Handwritten Punjabi Character Recognition Using Convolutional Neural Networks

A Thesis

submitted in partial fulfillment of the requirements for the award of the degree of

Master Of Engineering

in

Department of Computer Science and Engineering

by

Sonia Mittal

(801532052)

Under the supervision of

Dr. Karun Verma
Assistant Professor, CSED

Dr. Ravinder Kumar
Assistant Professor, CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA - 147004

July 2017

Certificate

I hereby certify that the work, which is being presented in the thesis, entitled **Handwritten Punjabi Character Recognition Using Convolutional Neural Networks**, in partial fulfillment of the requirements for the award of the degree of **Master Of Engineering** and submitted to the institution is an authentic record of my own work carried out during the period **July 2015** to **July 2017** under the supervision of **Assistant Professor Karun Verma and Assistant Professor Ravinder Kumar**. I have also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

The matter presented in this thesis has not been submitted elsewhere for the award of any other degree or diploma from any institution.

Date:

Sonia Mittal
Candidate

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

Date:
Dr. Karun Verma
Supervisor
Assistant Professor

Date:
Dr. Ravinder Kumar
Supervisor
Assistant Professor

The M.E. Viva-Voice examination of **Sonia Mittal**, has been held on 09th August, 2017
.

Acknowledgement

First of all, I would like to express my gratitude towards **Thapar University**, for providing me a platform to do my thesis work at such an esteemed institute.

I wish to express my respect, deep sense of gratitude and idebtedness to my guide **Mr. Karun Verma** and **Mr. Ravinder Kumar**, Assistant Professor, Computer Science And Engineering Department, Thapar University, Patiala for their invaluable and enthusiastic guidance, useful suggestions, unfailing patience and sustained encouragement throughout this work.

I would like to thank **Dr. Maninder Singh**, Head of Computer Science And Engineering Department, Thapar University, Patiala for kind help, guidance, encouragement and providing the necessary facilities to carry out my research. I am indebted to the faculty members of the department for valuable suggestions, friendly support and full cooperation rendered by all of them.

Last, but not the least, I am thankful to supreme power **"The GOD"** one who has always guided me to work on the right path of the life. Without his grace, this would never come to be today's reality. With special thanks, I dedicate this thesis to GOD.

Sonia Mittal

Abstract

Today, computers have influenced the life of human beings to a great extent. To provide the communication between computers and users, natural language processing techniques have proven to be very efficient way to exchange the information with less personnel requirement. In this thesis work, natural handwriting technique is used to recognize the online handwritten Punjabi characters as natural handwritten characters are less error prone as compared to the input taken via mouse or keyboard. This thesis describes the implementation of handwritten Punjabi character recognition using deep learning technique named as Convolutional Neural Networks (CNNs). The main problem occurs in the recognition of handwritten characters is due to the occurrence of variation in the handwriting style of different users because each person has their own style of writing and also the variability in the writing style of his/her own style due to change in mood, speed of writing at different instant of time.

Punjabi script is chosen for this research work as it comes on 14th position in the spoken languages and less work is done on Punjabi script as compared to work done on other scripts such as English, Devanagari, Gujarati, Chinese. CNN is chosen for the implementation as it is proven to be very efficient technique to recognize and classify the recognized handwritten characters into their respective classes as it concentrates on the dynamic features of the input handwritten character which is obtained from the random generated character matrices.

Here, we used 5-layer CNN having stride value of one for the classification of handwritten images into one of the large number of classes (430 classes) available. Punjabi script has total of 430 classes consisting of 35 consonants, 10 vowel identifiers and their corresponding combination characters. In our dataset, each class contains 100 images thereby providing a total of 43,000 number of character images dataset. We divide our dataset in the ratio of 65:25:10, 55:35:10, 45:45:10 training:testing:validation samples data respectively. Training, testing and validation accuracy at different number of epochs (consist of forward pass and backward pass) for these different sample ratios are calculated and thus compared.

Keywords: Convolutional Neural Networks (CNNs), Rectied Linear Unit (reLu), Dropout, Backpropagation, Tensorflow

Table of Contents

List of Figures

List of Tables

Chapter 1

A Brief Introduction Of Handwritten Recognition And Convolutional Neural Networks

1.1 Introduction

Handwritten character recognition has been an active area of research from the past few years and due to its diversive applications it continues to be a challenging research topic. The main aim of this thesis is to take the handwritten Punjabi input character as an input and then segment them into their corresponding akshara level after pre-processing and then recognize and classify them into their respective classes. Variation in the writing style of different users and also in the variation of user's own writing style due to change in mood, speed of writing has made it difficult to obtain higher recognition accuracy rate results and thus made the handwritten recognition process a challenging task that made it more interesting research topic to be considered.

Handwriting character recognition generally falls into two categories: recognition of online and offline handwritten characters. In recognition of offline handwritten characters, the handwritten characters are scanned and then these scanned images are used for training, testing and validating the neural network model and used for recognition process. Here we do not have stroke information. Whereas in online handwritten recognition process, the trajectories values are recorded in parallel while the write operation is performed. Also this trajectory information is stored in sequence of strokes in a specific temporal order by the combination of which individual akshara Punjabi character is made. In online handwritten character recognition, any pen pointing device such as digital pen, stylus pen are used for writing. The sensors are used to noted down the number of strokes used to write a Punjabi character. The point when pen tip touches the screen to the point when pen point is picked up from the screen is considered as one stroke.

Many applications has been provided by handwritten character recognition such as recognizing postal codes, signatures recognition, digital libraries, invoice and receipt processing. Many researches have been done in research areas such as image processing, pattern recognition, artificial intelligence and cognitive science etc. to solve handwritten charac-

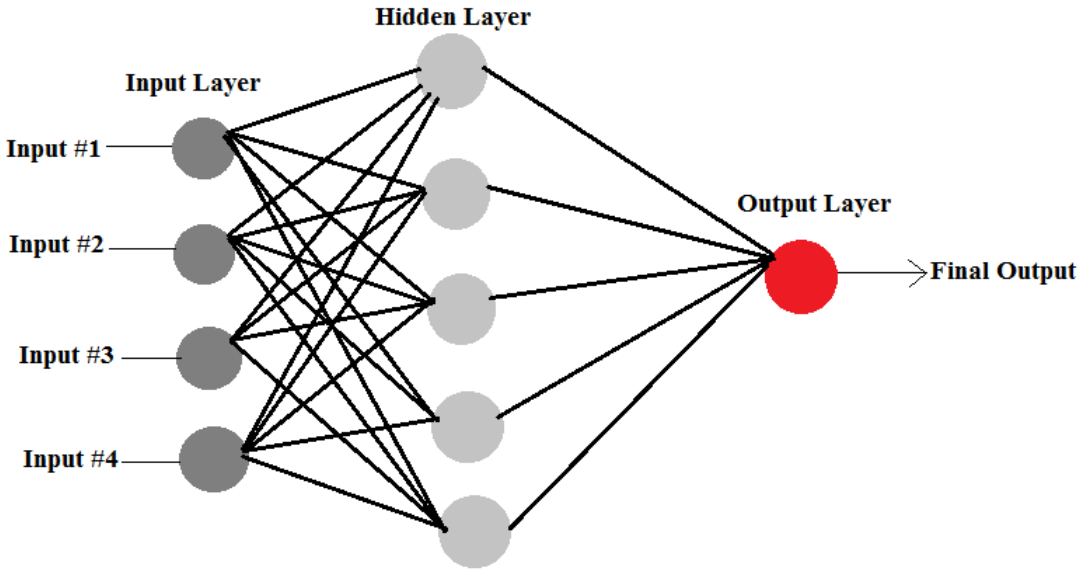


Figure 1.1: Simple neural network structure

ter recognition problem. Recognition of online handwritten Punjabi character recognition is the main objective of many research efforts in the pattern recognition field.

1.2 Why Neural Networks?

Artificial Neural Network (ANN) is proven to be a paradigm of information processing that follows the same process as followed by biological nervous system such as the human brain processes information. ANN consists of large number of interconnected processing elements called neurons that works in unison to solve complex problems. Many specific applications has been configured by the Artificial Intelligence (AI) like pattern recognition, following a learning process.

Neural networks takes different approaches to solve a problem than used by traditional computers. In order to solve a problem, traditional computers use algorithmic approach in which computer has to perform a specific set of instructions. If the specific steps that computer has to follow are not known to the computer, it cannot be able to resolve the problem which restricts the problem solving capability of conventional computers.

On the other hand, neural networks processes information in a similar way the human brain does. Neural network learn by example. Number of layers constitutes to form a neural network. Number of interconnected nodes combined to form neural network layers which contains an activation function. Various patterns are provided as an input to the

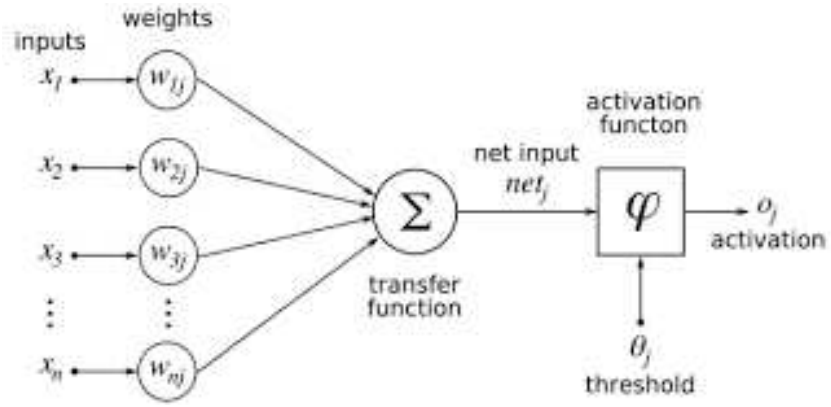


Figure 1.2: MultiLayer perceptron neural network

input layer that is further connected to one or more hidden layers which acts as the brain of neural network where the actual processing is done. The hidden layers then link to an output layer which gives us the required result.

A more sophisticated example of neuron is shown in Figure 1.1. Here the weighted inputs are used, decision of the neural network depends on the weight of the particular input. The inputs provided to the input layer are multiplied with their respective weight values which gives the required weighted input values. Summation of these weighted inputs are performed and then if computed sum is greater than pre-set threshold value, the neuron fires. If it is less than pre-set threshold value, the neuron does not fire. A simple example of how multilayer perceptron neural network works is shown in Figure 1.2 where

$$x_1, x_2, x_3, ..., x_n \tag{1.1}$$

are the inputs;

$$w_{1j}, w_{2j}, w_{3j}....., w_{nj} \tag{1.2}$$

are the respective weights.

Mathematically, the neuron fires if and only if;

$$x_1w_1 + x_2w_2 + x_3w_3 + ... + x_nw_n > T \tag{1.3}$$

where T denotes the specified pre-set threshold value.

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The multilayer perceptron has the ability to adapt to a particular situation by changing its weight and/or threshold value. Backpropagation can be used to update the values of weights and biases at the end of each epoch.

1.3 Why We Choose Convolutional Neural Network Over Simple Neural Network?

Regular Neural Networks receives an input and convert it using a cascading hidden layers to respective output. Each hidden layer consists of set of neurons where each neuron is attached to all neurons in the previous layer in fully connected manner and neurons present in a single layer function are independent completely and have not any connections in shared manner. The last fully connected layer used in the network named as output layer and in classification it represents the class scores. The problem occurs say if we have images of size $200 \times 200 \times 3$ that would lead to 1,20,000 weights which is wasteful and huge number of parameters often results in overfitting.

On the other hand, the main advantage of using Convolutional Neural Networks (CNNs) is that the images are provided as an input that constrain the architecture in a more sensible way. The layers of a convolutional network have neurons arranged in three dimensions: width, height and depth unlike from regular neural networks. Depth belongs to number of channels used. For grey scale images depth is equal to one and for colored images depth is equal to three: for red, blue and green colors. Here, the neurons in a layer will be connected not to all neurons in the previous layer in a fully connected manner, rather have connections with a small region of the layer proceeding it. The final output of the convolutional layer network will be a vector array of size equal to the total number of available classes, because by the end of convolutional network architecture, input full size image is converted into a single vector of class probabilities that are arranged along the depth dimension.

1.4 Layers And Other Hyper-parameters Used To Build Convolutional Neural Network

As shown in Figure 1.3, Convolutional layer network consists of number of layers which they used for calculating the results by computing the calculations via small region of the layer also called activation map window before it. Different layers used by the convolutional neural network are explained below.

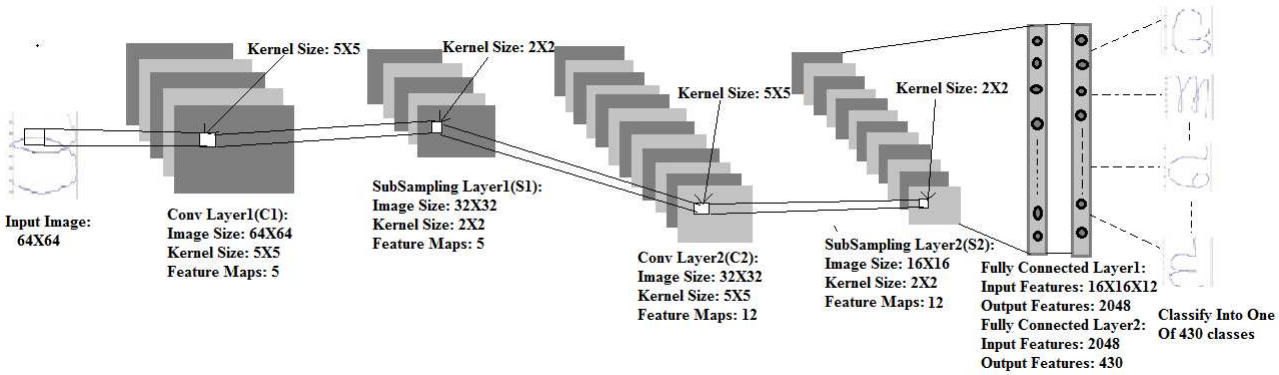


Figure 1.3: General Convolutional Neural Network (CNN) Architecture

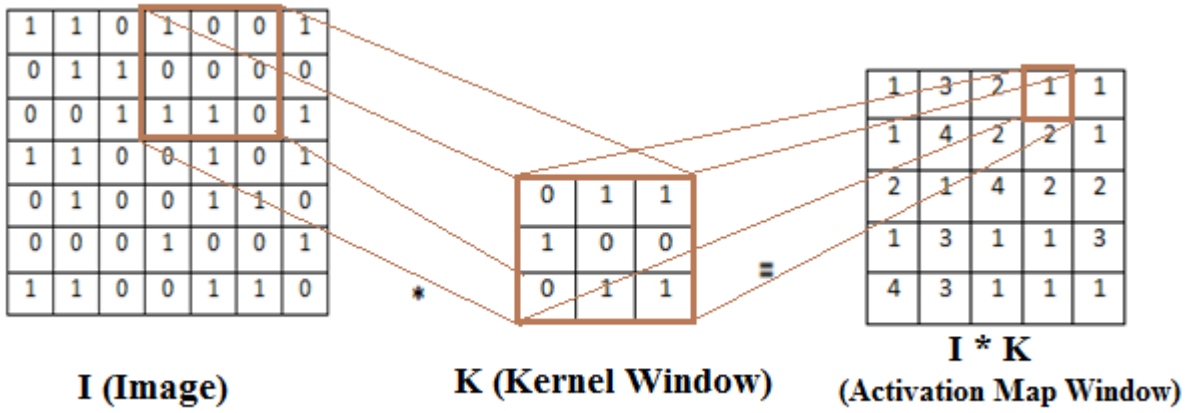


Figure 1.4: Example of convolutional layer

1.4.1 Convolutional Layer

The very first layer used in Convolutional Neural Networks (CNNs) is always a convolutional layer. As shown in Figure 1.4, in convolutional layer, a small window also called kernel size window slide across all the regions of the input image. This small window is also called kernel window or filter window and the region which is covered by it is called the receptive field. This field consists of array of numbers which are weight values or parameters. Depth of this filter will be same as that of depth of the input image. The first position of this filter window will be at the top left corner of the input image. As the filter is sliding or convolving around the input image, it is multiplying the values in the filter with the original pixel values of the image. Then these multiplications are summed up together so that you will get single number. We repeat this process for every location on the input volume. Every unique location on the input volume produces a number. After sliding the filter over all the possible locations of the input handwritten image, you get activation map or feature map. The second convolutional layer then works on this

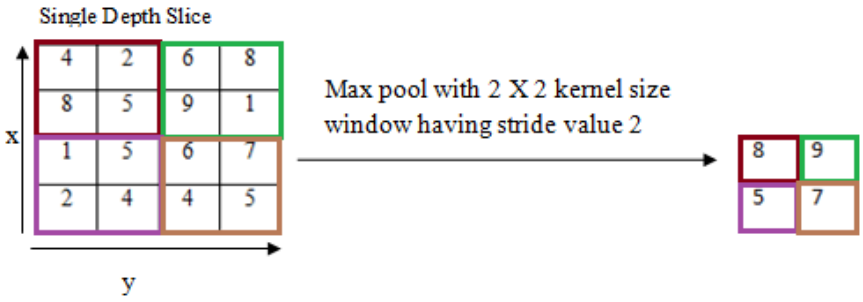


Figure 1.5: Example of max pooling layer having 2×2 kernel size window

layer output activation map instead of the original handwritten input image.

