“HAND WRITTEN CHARACTER RECOGNITION”

Mini Project Report

Submitted in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS & COMMUNICATION ENGINEERING

By

KEVIN BHENSHDADIYA(17BEC019)
NISHITH POSHIYA(17BEC059)

Department of Electronics &Communication Engineering
Institute of Technology
NIRMA UNIVERSITY
Ahmedabad 382 481

NOVEMBER 2019

CERTIFICATE

This is to certify that Mini Project-1 Report entitled “HAND WRITTEN CHARACTER RECOGNITION” submitted by Mr. Kevin Bhenshdadiya (17bec019) and Mr. Nishith Poshiya (17bec059) towards the partial fulfillment of the requirements for the award of degree in Bachelor of Technology in the field of Electronics & Communication Engineering of Nirma University is the record of work carried out by him/her under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this major project work to the best of our knowledge have not been submitted to any other University or Institution for award of any degree or diploma.

Date: 23/10/2019

Institute Guide

Dr. Bupendra Fataniya
Assistant Professor

Department of Electronics &
Communication Engineering
Institute of Technology
Nirma University
Ahmedabad

Head of Department
Department of Electronics &
Communication Engineering
Institute of Technology
Nirma University
Ahmedabad

Director

Institute of Technology
Nirma University
Ahmedabad

Undertaking for Originality of the Work

We, Kevin Bhenshdadiya and Poshiya Nishith, Roll No.17BEC019 & 17BEC059, give undertaking that the Mini Project entitled “HAND WRITTEN CHARACTER RECOGNITION” submitted by me, towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in the field of Electronics & Communication Engineering of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any other published work or any project report elsewhere; it will result in severe disciplinary action.

____

Signature of Student (1)

____

Signature of Student (2)

Date: _______________

Place: _______________

Endorsed by:

(Signature of Internal Guide)

Contents

Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.I am highly indebted to **Dr. Bupendra Fataniya** for their guidance and constant supervision as well as for providing necessary information regarding our topic.I would like to express my gratitude towards member of Nirma University for their kind co-operation and encouragement which help me in completion of this project. I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities."

Kevin Bhenshdadiya
Nishith Poshiya

ABSTRACT

Handwritten character recognition has been one of the active and challenging research areas in the field of image processing and pattern recognition. It has numerous applications which include, reading aid for blind, bank cheques and conversion of any hand written document into structural text form. In this paper an attempt is made to recognize handwritten characters for English, Hindi(Devanagari) alphabets without feature extraction using neural network. Each character data set contains 26 alphabets and 10 Digits.2000 different character data sets are used for training the neural network. The trained network is used for classification and recognition. In the proposed system, each character is resized into 24x24 pixels, which is directly subjected to training. That is, each resized character has 600 pixels and these pixels are taken as features for training the neural network. The results show that the proposed system yields good recognition rates which are comparable to that of feature extraction based schemes for handwritten character recognition.

Chapter 1

Introduction

1.1 Hand written character recognition

Handwriting recognition is an ability and technique of the system which receive the input from touch screen, electronic pen, scanner, images and paper documents. Offline handwriting recognition system is an art of identifying the word from images. As we know every person has their different writing style, so it is very difficult to recognize the correct handwritten characters and digits. Handwriting recognition system is developed to achieve the accuracy and reliable performance. So handwriting recognition is most challenging area if image and pattern recognition. Handwriting recognition is very useful in real world. There are many practical problems where handwriting recognition system is very useful like documentation analysis, mailing address interpretation, bank check processing, signature verification, postal addresses. Several approaches have been used in both online and offline handwriting recognition field like statistical methods, structural methods, neural network and syntactic methods. Some recognition system identify strokes, other apply recognition on single character or entire words. So handwriting recognition. In this project we worked on MNIST data set of 0-9 digits and Devanagari character of 46 classes using deferent CNN architecture and predicted the accuracy of the different models and tried to reduce the time taken.

Table 1. Architecture of OCR

Phase	Description	Approaches
Acquisition	The process of acquiring image	Digitization, binarization, compression
Pre-processing	To enhance quality of image	Noise removal, Skew removal, thinning, morphological operations
Segmentation	To separate image into its constituent characters	Implicit Vs Explicit Segmentation
Feature Extraction	To extract features from image	Geometrical feature such as loops, corner points Statistical features such as moments
Classification	To categorize a character into its particular class	Neural Network, Bayesian, Nearest Neighborhood

1.2 literature review

Optical character recognition system has been studied in last many decades. In 1914 Emanuel Goldberg developed a system that read handwritten character and digits and converted then into a telegraph code. At the same time Edmund Fournier d'Albe developed the Optophon, a handheld scanner that scan the printed page and produced the output. Goldberg continued to develop Handwriting recognition system for data entry. After some time he proposed matching the images with the templates containing the desire identification. This technique is known as template matching method.

After that Paul W.Handel also proposed a US patent on template matching handwriting technology in USA in 1933.In 1994 RCA engineers proposed first primitive computer type optical character recognition to help blind people. It designed to convert the handwritten report into punched cards for input in the computer for help in processing the shipment of 20-25 million books in a year. In 1965 Reader's Digest and RCA collaborated to build an optical character recognition system. In 1985 structural approaches were proposed with the statistical methods. In this systems broke the characters into set of patterns like horizontal and vertical lines and different curves. In this method system focused on shape of the characters. After 1990 the real progress is achieved using new techniques and methodologies in image processing and pattern recognition. In today's world more powerful computers and more accurate equipments like electronics pen, scanner and tablets are used. Many approaches like HMM, neural network, back propagation algorithm, fuzzy neural network are using to recognize handwritten documents.

