HANDWRITING DETECTION USING NEURAL NETWORK

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

in

Computer Science and Engineering

By

Manik Nijhawan (151322)

Paras Singal (151233)

Under the supervision of

Mr. Himanshu Jindal

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 **"**Handwriting Detection Using Neural Network**"** 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 August 2018 to May 2019 under the supervision of **Himanshu Jindal** (Assistant Professor(Grade-2) and CSE).

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

Manik Nijhawan, 151322

Paras Singal, 151233

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

(Supervisor Signature)

Mr. Himanshu Jindal

Assistant Professor(Grade-2)

CSE

Dated:

ACKNOWLEDGEMENT

My special gratitude to my project guide **Himanshu Jindal** for his inspiration, adroit guidance, constant supervision and constructive in successful completion of 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

LIST OF TABLES

ABSTRACT

This undertaking report compares three models for handwriting detection using MNIST and EMNIST dataset. The result is a system which can recognize any handwritten number or alphabet. The 3 models which we compared are artificial neural network, random forest and XGBoost. We additionally experimented with different hyperparameters to maximize test accuracy and reduce overfitting as much as we could. MNIST and EMNIST are most common dataset available on the internet for handwriting detection. MNIST dataset contains 60000 training images and 10000 test images. EMNIST dataset contains 124800 training images and 20800 test images.

CHAPTER 1

INTRODUCTION

1.1 Introduction to Handwriting Detection

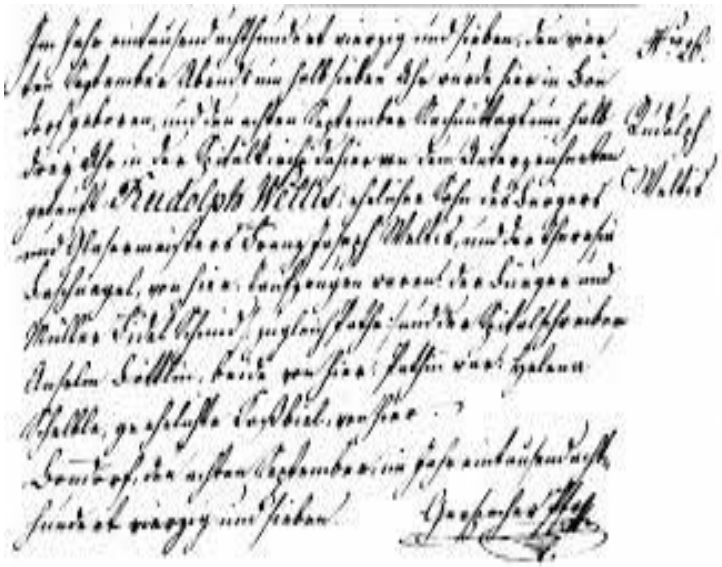
Handwriting detection is the ability of computer to receive and interpret handwritten input from sources such as photographs , touch screens, paper sheets, and other devices . The image of the written text may be sensed "off line" from a piece of paper using the principle of optical character recognition (OCR). Alternatively, the movements of the pen tip may be sensed "on line", for example by a pen-based computer screen surface, which is a generally easier task to perform as there are more clues available.

Its applications include storing written content digitally ,to convert digital signature into the one which computer can read and many other. Here we perform classification task using artificial neural network. The MNIST database of handwritten digits is considered for this task where MNIST is a large database of handwritten digits. It also handles formatting, segmentation into characters and finding the most suitable or the most suitable words.

Figure 1.1 Image of handwriting detection

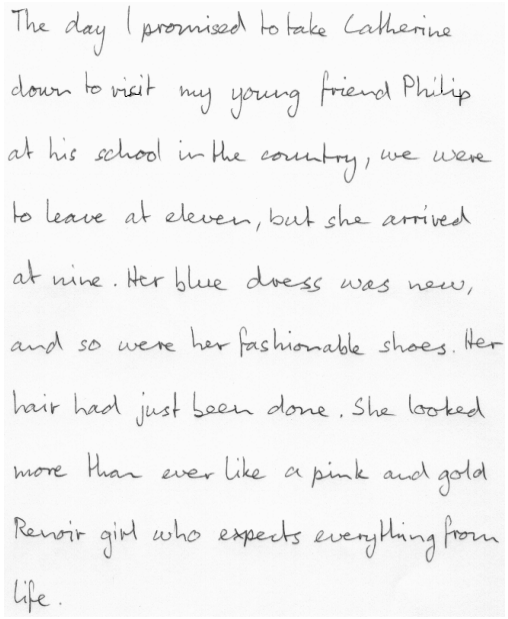
1.1.1 Examples of handwritten text

Here are a few examples of handwritten text. There are innumerous applications that can be developed with this technology.

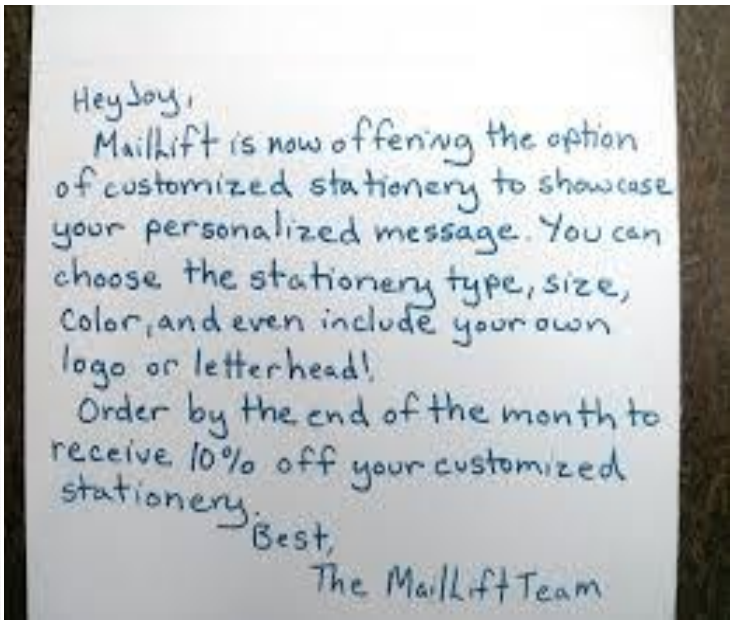


(a)

(b)



(c)



(d)

Figure 1.1.1 Handwritten text image (a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4

1.1.2 Use Case of handwriting detection in Postal Service

Hand Written Address Interpretation is a software system deployed by the United States Postal Service.Today more than 95% of the handwritten mail is sorted automatically. During the course of time almost all US states have adopted this technology. Here it shows one such use case of this technology.

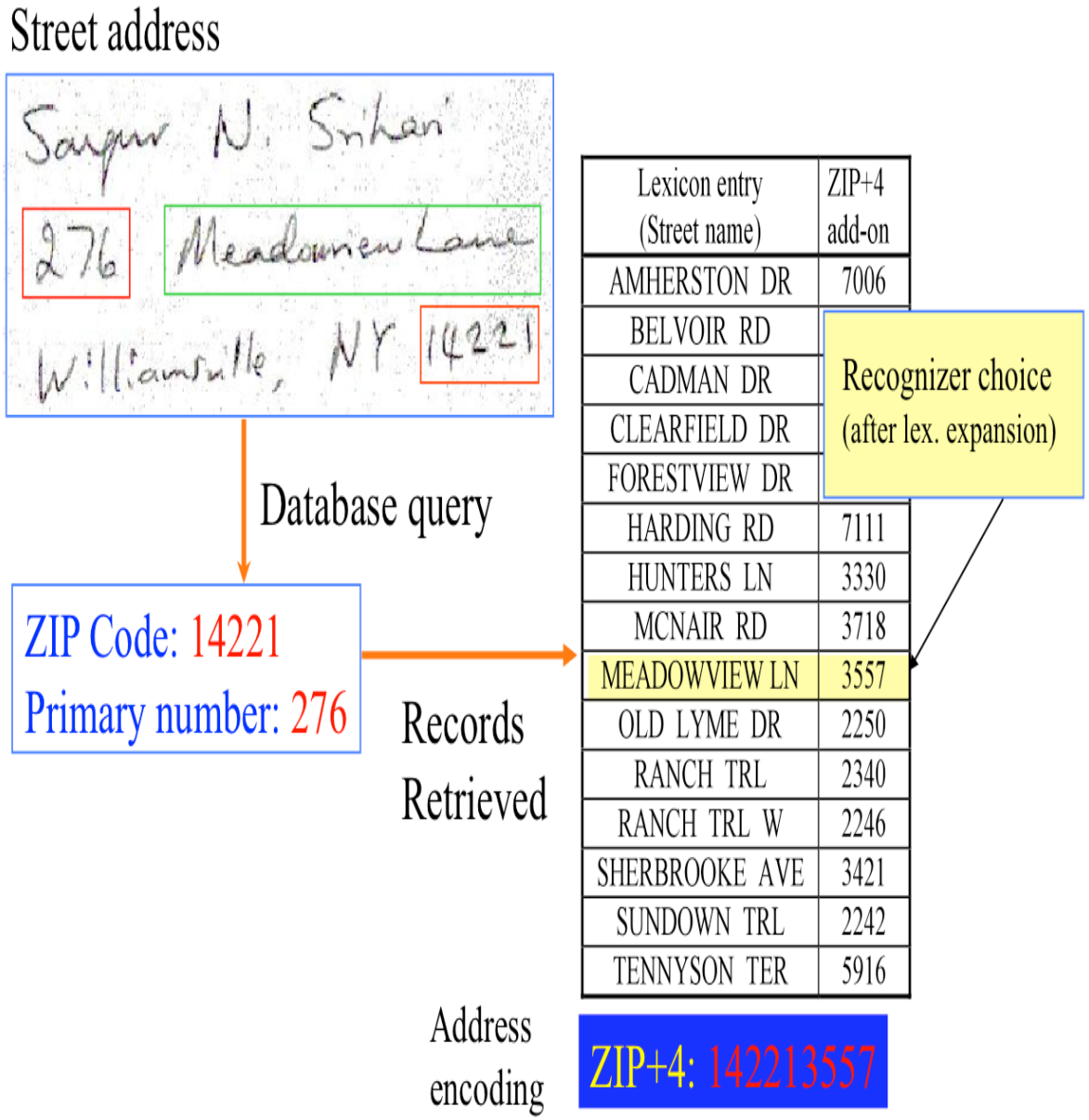


Figure 1.1.2 Use Case of handwriting detection in Postal Service

1.1.3 Use Case of handwriting detection in Author Identification

We an identify the author of a text by analyzing the given text. This is also known as handwritten biometric recognition.

1.2 Steps in handwriting detection

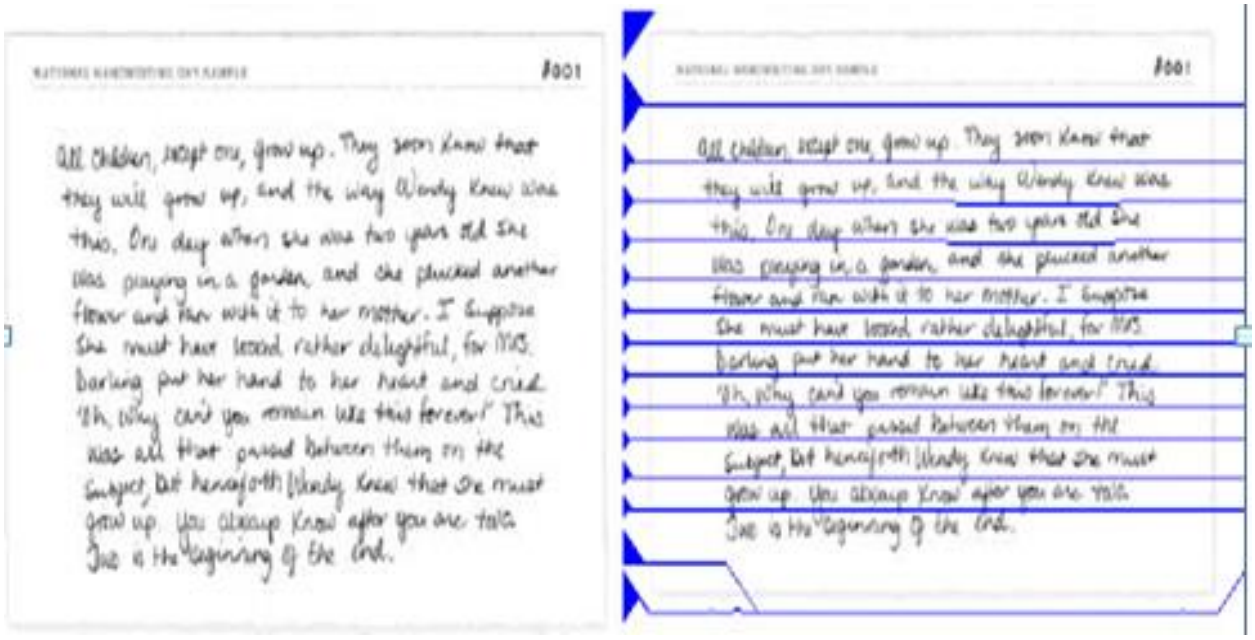
1. Line segmentation.