1.4.2 Pooling Layer

Pooling layer is placed in between the two convolutional layers. The function of pooling layer is to constituently reduce the spatial dimensions of handwritten input image to reduce the dimensionality of the input image and computations required in the network and hence control rate of overfitting. Every depth slice of the input image is taken into consideration by the pooling layer that resizes it spatially using MAX or AVERAGE operation. In MAX operation it picks up the highest values amongst the kernel size window say 2×2 window. It picks the average of the values coming under kernel size window in AVERAGE pooling operation.

In Figure 1.5, max pooling is used on input volume having kernel size window of 2×2. From each 2×2 kernel size window, maximum pixel value is picked up for consideration as it is considered as more relevant feature from its neighbourhood pixels that will be used efficiently to recognize and classify the input handwritten Punjabi character image.

Generally, Pooling layer used below parameters to reduce dimensionality of the input image.

1. Assume our input handwritten Punjabi character images are of following dimension:

   $W1 \times H1 \times D1$ which are width, height and depth of the input handwritten Punjabi character image respectively.

2. Two parameters are used :

   (a) Stride Value (S) and

   (b) Padding Value (F)

3. Produces images volume having following dimension:

   $W2 \times H2 \times D2$ whose values are computed as :

   (a) W2 = (W1 − F)/S + 1

   (b) H2 = (H1 − F) / S + 1

   (c) D2 = D1

1.4.3 Fully Connected Layer

Fully connected layer constitutes of neurons that are fully connected to all the neurons that are present in its previous layer same as in regular neural networks. Then matrix multiplication followed by bias offset is performed which is also called activation. The main difference that lies between the fully connected and convolutional layer is that in the convolutional layer, neurons are connected only to a small local region in the input and also they share parameters. However, the neurons in both the layers still compute dot products so their functional form is identical. Fully connected layer takes an input volume which will be the output of its previous layer and it outputs an N dimensional vector where N denotes number of classes that the program has to choose from. Each number in this N dimensional vector represents the probability of a certain class. The way in which fully connected layer works is that it looks at the output of the previous layer (activation maps of the high level features) and determines which features must correlate to a particular class. Class containing highest probability will be considered as the winning class in which we finally classify our input handwritten Punjabi character. In simple words, fully connected layer looks at what high level features most strongly correlate to a particular class and has particular weights so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.

1.4.4 Padding

The spatial dimensions of the input handwritten image goes on decreasing as we keep applying convolutional layers. We want to preserve as much information about the original input volume so that we can extract those low level features. Say we want to apply the same convolutional layer but we want the output volume to remain $32 \times 32 \times 3$. For this to happen, we can apply a zero padding to that layer. Zero padding pads the input with

zero around the border. Value of zero padding to be used can be calculated as :

$$\textbf{Zero Padding} = (\ k - 1)\ /\ 2 \tag{1.4}$$

Where, k is the filter window size, then the input and output volume will always have the same spatial dimensions. The formula used for calculating the output size for any given convolutional layer is given below.

$$\mathbf{O} = (\ (\ \mathbf{W} - \mathbf{K} + \mathbf{2P}\ )\ /\ \mathbf{S}\ ) + \mathbf{1} \tag{1.5}$$

Where O is the output height/length, W is the input height/length, K is the filter size, P is the padding and S is the stride.

0	0	0	0	0	0
0	2	5	1	8	0
0	9	1	8	5	0
0	1	6	7	7	0
0	4	2	1	9	0
0	0	0	0	0	0

Figure 1.6: Example of zero padding on 6×6 volume size image

1.4.5 Stride

Stride controls how the filter convolves around the input volume. If stride value is one, the filter convolves around the input volume by shifting one unit at a time. The amount by which the filter shifts is the stride.

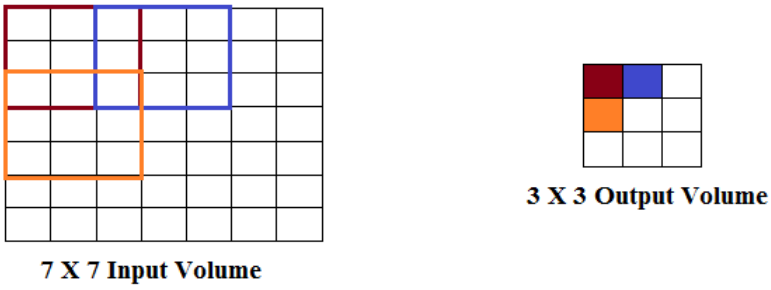


Figure 1.7: Example of stride value of 2 applied on 7×7 that will output 3×3 volume Image

In Figure 1.7, receptive field is shifting by 2 units now and the output volume shrinks as well.

1.4.6 Rectified Linear Units (reLu) Is Used As Activation Function

After each convolutional layer, it is convention to apply a nonlinear layer or activation layer immediately afterwards. The purpose of this layer is to introduce non-linearity to a system that basically has been just computing linear operations during the convolutional layers (elementwise multiplications and then summations). reLu activation functions are proven to be better than tanh and sigmoid function as reLu layers works far better because the network is able to train a lot faster without making a significant difference to the accuracy. The reLu layer applies the following function:

$$\mathbf{f(x) = max(0,x)} \tag{1.6}$$

to all of the values in the input volume. This layer just changes all the negative activations to zero. This layer increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the convolutional layer.

1.4.7 BackPropagation

Backpropagation can be separated into four distinct sections: the forward pass, loss function, backward pass and weight update. During the forward pass, you take a training image and pass it through the whole network. On our first training example, since all the weights or filter values were randomly initialized, the output will probably be like [.1 .1 .1 .1 .1 ..... .1], it means an output does not give preference to any number in particular. The network, with its current weights, is not able to look for those low level features or thus is not able to make any reasonable conclusion about what the classification might be. This goes to the loss function part of backpropagation. We are using training data that has both image and label. Loss function can be defined in terms of mean squared error (MSE) which is calculated as 1/2 **times (actual - predicted)** squared. Loss will be extremely high for the first couple of training images. And our goal is to get to a point where predicted label which is the output of the convolutional network is same as training label. In order to achieve this goal, we want to minimize the amount of loss value we have. We have to find out the inputs or weights which are most directly contributed to the loss or error of the network. So we perform backward pass through

the network which is determining which weights contributed more to the loss and finding ways to adjust them so that the loss decreases. Once we compute the derivative, weight update is performed. This is where we take all the weights of the filters and update them so that they change in the direction of the gradient.

Learning rate is a parameter that is chosen by the programmer. High learning rate means that bigger steps are taken in the weight updates and thus it takes less time to converge on an optimal set of weights. It also results in jumps that are too large and are not precise enough to reach the optimal point. The process of forward pass, loss function, backward pass and weight update is generally called one epoch. The program will repeat this process for a fixed number of epochs for each set of training images (a batch). Once parameter update is done on last training example, the network should be able to be trained well enough so that the weights of the layers are tuned correctly.

1.5 Steps Used In Handwritten Character Recognition

For recognition of any handwritten script, generally following process has been carried out. Each step followed during recognition of handwritten characters are discussed in brief below.

1.5.1 Preprocessing

Preprocessing includes centering of stroke, interpolation of missing points, smoothening of the curve using bezier curve implementation and rotation to clean the input image and smooth out the curves so that it can be recognized efficiently. Centering of stroke is needed as sometimes user writes at the edges of the screen so we have to centralise the handwritten characters. Interpolation of missing points is needed as sometimes few points are not captured by the screen between two consecutive points. Therefore, interpolation is used to insert that missing points. Smoothening of the curves is used to smooth out the curves using bezier curve implementation. Rotation is done if the input handwritten character image is tilted. Above specified preprocessing techniques are used to clean and smooth out our input handwritten character images so that the recognition and classification rate of the input images gets improved.

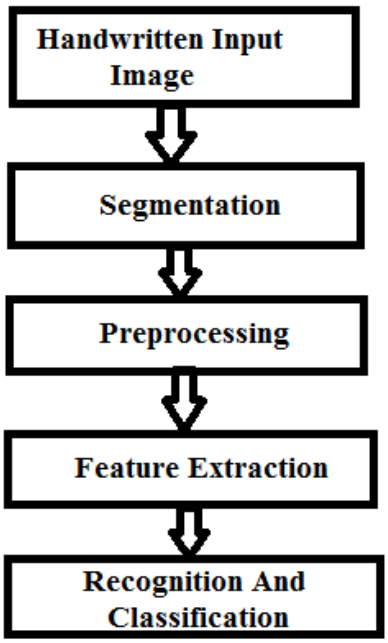


Figure 1.8: Steps Used In Handwritten Punjabi Character Recognition

1.5.2 Segmentation

It is the process of segmenting the handwritten Punjabi input word into their respective akshara characters. Horizontal projection (or profiling) and vertical projection (or profiling) is used to segment the Punjabi word into their respective akshara level after identifying the position of the headline stroke.

1.5.3 Feature Extraction

After segmentation, feature extraction computes the dynamic features (loops, curves, edges) of the image by convolving the filter size window over all possible regions of the input handwritten Punjabi character image. Random generated matrix each time when kernel window revolves around the handwritten input character image results in computing the features from the input image. More and more deeper we go inside the number of convolutional layers, more features has been extracted by identifying the curves, loops and edges of the whole input image.

1.5.4 Recognition and Classification

After segmentation, recognition the input handwritten Punjabi character image is performed and categorises in one of large number of classes available. After convolving around whole image, fully connected layer is used at end and then classifier is used to compute the probabilities which predicts the chances of input handwritten character to belong to a particular class. Class containing highest probability is the resultant class in which the handwritten input Punjabi character image is finally categorised at end.

1.6 Applications Of Handwritten Punjabi Character Recognition

Punjabi handwritten character recognition has many applications such as -

- Recognition of postal codes.
- Signatures recognition.
- Digital libraries.
- Invoice and receipt processing.

Chapter 2

Literature Survey

LeCun et al.[**1**] used backpropagation network for the recognition of handwritten digits. Preprocessing techniques are applied for cleansing and smoothening of the handwritten input images. The input provided to the developed neural network are the normalized handwritten isolated digit images. Error rate of 1% was achieved when neural network having backpropagation is applied on zipcode digits.

Verma[**2**] used multilayer perceptron and radial basis function neural networks for recognizing the handwritten characters of Hindi script. Backpropagation error algorithm is also used to improve the recognition rate of handwritten characters. In this proposed system, they compare the results obtained from multi-layer perceptron and radial basis fucntion (RBF) networks. Dataset consists of 245 samples written by five different users. The results so obtained shows that multilayer percepton (MLP) networks performs better than that of radial basis functions. But MLP networks training time is more as compared to radial basis function networks. Highest recognition rate so obtained using multilayer perceptron (MLP) networks and radial basis function are 85.01% and 70.8% respectively.

Bunke et al.[**3**] proposed a offline cursive handwritten character recognition system based on Hidden Markov Models (HMMs). Features that are computed in HMMs based on the arcs present on the skeleton graphs of the written words that has to be recognized. Dataset samples used in this proposed system was two dictionaries of 150 words written by various writers. Highest accuracy recognition rate achieved by this proposed system are 98% over word level.

Chi et al.[**4**] used self-organising maps and fuzzy rules for the recognition of handwritten numerals. SOM algorithm produces prototypes which alongwith variances are used to find out the fuzzy regions and related membership functions. Then fuzzy rules are generated from these learning training patterns. Input pattern is classified using fuzzy classifier in the recognition stage. And any pattern that remains unsure are classified using SOM classifier. Dataset sample used consists of 20,852 handwritten numerals that contains 10,426 training samples and 10,426 samples used for testing. Highest recognition results by using DNNC, SOM and FRSOM methods in this proposed system are 98%, 96.30%

and 96.80% respectively.

Verma[**5**] used multilayer perceptron and radial basis function neural networks for the recognition of handwritten Hindi character recognition. Error backpropagation algorithm was used to train the MLP networks. Dataset consists of two hundred forty five samples collected from 5 writers. They suffered long training time than that of RBF networks. Results prove that MLP networks trained by error backpropagation algorithm were superior in recognition accuracy and memory usage. Hindi language consists of 49 characters (13 vowels, 36 consonants) which is written from left to right. Each MLP network uses two-layer feed forward network with nonlinear sigmoidal functions. The output layer consist of 49 neurons each for one character. The maximum accuracy rate obtained was 84.10%.

Cho[**6**] used neural network for recognizing the handwritten constrained numeral characters. Pattern recognition problems are the main branch of Artificial Intelligence (AI). In this proposed system, three neural network classifiers are used to solve complex computational pattern recognition problems. First is multilayer perceptron (MLP) classifier. Second is Hidden Markov Model (HMMs). And third is structure adaptive self-organising map (SOM) classifier. Database of handwritten numerals is taken from Concordia University, Montreal, Canada. Highest recognition achieved by using these three different methods are 97.35%, 96.55% and 96.06% respectively.

Li and Yeung[**7**] used dominant points of strokes to recognize online handwritten alphanumeric characters. As all the alphanumeric characters consists of one or more strokes. When digitizing pad is used for writing, stroke information is stored i.e. the dominant trajectory points information is stored in database for each stroke in a specific temporal order. Directional information of the strokes resukts in pre-classification of the characters and positional oinformation will lead to final classification. Dataset samples consist of 62 character classes written in different styles by 21 people. Highest recognition rate results obtained from this proposed system is 91% having rejection rate of 1.1%.

Lawrence et al.[**8**] designed a system for face recognition using convolutional neural networks. As faces consist of multidimensional complex structure which makes it difficult to obtain high recognition results. Database consists of 10 images of each 40 individuals thereby a total of 400 images. This proposed system uses hybrid approach that uses the combination of local sample points and a self-organising map (SOM) neural network and also a convolutional neural network. Quantization of input images is done using SOM which results in reduction in images dimensionality. Convolutional network performs translation, rotation and scaling of input images. It extracts deeper features using cascading set of layers. Five images of each person is used for testing. An error rate of 3.8%

is achieved using this proposed system.

Lehal and Singh[9] provide a system for recognizing handwritten characters of Gurumukhi script. The major problems faced in the recognition of handwritten Gurumukhi characters is their unique features used in the script such as connectivity of the characters, position of the headline stroke, presence of similar shape characters. In this proposed system, recognition is done at sub-character level. Segmentation is done to divide the handwritten characters into sub-character level. Features are computed from this segmented zones and then binary decision trees and nearest neighbours are used for the classification. 96.60% accuracy recognition result rate was achieved from this proposed system.

Plamondon and Srihari[10] used convolutional neural networks to recognize the machine printed Gurumukhi numerals. Four convolutional layers which consists of two fully connected layers, single locally connected layer and two consecutive max pooling layers. Multilayer feed forward neural network is used to train the model. Each convolutional layer provides some dynamic features of the image by using max pooling layers of size 3×3. The highest recognition rate achieved was 94.14%.