1.3 Motivation

Many machine learning techniques able to learn and recognize hand written character to a greater extent. Handwritten character recognition (HCR) is a challenging task because of its variety of handwriting style by different type of writers in different type of language, this is the main challenge which is face by many researchers now a day in this field. Since all the information has shifted from handwritten documents to digital file format because of its reliability and durability. However a large amount of old documents are still in hand written forms so the attempt to convert them into digital form using manual typing to copy an existing file takes a lot of time and also require manpower to accurately manage each document and its copy as well to fulfill the task, this method of conversion is necessary for future handwritten documents also to make them digital documents. The main problem arises here is the hand writing styles since every different people has its own approach to handwriting in different language. Furthermore, many languages characters encompass a high range of characters that are morphologically complex. There is some other character also that made up of more than one character known as compound characters which make the task even more challenging. The above stated problem motivated us to build a system that will recognize characters in a way that reduces its manpower cost and also reduce the time to translate them.

Chapter: 2

Image preprocessing

Data in a paper document are usually captured by optical scanning and stored in a file of picture elements, called pixels. These pixels may have values: OFF (0) or ON (1) for binary images, 0–255 for gray-scale images, and 3 channels of 0–255 colour values for colour images. This collected raw data must be further analyzed to get useful information. Such processing includes the following:

3.1 Thresholding:

3.1.1 Simple Thresholding

A grayscale or colour image is reduced to a binary image. Here, the matter is straight forward. If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is **cv2.threshold**. First argument is the source image, which **should be a grayscale image**. Second argument is the threshold value which is used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than (sometimes less than) the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function

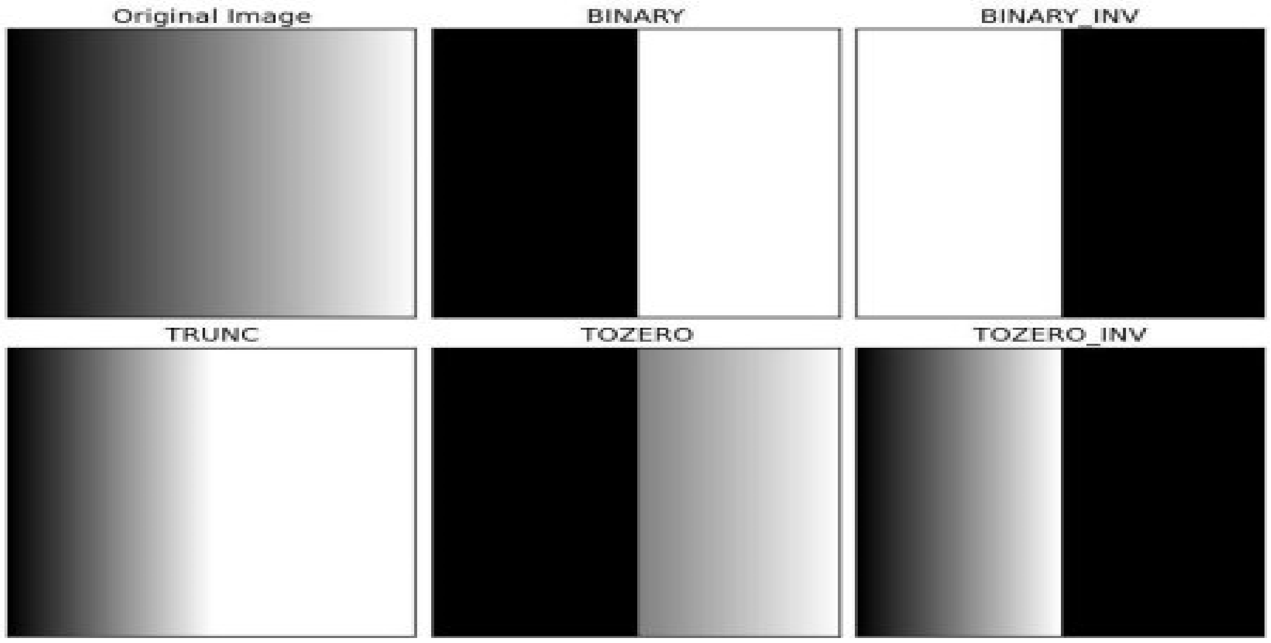


Fig. 1 Thresholding

3.1.2 Adaptive Thresholding:

In the previous section, we used a global value as threshold value. But it may not be good in all the conditions where image has different lighting conditions in different areas. In that case, we go for adaptive thresholding. In this, the algorithm calculate the threshold for a small regions of the image. So we get different thresholds for different regions of the same image and it gives us better results for images with varying illumination.

3.2 2D Convolution (Image Filtering)

As for one-dimensional signals, images also can be filtered with various low-pass filters (LPF), high-pass filters (HPF), etc. A LPF helps in removing noise, or blurring the image. A HPF filters helps in finding edges in an image.

OpenCV provides a function, **cv2.filter2D()**, to convolve a kernel with an image. As an example, we will try an averaging filter on an image. A 5x5 averaging filter kernel can be defined as follows:

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Filtering with the above kernel results in the following being performed: for each pixel, a 5x5 window is centered on this pixel, all pixels falling within this window are summed up, and the result is then divided by 25. This equates to computing the average of the pixel values inside that window. This operation is performed for all the pixels in the image to produce the output filtered image.