2. Character segmentation.

3. Pre-processing image to remove noise.

4. Apply suitable techniques to perform handwriting detection.

1.2.1 Line segmentation



(a) (b)

Figure 1.2.1 Line Segmentation (a) Original image (b) Resulting line segmentation image

Line segmentation is where you are separating the lines in a text. Suppose there are 10 lines in the text .First most task is to label the lines with numbering 1 to 10. Once you get the 10

different lines you perform various processes on each of the lines. This is crucial because the characters of handwritten text need to be isolated before applying recognition techniques.

Since different people have different styles of handwriting. Some people have slant handwriting and lines are not spaced uniformly, or when lines are very close to each other so all this become challenge in line segmentation.

Classical algorithms used here is edge detection techniques. Basically all the edge detection detection techniques can be modified to perform line segmentation for us.

1.2.2 Character segmentation

Figure 1.2.2 Character Segmentation

Once the line is segmented now we have only one line and the words in it and we need to seperated character by character from this line. Character segmentation is an operation that seeks to decompose an image of sequence of character into sub images. Some of the text are really huge while some are really small, some might have very close characters and some are far .So it all depends on individual and style of his/her handwriting. It is achieved by template matching ,extracting the distinguishing attribute of character image.

The scope of this project is you have a character and then you realize, then you learn and then you see which character it is and what letter it belong to. So we are not going to discuss line segmentation or character segmentation in this project. This is just a part of handwriting detection.

1.2.3 Pre-processing techniques

Talking about pre-processing techniques that need to be performed on handwritten text .The first pre-processing technique is noise removal technique. One of the most common noise these days is salt and pepper noise. This noise can occur in any form and can be seen as white and black dots on any image. When you capture a photo, these noise can occur due to environmental disturbances , due to dust particles in environment, due to dust on the camera lens and various other reasons.

(a) (b)

Figure 1.2.3 Median filtering (a) Original Image (b) Resulting Image

Effective noise reduction for this type of noise is Median filtering. This is most common used technique for this type of noise. It works by choosing a median of group of pixels and assigning its value to the neighbouring pixels.

For example :-

If we have a set of group of pixels “x”

x =[3 90 7 4]

So, the median filtered output signal “y” will be :-

y[1] = Median[3 3 90] = 3

y[2] = Median[3 90 7] = Median[3 7 90] = 7

y[3] = Median[90 7 4] = Median[4 7 90] = 7

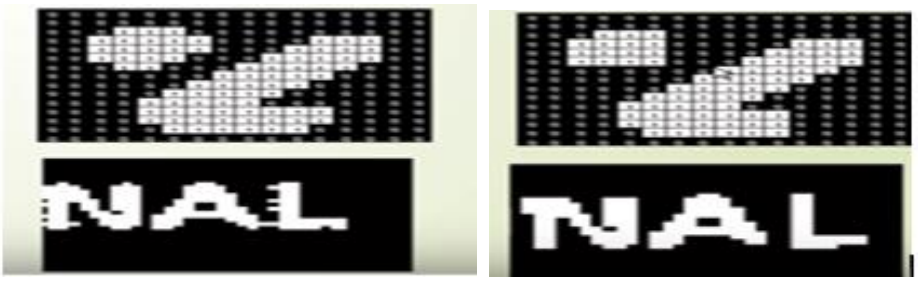
y[4] = Median[7 4 4] = Median[4 4 7] = 4

i.e. y=[3 7 7 4]

Hence in output signal ‘90’ get converted to ‘7’ and this will remove the white pixel on the image.

1.2.3.1 Morphological operation

The Second operation that can be performed for noise removal is morphological operation. It is used to remove or clean small protusions from the character and hence helps in smoothening of characters and thus helps in distinguishing the characters of characters. So it is one of the pre processing technique do after reducing.



(a) (b)

Figure 1.2.3.1 Morphological Operation (a) Original Image (b) Resulting Image

1.3 Introduction to Neural Network

Neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain.

Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.

NNs incorporate the two fundamental components of biological neural nets:

Neurons in brain is called nodes in NN

Synapses in brain is called weights in NN

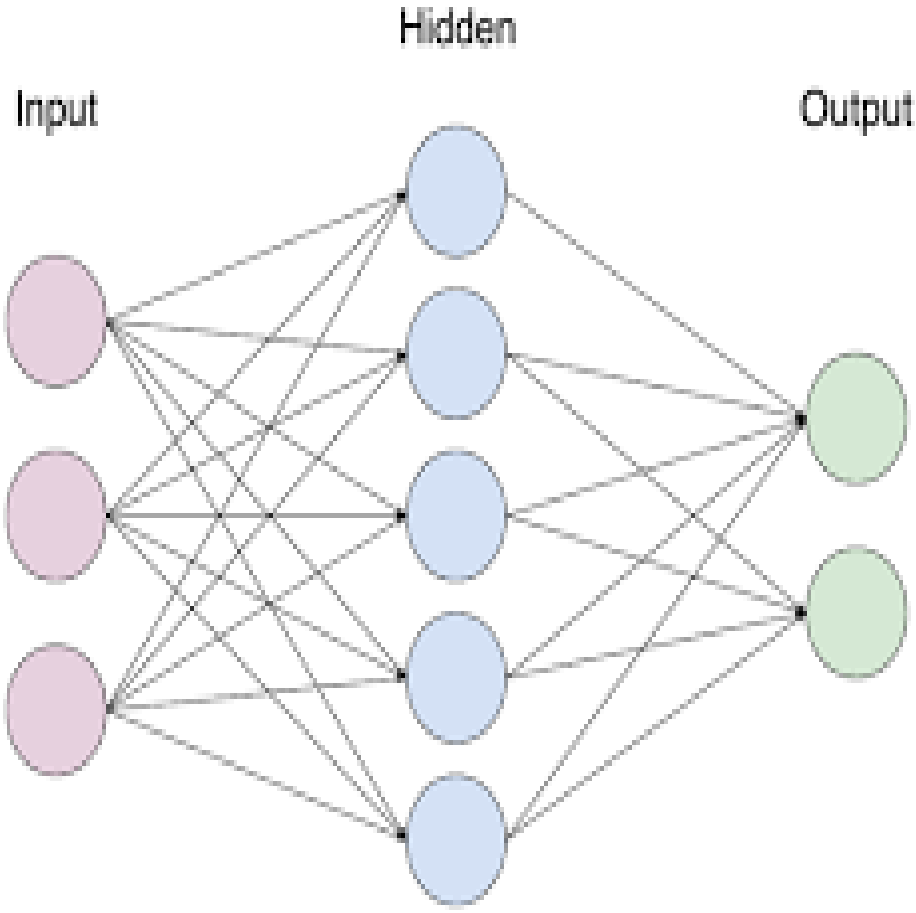


Figure 1.3 Neural Network

It is inspired by biological neural networks in the brain and works in similar way. It is like having multiple signal to a node and it has a decision making capacity and the outputs can be one or multiple again and this output is based on the previous logic and weight that is assigned. We can always assigned weight to the nodes and at the node we can perform some kind of calculation and the output of this would be a calculated weighted value of the set of inputs. So in this way using calculations we can expect it to do a logic for pattern recognition.

Neural network work in three main paradigms :-

1. Supervised learning :- It is where we know the cost function , input data and all the data i.e. given ,so we can calculate the cost function based on input data i.e. given. Cost function is basically the difference between the input and output data. It is basically a feedback network where we have the output, then you are taking part of output and feeding it the input and thus we calculate the cost function.

2. Un-supervised learning :- Here the cost function may or may not be the function of the input data.

3. Reinforcement learning :- How your input interacts with environment is known but you actually don't know what your input is. Only thing we know is how it behaves with the environment, what is the result ,and what are the effects of it. So, in this learning you can estimate a cost function but you might not be able to calculate cost function.

1.3.1 Basic Neural Network

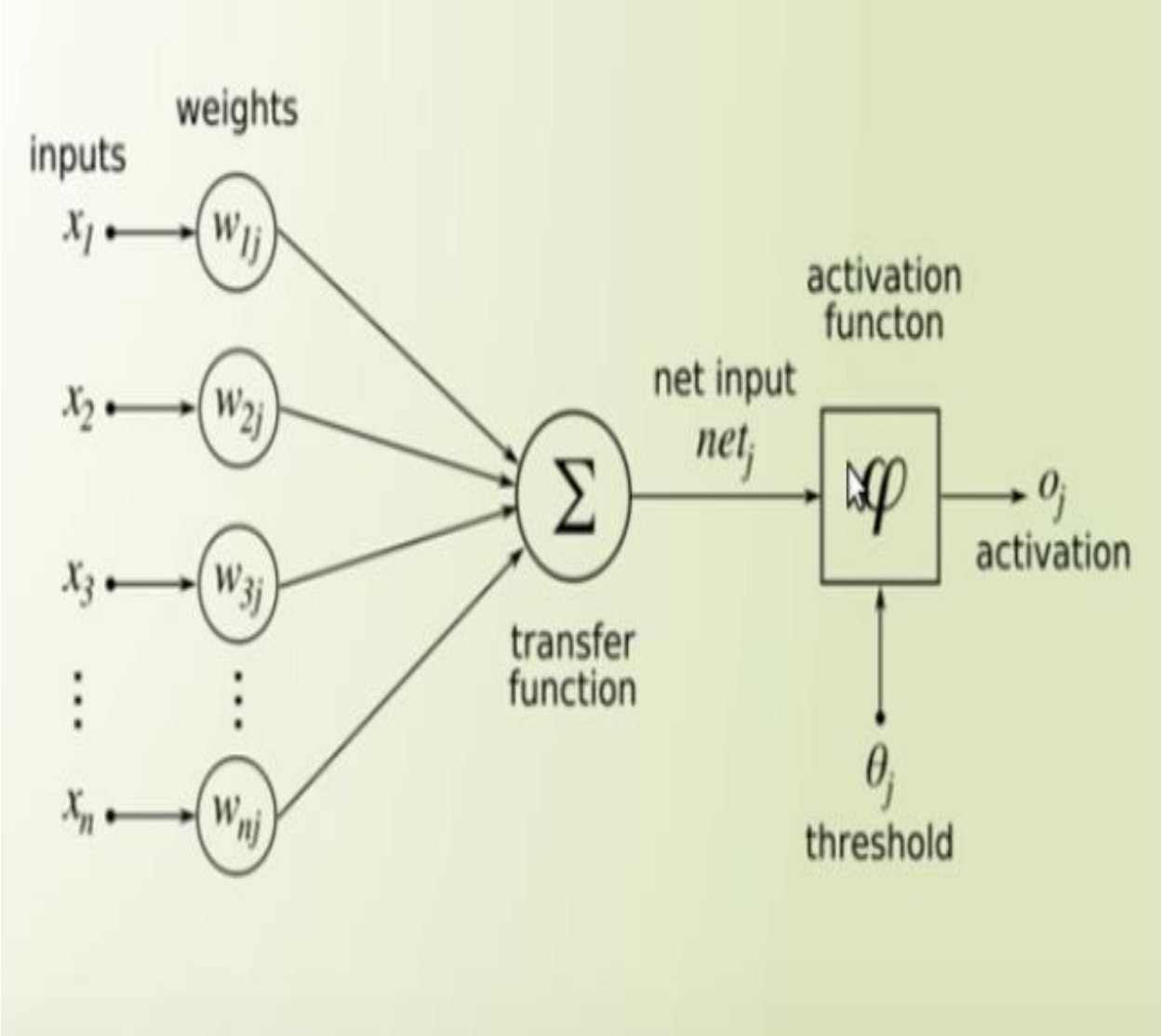


Figure 1.3.1 Basic Neural Network

This is the basic example of neural Network. We have x1,x2,x3, ----------,xn inputs and we have weights assigned for each input W1j,W2j,----------,Wnj ,we have the transfer function here that combines all its inputs and gives the output for this set of inputs. Transfer function is transferring or coverting the input function to output function. Activation function defines the output of a node.

1.3.2 Neural Network as Classifier Systems

Artificial neural networks are very effective classifiers. For example, we can train an ANN to classify cats versus dogs. In this example, cats and dogs are our 2 classes. It would be a binary classification problem.