Koerich et al.[11] used hybrid approach to combine neural networks alongwith Hidden Markov Models (HMMs) to recognize handwritten vocabulary words. The handwritten input data is provided to lexicon-driven word recognizer first that is based on HMMs that generates a list of the N best candidates whose word scores computation results are high. Afterwards, Neural Network (NN) classifier is used to compute the scores with respect to each segmented character and at the end, the scores from the HMM and NN classifiers are combined to give efficient results. Dataset used for this proposed system consists of total of 80,000 vocabulary words. From experimental results, it is proven that using hybrid approach of neural networks alongwith Hidden Markov Models (HMMs), there is around 10% improvement in the recognition accuracy results as compared to accuracy obtained over the HMM system alone. Highest accuracy obtained by HMM alone and hybrid approach of using HMM alongwith neural networks gives recognition accuracy results of 85.10% and 92.36% respectively.

Bhattacharya et al.[12] used multilayer perceptron classifiers for the recognition of handprinted Bangla numerals. Different feature extractors like presence of loops, terminal nodes, junctions are used to identify the characters and then multilayer perceptron neural network is used to classify the handwritten Bangla numerals based on these extracted features. Dataset of 1,800 Bangla numerals are used for training and testing the developed neural network model. 93.26% recognition accuracy rate was obtained from this proposed system and rejection rate of 1.71% was achieved on separate test data of 7,760 samples.

Also, 1,440 samples of validation data was used to predict the accuracy of the developed multilayer neural network.

Sharma et al.[13] used elastic matching method to recognize and classify online handwritten Gurumukhi characters. The recognition process is classified into two stages. In first stage, recognition of strokes is performed and in second stage, the input handwritten character is evaluated on the basis of recognised strokes found prior in first stage. The respective strokes used to write input handwritten Gurumukhi character is stored in the character database in a specific temporal order. The database used to store the strokes of a Gurumukhi character contains the information like script number, stroke number and respective stroke sample points for every xy trajectories of each used stroke. Dataset samples contains 41 Gurumukhi characters written by 60 different writers. 90.08% accuracy recognition rate was achieved from this proposed system.

Sharma et al.[14] proposed a system for the recognition of handwritten Gurumukhi characters using elastic matching method. Two stages are used in this proposed system to classify the images. Strokes are recognized in the first stage. And evaluation of characters is done based on the recognised strokes in the second stage. To improve the recognition accuracy results, features extraction is done. Stroke database is used to store the used strokes information i.e. stroke script number, stroke number and trajectories values corresponding to each stroke. Dataset samples are collected from 61 writers. Each writes 41 characters of Gurumukhi script. Highest recognition result rate obtained from this proposed system is 90.08%.

Graves and Schmidhuber[15] used multi-dimensional for recognizing the offline handwritten characters using recurrent neural networks. This proposed system takes the raw pixels image data as an input and then multidimensional recurrent neural network is built on this collected information. The three main things which are used in this proposed system to classify the handwritten characters are: a multidimensional recurrent neural network and LSTM, the connectionist temporal classification output layer and the heirarchical structure of the network. All these three components are combined to make a complete system. Highest accuracy recognition rate obtained using this proposed system is 87.20%.

Liu and Suen[16] proposed a system for the recognition of handwritten Bangla and Farsi numeral characters. Binary and gray-sacled images are used in this proposed system. Preprocessing techniques are applied on the handwritten image samples for the smoothening and cleansing of image data samples. Three databases are used in this system named as ISI Bangla numerals, CENPARMI Farsi numerals and IFHCDB Farsi numerals. After preprocessing, features are computed from these handwritten images and then fed to

the classifier for recognition and classification. Highest accuracy recognition rate results obtained on these three databases are 99.40%, 99.16% and 99.73% respectively.

Bhattacharya and Chaudhuri[17] provided system for handwritten numeral databases of indian scripts and also multistage recognition of mixed numeral is performed. Different handwritten scripts are used in this proposed system: Devanagari, Bangla and English. Database consists of 22,556 and 23,392 handwritten samples if isolated characters of Devanagari and Bangla collected from various real-life situations. After data collection, these are fed to the three multilayer perceptron classifiers in a cascaded manner. If rejection occurs at highest resolution, then another multilayer perceptron is used to recognize and classify the handwritten characters by combining output results of three classifiers of the previous stages. Handwritten numerals are collected from Indian postal mails and tabular form documents. Recognition accuracy rate of 99.26%, 98.47%, 97.52% for Bangla, Devanagari and English script was obtained.

Bhattacharya and Chaudhuri[18] used segmental hidden markov model to recognize and classify the online handwritten characters. Features are extracted by learning from samples used for training, learning from characters and adaptation according to the user specific requirements. This proposed system deals with two-dimensional Latian and Asian characters graphical shapes, symbols, and geometrical shapes. Highest recognition rate obtained from this proposed system is 97.30%.

Pal et al.[19] proposed a system for recognition of handwritten characters based on different features and classifiers and thereby compared their accuracy recognition results. Twelve different classifiers and four feature sets are used for handwritten Devanagari characters to compare their accuracy results. Different classifiers used in this proposed system are subspace method, support vector machines, projection distance, mirror image learning, linear discriminant function, nearest neighbour, Euclidean distance, k-nearest neighbour and compound modified quadratic discriminant function. Based on curvature and gradient information obtained from input gray-scaled images, feature sets are extracted from these input images. Highest accuracy recognition results obtained using different classifiers namely PD, SM, LD, SVM, MQDF, MIL, ED, NN, k-NN, MPD,CPD, CMQDF are 93.21%, 93.04%, 87.86%, 93.96%, 94.42%, 94.94%, 78.99%, 87.19%, 89.91%, 94.15%, 94.62% and 94.56% respectively.

Bhattacharya and Chaudhuri[20] used multistage approach for the recognition of handwritten Indian scripts databases and also database of mixed numerals. Dataset used contains total of 22,556 and 23,392 samples of isolated handwritten Devanagari and Bangla characters. In this proposed system, the input handwritten character is fed to three multiperceptron (MLP) classifier in a cascaded manner. If highest resolution rejects

the input handwritten image then other MLP classifier is used to classify the character by combining the results of all three cascading classifiers. Highest accuracy recognition results obtained by this proposed system are 65.20% , 70.85% for Devanagari and English numerals respectively. Also for recognition of handwritten English and Bangla samples, highest recognition results achieved are 98.64% and 98.05% respectively.

Shrivastava and Gharde[21] used support vector machine (SVM) for recognition of handwritten numeral characters of the Devanagari script. Database consists of 2,000 samples collected from 20 different people written in different writing style. Two techniques are used for features extraction: Moment invariant and affine moment invariant. 18 features are extracted from each image using these two techniques that are further fed into support vector machine (SVM) classifier for recognizing handwritten input characters. SVM uses binary classification techniques and also linear kernel function is used in SVM. Higher recognition rate result obtained using SVM in this proposed system is 99.48% that is proven to be highest recognition result obtained from all the previous work of recognizing handwritten numeral characters of Devanagari script.

Sharma and Jhajj[22] performed isolated handwritten Gurumukhi characters recognition. Zoning method is used for feature extraction in which the frame used to represent the character is segmented into various overlapping or non-overlapping zones. Densities of each zone is carried out and then computed features are fed into the classifiers. In this proposed system, two classifiers named as k-Nearest neighbor and support vector machine was used to recognize and classify the handwritten Gurumukhi characters. Euclidean distance is calculated between all the references point in k-nearest neighbor and then they are ordered in ascending order. SVM classifier used computed features to generate hyper-surfaces in the feature space thus helps in discriminating two classes of feature vectors. Highest recognition rate of 72.83% was achieved.

Patil and Shimpi[23] used neural networks for recognizing the handwritten characters of English script. In this proposed system, a character matrix generated form handwritten characters is used for feature extraction. Neural network uses feed forward approach to train the model and backpropagation technique to update the weights and bias values so as to obtain low error rate results. Multilayer perceptron (MLP) neural network is used having single hidden layer. Backpopagation techniques used in the developed neural network results in achieving 70% accuracy recognition results.

Espana-Boquers et al.[24] proposed a system for offline handwritten text recognition based on hybrid approach of neural networks and hidden Markov models. Preprocessing techniques like slant removal, normalization of size of the input images is performed to smooth out and clean the input images data using multilayer perceptrons. Structural part

of the proposed system are modeled using Markov chains and the multilayer perceptron (MLP) classifiers are used to estimate the required emitted probabilitites. 29.8% word error rate is obtained using this approach.

Maloo and Kale[25] used support vector machine (SVM) to recognize and classify the handwritten Gujarati numerals. Morpholical operations are used as preprocessing techniques. Then feature extraction is performed. Features are computed from the handwritten Gujarati numerals by first segmenting each isolated Gujarati numerals into blocks. Segmentation results in creation of four sets of features for each handwritten Gujarati numeral. After segmentation, affine invariant features are derived from these segmented blocks of each Gujarati numeral. Features computed from these four blocks of each Gujarati numeral are then provide as an input to support vector machine (SVM) classifier. 91% accuracy recognition rate was achieved from this proposed system.

Patel et al.[26] used neural networks for recognizing the handwritten characters. This system is designed to recognize the handwritten characters from a given scanned documents and then study the effects of changing the models of artificial neural networks. Neural networks are used for pattern recognition task. Multilayer feed forward network with backpropagation is used. Preprocessing techniques are used for segmentation of data, normalization of characters and de-skewing. The various parameters of the neural networks such as number of hidden layers to be used and number of epochs used are varied and accuracy is noted down at different instant of time. Highest accuracy obtained by this proposed system via using number of hidden layers as 78-26-78 is 91%.

Siddharth et al.[27] used number of feature sets and classifiers for the handwritten Gurumukhi numerals recognition. Three feature sets and three classifiers are used in this proposed system. First feature set is composed of distance profiles having 128 features. Second feature set is composed of different types of projection histograms having 190 features. Third feature set composed of zonal density and background directional distribution forming 144 features. Three classifiers used are SVM, PNN and K-NN. 5-fold cross validation accuracy is being observed for each feature set and classifier. Highest recognition results obtained using PNN, k-NN classifiers having third feature set are 98.33% and 98.51% respectively while highest accuracy recognition result obtained using SVM having second feature set is 99.20%. SVM classifier is used with Radial Basis Function (RBF) kernel.

Krizhevsky et al.[28] trained a large, deep convolutional neural network to classify 1.2 million high resolution images in the ImageNet LSVRC-2010 contest into 1,000 different classes. On the test data, they achieved top-1 and top-5 error rates of 37.50% and 17.01% respectively. Neural network developed contains 60 million parameters and

6,50,000 neurons. Neural network so developed constitutes of five convolutional layers, that are followed by max-pooling layers and three fully-connected layers with a final 1,000-way softmax. Dropout is used to reduce the rate of overfitting. In the second-best entry, they achieved error rate of 15.30% top-5 test error rate.

Bhattacharya et al.[29] used an efficient two stage approach for the recognition of handwritten Bangla characters. Database consists of 37,858 handwritten samples and accommodates a large spectrum of handwriting style by Bangla speaking population. Various preprocessing steps are binarization, size normalization, noise cleaning, headline truncation are applied on the input handwritten character to clean and smooth out the input images. Two stages has been used for the recognition of handwritten characters. In first stage, they used classifier for all underlying 50 basic character classes. Feature extraction is done via gradient descent approach in second stage by using rectangular grid that consists of horizontal and vertical lines over the character bounding box. Highest accuracy rate obtained for recognition of handwritten bangla characters was 95.84%.

Sinha et al.[30] used hybrid features extraction methods for recognition of offline isolated handwritten Gurumukhi numeral recognition. In this proposed system, hybrid approach of zoning is used which combines the centroid of zones and centroid of images. To calculate centroid of image, character zone is segmented into equal zones and centroid of image and average distances from centroid of characters to each zones present in the handwritten images are computed. And in zone centroid zone, character image is segmented into equal zones and then centroid of each zone and average centroid zone is calculated. An accuracy of 99.73% was achieved by this proposed system.

Mehrotra et al.[31] used convolutional neural networks for recognition of unconstrained online handwritten Devanagari characters. 42 Devanagari character classes are supported by this CNN architecture. Dataset samples of isolated Devanagari script are collected for this proposed system from different states of India. Using a hybrid approach, features are extracted from the collected words upto their respective character level that covers all the possible variations that occurs because of variation in writing styles of different persons. Experiments with 10 different configurations of CNN and for both exponential decay and inverse scale annealing approaches to convergence, show high promising results. Final layer outputs of top 3 configurations are averaged and classification decision achieved an accuracy of 99.82%.

Munish Kumar, R. K. Sharma and M. K. et al. Jindal [32] performed segmentation of lines and words for handwritten Gurumukhi script documents. After preprocessing techniques for any handwritten character recognition system, it is necessary to segment the text into lines, words and characters before the recognition of the text. Gurumukhi

script can be segmented into paragraphs, lines, words and characters. Gurumukhi script documents are segmented into lines with the use of strip based projection profile technique and to segment lines into words they used white space and pitch method. Horizontal profile projection is used for line segmentation and vertical profile projection is used for word segmentation. Using smearing technique consecutive black pixels in horizontal direction are smeared. If distance between white pixels is within threshold, then it is filled with black pixels. They achieved an accuracy rate of 98.40% for word segmentation and 93.70% for line segmentation in handwritten Gurumukhi script documents.

Vyas and Verma[33] provides a graphical independent platform technique for recognizing the online handwritten characters. Multiple segments has been obtained by segmenting the input handwritten character that is further fed to the recognition system. In this proposed system, segmentation of the input handwritten Gurumukhi characters are not dependent on their respective shapes. Rather it segments the input handwritten Gurumukhi character into points strokewise. It runs on any HTML5 enabled browser. As segmentation is not based on the input handwritten character shapes, rather use other tools to segment them thereby provides higher recognition results comparatively.

Chen et al.[34] Beyond Human Recognition proposed a system for the handwritten character recognition using convolutional neural networks. Due to varying writing style of different users, it is difficult to obtain high recognition rate results. In CNN based framework, CNN network structure containing cascading of layers is used to train the network model according to the input character. 0.18% error rate is achieved using this proposed system.

Maitra et al.[35] used CNN based common approach for recognizing the handwritten characters of multiple scripts. Convolutional Neural Network (CNN) is used as an efficient unsupervised feature vector extractor. Here they used 5 - layer CNN for moderately large class character recognition problem. For each case, a distinct Support Vector Machine (SVM) was used as corresponding classifier. CNN is trained using samples of standard 50 - class Bangla basic character database and features have been extracted for 5 different 10 - class numeral recognition problems of English, Devanagari, Bangla, Telugu and Oriya. The accuracy rate obtained for different scripts Bangla basic characters, Bangla Numerals, Devanagari Numerals, Oriya Numerals, Telugu Numerals and English Numerals was 95.60%, 98.38%, 98.54%, 97.20%, 96.50% and 99.10% respectively.

Kaur and Gurm[36] used convolutional neural networks for recognition of machine printed Gurumukhi numerals. Random generation matrix is one of the feature extraction method that is used in this proposed system. CNN concentrates on the dynamic features of the image. Accuracy of the work is measured with k-means and HOG algorithms.

By applying combination of multi-level pooling and convolutional layers, they achieved recognition rate of 95.67%. HOG and SVM method overcome the main challenges associated with the natural scene images like complex background in the images, different font styles and orientation of the text. Backpropagation is used to reduce the rate of overfitting and to update the values of weights and biases to achieve efficient recognition rates.

Chapter 3

Problem Statement

3.1 Problem Formulation

Computers have influenced the life of human beings to a great extent in today's world to make it flexible. Handwritten recognition techniques plays a vital role in various applications such as signatures identification, in banking applications and so on which is less error prone as compared to taking input via mouse or keyboard. Handwritten recognition system comes under the major field of Artificial Intelligence (AI) named as pattern recognition field.

The major problem faced in any handwritten character recognition system is the variation occurs in the writing styles of different users and also the variation occurs in the handwriting style of user's own style because of change in mood, speed of writing at different instant of time. Because of these variations in the writing style, it is difficult to obtain higher accuracy recognition results and thereby making this topic an interesting research topic to be considered.