3.3 Skew Correction :

A Skewed image is defined as a document image which is not straight. Skewed images directly impact the line segmentation of OCR engine which reduces its accuracy. We need to process the following steps to correct text skew.It Detect the text block with skew in the image.then it measure the Now calculate the angle of rotation.then according to the angle it Rotating the image to correct the skew.

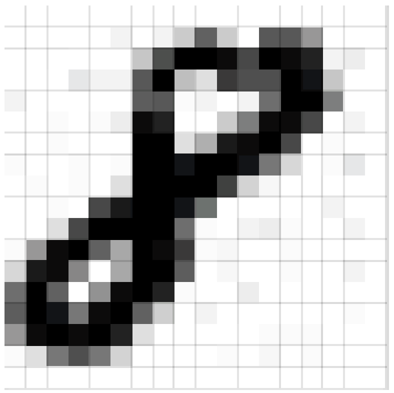
Chapter: 3

Convolutional neural network

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. CNNs use relatively little pre-processing compared to other image classification algorithms. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data.

3.1 Operations in the ConvNet

1) Convolution
   Channel is a conventional term used to refer to a certain component of an image. An image from a standard digital camera will have three channels – red, green and blue – you can imagine those as three 2d-matrices stacked over each other (one for each color), each having pixel values in the range 0 to 255.



ConvNets derive their name from the “convolution” operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. We will not go into the mathematical details of Convolution here, but will try to understand how it works over images.

Every image can be considered as a matrix of pixel values. Consider a 5 x 5 image whose pixel values are only 0 and 1 (note that for a grayscale image, pixel values range from 0 to 255, the green matrix below is a special case where pixel values are only 0 and 1):
the Convolution of the 5 x 5 image and the 3 x 3 matrix can be computed as shown in the animation in Figure 2 below:

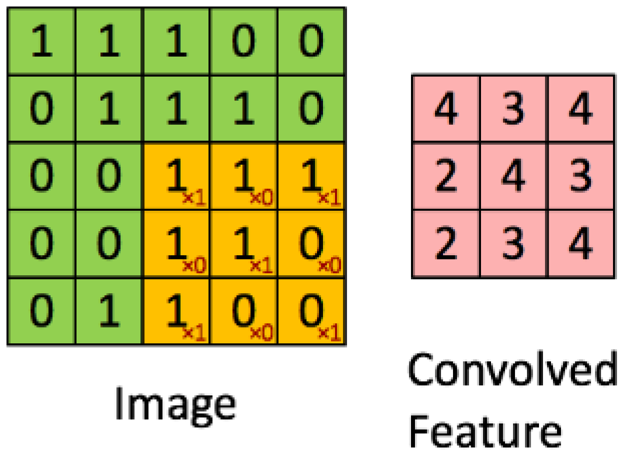


Fig. 2 convolution

the 3×3 matrix is called a 'filter' or 'kernel' or 'feature detector' and the matrix formed by sliding the filter over the image and computing the dot product is called the 'Convolved Feature' or 'Activation Map' or the 'Feature Map'. It is important to note that filters acts as feature detectors from the original input image.

In practice, a CNN learns the values of these filters on its own during the training process (although we still need to specify parameters such as number of filters, filter size, architecture of the network etc. before the training process). The more number of filters we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images.

- Depth: Depth corresponds to the number of filters we use for the convolution operation.
- Stride: Stride is the number of pixels by which we slide our filter matrix over the input input matrix.
- Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is also called wide convolution, and not using zero-padding would be a narrow convolution.

2) Non Linearity (ReLU)
An additional operation called ReLU has been used after every Convolution operation in Figure 3 above. ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by

3) Pooling or Sub Sampling:
Convolutional layers in a convolutional neural network systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input.
Convolutional layers prove very effective, and stacking convolutional layers in deep models allows layers close to the input to learn low-level features (e.g. lines) and layers deeper in the model to learn high-order or more abstract features, like shapes or specific objects.

A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a different feature map. This can happen with re-cropping, rotation, shifting, and other minor changes to the input image.
A common approach to addressing this problem from signal processing is called down sampling. This is where a lower resolution version of an input signal is created that still contains the large or important structural elements, without the fine detail that may not be as useful to the task.

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

Figure 10 shows an example of Max Pooling operation on a Rectified Feature map (obtained after convolution + ReLU operation) by using a 2×2 window.