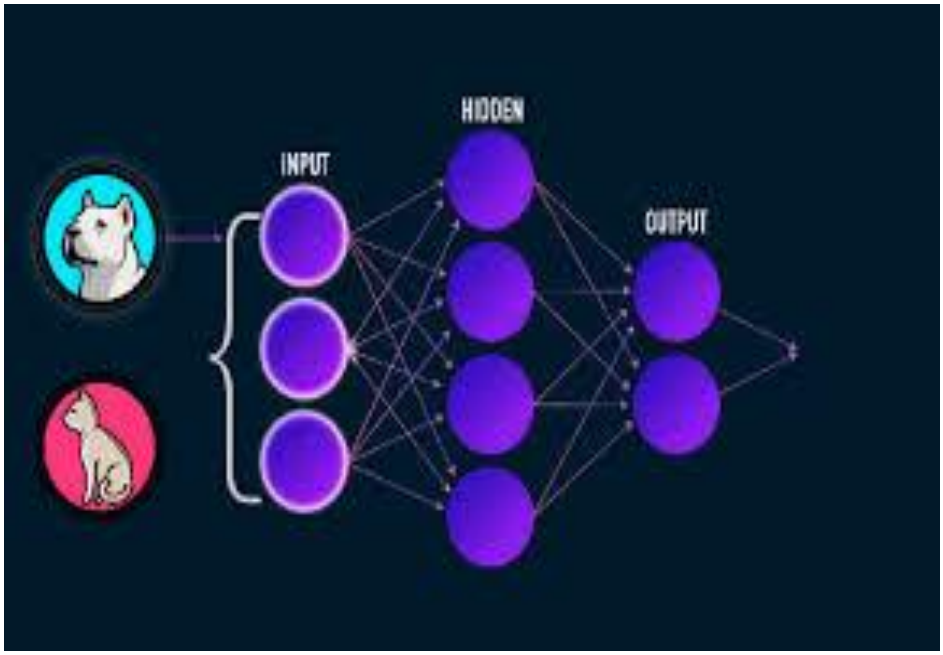


Figure 1.3.2 Neural network as classifier system

1.3.3 Why use Neural Network?

Neural network with its ability can be used to derive or determine the very meaning of complicated data that can be use to detect trend and also can be used to extract pattern that are too difficult for human to be notice or some other computer techniques. A trained neural network can be thought of as an genius or as an expert in the category of information it has been given to analyze. This trained neural network can be used to provide projections given new situations of interest and answer what if questions. Other Advantages Include:

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

2. Real Time Operation :- ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which is of this capability.

3. Self-Organization :- An ANN can create its own representation or organization of the information which receives during the learning time.

4. Fault Tolerance :- Partial destruction of network can leads to corresponding degradation of the performance. However, here some network capabilities can be retained even during major network damages.

1.3.4 Applications of Neural Network

Neural networks is where you have a set of samples of a particular alphabet ‘a’ and we suppose to take at least 5-10 different samples of ‘a’ i.e. 10 different handwritings of ‘a’. So we take 10 different samples and we perform neural network , and we write a neural network algorithm and we run the neural network algorithm on this 10 images and then when we have a 11^{th} image or something we know whether it is an ‘a’ another alphabet.

1.3.5 Two phase character recognition

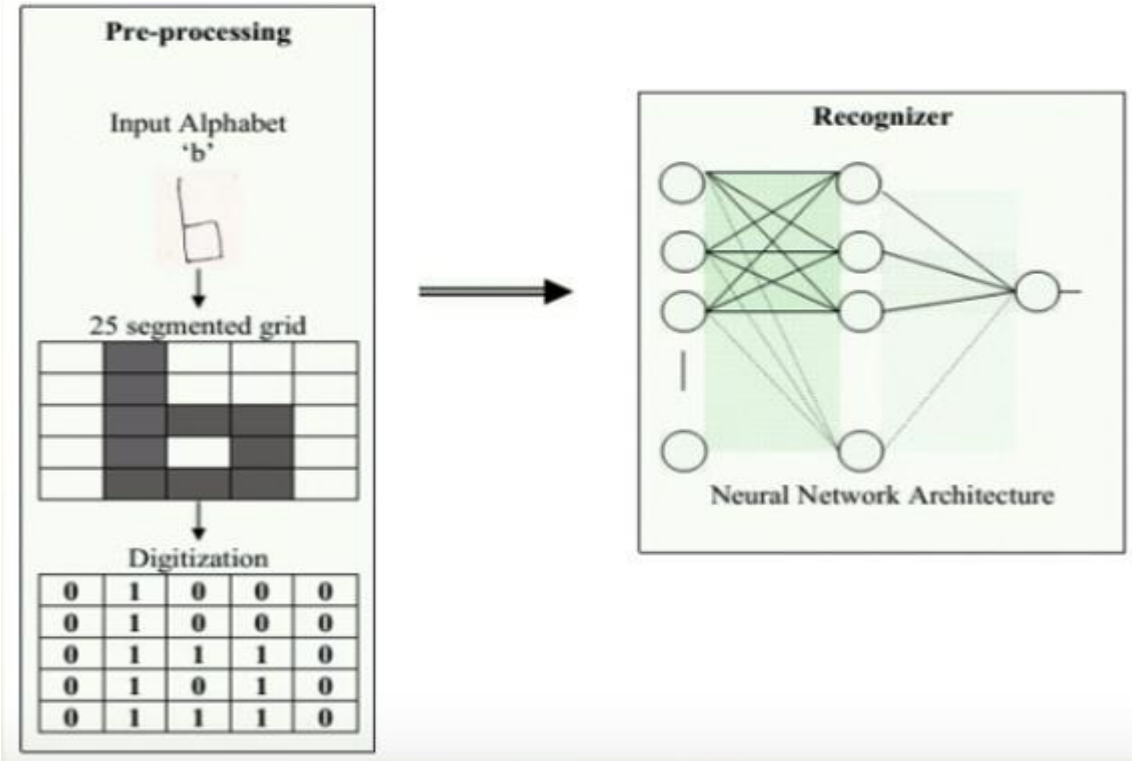


Figure 1.3.5 Two phase character recognition

These are the two phases of character recognition . The preprocessing technique as I said and you are separating the alphabet and you are doing preprocessing technique on it. This is one of the ways in which your image will be converted in a typical scenario in a MATLAB or opens a way whichever the software you are using. So this is represented in form of grids, and this is in digitized form. As shown in the above figure the grids which are filled with the alphabet are written as 1 in digitized form and the rest are written as 0. This is in fact fed to the neural network where all the processing is done. Neural network architecture has basically three layers - Input, Hidden and Output layer.

Hidden Layer could be any number of hidden layer nodes the more the number of hidden layer nodes and hidden layer, the more the complex the neural network is, the more computational requirement is necessary for this system that you are using. It all depends on how we go about

balancing, how many nodes you need for the hidden layers, how many hidden layer you want and how the network has to be.

1.3.6 Multi-layered perceptron (MLP) Classifier

The next technique we use here for character recognition is MLP classifier. This is a kind of classifier that can be used alternative to a neural network. Neural network could get computationally intensive based on kind of calculation and the complexity of the problem have to increase the number of hidden layers just to achieve higher accuracy. So in order to do that we go on increasing the hidden layers or the number of hidden nodes .These are number of hidden nodes in figure. We might want to have another layer like this ,we might have 3-4 hidden layer based on complexity of the program .Complexity is to do an character recognition for all the English alphabets. We have 26 different alphabets and to do this 26 different alphabets, neural network has to be complex. We can have various different sizes, different styles of handwriting. "A" simply could be written in thousands of styles and to understand all these things, we are working to store and to see how the pattern is changing between the various ways. We might want to have lot of this network. So this cannot can be decide. This is the limitation of Neural Network where your network size as to increasing. You could be guaranteed of getting good results or you may not but the thing is, if you are increasing the size of network your calculations are increasing . When once the calculations are increasing, your computational capacity need to be increased or you might not be even able to run it on home PC to just test how it works. So that is where multilayered perceptron classifier comes in.

1. It is a feed forward neural network :- Feed forward in the sense you do not have anything that is coming back from the output you are not using the output to basically control the input over here which is the not the case in neural network

2. Data is propagated from input to output units in a feed forward way :- i.e there is no feed back form the output

3. Data processing may extend over multiple layers :- So there can be multiple MLP classifier that can be used but there can be only one hidden layer

4. A one hidden layer MLP can approximate any function to any desired accuracy:- This is the advantage of MLP classifier ,we can use any number of MLP classifiers we can all process it in parallel but at the same time we are restricting the hidden layer to one hidden layer which reduces the competition calculations to a great extent and we can also be sure of getting any disaccurate results.

5. Because of these advantages MLP classifiers can avoid the bulk of calculations performed in conventional neural networks . As we demonstrated before in the above point.

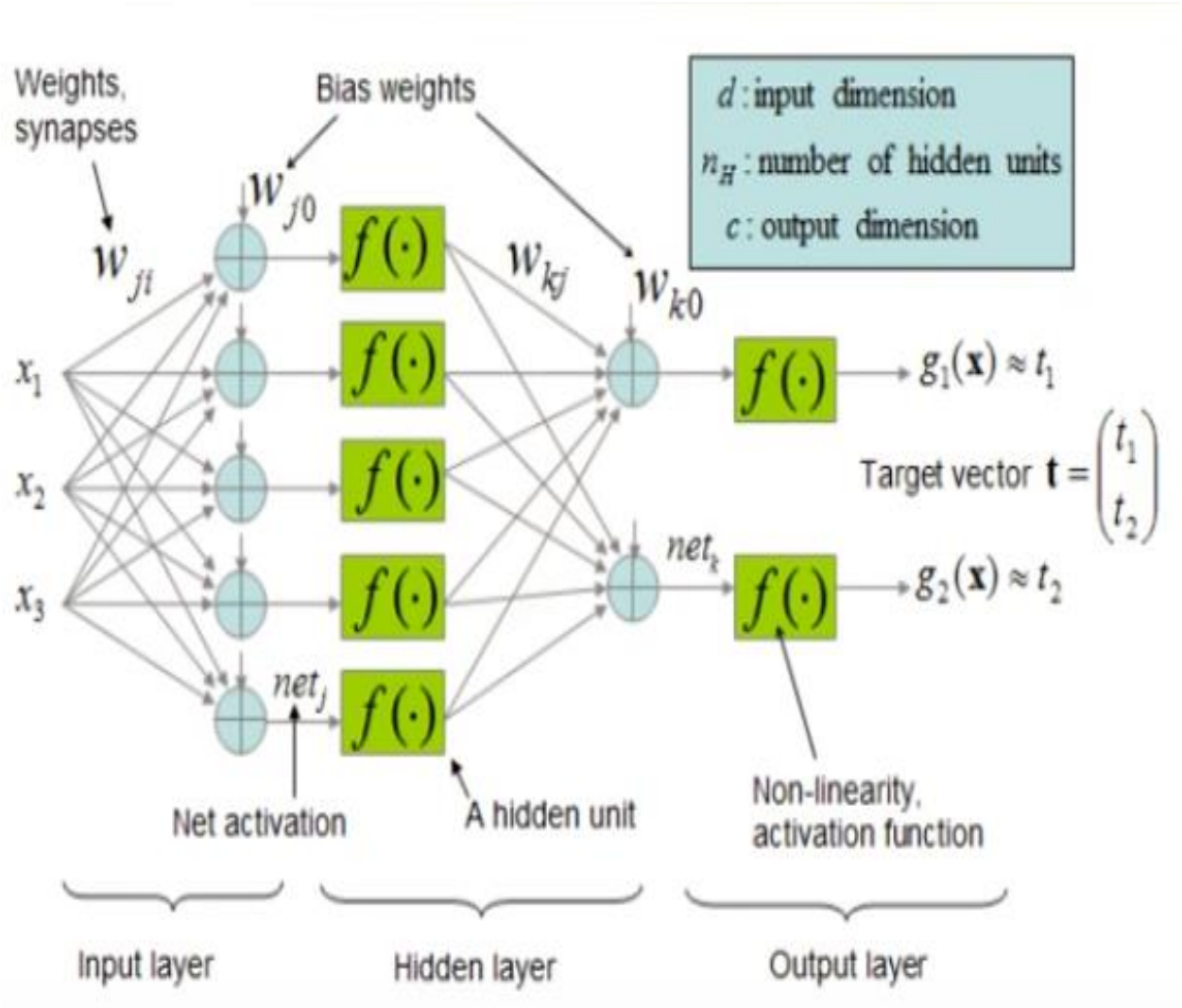


Figure 1.3.6 Multi-Layered Perceptron Classifier

So this is a typical diagram of a multilayers MLP classifier. This Wji represents weights synapses. Weight synapses refer to the strength or amplitude of connection between the nodes how strongly X1 and next node is connected. We have biased weight of each node and the function – The hidden unit . Net activation is the transfer function in the hidden unit. We have final activation function .

It is basically similar to neural network but only difference we do not have a feedback and we can have many multiple classifiers of hidden unit running parallely. So it is a feed forward network and it is much more faster because we just have one hidden layer.

1.4 Problem Statement

Currently handwriting detection is done using the Optical character recognition (OCR) approach Optical character recognition (OCR) is a technology that enables you to convert different types of documents, such as scanned paper documents, PDF files or images captured by digital camera editable and searchable data.

But Optical character recognition is not a complete success. Because Optical character recognition technology is not always perfect. In Optical character recognition hundred percent accuracy rate does not exist. The Optical character recognition engine makes use of pattern recognition and complex algorithms of mathematical nature to give a good text file output. We can optimize the settings of an Optical character recognition engine to help us but this may put us in many more problems as the solving of one problem can lead to the creation of another problem and many other such disadvantages are linked with Optical character recognition.