Neural networks are used for this approach as neural networks processes the information in same way as human brain does. Convolutional neural networks (CNNs) performs better even simple neural networks as CNN helps in reduction of overall complexity of the developed neural network as it results in reduction in number of mathematical computations required to be computed. As more and more convolutional layers we used to make convolutional neural network, more deeper features can be computed from the handwritten input character images. The major problem occurs during the implementation of convolutional neural networks is how to fine tune the different parameters of the convolutional neural network to train the developed neural network so as to increase the overall recognition accuracy rate high and thereby less error rate.

3.2 Research Gaps

Following are the gaps that are identified during literature survey:

- The already used techniques involves use of Hidden Markov Models (HMMs), zone density based classification , statistical and distribution feature methods for the classification of handwritten characters that includes computation of complex probabilities calculations at a large scale.
- Deep learning technique named as Convolutional Network Networks (CNNs) has been implemented on many handwritten scripts such as English, Devanagiri, Gurumukhi numerals. But comparatively less research work has been done on Punjabi script using convolutional neural networks.
- Tensorflow library is not used before for the implementation of convolutional neural networks over Punjabi script.
- All the present techniques used to classify handwritten input characters has been implemented for single CPU system. But using Tensorflow library, if dataset size is large, we can use multiple CPU systems using NVIDIA GPU's for parallel processing for better efficiency and computing execution results.

3.3 Research Objectives

The following research objectives are formulated:

- Use of deep learning techniques such as Convolutional Neural Networks (CNNs) for the recognition and classification of Punjabi handwritten characters.
- To use Google's open source library for the implementation of convolutional neural networks as Tensorflow provides many in-built functions for the computation of complex calculations. And using different sample ratios for training, testing and validation data and compare their respective results.
- As more and more deeper convolutional layers are used to build the convolutional neural networks, the complexity of computing mathematical calculations also gets reduced rapidly and more deeper features such as lines, curves, edges, headline are able to find out efficiently.

Chapter 4

Data Collection, Pre-Processing And Segmentation

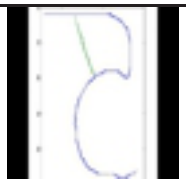
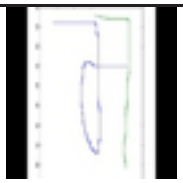
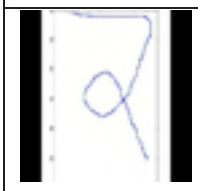
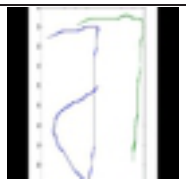
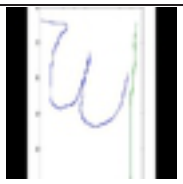
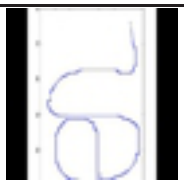
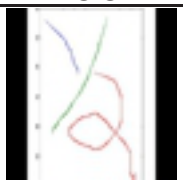
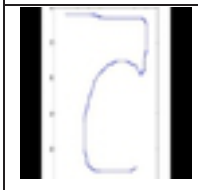
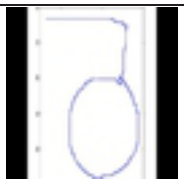
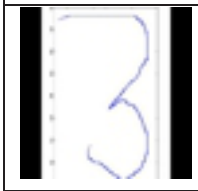
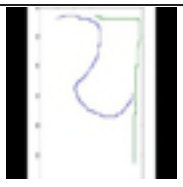
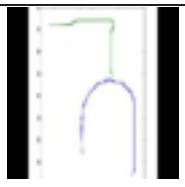
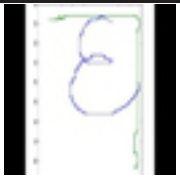
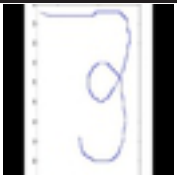
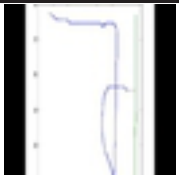
4.1 Overview Of Punjabi Script

Punjabi is mostly widely spoken language over millions of native speakers. It comes at 14th position in most widely spoken languages in the world. Punjabi word is composed of two parts: "Panj"and "ab"where "Panj"indicates five and "ab"refers to five major tributaries of the Indus river. Punjabi language originates from the Sanskrit language. Punjabi is oftenly written by the people of East Punjab in the Gurumukhi script. Punjabi script is written from left-to-right direction and in top down approach. Punjabi is a Sikh script modified, standardized and used by the second Sikh Guru, Guru Angad Dev Ji. Punjabi script has 35 consonants, 6 special characters and 10 vowel identifiers.

Punjabi script can be divided into three horizontal zones namely upper zone, middle zone and lower zone. The region above the headline, where some of the vowels and subparts of some other vowels resides represents upper zone, while the area below the headline where the consonants and some sub-parts of vowels are present represents middle zone. The middle zone is the busiest zone. The area below middle zone where some vowels and certain half characters lie in the foot of consonants represents lower zone.

Figure 4.1: Different zones and headline of Punjabi script

Table 4.1: Consonants used in Punjabi script

				
Oo'rha	Ai'rhaa	Ee'rhee	Sas'saa	Haa'haa
				
Kak'kaa	Khakh'khaa	Gag'gaa	Ghag'ghaa	Ngan'ngaa
				
Chach'chaa	Chhachh'chhae	Jaj'jaa	Jhaj'jhaa	Njan'njaa
				
Tain'kaa	Thath'thaa	Ddad'daa	Dhad'daa	Nhaa'nhaa
				
Tat'taa	Thath'thaa	Dad'daa	Dhad'ddaa	Nan'naa
				
Pap'paa	Phaph'phaa	Bab'baa	Bhab'baa	Mam'maa
				
Yay'yaa	Ra'raa	Lal'laa	Vav'vaa	Rhar'rhaa

4.2 Punjabi consonants and vowel identifiers

Table 4.1 shown below represents 35 consonants of Punjabi script. The present study is focused to recognize these characters in online handwritten recognition system. There are six special characters and ten vowel identifiers in Punjabi script. Therefore by including these Punjabi alphabets images and with the combination of ten vowel identifiers along with each Punjabi character, we get a total of 430 classes. And for each class, we collected 100 images dataset. Thereby getting a total of 43,000 images dataset.

Punjabi script also uses vowel identifiers which are also pronounced as matras. Various vowel identifiers used in Punjabi script is shown in Table 4.2.

Table 4.2: Some vowel identifiers used in Punjabi script

Vowel Symbols Used	Vowel Identifier Name	Vowel Symbols Used	Vowel Identifier Name
	Auṅkar		Dulaiṅkar
	Adhdhak		Bihārī
	Lāṃvāṃ		Dulāṃvām
	Kannā		Hōṛā
	Kanauṛā		Ṭippī

By using combinations of these vowel identifiers alongwith the consonants of the Punjabi Script, we can have number of classes in the Punjabi script. In this proposed system, we are using in our dataset: Punjabi consonats, 6 special characters and combination of all the Punjabi consonats with each of the vowel identifiers. Sample images of Punjabi character "haa'haa" with the combinations of different vowel identifiers written by the same user at different instant of time is shown in Figure 4.2 and Figure 4.3. As there is variation in user's writing style at different instant of time due to change in mood, speed of writing used, it is difficult to obtain higher recognition results. The different vowel identifiers used in our proposed system dataset are named as : Adhdhak, Auṅkar, Bihārī, Dulāṃvām, Dulaiṅkar, Hōṛā, Kanauṛā, Kannā, Lāṃvāṃ and Ṭippī.

In our proposed system, we have used these 35 consonants of Punjabi script, 6 special characters and the combination of all these consonant characters each with respective vowel identifiers specified in Table 4.2. At the end we get total of 430 classes in Punjabi script. Each class contains 100 images in which variation occurs due to difference in the writing style of different users and also in the user's own writing style due to mood, speed of writing at different instant of time.

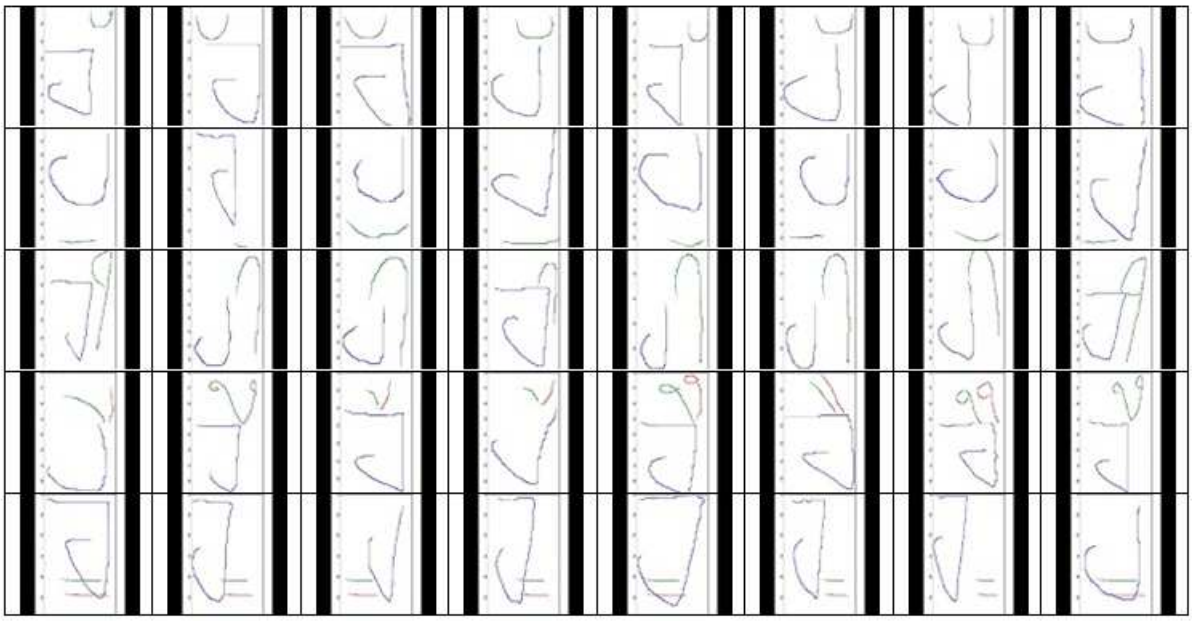


Figure 4.2: Sample of Haa'haa Punjabi character with vowel identifiers (Adhdhak, Auṅkar, Bihārī, Dulāṃvām and Dulaiṅkar

4.3 Overview Of Data Collection And Pre-Processing

Data collection and preprocessing techniques are the two phases required before the recognition phase in online handwriting recognition process. The following section includes collection of strokes information collected from the generated xml files for handwritten Punjabi words, and then data processing algorithms on this collected data is applied to refine the data so that it helps in the improvement of accuracy for recognition results. This chapter concentrates on data collection process, preprocessing algorithms and its various stages. These phases of online handwritten recognition system are not independent of each other and thus should be planned together. These stages should be followed as this will help in obtaining higher recognition accuracy results for classifying input handwritten Punjabi character into one of the available 430 classes.

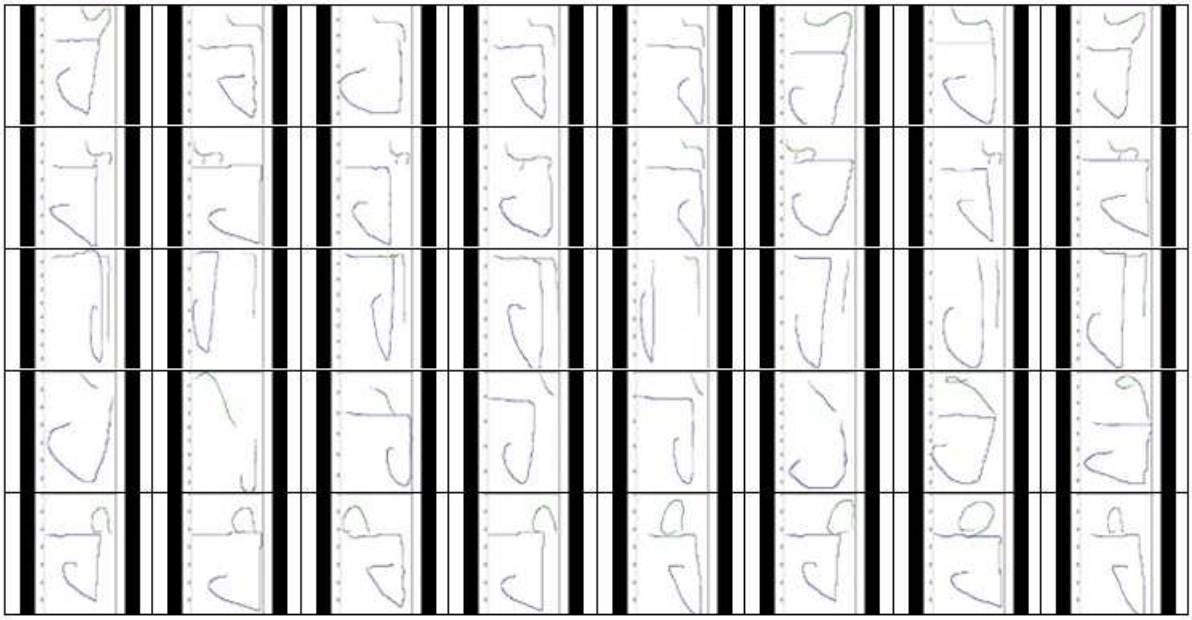


Figure 4.3: Sample of Haa'haa Punjabi character with vowel identifiers (Hōṛā, Kanauṛā, Kannā, Lāṃvāṃ and Ṭippī

4.4 Data Collection

Generally, the format used for online handwritten Punjabi character recognition is a sequence of coordination points or trajectories information of the moving pen point. The point when pen point touches the screen upto the point when it is picked up from the screen is considered as one stroke. Sampling of pen trace is done at a constant rate so that data points are collected will be distributed evenly in time but not in space. The sample points are located densely on the true pen trace when writing speed is slow whereas sparsely located points are obtained when quick writing is performed. Usually speed of writing typically slows down on sharp corners, in the beginning of the stroke and at the end of the stroke. Pen movements are generated when it touches the screen and picked up from the screen i.e. strokewise information is stored in the xml files for each user. We have taken 100 Punjabi words from 20 users i.e. we have 20 users xml files in which the respective stroke information of the handwritten Punjabi words written by them are stored in a temporal order.

Here to write a Punjabi character "Kak'kaa", user has used two strokes. Therefore, the data stored in xml file will be like: First the xy coordinates corrorsponding to stroke one are stored with corresponding akshara character and its unicode value. And then stroke two xy trajectories are stored with its corresponding akshara character and unicode value.