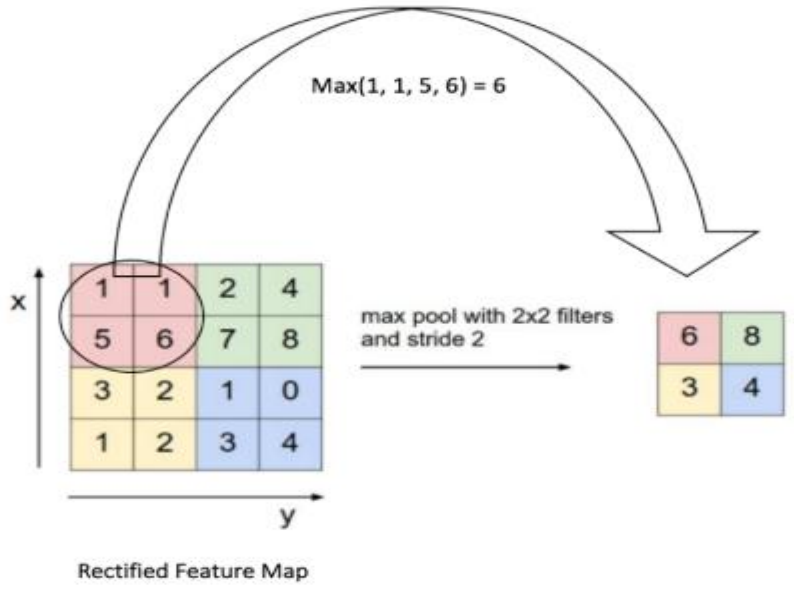


Fig.3 Max pooling

We slide our 2 x 2 window by 2 cells (also called 'stride') and take the maximum value in each region. As shown in Figure 10, this reduces the dimensionality of our feature map.

In the network shown in Figure 11, pooling operation is applied separately to each feature map (notice that, due to this, we get three output maps from three input maps).

4) Fully Connected Layer

The output feature maps of the final convolution or pooling layer is typically flattened, i.e., transformed into a one-dimensional (1D) array of numbers (or vector), and connected to one or more fully connected layers, also known as dense layers, in which every input is connected to every output by a learnable weight. Once the features extracted by the convolution layers and downsampled by the pooling layers are created, they are mapped by a subset of fully connected layers to the final outputs of the network, such as the probabilities for each class in classification tasks. The final fully connected layer typically has the same number of output nodes as the number of classes. Each fully connected layer is followed by a nonlinear function, such as ReLU, as described above.

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used, but will stick to softmax in this post). The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. I recommend reading this post if you are unfamiliar with Multi Layer Perceptrons.

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

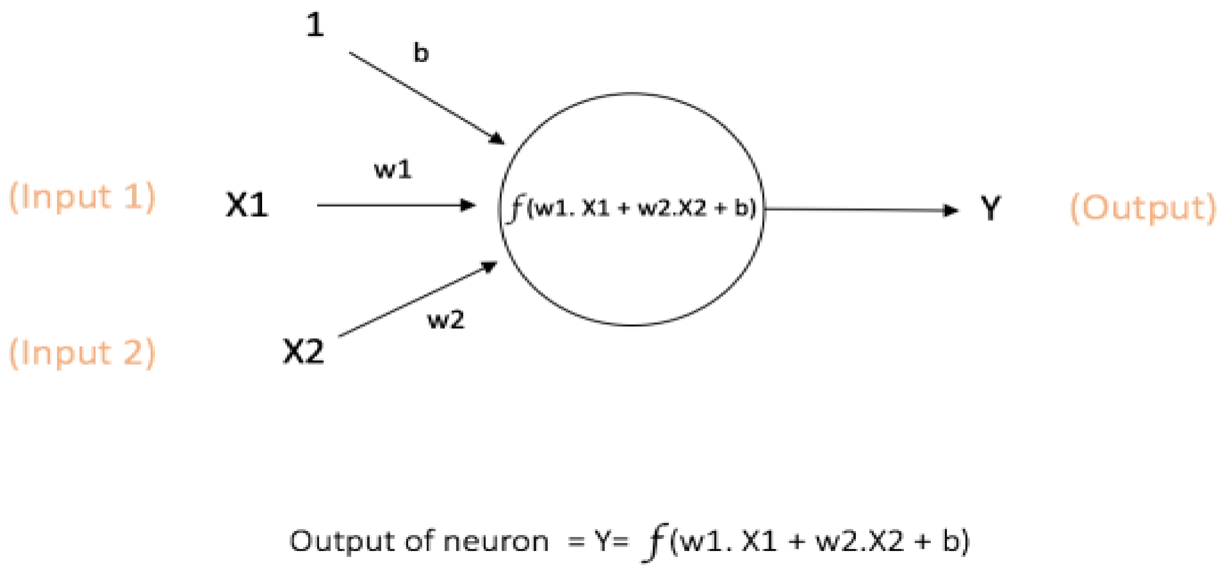


Fig.4 Single nueron

The above network takes numerical inputs **X1** and **X2** and has weights **w1** and **w2** associated with those inputs. Additionally, there is another input **1** with weight **b** (called the **Bias**) associated with it. We will learn more details about role of the bias later.

The output **Y** from the neuron is computed as shown in the Figure 1. The function ***f*** is non-linear and is called the **Activation Function**. The purpose of the activation function is to introduce non-linearity into the output of a neuron. This is important because most real world data is non linear and we want neurons to *learn* these non linear representations.

