490 - val_loss: 1.2154 - val_acc: 0.6545
Epoch 7/10
42840/42840 [==============================] - 29s 688us/step - loss: 1.1653 - acc: 0.6640 - val_loss: 1.1561 - val_acc: 0.6699
Epoch 8/10
42840/42840 [==============================] - 30s 690us/step - loss: 1.1111 - acc: 0.6797 - val_loss: 1.1094 - val_acc: 0.6834
Epoch 9/10
42840/42840 [==============================] - 30s 706us/step - loss: 1.0655 - acc: 0.6915 - val_loss: 1.0694 - val_acc: 0.6902
Epoch 10/10
42840/42840 [==============================] - 27s 619us/step - loss: 1.0260 - acc: 0.7028 - val_loss: 1.0302 - val_acc: 0.7039
```

9. Displaying the accuracy of CNN model

```
scores = cnn.evaluate(x_test, T_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 70.39%
```

10. The features extracted using CNN are fed to the classifiers.

11. Create a generic function to run the classifier and predict accuracy.

12. Pass features to Random Forest classifier for prediction.

```
Random Forest Classifier starting ...
Time taken for prediction = 12.111731 seconds
CNN-Random Forest Accuracy: 71.7%
acc= 71.6542172425
```

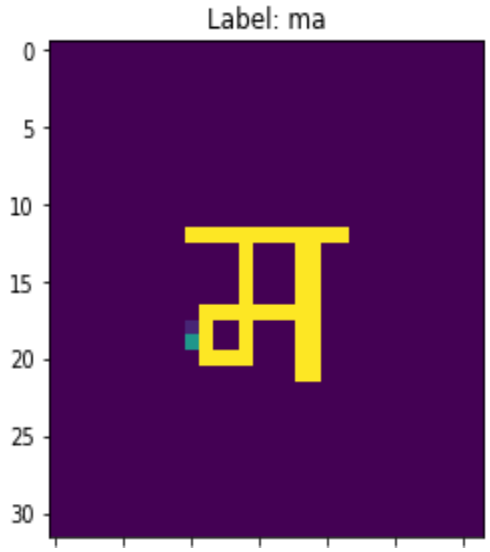
13. Pass features to Multi-Layer Perceptron Classifier for prediction

```
Multi-Layer Perceptron Classifier starting ...
Time taken for prediction = 77.378430 seconds
CNN-Multi-Layer Perceptron Accuracy: 77.3%
acc= 77.2642390289
```

14. Prediction of characters by the two classifiers:

```
img=Image.open('./testFiles/word1_1.png')
img=img.convert('L')
img=img.resize((32,32))
imgData = np.array(img)
plt.title("Label: ma")
plt.imshow(imgData)
img.size
flat=imgData.flatten()
np.shape(flat)
flat=flat.reshape(1,-1)
flat.shape
```

```
(1, 1024)
```

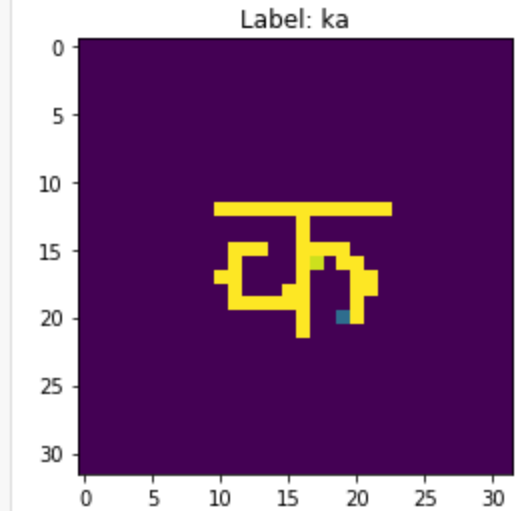



```
y_pred=partialDataRandomForestClassifier.predict(flat)
le.inverse_transform(y_pred[0])
```

```
'ma'
```

```
y_pred=partialDataMLPClassifier.predict(flat)
le.inverse_transform(y_pred[0])
```

```
'ga'
```




```
y_pred=partialDataRandomForestClassifier.predict(flat)
le.inverse_transform(y_pred[0])
```

```
'ba'
```

```
y_pred=partialDataMLPClassifier.predict(flat)
le.inverse_transform(y_pred[0])
```