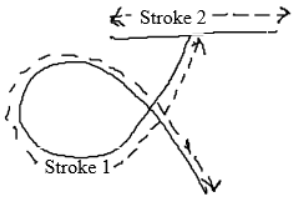


Figure 4.4: Character "Kak'kaa" written using two strokes

When the user write using stylus pen on the screen, the corresponding xy trajectories are stored in an xml file in a specific temporal order in which strokes are used to write a Punjabi word. The sample of xml file generated when a user wrote some Punjabi words are shown from Figure 4.5 to Figure 4.8.

```
<?xml version="1.0" encoding="UTF-8"?>
- <OHWRSchema>
    <dataSetDef/>
    <writerDef/>
    <annotationSchema>page textBlock line word stroke subStroke akshara strokeGroup headLine</annotationSchema>
  - <annotationDef>
      <pointOfOrigin/>
    - <document traceDimension="2" pageCount="1" encodingType="UTF-8">
      - <page textBlockCount="1" pageNumber="1" pageId="PAN_TUP_User_001">
        - <textBlock truthLevel="labeled" textBlockNumber="1" paragraphCount="1" noOfCorners="4">
            <polygon/>
          - <paragraph truthLevel="labeled" sentenceCount="1" paragraphNumber="1" lineCount="1">
            - <line truthLevel="labeled" wordCount="1" lineNumber="1">
              - <labelDesc>
                  <desc>ਅਣਪਛਾਤੇ</desc>
                - <annotationDetails numberOfCodeChars="7">
                    <codeSequence>0A05 0A23 0A2A 0A1B 0A3E 0A24 0A47</codeSequence>
                  </annotationDetails>
                </labelDesc>
              - <word truthLevel="labeled" wordNumber="4" wSentenceIndex="0" subStrokeCount="11" strokeCount="11" aksharaCount="5">
                - <labelDesc>
                    <desc>ਅਣਪਛਾਤੇ</desc>
                  - <annotationDetails numberOfCodeChars="7">
                      <codeSequence>0A05 0A23 0A2A 0A1B 0A3E 0A24 0A47</codeSequence>
                    </annotationDetails>
                  </labelDesc>
                  <qualityBits>1111</qualityBits>
                - <stroke truthLevel="labeled" subStrokeCount="1" strokeNumber="1">
                  - <labelDesc>
                      <desc>144</desc>
                    </labelDesc>
                  - <subStroke truthLevel="labeled" subStrokeNumber="1" subStrokeID="1">
                    - <labelDesc>
                        <desc>0</desc>
                      </labelDesc>
                    - <hwTrace NumberOfPoints="221">
                        <trace>218 41 218 41 222 43 223 44 224 45 225 45 225 46 226 47 227 48 227 49 228 50 228 51 228 52 229 52 229 53 229 54 229
                          55 230 56 230 57 230 58 231 58 231 59 231 60 231 61 231 62 232 63 232 64 232 65 232 66 232 67 233 68 233 69 233 70 233 71
```

Figure 4.5: Sample of Punjabi word "ANNAPCHHATE" (Sample of XML file generated for user1: Contains stroke information)

4.5 Preprocessing Phase

Before recognition and classification process, preprocessing phase should be applied on the collected handwritten Punjabi characters to remove noise and missing points present in the input text due to hardware and software limitations. Presence of sharp edges, non-centered text, uneven sizes of text and missing points in the text trajectories due to high speed of handwriting and slants in characters results in occurrence of noise. Preprocessing

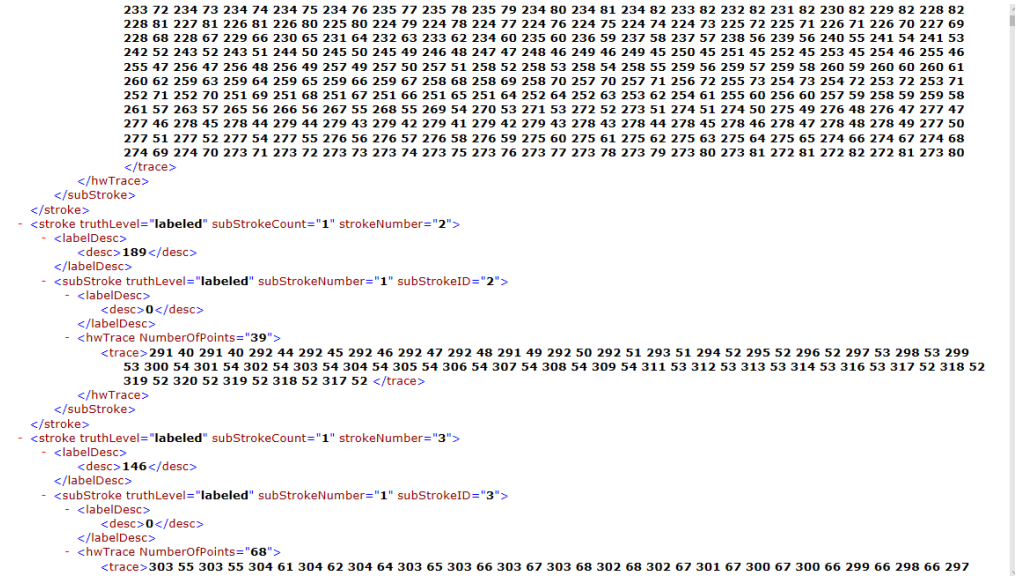


Figure 4.6: Sample of Punjabi word "ANNAPCHHATE" (XML file generated for user1: Contains stroke information) (Cont.)

Figure 4.7: Sample of Punjabi word "ANNAPCHHATE" (XML file generated for user1: Contains stroke information) (Cont.)

Figure 4.8: Sample of Punjabi word "ANNAPCHHATE" (XML file generated for user1: Contains stroke information) (Cont.)

steps are preferred as if applied they results in improvement of recognition accuracy rate of online handwritten Punjabi characters.

The following preprocessing steps have been applied on collected online handwritten Punjabi characters.

4.5.1 Size normalization and Centering Of Stroke

Different users have their own style of writing. Also variation can occur in the handwriting style of the same users due to speed of writing, change in mood at different instant of time. Depending upon the user way of writing, size of the input strokes also varies which depends on the movement of the pen on the writing pad. Also some users writes at the corners of the writing pad which are not centered as pen is moving along the border frame with assumed fixed frame size. Therefore we use size normalization so as we get same sized images and centering of strokes so that there will be lessen number of missing points due to sharp corners of the fixed frame. Below algorithm is used for size normalization and centering of strokes.

In this algorithm, we used a origin of frame which will be considered as frame of reference. Say this origin point trajectories values are represented by (x_0,y_0) and set of pixels which

our online handwritten Punjabi characters drawn should follow this rule:

$$(x, y) : 0 <= x <= L_x, 0 <= y <= L_y; \tag{4.1}$$

where L_x and L_y denotes maximum size of the fixed size frame window along x and y axis respectively. Assume that number of pixels used for writing a Punjabi character is n.

Algorithm

1. Specify frame size window dimensions to be used.

$$L_x = 64\ (pixels), L_y = 64\ (pixels) \tag{4.2}$$

   Where L_x and L_y denotes the maximum frame size dimension along x and y axis respectively. This is used for normalized the size of the input handwritten Punjabi character image.

2. Now, rearrange the pixel values of the strokes used for writing a Punjabi character so that they will lie within the range of frame window dimensions by taking the frame reference point as origin which is denoted by (x_0,y_0). Set

$$P_{ix} = P_{ix} \times (l_x/L_x) \tag{4.3}$$

$$P_{iy} = P_{iy} \times (l_y/L_y) \tag{4.4}$$

   for all points P_i in list $i = 1, 2, ..., n$ Here l_x and l_y are the xy trajectories corresponding to each stroke used in Punjabi character. P_{ix} and P_{iy} represents new xy trajectories of the corresponding stroke.

3. Now, rearrange the points by considering (x_0,y_0) as the frame reference point so that all the corresponding strokes pixel values will be centered inside the frame size window. Set

$$P_{ix} = P_{ix} \pm x_0 \tag{4.5}$$

$$P_{iy} = P_{iy} \pm y_0 \tag{4.6}$$

   Above algorithm normalizes the input images to 64×64 sized input handwritten images. And also images are centered in the frame window now by taking (x_0,y_0) as an origin reference point.

4.5.2 Interpolation of Missing Points

When the user writes on the writing speed with a pen with very high speed, then there will be missing points present in the input handwritten Punjabi word. These points can be calculated using Bezier and B-spline curves implementation. In this proposed system, we used piecewise bezier curve interpolation because it interpolates points among fixed number of points and also maintains equidistance between the points. In this piecewise bezier interpolation curve implementation, a set of four consecutive points are taken into consideration for obtaining bezier curve. Then the next set of four consecutive points gives next bezier curve and so on for all the points of all the strokes one by one. Below algorithm is used for interpolating missing points using piecewise bezier curve interpolation.

Algorithm

1. Create an empty list L which is used for storing the points which will generate from Bezier interpolation function.
2. Set t = Total number of strokes in the list which is used for writing Punjabi character. Set h = 1.
3. Repeat Step 4 for each stroke h, until $h <= t$ i.e. total number of strokes.
4. (a) Calculate n as the total number of points used in the current stroke h.

   (b) If ($n >= 4$) then CALL

   $$BezierCurve(P_i, P_{i+1}, P_{i+2}, P_{i+3}) \; ; \tag{4.7}$$

   for all points P_i; $i = 1, 2, ..., n-3$.

   (c) Else Set $h = h + 1$

   (d) End if

   (e) Update list L by incorporating the new points as the consecutive points which are obtained from the BezierCurve function.

   (f) Set $h = h + 1$
5. Exit.

Function BezierCurve

1. q is a variable such that $0 <= q <= 1$.
2. Set $q = 0.20$ and delta (u) = 0.20.

3. Repeat steps 4 and 5 until $u <= 1$.

4. Calculate x coordinate of new point as :

$$P_{ix} \times (1-q)^3 + P_{(i+1)x} \times 3 \times q \times (1-q)^2 + P_{(i+2)x} \times 3 \times q^2 \times (1-q) + P_{(i+3)x} \times q^3 \quad (4.8)$$

   and calculate new y coordinate point value as :

$$P_{iy} \times (1-q)^3 + P_{(i+1)y} \times 3 \times q \times (1-q)^2 + P_{(i+2)y} \times 3 \times q^2 \times (1-q) + P_{(i+3)y} \times q^3 \quad (4.9)$$

5. Set $q = q$ + delta (q).

6. Return.

Above algorithm helps in smoothening out of curve, edges which are missing because of high speed writing of the user.

4.5.3 Resampling of Points

Resampling of points is required so as to maintain equal distances between the points in the list. For any pair of points in the list having a distance greater than one, we add new point between such pairs. Any pair which is already having distance between them less than one remains untouched. Below algorithm is used for resampling of points between every possible pair in the list.

Algorithm

1. For all points in the list,
   If ($d > 1$) then, where d is distance between the two points, CALL
   BezierCurve (For interpolating missing points)
   End if.

2. For all points of a stroke in the list, remove points at constant distance with respect to total number of points in the stroke. This acts as filter which tells maximum fixed number of points to be used for writing a stroke and also helpful in retaining the shape of the stroke.

3. Exit

4.6 Segmentation

Segmentation plays a very vital role in the recognition of online handwritten character. Handwritten Punjabi words are divided into their akshara level based on their stroke information. The main challenge faced during the segmentation of characters is the identification of the headline and accordingly find our the lines. Curves and edges present in the three zones of the Punjabi characters which helps in the improvement of recognition accuracy rate. Horizontal and vertical profile projection techniques are used for the segmentation of handwritten Punjabi characters. Using Horizontal profile projection, we can calculate the total number of black pixels for each row i.e. upto the height of the character images. As we are using images of size 64×64, it means horizontal projection is calculated as :

$$HP(k),\ k = 1, 2, 3, 4....., N \tag{4.10}$$

Where N represents the height of the character image and k denotes the total number of black pixels for each row. The row containing highest number of black pixels are recognized as headline stroke.

Then, vertical profile projection is done to segment the characters into their corresponding akshara level. The vertical projection is then calculated as :

$$VP(i),\ i = 1, 2, 3, 4, ..., M \tag{4.11}$$

Where VP(i) denotes the total number of black pixels in the respective ith coloumn. Coloumn containing lowest number of black pixels denotes that there is space between the two characters which indicates that we can segment the character from this point after removing the headline stroke found after the horizontal profile projection.

Chapter 5

Implementation And Experimental Results

The main purpose of using Tensorflow for the implementation of Convolutional Neural Networks (CNNs) is that it uses a computational graph for execution purpose which works more efficiently as compared to the calculations that are performed in Python directly. Also Tensorflow is more efficient than numpy as in numpy, the computation of a single mathematical operation is known at a instant of time. On the other hand, Tensorflow use entire computational graph that will be executed.

Tensorflow also provides efficient inbuilt mathematical techniques to calculate gradient values that results in the optimization of the variable values to improve the accuracy of the developed model. Tensorflow is open source library provided by Google for performing numerical computations via data flow graphs. Graphs consists of edges and nodes where nodes indicates mathematical operations and edges represent multidimensional data tensors or arrays to provide communication between these nodes. Also, Tensorflow can be used for parallel processing in distributed environment by taking the help of multi core CPU's as well as GPU's.

The main things which a Tensorflow graph includes to make a computational graph are listed below:

1. For providing input to the computational graph, some placeholder variables are required.
2. Weights and bias variable values that are needed to be optimized to improve the accuracy recognition rate of the developed convolutional neural networks.
3. Cost measure function to indicate the difference between the actual and predicted values so as to improve the optimization of the convolutional neural network.
4. To update the values of the weights and bias variables, an optimization method is needed.

5.1 How Tensorflow works?

The process flow representing how optimization takes place using Tensorflow computational graph is show in Figure 5.1. Initially the input matrix that consists of pixel values ranged between 0 and 255 inclusive of the respective input handwritten character image is multiplied by the weight matrix which is randomly generated initially. After multiplying pixel values with their corresponding weight values then non linearity function value i.e. bias values are added to these summation of multiplication values. And then an activation function is applied on this summation result so as to reduce the number of mathematical computations needed. Different activation functions can be used here. For example, rectified linear unit (reLu) function is proven to be more effective than that of tanh function as reLu activation functions dumps all the negative valued functions and only considers the non negative valued function values. After that predicted values are calculated using sigmoid function and fully connected network at the end of the convolutional layers. Cost function value is measured by noting the difference between the actual and predicted values. Then an optimization function such as Adam optimizer, Gradient Descent optimizer or so on can be used to update the weight and bias values so as to reduce the cost function value or error rate and thereby results in improvement of handwritten recognition accuracy results.

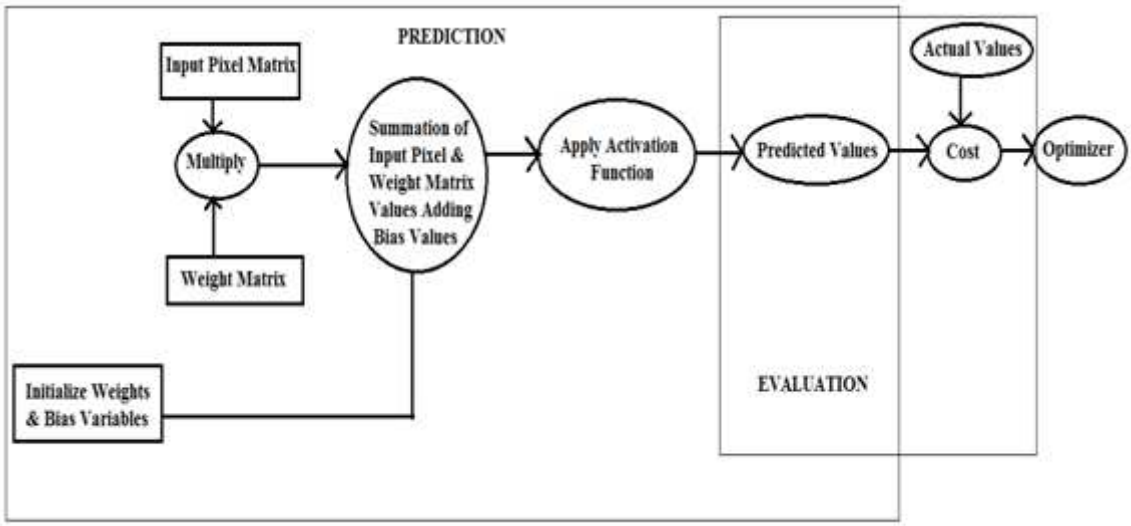


Figure 5.1: How optimization flow occurs

5.2 Convert handwritten Punjabi character images to Tensorflow format

Firstly, your all input images should be in .jpeg or .png format. Previously each of 430 Punjabi alphabet classes contains 100 images. Divide the dataset in 65:35 training set:testing set respectively. Make the directory structure as /train/urha/*.jpeg,....., /train/rahrha/*.jpeg for training data and similarly for testing data make directory structure as /test/urha/*.jpeg,....., /test/rahrha/*.jpeg. Tensorflow script is used that converts the training and evaluation data into a sharded dataset that consists of TFRecord files having directory structure like trainDirectory/train-00000-of-01024, trainDirectory/train-00001-of-01024 ,....., trainDirectory/train-01023-of-01024 and structure used for validation directory will be like validationDirectory/validation-00000-of-1028, validationDirectory/validation-00001-of-01028,....., validationDirectory/validation-01027-of-01028. Here in this example we have choosen 1024 and 128 shards for each data set. Each record within the TFRecord file contains following fields:

1. **Image/encoded:** String that contains JPEG encoded image having RGB colorspace.
2. **Image/height:** Integer value specifying image height that are in pixels.
3. **Image/width:**Integer value specifying image width that are in pixels.
4. **Image/colorspace:** String representing the RGB colorspace.
5. **Image/channels:** Integer value indicating the total number of channels used as 3 for RGB channels, 1 for grayscale images.
6. **Image/format:** String that specifies the format of the images say .jpeg.
7. **Image/filename:** String that specifies the basename of the images file used. For example “urha1.jpeg”, “kukka10.jpeg”.
8. **Image/class/label:** Index that specifies the index in a classification layer that belongs to the corresponding label value of the image used.
9. **Image/class/text:** String that specifies the corresponding label of the image used which is user friendly and easily readable by the user. For example : “urha”, “kukka”.