1.4.1 Solution

The above problem can be solved by applying New feature extraction techniques and by using artificial neural network. By applying and using above techniques we can expect hundred percent accuracy of character recognition and can sort out all kind of problems which occurs due to the Optical character recognition. We have done comparison among Artificial neural network, random forest, XGBoost to see which techniques perform better.

1.5 Objective

1. To Understand problem of handwriting detection

2. Here our main aim is to covert old literature into digitized form manually by reducing man power

3. System serve or act as guide and working in character recognization areas

4. To describe solution using Artificial neural network, Random forest, XGBoost and compare the results among them to see which algorithm performs better.

5. Applying solution to MNIST and EMNIST dataset

6. To demonstrate various techniques that are used for performing handwriting detection.

7. Demonstrate the usage of techniques used in this project.

8. Emphasize the challenges faced while performing these specific tasks.

9. Emphasize on the limitation of various techniques.

CHAPTER 2

LITERATURE SURVEY

In this chapter, we have looked into different research papers that have investigated the different studies regarding handwriting detection using Neural Network.

Gil Levi and Tal Hassner, Haider A. Alwzwazy1 [1,2] Neural networks are playing a major role in machine learning applications, since neural networks work like human brain they try to mimic the working of neurons in human brain that is why they are nowadays used to build self driven cars that can park themselves without any driver. Google deep minds are also working in this field to develop board playing games.

Dr. Kusum Gupta [3] In machine learning applications, we need different features for the prediction. So the feature recognition helps us in extracting features and their parameters. If we have more features then it will sometimes lead to over fitting. When there is large difference between train accuracy and test accuracy, then model will over fit that is why feature specification is very important. We have to select those feature only which will help us in better prediction in the application.

Faisal Tehseen Shah, Kamran Yousaf [4] Before 1980's and 1990's the data sets are very less and the neural networks are fringe but in 21st century the data sets are present in large amount that is why neural networks are making a big come back and neural networks helps us to make large number of machine learning applications. Later in 1980's developing of speech recognition was a very difficult task but now this is not a very difficult task because of the increase in the data set.

Sherif Abdel Azeem, Maha El Meseery, HanyAhmed [5] Neural networks can be used as powerful tool for signal and image processing. In this research paper the online Arabic handwritten digits recognition is done. The neural networks have lot more potential to do things by increasing the number of hidden layer in the network, all the processing is done by the hidden layer nodes.

Yusuf Perwez, Ashish Chaturvedi [6] To recognize the alphabets we can represent the English alphabets in the binary format in which the input is given to the simple feature extraction system and the output is fed the neural network that we have created.

Gregory Cohen, Saeed Afshar, Jonathan Tapson [7] In this research paper MNIST is introduced which is a dataset which is extended version of the NIST. This dataset contains about 60k images and the images are only the black and white images and because of its large data set this dataset help us to train our algorithm more accurately .MNIST also solves the problem of data set for the neural networks.

Diederik Kingma, Jimmy Ba [8] Adam Optimizer is a variant of stochastic gradient descent algorithm. Adam is an abbreviation for adaptive moment estimation. It is the current best choice among gradient based convex optimization algorithm. One advantage is that hyperparameters require less tuning. This method is very efficient, requires very little memory requirements and is best for problems which are having large data or parameters.

C. Zhang & P. C. Woodland [9] The creation of the hidden layers is always an issue and the function that can be used in the hidden layer is also an issue , the most common function are the sigmoid function and ReLU function. In this paper we have studied about the various parameters of the ReLu function that has also helped in our project.

X. Glorot and Y. Bengio [10] In this research paper we have studied that there are some problems in finding the local minima of the algorithm and there can be problem of over fitting and under fitting, the training of deep neural networks is also a major issue for this we have to do random initialization.

G. E. Hinton and R. Salakhutdinov [11] In this research paper we have studied that sometimes the dimensionality of the features may be a major issue we have to keep the features as small as possible for this there are many dimensionality reduction techniques to do a better prediction.

D.R. Hush ,B.G. Horne [12] In this research paper we have studied that the supervised learning can be of two types static and dynamic this paper tells us that the dynamic supervised learning algorithms have memory while the other one that is static supervised learning algorithm have no memory.

Y. Le Cun, B. Boser [13] In this paper an application for the handwriting using backpropgation is given and this method is shown to have 1% error and 9% reject rate. We have to perform back propagation for output layer to the input layer.

B. El Kessab1 , C. Daoui [14] In this research paper the hand writing detection is shown through the optical character reorganization (OCR) and the success rate through this will be 80%.

Y. LeCun, L. Bottou {15] In this research paper it is shown that the number of layers are varied in the neural network and the gradient descent back propagation algorithm is applied for the application.

ErnstKussul, TatianaBaidyk [16] In this research paper it is shown that the how we can improve the handwriting algorithm using the MNIST database. In the best cases the error rates can be 0.7%,0.42% and 0.63%.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 MNIST dataset

1. It has training set of 60,000 images

2. It has test set of 10,000 images

3. It has black and white images

4. It has 28x28 pixel bounding box

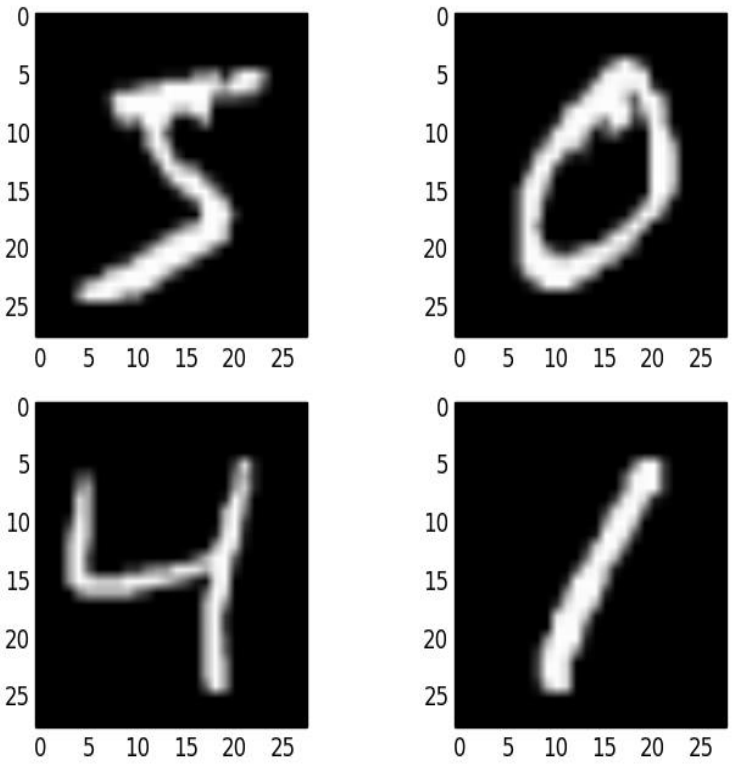


Figure 3.1 MNIST dataset image

3.2 EMNIST dataset

1. It has training set of 1,24,800 images

2. It has test set of 20,800 images

3. It has black and white images

4. It has 28x28 pixel bounding box

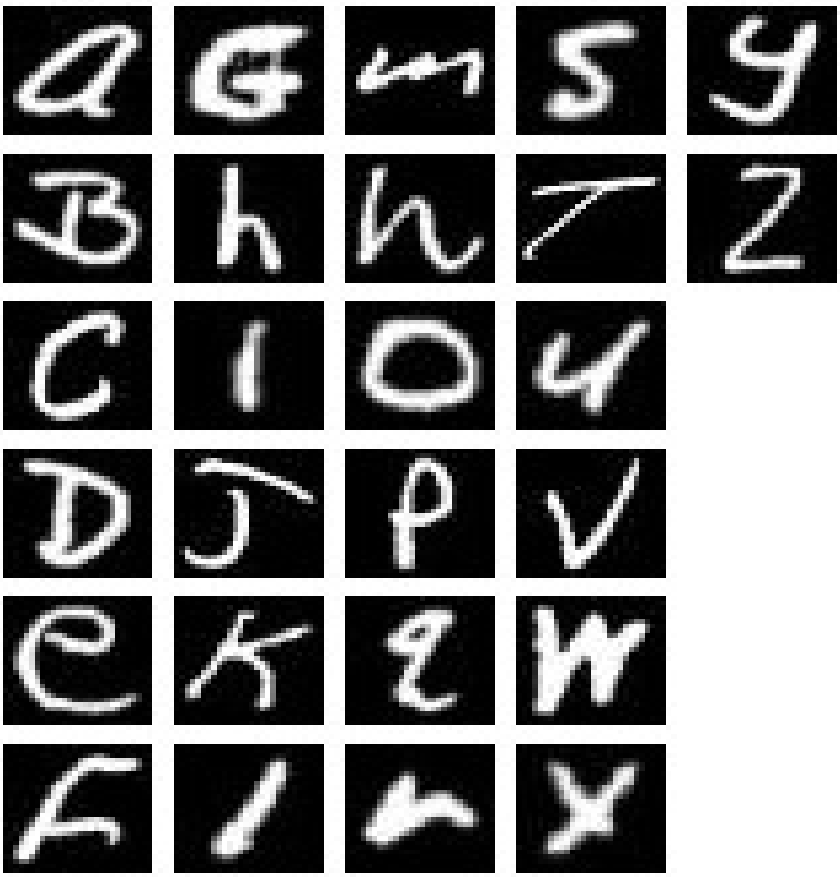
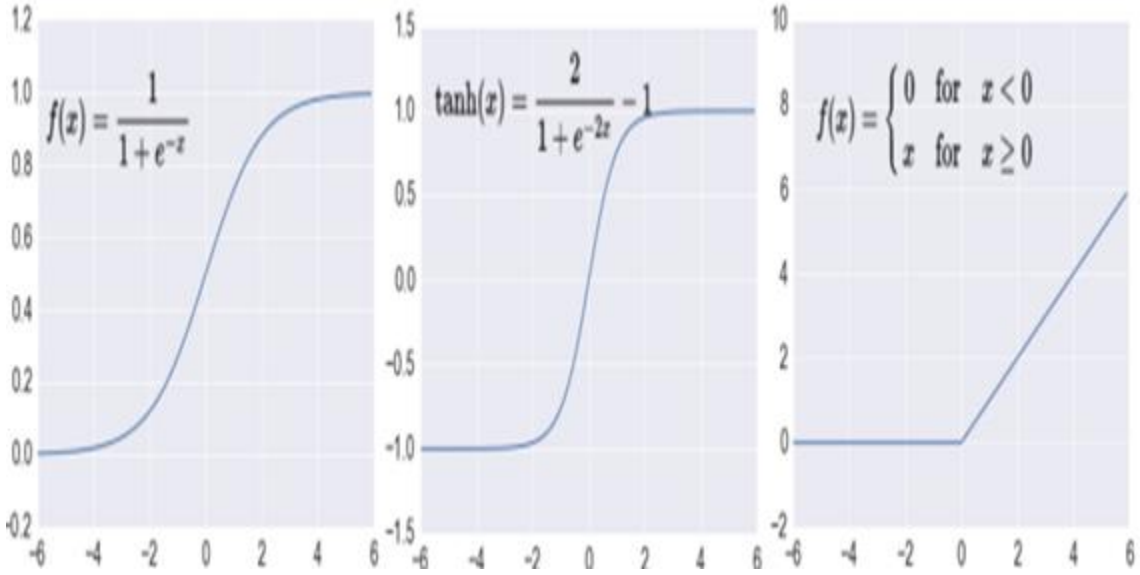


Figure 3.2 EMNIST dataset image

3.3 Activation function

1. They introduce non-linear properties to our Network.

2. They convert a input signal of a node in a NN to an output signal.



(a) (b) (c)

Figure 3.3 Activation function (a) Sigmoid (b) TanH (c) ReLU

3.4 Design of system for MNIST dataset

Data set is divided into two parts-training set and test set. Training set has 60000 images of 28*28 grid size and test set has 10000 images of same size.