```
'ka'
```

It can be noticed here that model is committing significant mistakes. This is due to the devanagari characters being similar to one another.

4.3 Performance Analysis of Random forest Classifier

- A random forest is an algorithm which fits various classifiers which are present in the form of decision trees on a number of sub samples of the dataset. It uses averaging technique to improve the efficiency of prediction.
- For a model which was made to learn and tested on less no. of devanagari handwritten characters the accuracy of Random Forest Classifier came out to be between 85% and 90%.
- However for a model which was made to learn and tested on all the 36 devanagari handwritten characters accuracy of Randome Forest Classifier came out to be between 68% and 70%.

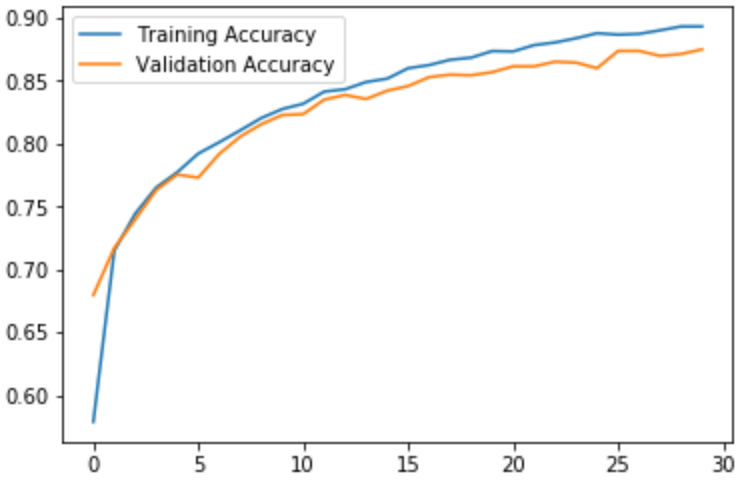


Fig Performance Analysis of Random forest Classifier

4.4 Performance analysis of Multi-Layer Perceptron Classifier

- It is amongst the most widely accepted classifiers. It is used for multiclass classification. It uses stochastic gradient descent for the purpose of optimizing log-loss function.
- A model which was trained and tested on few no. of devanagari handwritten characters displayed the accuracy of Multi-Layer Perceptron Classifier between 87% and 92%.
- However for a model which was trained and tested on all the 36 devanagari handwritten characters accuracy of Randome Forest Classifier came out to be between 74% and 80%.

Hence it can be concluded that Multi-Layer Perceptron Classifier predicts the handwritten Devanagari Characters in a better and a more efficient manner than the Random Forest Classifier.

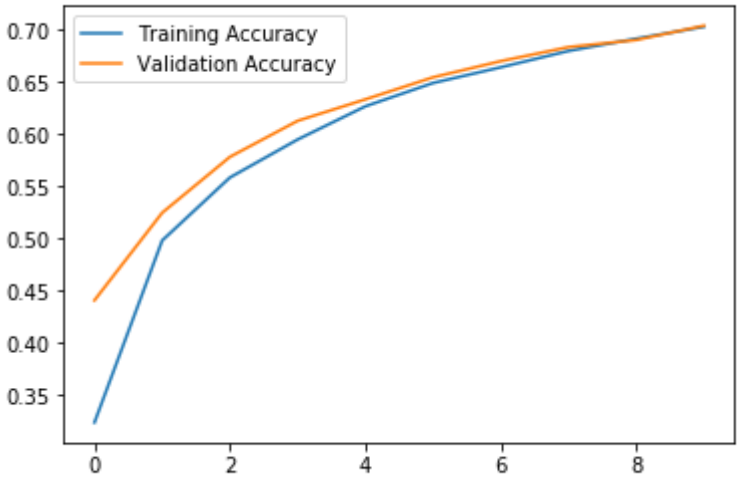


Fig. Performance analysis of MLP classifier

CHAPTER 5
CONCLUSION

5.1 Conclusion

In our country and other countries where devanagari script is used, a great no of historical documents and books are there which need to be converted into a digital form for following puposes:

- Improving accessibility
- Indexing and sharing
- Learning about the history and culture

Digitization or a process for the recognizing and predicting the handwritten devanagari characters would also help other communities of research in India which are particularly working in the fields of humanities and social sciences.

Handwritten character recognition continues to be a burning area of research. Recognition of handwritten Devanagari characters is quite a challenging job because of the similarities between the various characters. By using Neural Networks for the pupose of extraction of impotant features of the character image the task of classification performed by multi class classifiers has been simplified to a large extent.

To conclude, Image Processing, Neural Networks, Handwriting recognition, feature extraction are different popular fields of research and there is a great scope for improvement in these areas.

5.2 Future Scope

- The algorithms used for the prediction and recognition of handwritten devanagari characters can be improved and modified for better accuracy.
- There is a scope for the addition of new features for the improvrmrnt of accuracy of the model.
- Testing of these algorithms on large devanagari handwritten character dataset can be performed.
- The model can also be extended for the recognition of degraded text or characters which are broken.

- Digits recognition, recognition of characters which are half or compounded can also be incorporated to improve the efficiency of the model.

REFERENCES

- Ashwin S Ramtake, Millind E Rane **"Survey on Offline Recognition of**

Handwritten Devanagari Script"

- Aradhana Malankara 1, Prof. Mitul M Patel 2 **"Handwritten Devanagari Script Recognition: A Survey"**
- Kapil Mehrotra, Saumya Jetley, Akash deshmukh, Swapnil Belhe **"Unconstrained Handwritten Devanagari Character Recognition using Convolutional Neural Networks"**
- https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/
- https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

DEVANAGARI CHARACTER RECOGNITION USING NEURAL NETWORKS

ORIGINALITY REPORT

13%	6%	7%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Jaypee University of Information Technology Student Paper	2%
2	Natasha Jeppu, K. Chandrasekaran. "Extending Denoising AutoEncoders for Feature Recognition", 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018 Publication	1%
3	juxt.pro	1%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 14-7-2020

Type of Document (Tick): **PhD Thesis** | **M.Tech Dissertation/ Report** | **B.Tech Project Report** | **Paper**

Name: Tavishi Sharma **Department:** Computer Science **Enrolment No** 161310

Contact No. 9459217999 **E-mail.** tavishi.nmd@gmail.com

Name of the Supervisor: Dr Mrityunjay Singh

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters):

DEVANAGARI HANDWRITTEN CHARACTER RECOGNITION USING NEURAL NETWORKS

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages = 52
- Total No. of Preliminary pages = 8
- Total No. of pages accommodate bibliography/references = 2

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 13 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Abstract & Chapters Details**	
	• All Preliminary Pages • Bibliography/ Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Page counts	
			File Size	

Checked by
Name & Signature **Librarian**

Please send your complete Thesis/Report in (PDF) & DOC (Word File) through your Supervisor/Guide at plagcheck.juit@gmail.com