Create a separate .txt file that contains all the label name values that are used in your dataset. In this proposed system, we are using 430 classes. Then use buildscript.py file

by specifying the training data location, testing data location, labels.txt file in which all label names are written, outputDirectory where the Tensorflow data is to be stored that will obtain after conversion, number of threads used. You will get all your dataset images in their respective Tensorflow TFRecord file format. Then you can use these tensor objects file for training and testing the neural network.

5.3 Hardware And Software Requirements

The various pre-requisites for the implementation of proposed system are specified below:

- **CPU Architecture:** x86_64
- **System Memory:** 8-32 GB
- **OS:** Ubuntu 14.04
- **Python 2.7**
- **Numpy Library Of Python:** Used for performing mathematical computations.
- **Matplotlib Library Of Python:** Used for plotting handwritten input character images and resultant graphs.
- **Tensorflow Library:** Google's open source library used for the implementation of convolutional neural networks.

5.4 Tensorflow Installation

In this proposed system, Ubuntu environment is used for installation of Tensorflow library for the implemntation of Convolutional Neural Networks (CNNs). The following steps has been used for the installation of Tensorflow library:

- Firstly, install any virtual environment like VirtualBox, VMWare in which you put the image the Ubuntu 14.04 later on.
- Before installing Tensorflow, the pre-requisites are: Python and pip should be there in your Ubuntu environment. By default Python is already installed in your Ubuntu environment. To check its installed version, use below command:

$python - V$ for Python version and

$pip - V$ for pip version.
Note: We recommend that either pip or pip3 is installed successfully i.e. version 8.1 or higher version of pip should be installed. Pyhton 2.7 or Python 3.3+ should be installed. If lower versions are setup, then upgrade your Python and pip version using pip upgrade command in the prior step.

- After successful installation of Python and pip versions, install Tensorflow using below commands:
  - **pip install tensorflow**
  - Set

    TFPYTHONURL = https://storage.googleapis.com/tensorflow/linux/cpu /tensorflow-1.1.0-cp27-none-linux_x86_64.whl
  - Use below command:

    pip install –upgrade TFPYTHONURL

    This command will successfully install Tensorflow and all the required packages it needs automatically in your Ubuntu environment.
- To validate whether Tensorflow is installed in your system or not, you can use below sample code:
  - **import tensorflow as tf;**
  - **res = tf.constant("Hi I Am Working..!!")**
  - **sess=tf.Session();**
  - **print sess.run(res);**

  If tensorflow is successfully installed in your Ubuntu environment, then it will output: **"Hi I Am Working..!!"** in your terminal window.

5.5 Different Layers Used To Build Convolutional Neural Network In Our Proposed System

In the proposed approach, CNN[35] architecture is used for character recognition. We are not recommending here to train a separate CNN for each individual problem as it takes a considerable amount of time and effort to tune its parameters. Rather we use only one CNN for large number of characters to train them in batches. Number of iterations are used to train our model so that it can efficiently recognize the new random input handwritten character. To train our model, we are using deep learning toolbox of TensorFlow[37] library which is open source library provided by Google. This is a Python library used for the implementation of CNN. It provides many inbuilt methods to optimize, train and evaluate mathematical expressions that involves multi-dimensional arrays representing the height, width and depth of the input handwritten character. Depth indicates total number of channels. For grey-scale image depth is equal to one. In this proposed system, we are using 5 convolutional layers followed by two fully connected layers. Activation maps and optimizers are used to improve the recognition rate of the developed model.

5.5.1 Convolutional Layer 1 And Subsampling Layer 1

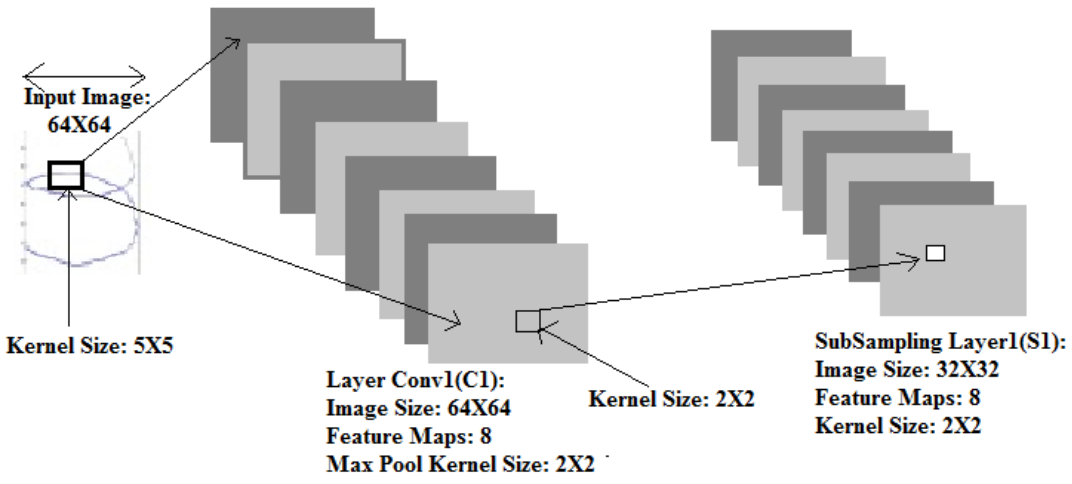


Figure 5.2: Computation From Input Image To First Convolutional C1 Layer And Subsampling S1 Layer)

Firstly, the handwritten input Punjabi character image is given as an input to first convolutional layer. The pixel values of the handwritten input Punjabi character are

passed as an vector array of size 64×64 as we are normalized our all handwritten Punjabi character image to a normalized phase of 64×64 in the preprocessing phase.

Weights and bias values are initially randomly assigned values. As shown in Figure. 5.2, a small window also called kernel size window of size 2×2 is allowed to convolve around the whole input handwritten Punjabi image by starting from top left corner of the input image and each time shifts that kernel window by stride value. Product between the kernel size window values and their corresponding pixel values present in the input matrix are thus computed and calculated. 8 filter layers are used at convolutional layer 1 of same size so that more deeply the edges, loops and lines present on the handwritten input Punjabi character image are detected. Weight values and bias values for all filter layers are same so that number of computations will not get increased.

After convolutional layer, subsampling layer is used to decrease the spatial dimensionality of the handwritten input character image. Kernel size window of 2×2 is used with stride value of one. MAX pooling is used which extracts the maximum pixel value amongst the 2×2 kernel size window each time when kernel size window is shifted by stride value. It means we are picking up the most relevant feature form the whole image which helps in detection of input handwritten character image by detecting loops, edges and lines more efficiently. After applying subsampling layer 1, we obtained 8 feature maps each having input images of size 32×32.

5.5.2 Convolutional Layer 2 And Subsampling Layer 2

As shown in Figure 5.3, convolutional layer 2 takes as an input which are the output of the subsampling layer 1. Kernel window of size 2×2 is used to convolve over the activation window map obtained from the first subsampling layer instead of convolving over the original input handwritten Punjabi character image thereby results in reduction in number of computations needed because of the presence of subsampling layer. 16 feature maps are used in convolutional layer 2 to compute more deeper features from the handwritten Punjabi character image.

After convolutional layer 2, subsampling layer 2 is applied on the output obtained from the convolutional layer 2. 2×2 kernel size window is used. MAX pooling extracts maximum value amongst the 2×2 matrix thereby further results in reduction of the spatial dimensionality of the input handwritten Punjabi character image from 32×32 to 16×16.

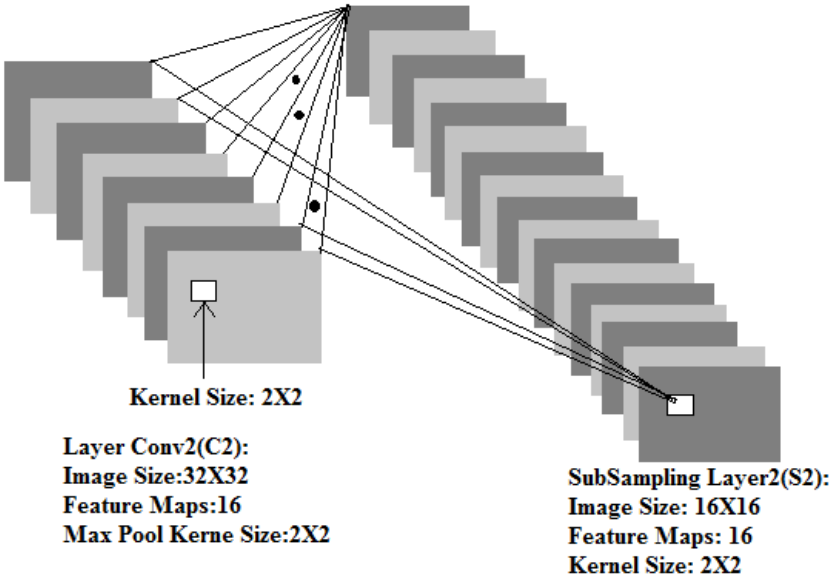


Figure 5.3: Computation From Second Convolutional C2 Layer To Second Subsampling S2 Layer

5.5.3 Convolutional Layer 3 And Subsampling Layer 3

As shown in Figure 5.4, the output obtained after applying the subsampling layer 2 is an activation window of size 32×32 and is provided as an input to the convolutional layer 3. 32 feature maps are used at this third convolutional layer. Each of 32 features are slided over the activation window obtained after subsampling layer2 to compute deeper features like edges, curves and edges of the input handwritten Punjabi character image. Kernel window of size 2×2 having stride value of one is used.

Subsampling layer 3 is applied on the output obtained from the convolutonal layer 3. MAX pooling is used to pick the most relevant feature which helps in detection of handwritten input image. 2×2 kernel window is used. Subsampling layer 3 results in reduction of spatial dimensionality from 16×16 to 8×8 that results in reduction in complexity of the developed neural network and also less number of computations is required now as compared to number of computations if set of convolutional and max pooling layers were not used as spatial dimension of the input images will not get reduced otherwise. More and more deeper we applied convolutional layers, more relevant features we obtained which helps in recognition of handwritten Punjabi character to improve the resultant accuracy recognition results.

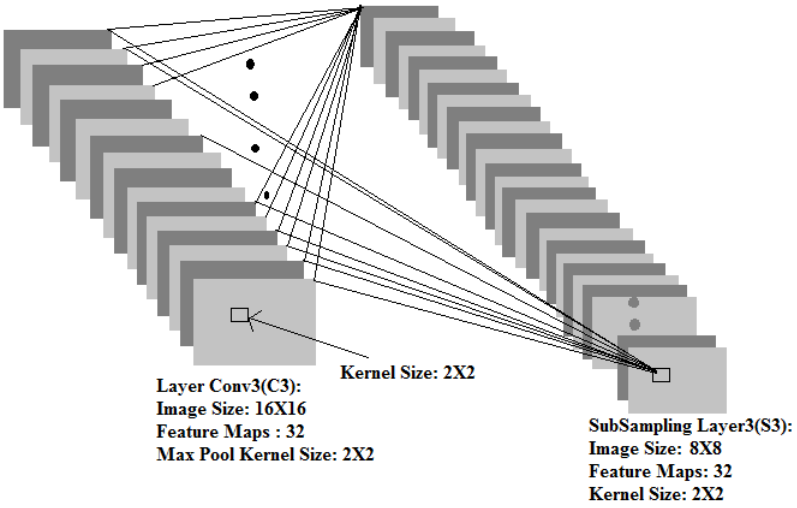


Figure 5.4: Computation From Third Convolutional C3 Layer To Third S3 Subsampling Layer

5.5.4 Convolutional Layer 4 And Subsampling Layer 4

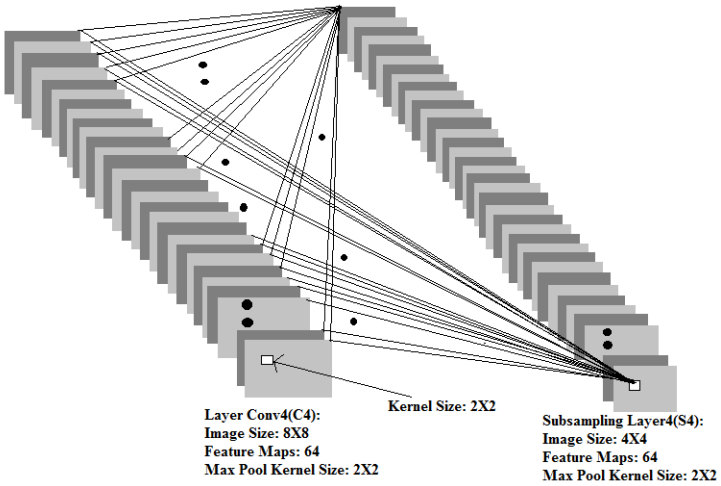


Figure 5.5: Computation From Fourth C4 Convolutional Layer To Fourth S4 Subsampling Layer

As shown in Figure 5.5, convolutional layer 4 takes as an input 8×8 sized images which are the output of the third subsampling layer 3. Kernel size window of size 2×2 is used which computes dot product between the kernel size window with their respective pixel values of the activation map obtained from third subsampling layer. 64 feature maps are used at convolutional layer 4 to compute the relevant features.

At subsampling layer 4, output of convolutional layer 4 is provided as an input to the subsampling layer 4. MAX pooling having kernel window of size 2×2 is used to extract

most relevant features. Subsampling layer 4 results in reduction of spatial dimensionality of the images from 8×8 to 4×4.

5.5.5 Convolutional Layer 5 And Subsampling Layer 5

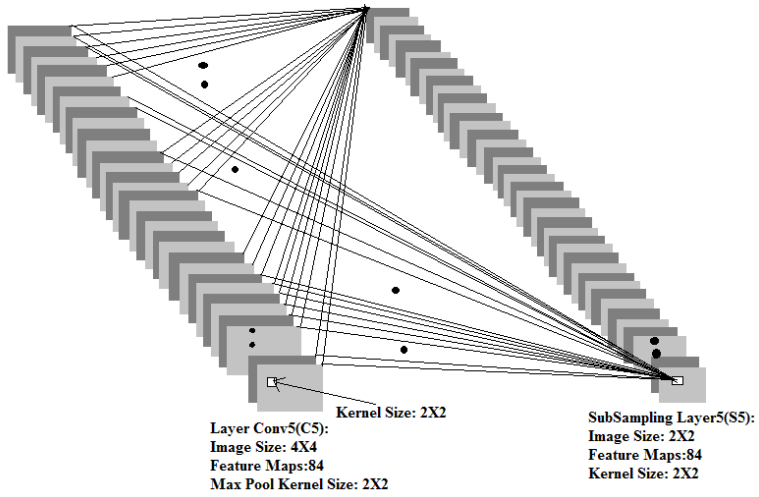


Figure 5.6: Computation From Fifth Convolutional C5 Layer To Fifth Subsampling S5 Layer

As shown in Figure 5.6, convolutional layer 5 takes as an input 4×4 sized images which are the output of the fourth subsampling layer S4. Kernel size window of size 2×2 is used which computes dot product between the kernel size window with their respective pixel values of the activation map obtained from fourth subsampling layer. 84 feature maps are used at convolutional layer 5 to compute the most relevant features.