Every activation function (or *non-linearity*):

single number and performs a certain fixed mathematical operation on it . There are several activation functions you may encounter in practice:

- **Sigmoid:** takes a real-valued input and squashes it to range between 0 and 1

$$\sigma(x) = 1 / (1 + \exp(-x))$$

- **tanh:** takes a real-valued input and squashes it to the range [-1, 1]

$$\tanh(x) = 2\sigma(2x) - 1$$

- **ReLU**: ReLU stands for Rectified Linear Unit. It takes a real-valued input and thresholds it at zero (replaces negative values with zero0

  ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

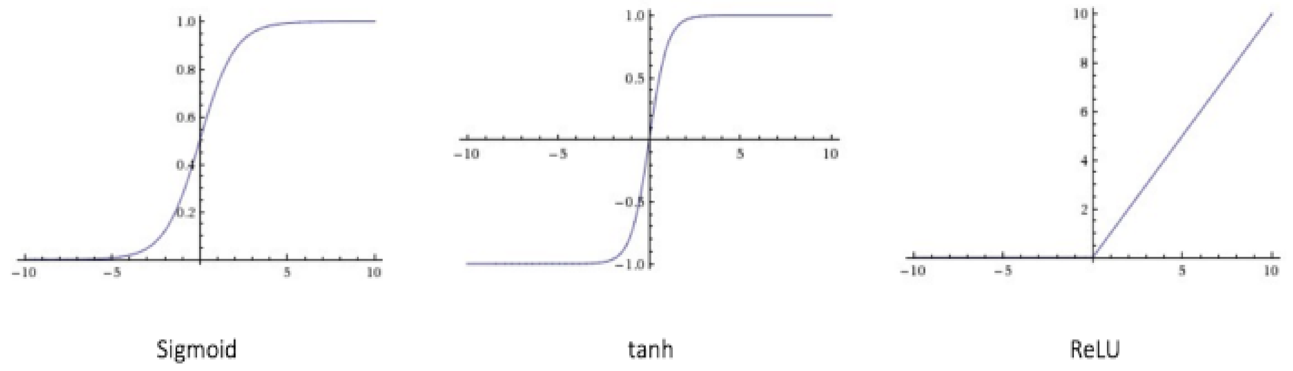


fig. 5 Activation function

Feedforward Neural Network

The feedforward neural network was the first and simplest type of artificial neural network devised . It contains multiple neurons (nodes) arranged in **layers**. Nodes from adjacent layers have **connections** or **edges** between them. All these connections have **weights** associated with them..

A feedforward neural network can consist of three types of nodes:

1. **Input Nodes –** The Input nodes provide information from the outside world to the network and are together referred to as the "Input Layer". No computation is performed in any of the Input nodes – they just pass on the information to the hidden nodes.
2. **Hidden Nodes –** The Hidden nodes have no direct connection with the outside world (hence the name "hidden"). They perform computations and transfer information from the input nodes to the output nodes. A collection of hidden nodes forms a "Hidden Layer". While a feedforward network will only have a single input layer and a single output layer, it can have zero or multiple Hidden Layers.
3. **Output Nodes –** The Output nodes are collectively referred to as the "Output Layer" and are responsible for computations and transferring information from the network to the outside world.

In a feedforward network, the information moves in only one direction – forward – from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network (this property of feed forward networks is different from Recurrent Neural Networks in which the connections between the nodes form a cycle).

Two examples of feedforward networks are given below:

1. Single Layer Perceptron – This is the simplest feedforward neural network [4] and does not contain any hidden layer.
2. Multi Layer Perceptron – A Multi Layer Perceptron has one or more hidden layers. .

Multi Layer Perceptron

A Multi Layer Perceptron (MLP) contains one or more hidden layers (apart from one input and one output layer). While a single layer perceptron can only learn linear functions, a multi layer perceptron can also learn non – linear functions.

Figure 4 shows a multi layer perceptron with a single hidden layer. Note that all connections have weights associated with them, but only three weights (w0, w1, w2) are shown in the figure.

Input Layer: The Input layer has three nodes. The Bias node has a value of 1. The other two nodes take X1 and X2 as external inputs (which are numerical values depending upon the input dataset). As discussed above, no computation is performed in the Input layer, so the outputs from nodes in the Input layer are 1, X1 and X2 respectively, which are fed into the Hidden Layer.

Hidden Layer: The Hidden layer also has three nodes with the Bias node having an output of 1. The output of the other two nodes in the Hidden layer depends on the outputs from the Input layer (1, X1, X2) as well as the weights associated with the connections (edges). Figure 4 shows the output calculation for one of the hidden nodes (highlighted). Similarly, the output from other hidden node can be calculated. Remember that ***f*** refers to the activation function. These outputs are then fed to the nodes in the Output layer.