First 25 images from training set are :-

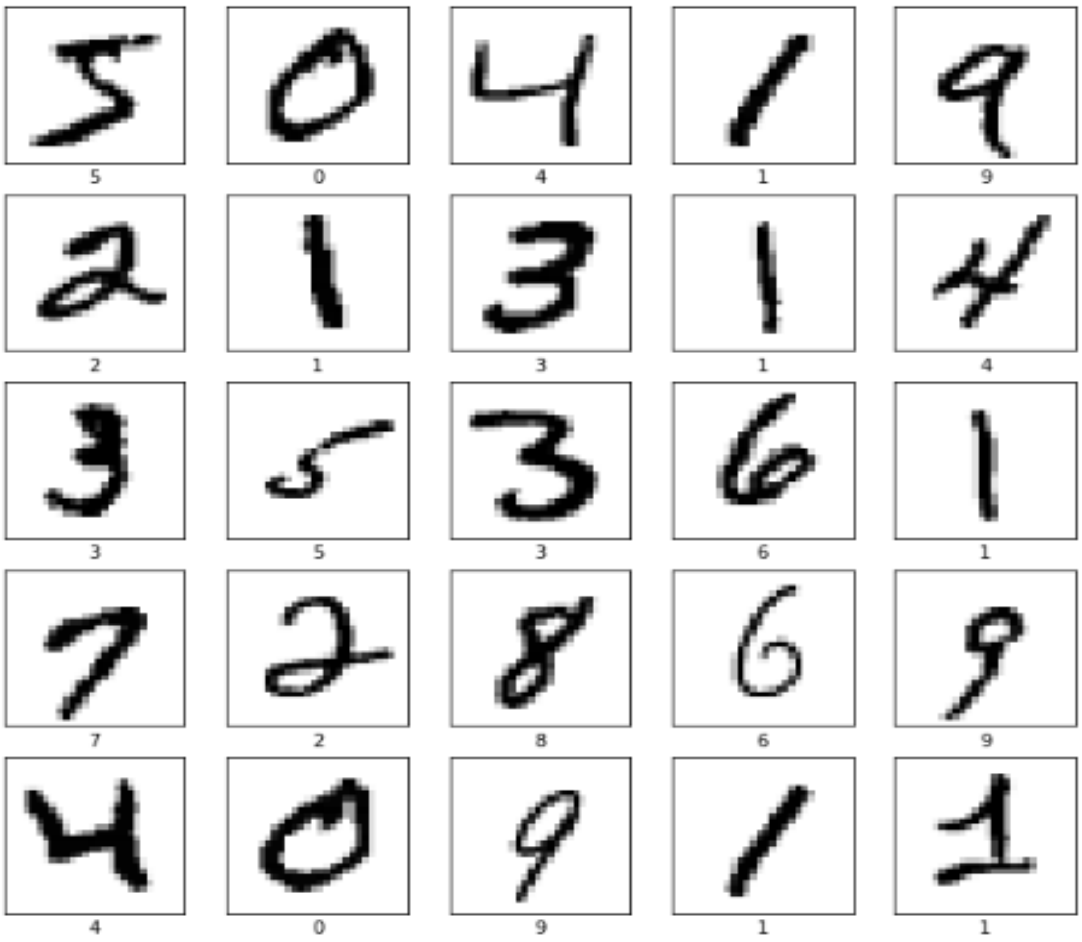


Figure 3.4 First 25 images of training set for MNIST dataset

3.4.1 Artificial Neural Network

Two layer neural network

1. Input layer :- It has input shape 28*28.

2. Hidden layer :- Only one hidden layer is used with 128 nodes and relu activation function is used.

3. Output layer :- It has 10 nodes. Each node represents a class. The 10 classes are digits (0-9).

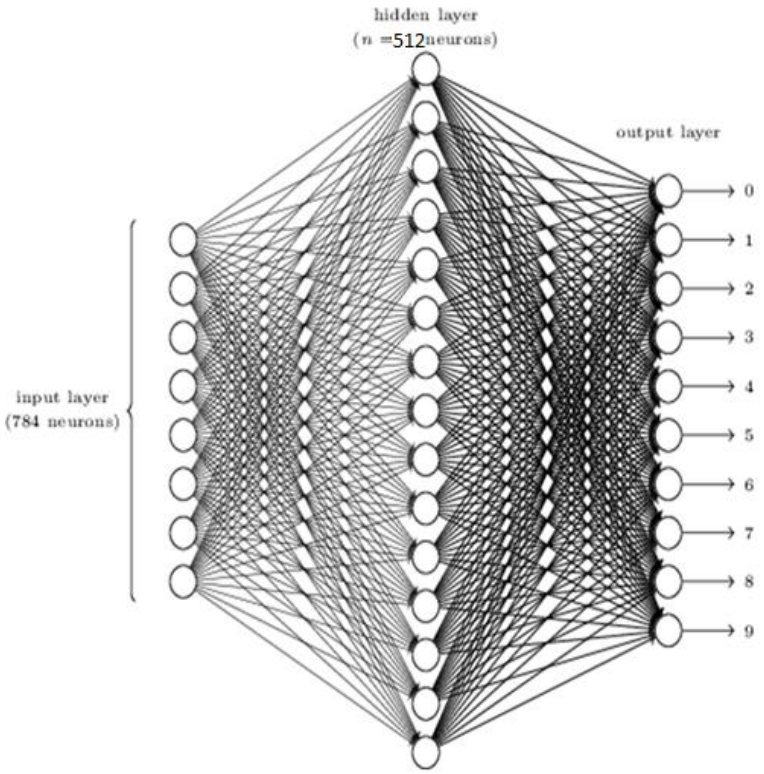


Figure 3.4.1 Two layer neural network

Optimizer

Adam optimizer is used. Adam Optimizer is a variant of stochastic gradient descent algorithm. Adam is an abbreviation for adaptive moment estimation. It is the current best choice among gradient based convex optimization algorithm. One advantage is that hyperparameters require less tuning.

Loss function

Logarithmic loss function (sparse_categorical_crossentropy) is used. The goal of each training step is to minimize this loss function. This loss function is used when the classes we are predicting are mutually exclusive.

Metric of evaluation

Accuracy metrics or classification accuracy is used for evaluating performance of our model.

3.4.2 Random Forest

Before implementing random forest, reshape function is used. Reshape function gives new shape to an array without changing its data. In our project, reshape function is used to convert three dimensional array to two dimensinal array. In our project, 6 hyperparameters are used in random forest.

1. n_estimators :- n_estimators defines the no. of trees in the forest. The default value of n_estimators is 100.

2. max_depth :- The max_depth is the maximum depth of tree. The default value of max_depth is none.

3. max_features :- The number of features to consider when looking for the best split:

a) If int, then consider max_features features at each split.

b) If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.

c) If "auto", then max_features=sqrt(n_features).

d) If "sqrt", then max_features=sqrt(n_features) (same as "auto").

e) If "log2", then max_features=log2(n_features).

f) If None, then max_features=n_features.

4. max_leaf_nodes :- Grow trees with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes. The default value of max_leaf_nodes is none.

5. n_jobs :- The number of jobs to run in parallel for both fit and predict. None means program is using only 1 job and -1 using all processors. The default value of n_jobs is none.

6. random_state :- random_state is the seed required by the random no. generator. It is used to make the results of the experiment reproducable.

3.4.3 XGBoost

Before implementing XGBoost, reshape function is used. Reshape function gives new shape to an array without changing its data. In our project, reshape function is used to convert three dimensional array to two dimensional array. In our project, 7 hyperparameters are used in XGBoost.

1. n_estimators :- n_estimators defines the no. of trees in the forest. The default value of n_estimators is 100.

2. max_depth :- The max_depth is the maximum depth of tree. Increasing this value will make the model more complex more likely to overfit. The default value of max_depth is 3.Range of max_depth is 0 to infinity.

3. subsample :- Subsample ratio of the training instances. Setting the value of subsample to 0.5 means that XGBoost randomly sample half of the training data prior to growing trees and this will prevent overfitting. Subsampling will occur once in every boosting iteration. The default value of subsample is 1.Range of subsample is (0,1].

4. col_sample_bytree :- It is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed. This is used for subsampling of columns. The default value of col_sample_bytree is 1.Range of col_sample_bytree is (0,1].

5. learning_rate :- It is step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly gets the weights of new features and learning_rate shrinks the feature weights to make the boosting process more conservative. The default value of col_sample_bytree is 0.1.

6. random_state :- random_state is the seed required by the random no. generator. It is used to make the results of the experiment reproducable.

7. n_jobs :- The number of jobs to run in parallel for both fit and predict. None means program is using only 1 job and -1 using all processors. The default value of n_jobs is 1.

3.5 Design of system for EMNIST dataset

Data set is divided into two parts-training set and test set. Training set has 124800 images of 28*28 grid size and test set has 20800 images of same size.

First 25 images from training set are :-

Figure 3.5 First 25 images of training set for EMNIST dataset

3.5.1 Artificial Neural Network

Three layer neural network

1. Input layer :- It has input shape 28*28.

2. Hidden layer :- Two hidden layer is used with 256 nodes and relu activation function is used.

3. Output layer :- It has 26 nodes. Each node represents a class. The 10 classes are alphabets (A-Z).

Optimizer

Adam optimizer is used. Adam Optimizer is a variant of stochastic gradient descent algorithm. Adam is an abbreviation for adaptive moment estimation. It is the current best choice among gradient based convex optimization algorithm. One advantage is that hyperparameters require less tuning.

Loss function

Logarithmic loss function (sparse_categorical_crossentropy) is used. The goal of each training step is to minimize this loss function. This loss function is used when the classes we are predicting are mutually exclusive.

Metric of evaluation

Accuracy metrics or classification accuracy is used for evaluating performance of our model.

3.5.2 Random forest

System design for random forest is same as described for MNIST.

3.5.3 XGBoost

System Design for XGBoost is same as described for MNIST.

CHAPTER 4

PERFORMANCE ANALYSIS

4.1 Hyperparameter tuning of Neural Network for MNIST dataset

Hyperparameter tuning for MNIST dataset					
Expt no.	hidden layer nodes	Train accuracy	Train loss	Test accuracy	Test Loss
1	16	0.9523	0.1661	0.9452	0.1832
2	32	0.973	0.0936	0.9648	0.1193
3	64	0.9851	0.0526	0.9738	0.0875
4	128	0.9907	0.0332	0.9768	0.0747
5	256	0.9956	0.0174	0.9798	0.0691
6	512	0.9975	0.0101	0.9815	0.0648
7	1024	0.9966	0.0109	0.9809	0.0688

Table 4.1 Hyperparameter tuning of Neural Network for MNIST dataset

4.2 Hyperparameter tuning of Neural Network for EMNIST dataset

Hyperparameter tuning for EMNIST dataset						
Expt no.	hidden layer nodes1	hidden layer nodes 2	Train accuracy	Train loss	Test accuracy	Test Loss
1	16	16	0.7725	0.7703	0.7677	0.788
2	32	32	0.8527	0.4844	0.8426	0.5208
3	64	64	0.9022	0.3033	0.8812	0.3839
4	128	128	0.9311	0.2012	0.899	0.3306
5	256	256	0.9514	0.1296	0.9117	0.3072
6	512	512	0.9602	0.1007	0.9164	0.3109
7	1024	1024	0.9634	0.0926	0.9162	0.3315

Table 4.2 Hyperparameter tuning of Neural Network for EMNIST dataset

4.3 Hyperparameter tuning of Random Forest for MNIST dataset

Hyperparameter tuning of random forest for MNIST dataset						
Expt No.	n_estimators	max_depth	max_features	max_leaf_nodes	Train accuracy	Test accuracy
1	100	None	None	None	1	0.9676
2	200	None	None	None	1	0.9704
3	300	None	None	None	1	0.9714
4	500	None	None	None	1	0.9711
5	300	6	None	None	0.8925	0.8973
6	300	7	None	None	0.9158	0.918
7	300	8	None	None	0.9341	0.9279
8	300	9	None	None	0.9506	0.9414
9	300	10	None	None	0.9653	0.9476
10	300	12	None	None	0.987	0.9593
11	300	15	None	None	0.9973	0.9675
12	300	20	None	None	0.9993	0.9709
13	300	25	None	None	0.9998	0.9705
14	300	None	0.8	None	1	0.9636
15	300	20	0.8	None	0.9979	0.9635
16	300	None	None	15	0.8176	0.8254
17	300	None	None	28	0.8651	0.871
18	300	None	None	56	0.898	0.9027

Table 4.3 Hyperparameter tuning of Random Forest for MNIST dataset

4.4 Hyperparameter tuning of Random Forest for EMNIST dataset

Hyperparameter tuning of random forest for EMNIST dataset						
Expt No.	n_estimators	max_depth	max_features	max_leaf_nodes	Train accuracy	Test accuracy
1	100	None	None	None	1	0.8837
2	200	None	None	None	1	0.8876
3	300	None	None	None	1	0.8885
4	100	6	None	None	0.6476	0.6452
5	100	8	None	None	0.7503	0.7393
6	100	10	None	None	0.8336	0.8044
7	100	12	None	None	0.9078	0.8457
8	100	15	None	None	0.9698	0.8724
9	100	20	None	None	0.9885	0.884
10	100	25	None	None	0.996	0.8859
11	100	None	None	15	0.502	0.5036
12	100	None	None	28	0.573	0.5728
13	100	None	None	56	0.6427	0.6412
14	100	None	None	100	0.6977	0.6939

Table 4.4 Hyperparameter tuning of Random Forest for EMNIST dataset

4.5 Results of MNIST using Neural Network

4.5.1 Training step

60000 images are used in training step.

```
Epoch 1/10
60000/60000 [==============================] - 4s 75us/sample - loss: 0.2621 - acc: 0.9252
Epoch 2/10
60000/60000 [==============================] - 5s 80us/sample - loss: 0.1076 - acc: 0.9681
Epoch 3/10
60000/60000 [==============================] - 4s 70us/sample - loss: 0.0704 - acc: 0.9800
Epoch 4/10
60000/60000 [==============================] - 4s 71us/sample - loss: 0.0503 - acc: 0.9852
Epoch 5/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.0377 - acc: 0.9886
Epoch 6/10
60000/60000 [==============================] - 5s 80us/sample - loss: 0.0273 - acc: 0.9923
Epoch 7/10
60000/60000 [==============================] - 5s 82us/sample - loss: 0.0204 - acc: 0.9947
Epoch 8/10
60000/60000 [==============================] - 5s 80us/sample - loss: 0.0152 - acc: 0.9961
Epoch 9/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.0120 - acc: 0.9973
Epoch 10/10
60000/60000 [==============================] - 5s 84us/sample - loss: 0.0101 - acc: 0.9975
```

Figure 4.5.1 Train loss and train accuracy for MNIST dataset

In this loss is 0.0101 and accuracy is 0.9975.