At subsampling layer 5, output of convolutional layer 5 is provided as an input to the subsampling layer 5. MAX pooling having kernel window of size 2×2 is used to extract most relevant features. Subsampling layer 5 results in reduction of spatial dimensionality of the images from 4×4 to 2×2.

5.5.6 Fully Connected Layer 1

After five convolutional layers, we are using fully connected layer which provides the same structure as that of simple neural networks i.e. fully connected. As shown in Figure 5.7, the output of the fifth convolutional layer is provided as an input to the first fully connected layer. Number of input features provided to the first fully connected layer $2 \times 2 \times 84$ which are the width, height and number of channels or filters used in the

outcome of last subsampling layer S5. Number of output features provided by the fully connected layer 1 in our proposed system are 1024. This fully connected layer is fully connected with its previous subsampling layer S5 layer.

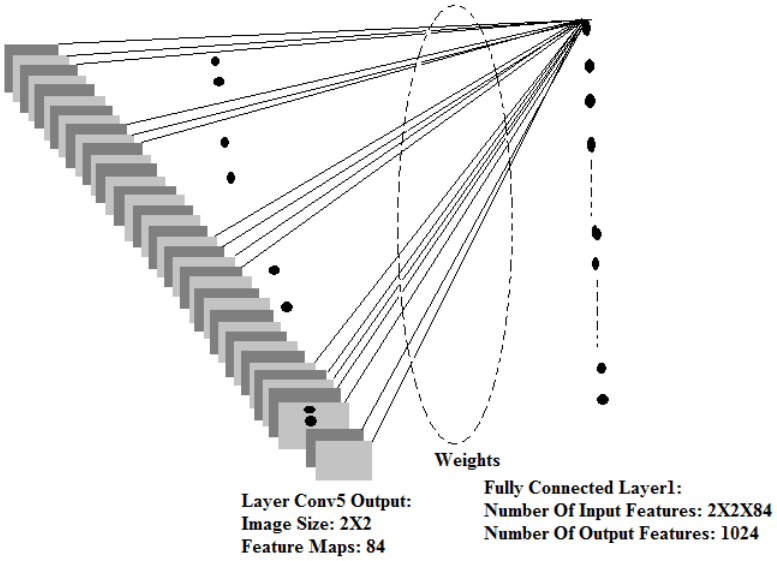


Figure 5.7: Computation From Fifth Convolutional C5 Layer To First Fully Connected FC1 Layer

5.5.7 Fully Connected Layer 2

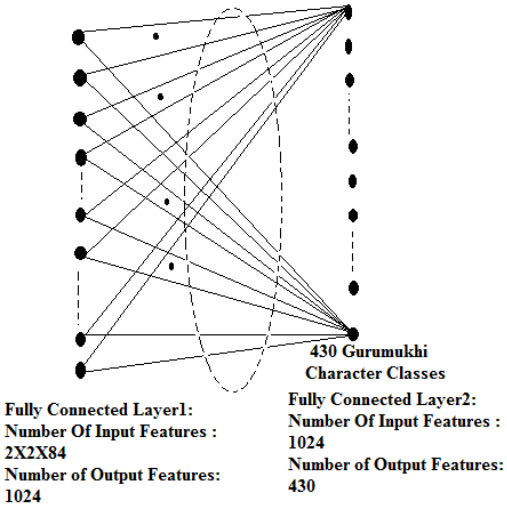


Figure 5.8: Computation From First Fully Connected FC1 Layer To Second Fully Connected FC2 Layer

As shown in Figure 5.8, number of output features obtained from the first fully connected

layer FC1 is given as input to the second fully connected layer FC2. Connection between the fully connected layer 1 FC1 and fully connected layer2 FC2 is fully connected. Number of input features at fully connected layer 2 is 1024 feature maps and output is equal to total number of classes available i.e. 430 classes. Here in these layers, probabilities are calculated by summation of input values and their corresponding weight values and later add bias values to provide non linearity in the developed model. Probability value present in each class for the input handwritten Punjabi character image indicated how likely is the input handwritten Punjabi character image to be classified in that respective class. At the end, starting from four dimensional convolutional neural network parameters we obtained one dimensional vector array representing the probability values of each class for input handwritten Punjabi character. Size of vector array will be equal to the total number of classes available. Here, in our proposed system, we have a total of 430 classes. Class containing highest probability value will be considered as the winning class in which the input handwritten character image is finally categorised.

5.6 Activation Functions

5.6.1 Why We Used Activation Functions?

To provide non linearity in the neural network developed, activation functions are used. The need of introducing non linearity in the network model as it allows the developed model to respond to a response variable that can be target variable, class label or score that varies non-linearly with respect to their explanatory variables. Non-linearity actually means that the output variable cannot be able to reproduced from a linear combination of the inputs. If we do not use activation functions in the developed neural network, irrespective of how many layers are used to develop this model, it will behave like a single-layer perceptron because after summation of these layers result will lead to creation of another linear function only. In other words, in simple linear regression, the main aim is to find optimal values of weights and biases that results in minimum error rate function i.e. to minimize the difference between the explanatory and target variables. Top graph represents linear regression line to show how it classify the classes if no activation function are used. Middle graph shows when the problem is non-linear. But we are implementing linear regression without using activation functions. From Figure 5.10 graph, it is clear that it misclassified a lot of samples in the wrong classes as without using activation functions, no matter how many layers you add in the neural network, it will behave like a single perceptron model.

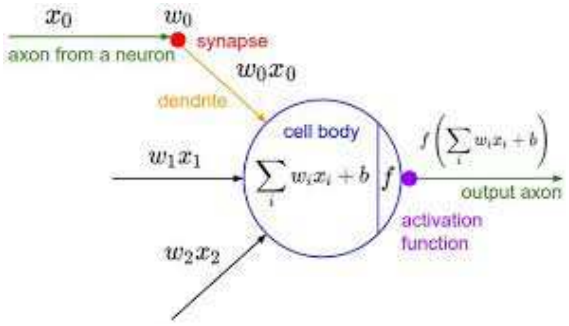


Figure 5.9: How Activation Function Works

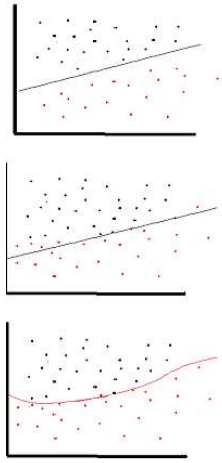


Figure 5.10: Graphs Representing How Activation Function Is Helpful

But as most of the real life problems are non linear. Therefore we use activation functions to add some non linearity in the developed network model to make incoming data non linear. An activation function acts as a decision making function that determines whether a particular neural feature at that point is present or not. The mapping of values is done between 0 and 1, where 0 indicates the feature is not present there and 1 indicates feature is present. Small changes that occurs in the weights and bias values by taking their derivatives with respect to error rate function values cannot be reflected in the activation values as it takes the value either 0 or 1. Therefore non linear functions are added to make them continuous and differentiable between this range. The main purpose of adding non linearity via activation functions in developed neural network is to make non linear decision boundaries using non linear combinations of the input weights and bias values. Last figure shows network model has non linearity regression line to classify the samples in their respective classes via activation functions i.e. more able to handle non linear combinations of input variables to obtain non linearity function value to give better results.

5.6.2 Types Of Activation Functions

The various types of activation functions that can be used in the neural network model are explained below:

1. **Identity Functions:** The function used by identity functions is mathematically represented as

$$f(x) = x \ ; \ for \ all \ x. \tag{5.1}$$

   Single layered neural network use step function while converting continuously varying input variable functions to a binary output either 0 or 1.

2. **Binary Step Function:** Threshold value is used by the binary step function. Binary step function that is used can be mathematically represented by

$$F(x) = 1 \ ; \ if \ x >= T \tag{5.2}$$

$$F(x) = 0 \ ; \ if \ x <= T \tag{5.3}$$

   It means, the neuron is fired only when their summation(product of weights and input values along with bias values) result value is greater than the specified threshold value, otherwise it will not fire.

3. **Sigmoid Functions:** Sigmoid functions are S shaped functions which are used as an activation function. The most commonly sigmoid functions are logistic and hyperbolic tangent functions. Generally, in backpropagation neural networks sigmoid functions are used as they help in reduction of burden of complication that is involved in the training phase of the network model.

4. **Binary Sigmoid:** Logistic function is also used as a sigmoid function having value between 0 and 1 are used to develop a neural network model whose output values are either binary output 0 or 1 indicating whether the feature is present or not respectively or either varies between 0 and 1. It is also named as binary sigmoid.

5. **Bipolar Sigmoid:** Logistic sigmoid function can also be scaled to solve a particular problem by having any range of values. Most commonly range that is used is $-1 : 1$. The id used is called bipolar sigmoid.

5.7 BackPropagation

Backpropagation is considered as a very effective technique of training neural networks and used along with optimization methods like gradient descent. Backpropagation algorithm consists of two phase cycle which are propagation and weight update. Process followed by backpropagation algorithm is explained ahead.

It consists of two phases: forward pass and backward pass. In forward pass, an input vector is presented to the network which is propagated forward through the network layer by layer until it reaches the output layer. After the output of the network is computed and calculated, then the output of the developed network model is compared to the actual output to compute error loss function for each of the neurons in the output layer. These calculated error values are then propagated backwards, starting from the output to the first neuron input layer until each neuron have their associated error function value that represents their respective contribution in giving the output value. Backpropagation makes use of the calculated error function values to calculate gradient of the loss function with respect to the weights used in the network. While in the second phase named as backward phase, this computed gradient value id fed to an optimization method which will update the values of weights and biases in order to minimize the error rate function value and thereby to improve the performance rate of the network model.

So as when later on, a new input pattern that may contain noise or incomplete, neurons that are present in the hidden layer of the developed network model respond by giving an active output and thereby recognized efficiently.

5.8 Dropout

One of the regularisation technique that can be used in neural network is dropout. Dropout is referred to as technique where some of the randomly selected neurons gets dropped out. It means that the contribution made in the activation of these downstream neurons get temporally removed during the forward pass and also weights and bias values of these neurons are not updated during the backward pass.

During training, if some neurons are randomly dropped off, then other neurons will have to take care of making predictions even for missing neurons that results in multiple independent internal representations being learned by the network model. Use of dropout technique makes the network model less sensitive to the specific weights of neurons which results in building a network that is capable of better recognition results and is less likely

to overfitting of training data. You can judge by seeing the results that is your model is overfitted or not. If network model gives good results on cross validation on training samples but not on independent test set then there is need to add dropout layer to reduce the rate of dependency of developed model on training data set.

5.9 Experimental Results

In this proposed system, we have to recognize the handwritten input Punjabi character into one of the 430 classes of Punjabi script. CNN[8] computes dynamic features of the handwritten images using the dynamically generated random matrices. More and more deeper convolutional layers used, more accurate our accuracy results will be. Because as we go on deeper in convolutional layers, it works on activation maps produced by previous convolutional layers to evaluate the features i.e. find deeper features such as lines, curves and edges. The forward pass in which training is done and the backward pass in which the weights and bias values are updated is called one epoch.

Table 5.1: Training, Testing And Validation accuracy having samples data in 45:45:10 at different number of epochs

Number of epochs	Training Accuracy	Testing Accuracy	Validation Accuracy
3000	49.09%	36.07%	30.24%
4500	57.29%	49.40%	41.60%
6000	64.02%	54.22%	50.62%
7500	73.50%	62.02%	59.01%
9000	82.67%	73.32%	68.67%
10500	90.89%	86.22%	79.07%
12000	97.02%	94.26%	90.67%

We ran the CNN implementation over the training, testing and validation samples for number of epochs and the results obtained at different instances of time is represented in Table 5.1, Table 5.2, Table 5.3 for different ratio of 45:45:10,55:35:10, 65:25:10 training:testing:validation samples data respectively. Highest training, testing and validation accuracy recognition results so obtained are 99.04%, 98.07% and 93.05% respectively. Resultant graphs of comparisons between the training, testing and validation accuracies are shown in Figure 5.11, Figure 5.12 and Figure 5.13 for 45:45:10, 55:35:10, 65:25:10 samples data respectively.

Dropout[38] and Backpropagation[39] techniques has been used for improving the accuracy of the trained model. Dropout value of 0.5 is used during the training process and

Table 5.2: Training, Testing And Validation accuracy having samples data in 55:35:10 at different number of epochs

Number of epochs	Training Accuracy	Testing Accuracy	Validation Accuracy
3000	42.04%	30.01%	25.04%
4500	51.81%	44.12%	34.42%
6000	64.44%	52.07%	49.21%
7500	71.11%	67.24%	55.25%
9000	80.01%	77.77%	67.68%
10500	89.07%	88.02%	81.09%
12000	96.02%	93.24%	90.04%

Table 5.3: Training, Testing And Validation accuracy having samples data in 65:25:10 at different number of epochs

Number of epochs	Training Accuracy	Testing Accuracy	Validation Accuracy
3000	45.04%	32.02%	28.07%
4500	54.92%	47.20%	42.27%
6000	65.01%	54.40%	47.06%
7500	72.34%	61.32%	56.07%
9000	84.40%	70.29%	67.50%
10500	93.56%	85.08%	81.22%
12000	99.04%	98.07%	93.05%

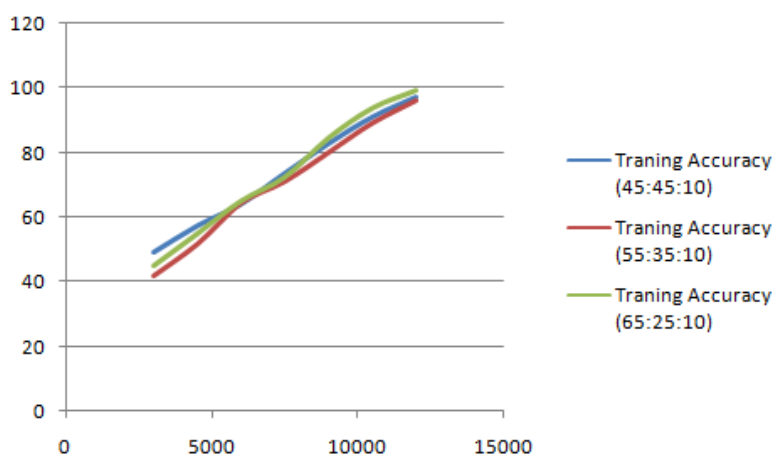


Figure 5.11: Graphs representing training accuracy results obtained for different samples ratio of 65:25:10, 55:35:10, 45:45:10 Training:Testing:Validation samples data at different instant of time

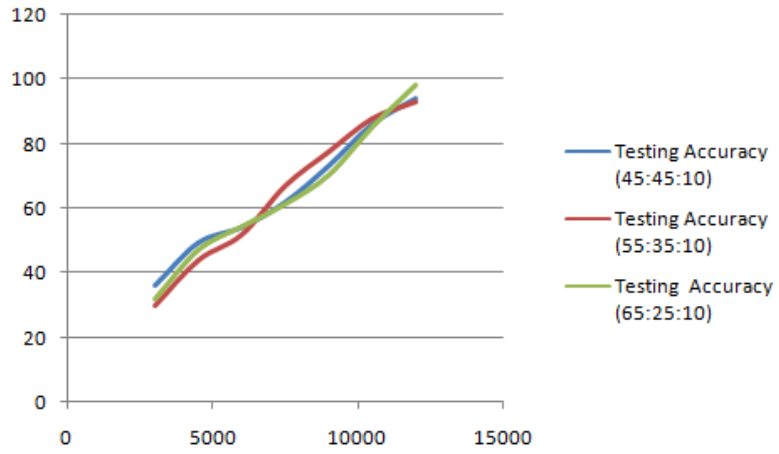


Figure 5.12: Graphs representing testing accuracy results obtained for different samples ratio of 65:25:10, 55:35:10, 45:45:10 Training:Testing:Validation samples data at different instant of time