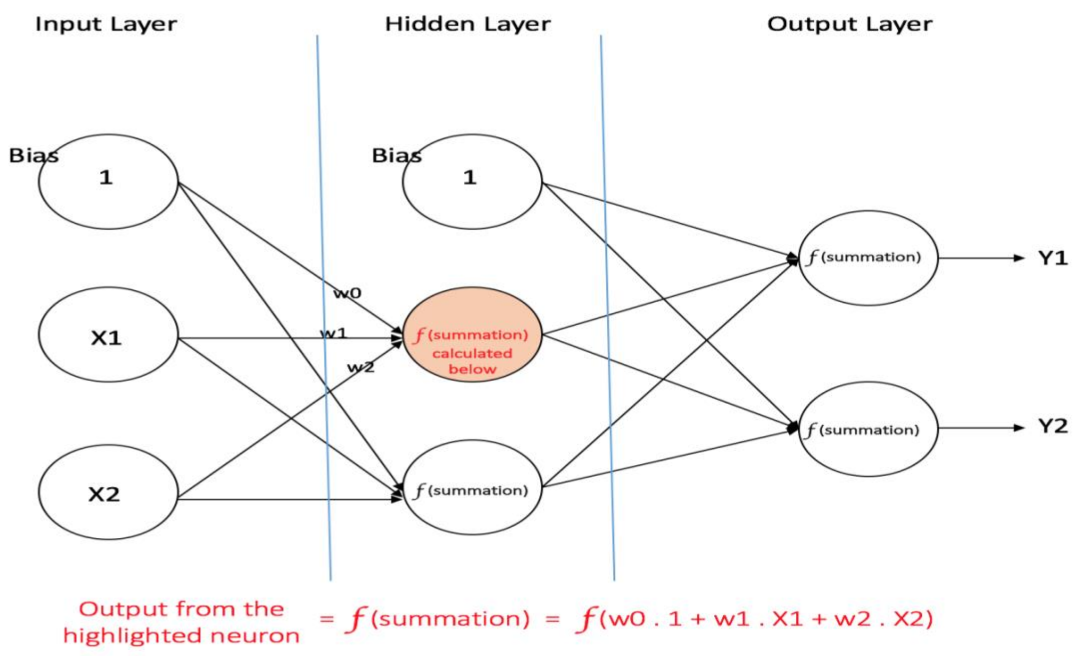

Input Layer
Hidden Layer
Output Layer
Bias
1
X1
X2
Bias
1
w0
w1
w2
f (summation) calculated below
f (summation)
f (summation)
f (summation)
Y1
Y2
Output from the highlighted neuron = f (summation) = f (w0 . 1 + w1 . X1 + w2 . X2)

Fig 5: a multi layer perceptron having one hidden layer

Output Layer: The Output layer has two nodes which take inputs from the Hidden layer and perform similar computations as shown for the highlighted hidden node. The values calculated (Y1 and Y2) as a result of these computations act as outputs of the Multi Layer Perceptron.

Training our MLP: The Back-Propagation Algorithm

The process by which a Multi Layer Perceptron learns is called the Backpropagation algorithm.

Backward Propagation of Errors, often abbreviated as BackProp is one of the several ways in which an artificial neural network (ANN) can be trained. It is a supervised training scheme, which means, it learns from labeled training data (there is a supervisor, to guide its learning).

To put in simple terms, BackProp is like "**learning from mistakes**". The supervisor ***corrects*** the ANN whenever it makes mistakes.

An ANN consists of nodes in different layers; input layer, intermediate hidden layer(s) and the output layer. The connections between nodes of adjacent layers have "weights" associated with them. The goal of learning is to assign correct weights for these edges. Given an input vector, these weights determine what the output vector is.

In supervised learning, the training set is labeled. This means, for some given inputs, we know the desired/expected output (label).

BackPropAlgorithm:

Initially all the edge weights are randomly assigned. For every input in the training dataset, the ANN is activated and its output is observed. This output is compared with the desired output that we already know, and the error is “propagated” back to the previous layer. This error is noted and the weights are “adjusted” accordingly. This process is repeated until the output error is below a predetermined threshold.

Once the above algorithm terminates, we have a “learned” ANN which, we consider is ready to work with “new” inputs. This ANN is said to have learned from several examples (labeled data) and from its mistakes (error propagation).

Now that we have an idea of how Backpropagation works, lets come back to our student-marks dataset shown above.

In classification tasks, we generally use a Softmax function as the Activation Function in the Output layer of the Multi Layer Perceptron to ensure that the outputs are probabilities and they add up to 1. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one. So, in this case,

$$\text{Probability (Pass)} + \text{Probability (Fail)} = 1$$

Step 1: Forward Propagation

All weights in the network are randomly assigned. Lets consider the hidden layer node marked **V** in Figure 5 below. Assume the weights of the connections from the inputs to that node are w1, w2 and w3 (as shown).

The network then takes the first training example as input (we know that for inputs 35 and 67, the probability of Pass is 1).

- Input to the network = [35, 67]
- Desired output from the network (target) = [1, 0]

Then output V from the node in consideration can be calculated as below (***f*** is an activation function such as sigmoid):

$$V = \boldsymbol{f}\,(1{*}w1 + 35{*}w2 + 67{*}w3)$$

Similarly, outputs from the other node in the hidden layer is also calculated. The outputs of the two nodes in the hidden layer act as inputs to the two nodes in the output layer. This enables us to calculate output probabilities from the two nodes in output layer.

Suppose the output probabilities from the two nodes in the output layer are 0.4 and 0.6 respectively (since the weights are randomly assigned, outputs will also be random). We can see that the calculated probabilities (0.4 and 0.6) are very far from the desired probabilities (1 and 0 respectively), hence the network in Figure is said to have an ‘Incorrect Output’.