4.5.2 Testing step

10000 images are used in testing step.

In this loss is 0.0648 and accuracy is 0.9815.

4.5.3 Visualizing predictions

First 25 images from test set along with predictions are shown as below :-

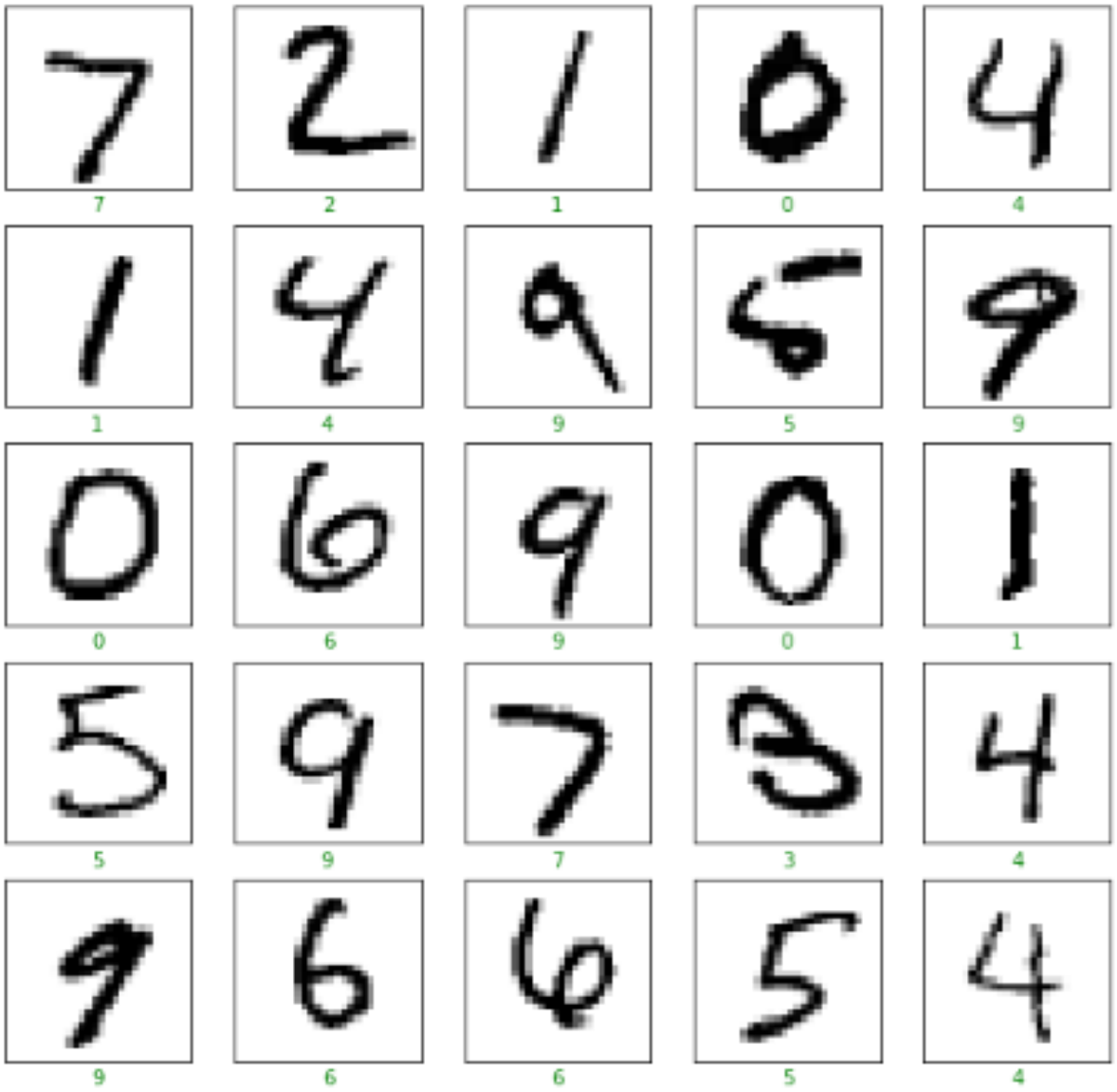


Figure 4.5.3 First 25 images of test set for Neural network

4.6 Results of MNIST using Random Forest

4.6.1 Training step

60000 images are used in training step. In this accuracy is 1.0

4.6.2 Testing Step

10000 images are used in testing step.

In this accuracy is 0.9714.

4.6.3 Visualizing Predictions

25 random images from test set along with predictions are shown as below :-

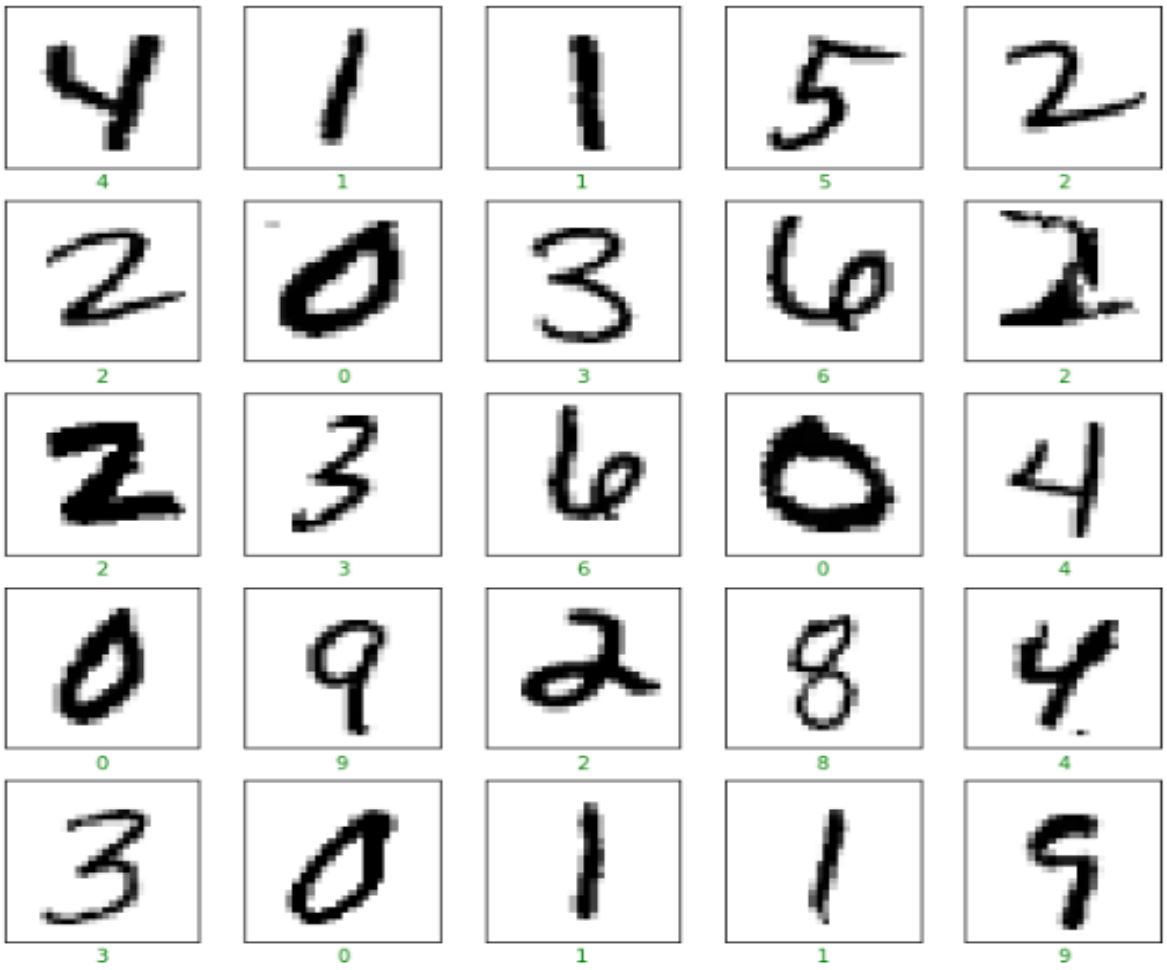


Figure 4.6.3 - 25 random images of test set for Random forest

4.7 Results of MNIST using XGBoost

4.7.1 Training step

60000 images are used in training step. In this accuracy is 0.9434.

4.7.2 Testing Step

10000 images are used in testing step.

In this accuracy is 0.9368.

4.7.3 Visualizing Predictions

25 random images from test set along with predictions are shown as below :-

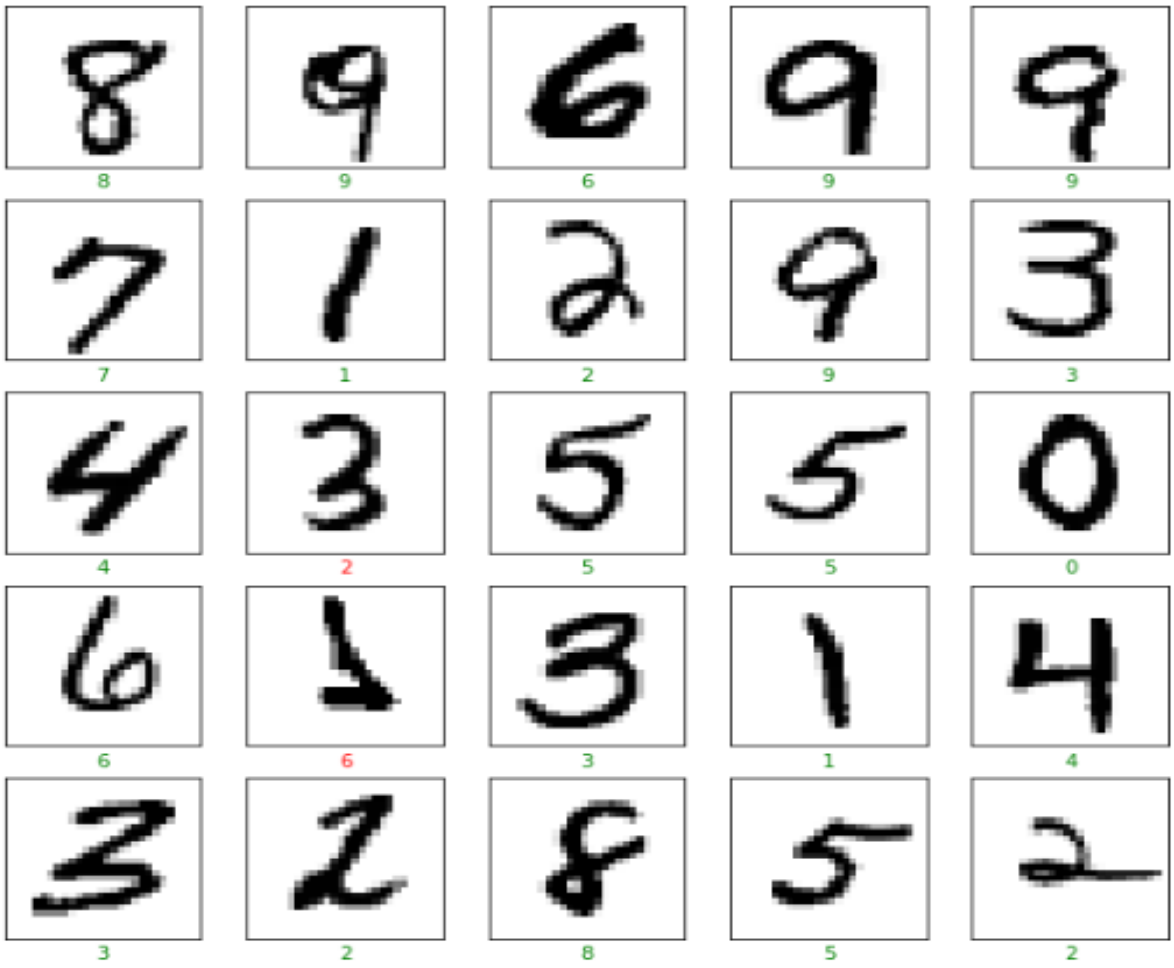


Figure 4.7.3 - 25 random images of test set for XGBoost

4.8 Results of EMNIST using Neural Network

4.8.1 Training step

124800 images are used in training step.

```
Epoch 1/10
124800/124800 [==============================] - 10s 77us/sample - loss: 0.6727 - acc: 0.7989
Epoch 2/10
124800/124800 [==============================] - 9s 73us/sample - loss: 0.3410 - acc: 0.8911
Epoch 3/10
124800/124800 [==============================] - 9s 71us/sample - loss: 0.2746 - acc: 0.9098
Epoch 4/10
124800/124800 [==============================] - 9s 72us/sample - loss: 0.2360 - acc: 0.9201
Epoch 5/10
124800/124800 [==============================] - 9s 69us/sample - loss: 0.2086 - acc: 0.9273
Epoch 6/10
124800/124800 [==============================] - 9s 70us/sample - loss: 0.1865 - acc: 0.9344
Epoch 7/10
124800/124800 [==============================] - 9s 70us/sample - loss: 0.1703 - acc: 0.9389
Epoch 8/10
124800/124800 [==============================] - 9s 70us/sample - loss: 0.1531 - acc: 0.9441
Epoch 9/10
124800/124800 [==============================] - 9s 69us/sample - loss: 0.1423 - acc: 0.9475
Epoch 10/10
124800/124800 [==============================] - 9s 70us/sample - loss: 0.1296 - acc: 0.9514
```

Figure 4.8.1 Train loss and train accuracy for EMNIST dataset

In this loss is 0.1296 and accuracy is 0.9514.