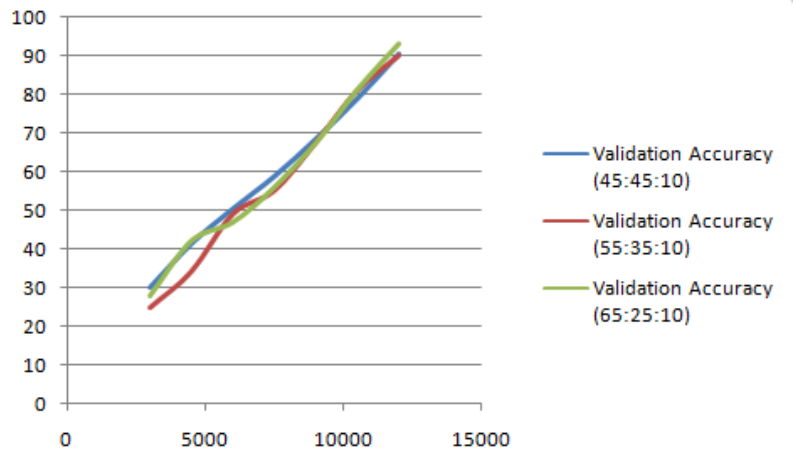
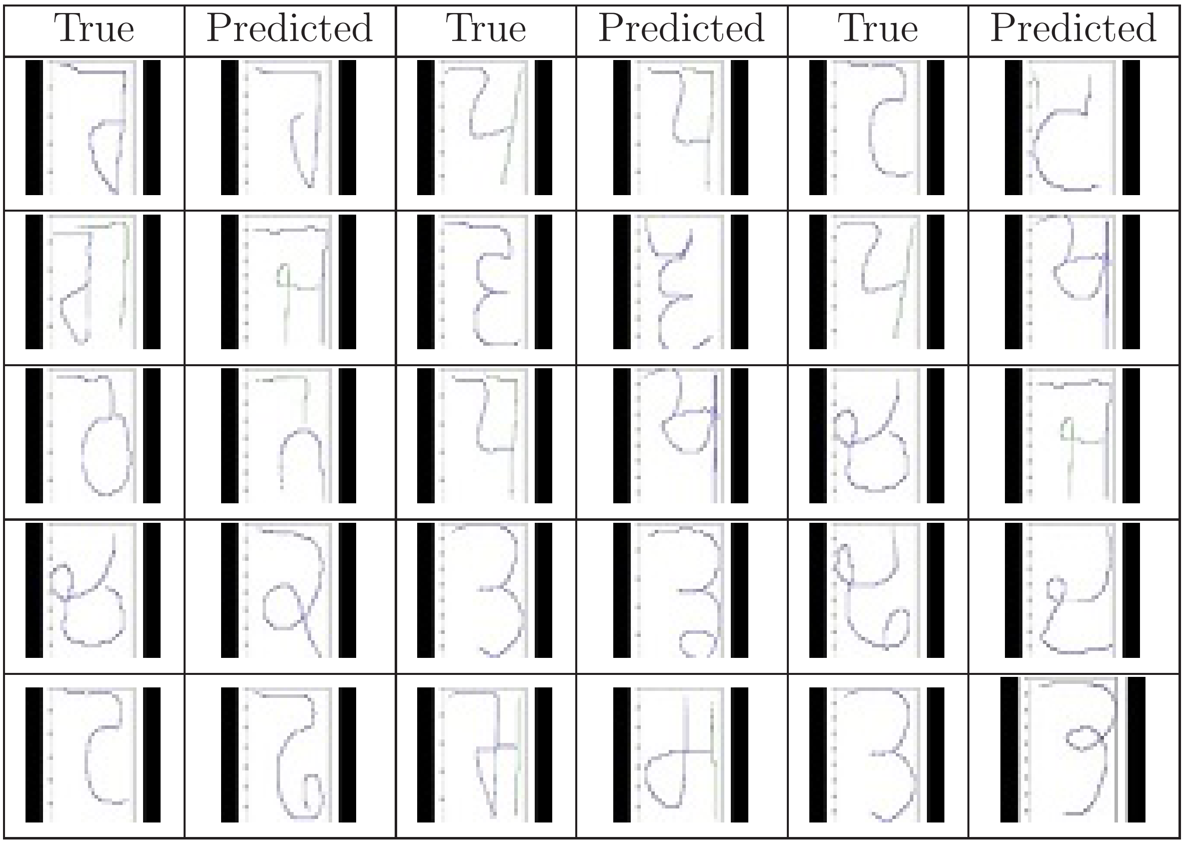


Figure 5.13: Graphs representing validation accuracy results obtained for different samples ratio of 65:25:10, 55:35:10, 45:45:10 Training:Testing:Validation samples data at different instant of time

Table 5.4: Example of some miss-classified classes



1.0 dropout value is used during the testing process. Gradient descent optimizer is used to train our model over the samples of handwritten Punjabi script.

Some of the miss-classied samples over the Punjabi script are shown in Table 5.4. Here our trained model classies the handwritten input Punjabi character into some dierent class rather than the class in which it actually lie.

As shown in Table 5.4, our developed neural network has miss classified the handwritten input Punjabi character. There are many similar words exist in Punjabi script that the developed neural network is not not able to classify correctly. For example, input handwritten Punjabi character “Ra'raa”is classified as “Haa'haa”; “Pap'paa”is classified as “Dhad'ddaa”and so on. Thus, the varying writing styles of different users and also the variation occurs in writing style of user's own style has made the recognition of handwritten Punjabi characters a tedious task and thereby made it difficult to obtain higher recognition results. Using deep learning techniques such as convolutional neural networks, we are able to compute deeper feature from the handwritten input images, as we go more and more deeper in the convolutional layers, it works on the activation window obtained form its previous layer thereby results in computing more deeper features (lines,

Table 5.5: Comparison Of Results With Earlier Approaches

References	Classification Techniques Used	Testing Character Accuracy
Plamondon and Srihari[10]	4-Layer CNN With 3×3 Kernel Size	94.14%
Sharma et al.[14]	Elastic Matching	90.08%
Sharma et al.[40]	HMMs	91.90%
Verma and Sharma[41]	SVMs, HMMs	96.70%
Verma and Sharma[42]	Zone Based Identification With SVM Kernels	93.70%
Current Work	5-Layer CNN With 5×5 Kernel Size Having 2×2 Maxpooling	98.07%

curves, loops). In Table 5.5, the accuracies obtained from present approaches and current work are compared.

Chapter 6

Conclusion And Future Work

6.1 Conclusion

The main aim of this proposed system is to implement feature extraction techniques to classify the handwritten input Punjabi words using open source library provided by Google named as Tensorflow library. The main advantage of using CNN for the implementation of this system is that CNN uses dynamic generated random matrices to compute the dynamic features of the handwritten input Punjabi character images. CNN is used over simple networks as it reduce the complexity of the overall developed neural network by reducing the number of mathematical computations required between input image pixel values and weight and bias values via max pooling or average pooling.

Also Tensorflow provides many in-built methods to perform mathematical computations to improve the recognition accuracy results and thereby helps in reduction of training and testing time. There is no need to code all the functionalities deeply. Tensorflow provides you readymade methods having appropriate parameters on which you can train and test the neural network model by using different optimizers and thereby compared their respective computed results and accuracy recognition rates. This is the first time implementation of handwritten recognition of Punjabi characters via open source Tensorflow library provided by Google.

As Tensorflow is open source library provided by Google therefore in respect of overall cost of the developed neural network, it is very effective as you need not to pay anything for using this library. Tensorflow provides you both the environment single processing units and multiprocessing units i.e. according to your dataset size, you can implement convolutional neural network either on single processing unit means single CPU system or you can implement convolutional neural network on distributed system using CUDA graphical processing units (GPU).

6.2 Summary Of Contributions

Following are the summary of contributions which are formulated in this work:

- As Tensorflow is a Google's open source library so it will be cost effective to use.
- Tensorflow provides many in-built methods to solve complex mathematical computations needed and uses a computtaional graph for computations.
- Numpy and matplotlib library are used along with Tensorflow library for mathematical compuations and to plot graph results.
- Using Tensorflow, can implement the classification and recognition of handwritten characters on multi-CPU systems using NVIDIA GPU's parallel processing.

6.3 Future Work

- In future work, we can also include Punjabi numerals to make alphanumeric dataset for recognition and classication using Tensorflow library. Also we can fine-tune the parameters of the different layers used for the implementation of CNN to calculate the accuracy and compared with our results.
- Also Tensorow supports distributed environment via the use of NVIDIA CUDA GPUs. So in future we can train our model on distributed environment for parallel processing over Punjabi alphanumeric dataset and thus compare the results so obtained with the results obtained without using CUDA GPU accuracy results.

References

[1] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[2] Birijesh K Verma. Handwritten hindi character recognition using multilayer perceptron and radial basis function neural networks. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 2111–2115. IEEE, 1995.

[3] Horst Bunke, Markus Roth, and Ernst Günter Schukat-Talamazzini. Off-line cursive handwriting recognition using hidden markov models. *Pattern recognition*, 28(9):1399–1413, 1995.

[4] Zheru Chi, Jing Wu, and Hong Yan. Handwritten numeral recognition using self-organizing maps and fuzzy rules. *Pattern Recognition*, 28(1):59–66, 1995.

[5] Birijesh K Verma. Handwritten hindi character recognition using multilayer perceptron and radial basis function neural networks. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 2111–2115. IEEE, 1995.

[6] Sung-Bae Cho. Neural-network classifiers for recognizing totally unconstrained handwritten numerals. *IEEE Transactions on Neural Networks*, 8(1):43–53, 1997.

[7] Xiaolin Li and Dit-Yan Yeung. On-line handwritten alphanumeric character recognition using dominant points in strokes. *Pattern recognition*, 30(1):31–44, 1997.

[8] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[9] Gurpreet Singh Lehal and Chandan Singh. A gurmukhi script recognition system. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 557–560. IEEE, 2000.

[10] Réjean Plamondon and Sargur N Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):63–84, 2000.

[11] Alessandro L Koerich, Yann Leydier, Robert Sabourin, and Ching Y Suen. A hybrid large vocabulary handwritten word recognition system using neural networks with hidden markov models. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 99–104. IEEE, 2002.

[12] Ujjwal Bhattacharya, Tanmoy Kanti Das, Amitava Datta, Swapan Kumar Parui,

and Bidyut Baran Chaudhuri. A hybrid scheme for handprinted numeral recognition based on a self-organizing network and mlp classifiers. *International journal of pattern recognition and artificial intelligence*, 16(07):845–864, 2002.

[13] Anuj Sharma, Rajesh Kumar, and RK Sharma. Online handwritten gurmukhi character recognition using elastic matching. In *Image and Signal Processing, 2008. CISP'08. Congress on*, volume 2, pages 391–396. IEEE, 2008.

[14] Anuj Sharma, Rajesh Kumar, and RK Sharma. Online handwritten gurmukhi character recognition using elastic matching. In *Image and Signal Processing, 2008. CISP'08. Congress on*, volume 2, pages 391–396. IEEE, 2008.

[15] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[16] Cheng-Lin Liu and Ching Y Suen. A new benchmark on the recognition of handwritten bangla and farsi numeral characters. *Pattern Recognition*, 42(12):3287–3295, 2009.

[17] Ujjwal Bhattacharya and Bidyut Baran Chaudhuri. Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals. *IEEE transactions on pattern analysis and machine intelligence*, 31(3):444–457, 2009.

[18] Thierry Artieres, Sanparith Marukatat, and Patrick Gallinari. Online handwritten shape recognition using segmental hidden markov models. *IEEE transactions on pattern analysis and machine intelligence*, 29(2), 2007.

[19] Umapada Pal, Tetsushi Wakabayashi, and Fumitaka Kimura. Comparative study of devnagari handwritten character recognition using different feature and classifiers. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 1111–1115. IEEE, 2009.

[20] Ujjwal Bhattacharya and Bidyut Baran Chaudhuri. Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals. *IEEE transactions on pattern analysis and machine intelligence*, 31(3):444–457, 2009.

[21] Shailedra Kumar Shrivastava and Sanjay S Gharde. Support vector machine for handwritten devanagari numeral recognition. *International journal of computer applications*, 7(11):9–14, 2010.

[22] Dharamveer Sharma and Puneet Jhajj. Recognition of isolated handwritten characters in gurmukhi script. *International Journal of Computer Applications*, 4(8):9–17, 2010.

[23] Vijay Patil and Sanjay Shimpi. Handwritten english character recognition using neural network. *Elixir Comput Sci Eng*, 41:5587–5591, 2011.

[24] Salvador Espana-Boquera, Maria Jose Castro-Bleda, Jorge Gorbe-Moya, and Fran-

cisco Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *IEEE transactions on pattern analysis and machine intelligence*, 33(4):767–779, 2011.

[25] Mamta Maloo and KV Kale. Support vector machine based gujarati numeral recognition. *International Journal on Computer Science and Engineering*, 3(7):2595–2600, 2011.

[26] Chirag I Patel, Ripal Patel, and Palak Patel. Handwritten character recognition using neural network. *International Journal of Scientific & Engineering Research*, 2(5):1–6, 2011.

[27] Kartar Singh Siddharth, Renu Dhir, and Rajneesh Rani. Comparative recognition of handwritten gurmukhi numerals using different feature sets and classifiers. In *Proceedings of International Conference on Image Information Processing (ICIIP 2011)*, 2011.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[29] Ujjwal Bhattacharya, Malayappan Shridhar, Swapan K Parui, PK Sen, and BB Chaudhuri. Offline recognition of handwritten bangla characters: an efficient two-stage approach. *Pattern Analysis and Applications*, 15(4):445–458, 2012.

[30] Gita Sinha, Rajneesh Rani, and Renu Dhir. Handwritten gurmukhi numeral recognition using zone-based hybrid feature extraction techniques. *International Journal of Computer Applications*, 47(21), 2012.

[31] Kapil Mehrotra, Saumya Jetley, Akash Deshmukh, and Swapnil Belhe. Unconstrained handwritten devanagari character recognition using convolutional neural networks. In *Proceedings of the 4th International Workshop on Multilingual OCR*, page 15. ACM, 2013.

[32] Munish Kumar, MK Jindal, and RK Sharma. Segmentation of isolated and touching characters in offline handwritten gurmukhi script recognition. *International Journal of Information Technology and Computer Science (IJITCS)*, 6(2):58, 2014.

[33] Mayur Vyas and Karun Verma. A platform independent methodology for on-line handwritten character segmentation. In *Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on*, pages 1480–1483. IEEE, 2014.

[34] Li Chen, Song Wang, Wei Fan, Jun Sun, and Satoshi Naoi. Beyond human recognition: A cnn-based framework for handwritten character recognition. In *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*, pages 695–699. IEEE, 2015.

[35] Durjoy Sen Maitra, Ujjwal Bhattacharya, and Swapan K Parui. Cnn based common approach to handwritten character recognition of multiple scripts. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1021–1025. IEEE, 2015.

[36] Davinder Kaur and Mrs Rupinder Kaur Gurm. Machine printed gurumukhi numerals recognition using convolutional neural networks. 2016.

[37] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[38] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[39] Y LeCun, B Boser, JS Denker, D Henderson, RE Howard, W Hubbard, and LD Jackel. Handwritten digit recognition with a back-propagation network, 1989. In *Neural Information Processing Systems (NIPS)*.

[40] Anuj Sharma, Rajesh Kumar, and RK Sharma. Hmm based online handwritten gurmukhi character recognition. *ACM digital library machine graphics and vision international journal*, 19:439–449, 2010.

[41] Karun Verma and Rajendra Kumar Sharma. Comparison of hmm-and svm-based stroke classifiers for gurmukhi script. *Neural Computing and Applications*, pages 1–13, 2016.

[42] Karun Verma and RK Sharma. Recognition of online handwritten gurmukhi characters based on zone and stroke identification. *Sādhanā*, 42(5):701–712, 2017.

List of Publications

1. Sonia Mittal, Karun Verma, Ravinder Kumar, "Handwritten Punjabi Character Recognition Using Convolutional Neural Networks", Neural Computing And Applications, Springer. [IF=2.5]
   [Communicated]