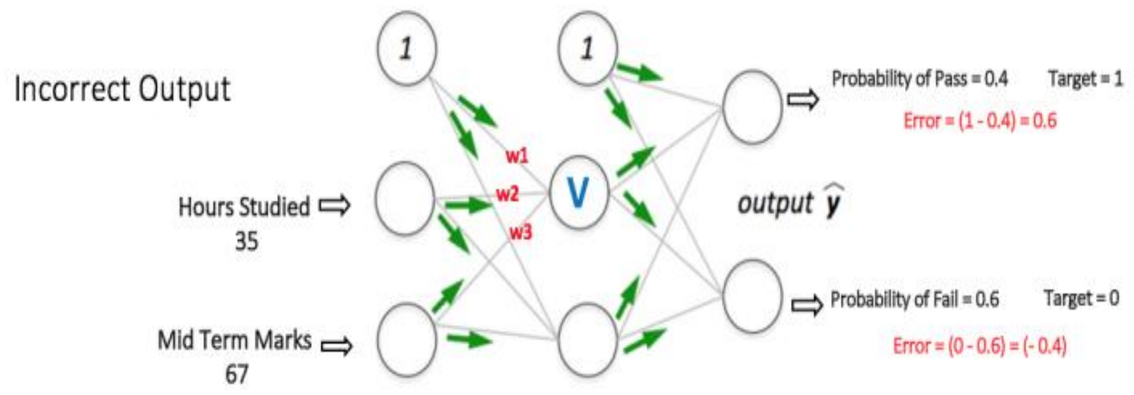


Figure 5: forward propagation step in a multi layer perceptron

Step 2: Back Propagation and Weight Updation

We calculate the total error at the output nodes and propagate these errors back through the network using Backpropagation to calculate the *gradients*. Then we use an optimization method such as *Gradient Descent* to ‘adjust’ **all** weights in the network with an aim of reducing the error at the output layer. This is shown in the Figure below (ignore the mathematical equations in the figure for now).

Suppose that the new weights associated with the node in consideration are w4, w5 and w6 (after Backpropagation and adjusting weights).

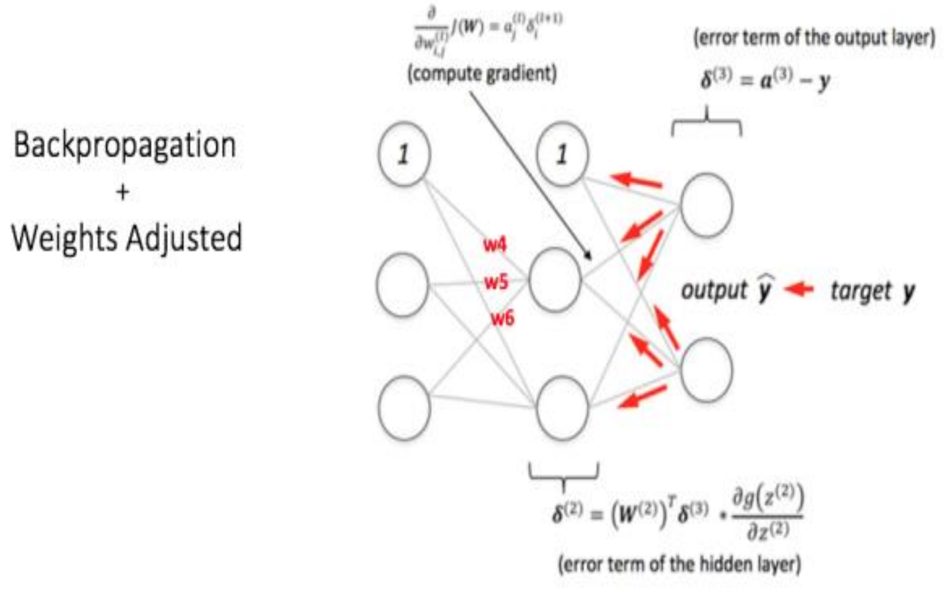

Backpropagation
+
Weights Adjusted
$\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$
(compute gradient)
(error term of the output layer)
$\delta^{(3)} = a^{(3)} - y$
w4
w5
w6
output $\hat{y}$
target y
$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} \cdot \frac{\partial g(z^{(2)})}{\partial z^{(2)}}$
(error term of the hidden layer)

Figure 6: backward propagation and weight updation step in a multi layer perceptron

If we now input the same example to the network again, the network should perform better than before since the weights have now been adjusted to minimize the error in prediction. As shown in Figure 7, the errors at the output nodes now reduce to [0.2, -0.2] as compared to [0.6, -0.4] earlier. This means that our network has learnt to correctly classify our first training example