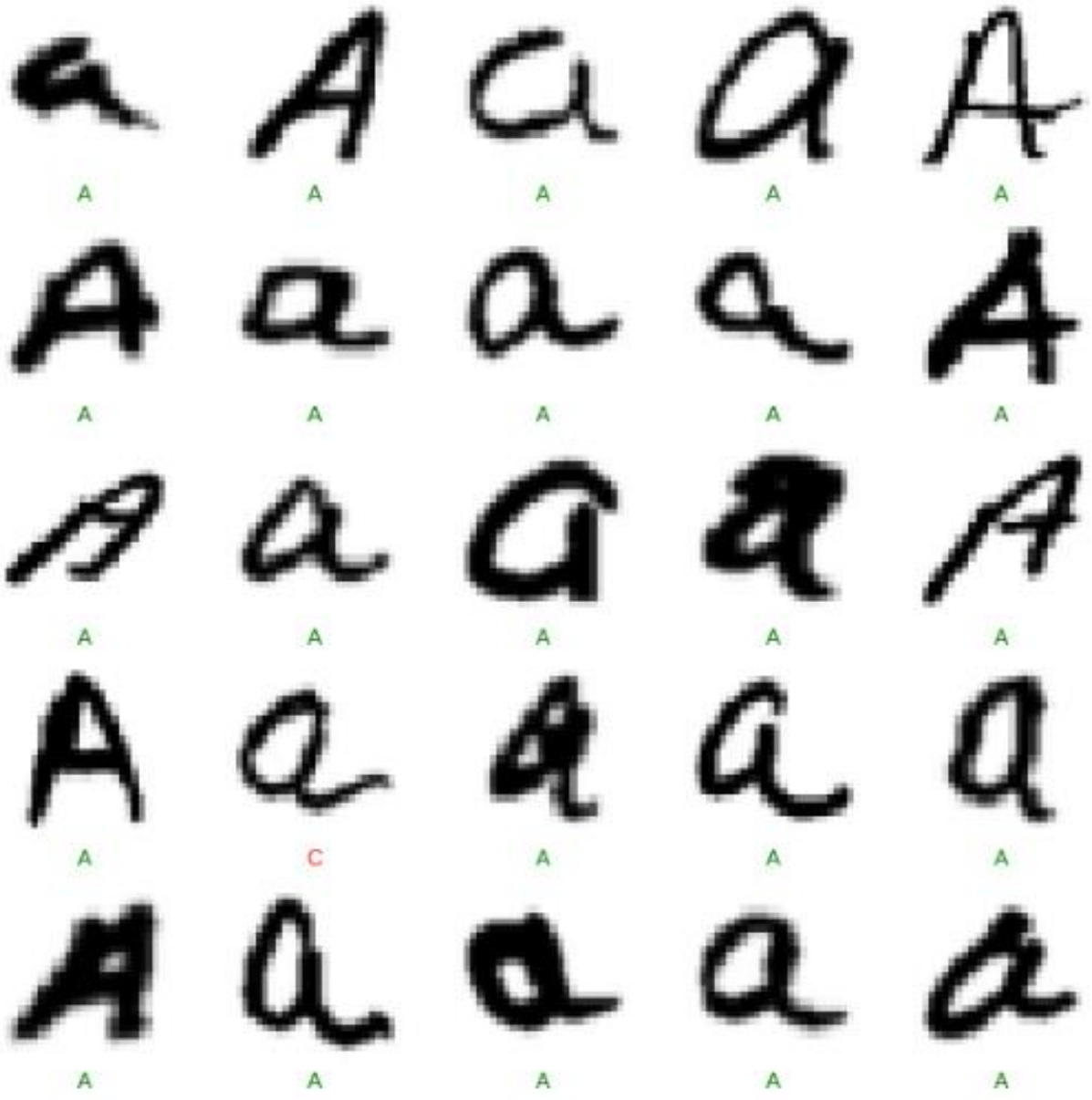
4.8.2 Testing step

20800 images are used in testing step.

In this loss is 0.3072 and accuracy is 0.9117.

4.8.3 Visualizing predictions

First 25 images from test set along with predictions are shown as below :-



(a)

25 random images from test set along with predictions are shown as below :-

(b)

Figure 4.8.3 Visualizing predictions of Neural network (a) First 25 images of test set

(b) 25 random images of test set

4.9 Results of EMNIST using Random Forest

4.9.1 Training step

124800 images are used in training step. In this accuracy is 1.0

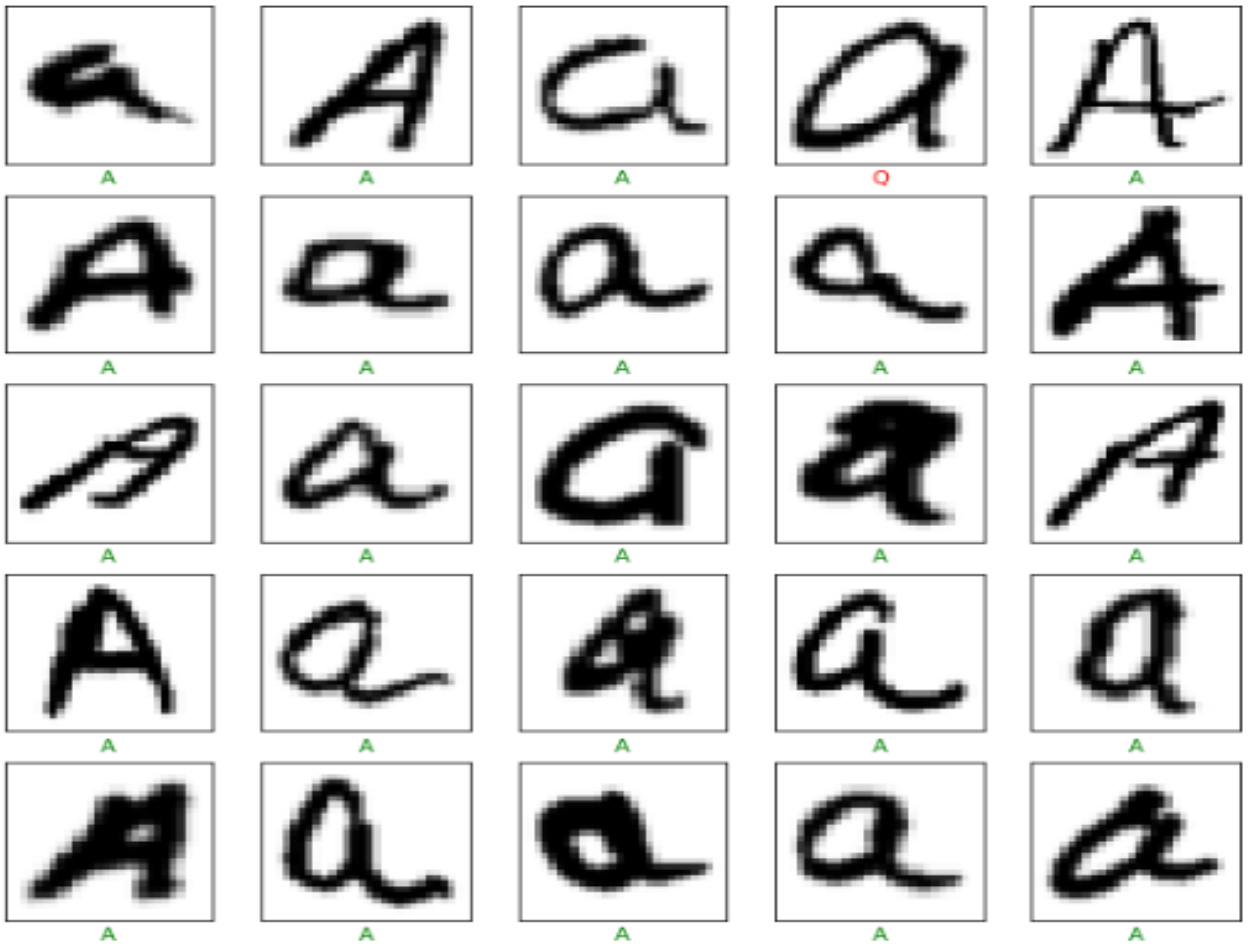
4.9.2 Testing Step

20800 images are used in testing step.

In this accuracy is 0.8885.

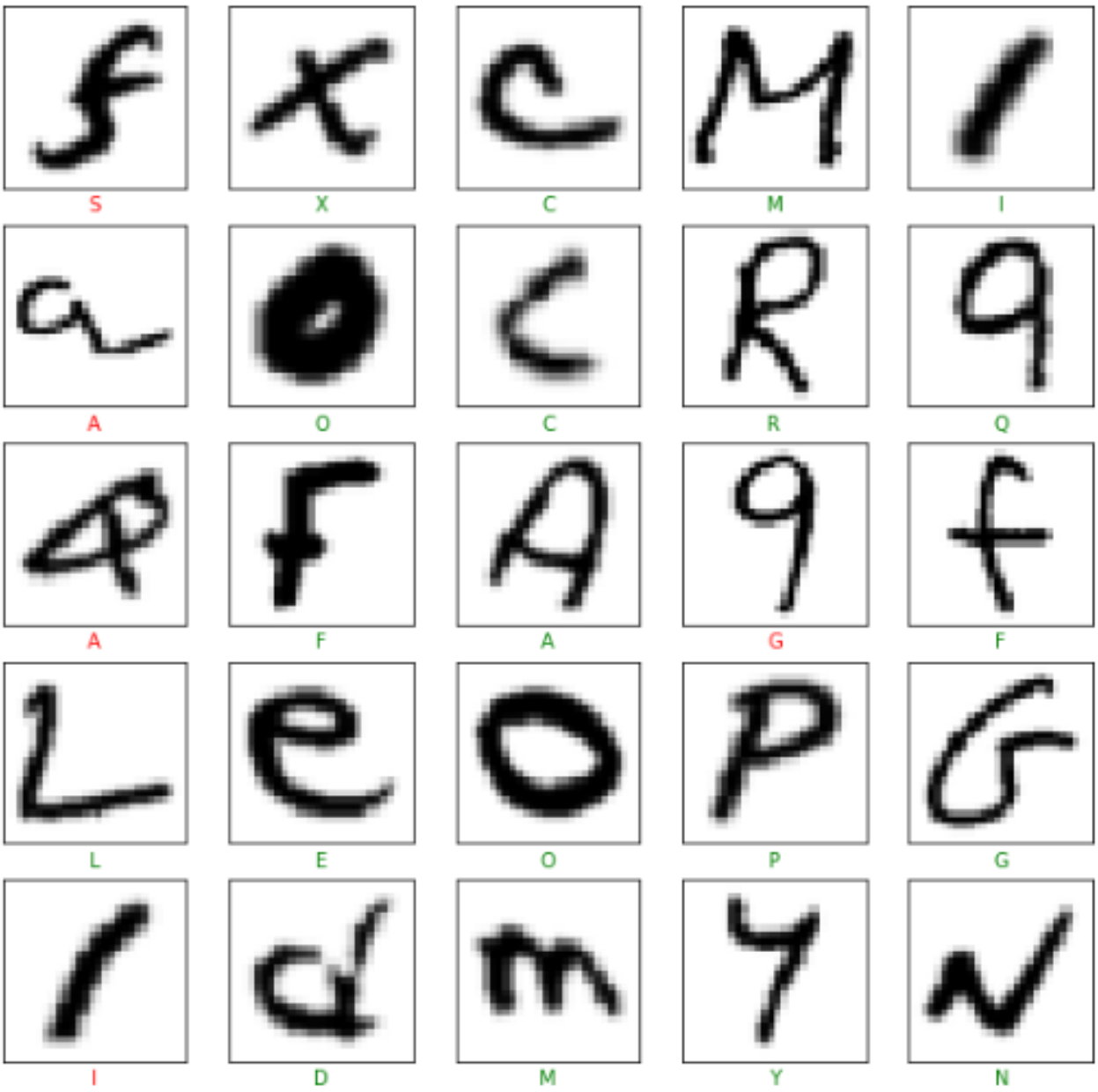
4.9.3 Visualizing Predictions

First 25 images from test set along with predictions are shown as below :-



(a)