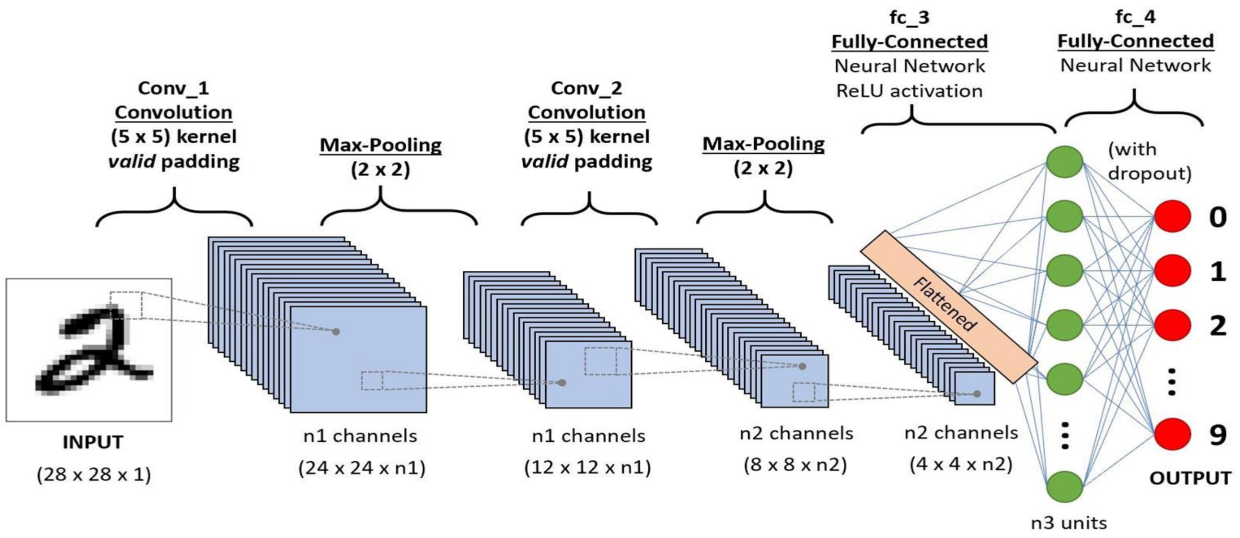


Fig. 7 Architecture of CN

Chapter 4
OBSERVATIONS

4.1 Digit recognition

4.1.1 MNIST Handwritten Digit Classification Dataset

The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset**.**

It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9. It has 6000 samples of each class.It is a widely used and deeply understood dataset and, for the most part, is "solved." Top-performing models are deep learning convolutional neural networks that achieve a classification accuracy of above 99%, with an error rate between 0.4 %and 0.2% on the hold out test dataset.

4.1.2 Model

Table 2. Model of Digit recognition

Layer (type)	**Output Shape**	**Parameters**
Conv2d_1 (Conv2D)	(None, 28, 28, 32)	320
Max_pooling2d_1 (maxpooling2) Conv2d_2 (Conv2D)	(None, 14, 14, 32)	0
Conv2d_2 (Conv2D)	(None, 14, 14, 64)	18496
Max_pooling2d_2 (maxpooling2	(None, 7, 7, 64)	0
Conv2d_3 (Conv2D)	(None, 7, 7, 64)	36928
Flatten_4 (Flatten)	(None, 3136)	0
dense_1 (dense)	(None, 64)	200768
Activation_1	(None, 64)	0
Dense_2	(None, 10)	650

Table 3. Comparison of different CNN models

Evaluation of models	**Accuracy**
CNN	**97.21%**
VGG16	**99.61%**
Resnet164	**99.72%**
Wideresnet28_10	**99.72%**
mobileNet	**99.63%**

4.2. Devanagari character recognition

4.2.1 Devanagari data set

It is a dataset of 92,000 small square 32×32 pixel grayscale images of handwritten single digits between 0 and 9 and ka to gna in hindi chracter. It has 2000 samples of each class.It is a widely used and deeply understood dataset and, for the most part, is “solved.”

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

Table 4. Model of Devanagari character recognition

Layer (type)	**Output Shape**	**Parameters**
Conv2d_1 (Conv2D)	(None, 30, 30, 32)	320
Conv2d_2 (Conv2D)	(None, 28, 28, 64)	18496
Max_pooling2d_1 (maxpooling2)	(None, 14, 14, 64)	0
Conv2d_3 (Conv2D)	(None, 12, 12, 64)	36928
Max_pooling2d_2 (maxpooling2	(None, 6, 6, 64)	0
Conv2d_4 (Conv2D)	(None, 4, 4, 64)	36928
Conv2d_5 (Conv2D)	(None, 2, 2, 64)	36928

Dropout_1 (Dropout)	(None, 1, 1, 64)	0
Flatten_4 (Flatten)	(None, 64)	0
Dense_1 (Dense)	(None, 128)	8320
Dense_2	(none, 64)	8256
Dense_3 (Dense)	(None, 46)	2990

Total Parameters: 149,166
Trainable Parameters: 149,166
Non-trainable parameters: 0

Table 5. comparison of Devanagari character recognition models

	Parameters	Random forest	Multilayer perceptron	k-nearest neibour
Predict full image by model Trained on full image	Accuracy	71.7%	77.3%	80.6%
	Time	12.11	77.3	548
Predict full image by model Trained on cropped image	Accuracy	55.6%	66.7%	55.6%
	Time	0.015	4.86	2.3
Predict cropped image by model Trained on full image	Accuracy	71.2%	77.2%	81.4%
	time	9.72	97.13	673

Results discussion about cropped images training and prediction

Key points about the above implementation are:

- The model was trained on cropped character images and then used to predict the full character images.
- On this setup the model was able to predict atleast some characters correctly if not all.
- Model was then trained on full character images and then used to predict the cropped images of characters.
- This caused the degraded model performance due to increased misclassification.
- It is an expected result since croppping of the images may alter their characteristics to transform them into similar looking other characters thereby causing the model to predict them incorrectly.
- For eg. if character 'ka' is cropped vertically into exact half then left half is exactly same as the character 'waw' this causes the misclassificatio degrading the model performance.

CONCLUSION:

Handwritten character could be improved by chosing the appropriate layers in CNN network. Also it was observed that because of convolusion layer the performance time decreased by more then expected.It was found that model prediction was good when trained on cropped character images and used to predict full characters since the model could look for the features in cropped images and locate them in the full images.However if the model was trained on full character images and then a cropped character image was given to it, the accuracy of the model was hampered with possible reason being that the cropped image may look similar to some other character.

Refrences

[1] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[2]https://scholar.google.co.in/scholar?q=ocr+of+handwritten+data+using+svm&hl=en&as_sdt=0&as_vis=1&oi=scholart

[3] https://datascience.stackexchange.com/questions/6107/what-are-deconvolutional-layers

[4] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

[5]https://medium.com/@himanshubeniwal/handwritten-digit-recognition-using-machine-learning-ad30562a9b64

[6]ML-Devanagari-Character-Recognition/blob/master/Devanagari_Character_Recoginition