25 random images from test set along with predictions are shown as below :-



(b)

Figure 4.9.3 Visualizing predictions of Random forest (a) First 25 images of test set

(b) 25 random images of test set

4.10 Results of EMNIST using XGBoost

4.10.1 Training step

124800 images are used in training step. In this accuracy is 0.7864

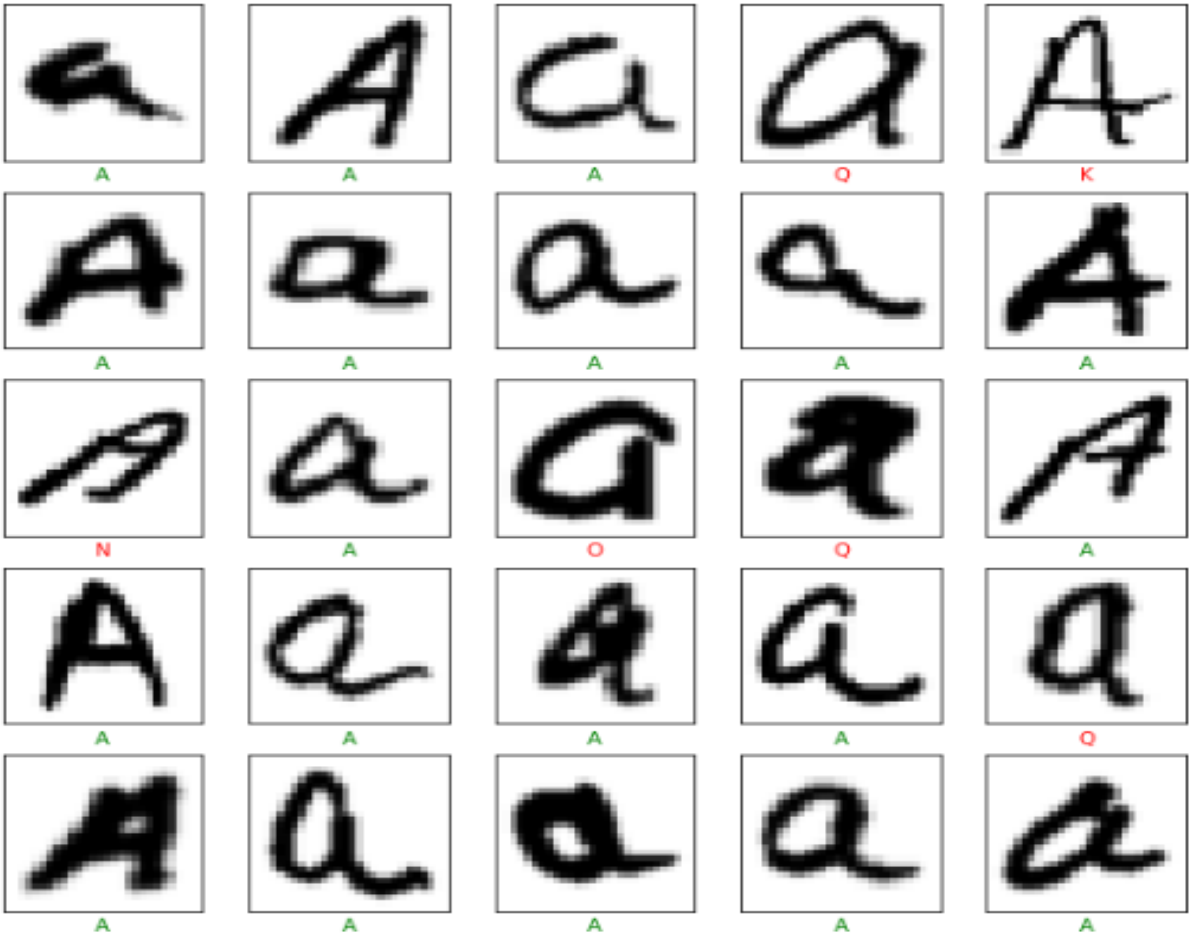
4.10.2 Testing Step

10000 images are used in testing step.

In this accuracy is 0.7682.

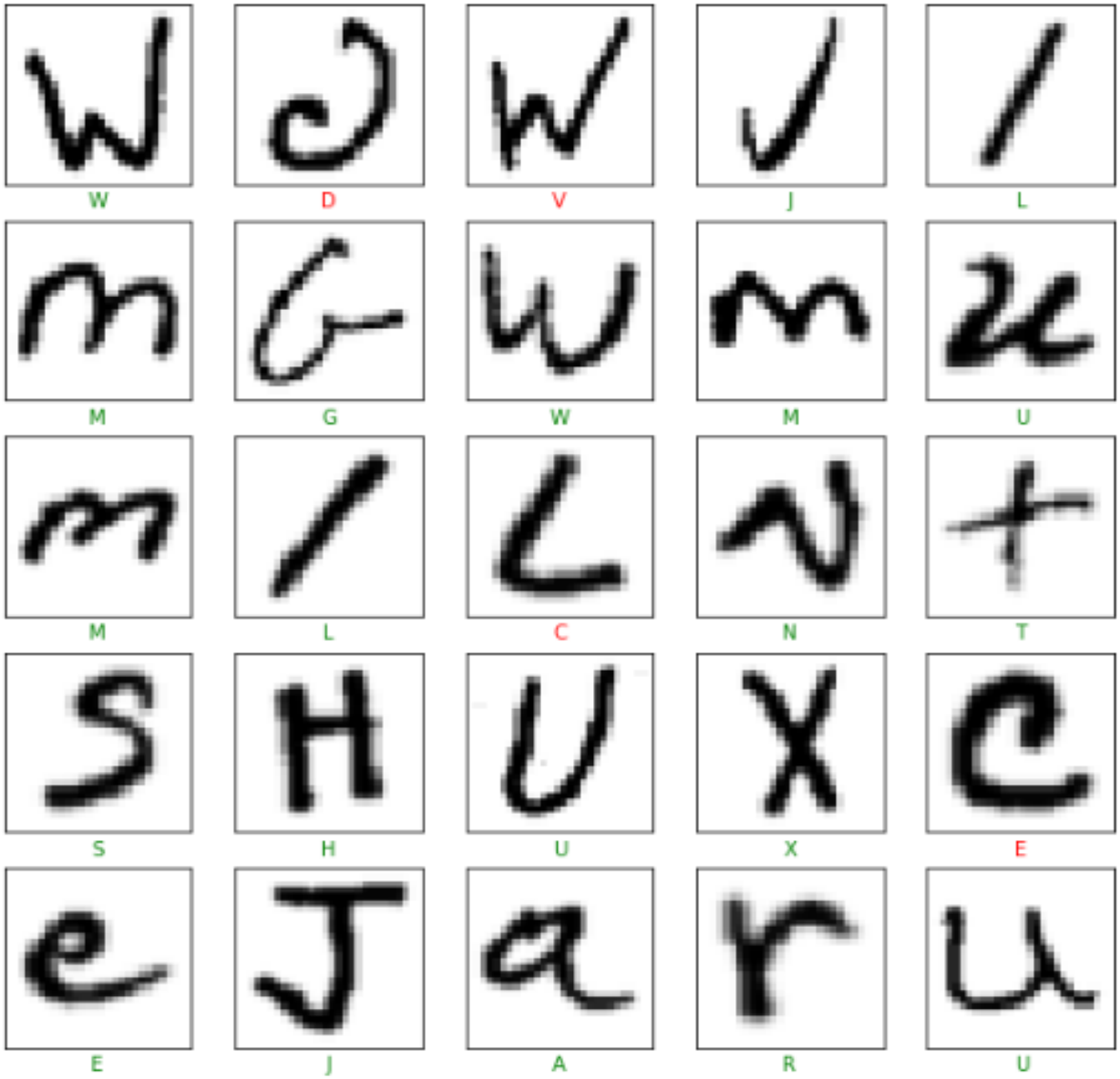
4.10.3 Visualizing Predictions

First 25 images from test set along with predictions are shown as below :-



(a)

25 random images from test set along with predictions are shown as below :-



(b)

Figure 4.10.3 Visualizing predictions of XGBoost (a) First 25 images of test set

(b) 25 random images of test set

4.11 Result Comparison

We use MNIST and EMNIST dataset in our programs to create a handwriting detection system. Using MNIST dataset, it can learn to recognize numbers from 0 to 9. Using EMNIST dataset, it can learn to recognize alphabets from A to Z. We compared 3 different models namely artificial neural network, random forest and xgboost on both MNIST and EMNIST dataset. We experimented with different hyperparameters to tune these models. The results are as follows :-

Result Comparison				
S.No	Dataset	Model	Best Hyperparameters	Test accuracy
1	MNIST	Artificial Neural Network	1 hidden layer, no. of nodes in hidden layer=512	0.9815
2	MNIST	Random Forest	n_estimators=300, max_depth=None, max_features=None, max_leaf_nodes=None	0.9714
3	MNIST	XGBoost	n_estimators=100, max_depth=3, subsampe=1, colsample_bytree=1, learning_rate=0.1	0.9368
4	EMNIST	Artificial Neural Network	2 hidden layer, no. of nodes in hidden layer=256	0.9117
5	EMNIST	Random Forest	n_estimators=300, max_depth=None, max_features=None, max_leaf_nodes=None	0.8885
6	EMNIST	XGBoost	n_estimators=100, max_depth=3, subsample=1, colsample_bytree=1, learning_rate=0.1	0.7682

Table 4.11 Result Comparison

CHAPTER 5

CONCLUSION

5.1 Conclusion

For MNIST dataset, the performance of Artificial Neural Network is 98.15% which is highest among three algorithms that we tested upon.

For EMNIST dataset, the performance of Artificial Neural Network is 91.17% which is highest among three algorithms that we tested upon.

Going further we can use Artificial Neural Network for handwriting detection on more challenging cases.

5.2 Future scope

There are several new techniques like Convolutional neural networks, Recurrent neural networks, Long-Short term Memory units etc.

We can also use convolution neural network for handwriting detection.

REFERENCES

[1] Gil Levi and Tal Hassner, "Offline handwritten digit recognition using neural network", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 2, no. 9, pp. 4373-4377, 2013.

[2] Haider A. Alwzwazy1 , Hayder M. Albehadili2 , Younes S. Alwan3 , Naz E. Islam4 ,"Handwritten Digit Recognition Using Convolutional Neural Networks", International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, no. 2, pp. 1101-1106, 2016.

[3] DR.KUSUMGUPTA2 ,"A comprehensive review on handwritten digit recognition using various neural network approaches", International journal of enhanced research in management & computer applications, vol. 5, no. 5, pp. 22-25, 2016.

[4] Faisal Tehseen Shah, Kamran Yousaf,"Handwritten Digit Recognition Using Image Processing and Neural Networks", Proceedings of the World Congress on Engineering, vol., 2007.

[5] Sherif Abdel Azeem, Maha El Meseery, HanyAhmed, ”Online Arabic Handwritten Digits Recognition “,Frontiers in Handwriting Recognition (ICFHR), 2012

[6] Yusuf Perwez, Ashish Chaturwedi, ”Neural Networks for Handwritten English Alphabet Recognition”, 2011

[7] Gregory Cohen, Saeed Afshar, Jonathan Tapson, Andre van Schaik, “EMNIST: an extension of MNIST to handwritten letters”, 2017

[8] Diederik Kingma, Jimmy Ba, “Adam: A Method for Stochastic Optimization”, 2017

[9] C. Zhang & P. C. Woodland, “Parameterised Sigmoid and ReLU Hidden Activation Functions for DNN Acoustic Modelling”, 2015

[10] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” Proc. International conference on artificial intelligence and statistics, 2010.

[11] G. E. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, Vol. 313, no. 5786, pp. 504–507, Jul 2006.

[12] D.R. Hush ; B.G. Horne, “Progress in supervised neural networks”, 1993

[13] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Handwritten Digit Recognition with a Back-Propagation Network”, 1992

[14] B. El Kessab, C. Daoui, B. Bouikhalene, M. Fakir and K. Moro, “Extraction Method of Handwritten Digit Recognition Tested on the MNIST Database”, 2013

[15] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", Intelligent Signal Processing, 2001, pp. 306-351

[16] ErnstKussul, TatianaBaidyk, “Improved method of handwritten digit recognition tested on MNIST database”, 2004