IMPLEMENTATION OF HANDWRITTEN MULTI DIGITS RECOGNITION USING CONVOLUTION NEURAL NETWORK

SUBMITTED BY:

KUNDAN KUMAR ROY

EXAMINATION ROLL NO: M4SWE19005

REGISTRATION NUMBER: 140963 OF 2017-18

A THESIS SUBMITTED TO THE FACULTY OF ENGINEERING & TECHNOLOGY OF JADAVPUR UNIVERSITY IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

MASTER OF ENGINEERING

IN

SOFTWARE ENGINEERING

UNDER THE SUPERVISION OF

MS. MUNMUN BHATTACHARYA

ASSISTANT PROFESSOR

Department of Information Technology

JADAVPUR UNIVERSITY

2019

Department of Information Technology

Faculty of Engineering & Technology

Jadavpur University

<u>*Certificate of Submission*</u>

I hereby recommend that the thesis, entitled ***"Implementation of Handwritten Multi Digits Recognition Using Convolution Neural Network"*** *prepared by* ***Mr. Kundan Kumar Roy (****Registration No.140963 of 2017-18) under my supervision, be accepted in partial fulfillment of the requirement for the degree of* ***Master of Engineering*** *in* ***Software Engineering*** *from the Department of* ***Information Technology*** *under* ***Jadavpur University.***

Ms. Munmun Bhattacharya
Assistant Professor
Department of Information Technology
Jadavpur University

Countersigned By:

----------------------------	---------------------------
Head of the Department	**Dean**
Information Technology	**Faculty of Engineering**
Jadavpur University	**Jadavpur University**

JADAVPUR UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY FACULTY OF ENGINEERING & TECHNOLOGY

CERTIFICATE OF APPROVAL

The thesis at instance is hereby approved as a creditable study of an Engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis for the purpose for which it is submitted.

-------------------------   ---------------------------

Signature of External Examiner   **Signature of Supervisor**

Ms. Munmun Bhattacharya
Assistant Professor
Department of Information Technology
Jadavpur University

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis contains literature survey and original research work by me, as a part of my Master of Engineering in Software Engineering studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name**: Kundan Kumar Roy**

Roll Number**: M4SWE19005**

Thesis Title**: Implementation of Handwritten Multi Digits Recognition Using Convolution Neural Network**

Signature with date

Acknowledgement

Firstly, I offer my sincere gratitude to the DEPARTMENT OF INFORMATION TECHNOLOGY, JADAVPUR UNIVERSITY, for giving me the opportunity to pursue this thesis during the degree course of Master of Engineering in Software Engineering and thereby helping me to improve my career.

I am grateful to my guide, Prof. Munmun Bhattachrya for his continuous support, co- ordination, guidance and invaluable help during the entire duration of the project. Without his disposition, spirit of accommodation, frankness, timely clarification and above all faith in me, this task could not have been completed in due time. His readiness to discuss all important matters at work deserves special attention.

I also want to express my heartfelt gratitude to my friends, classmates and seniors for their support and assistance in many ways in completion of the project.

And finally, I want to express my deep sense of gratitude with profound obeisance and veneration to my beloved parents, for their constant encouragement, support and blessings which enabled me to complete the project in time.

(Kundan Kumar Roy)
ME in Software Engineering
Jadavpur University

Dedicated to my beloved Parents

&

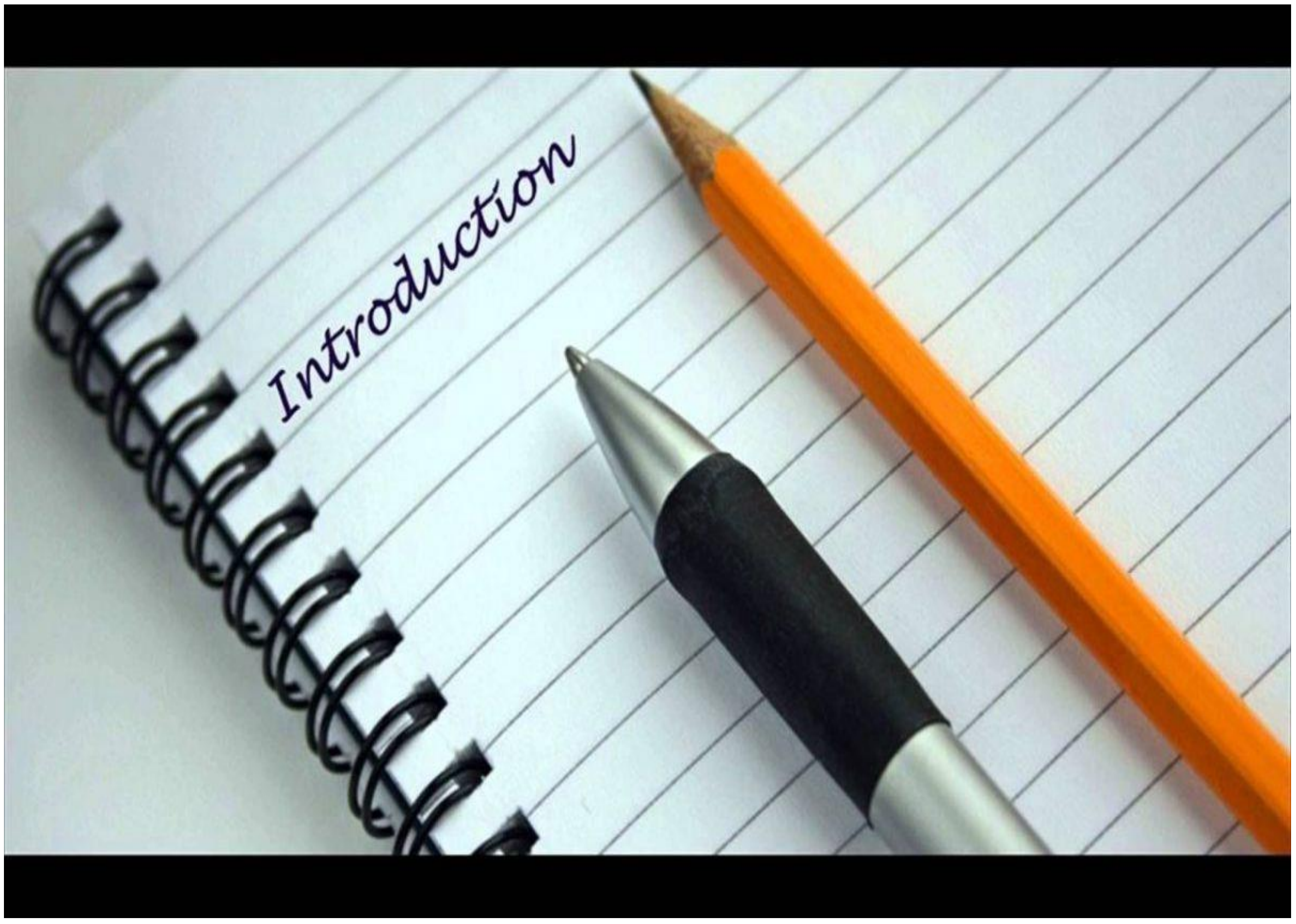
Almighty

Abstract

In this paper we have developed a convolution neural network that can recognize a handwritten digit. We take a multi handwritten digit and segment this image into individual digit and then this digit is fed into the trained model on MNIST [1] dataset. Handwritten digit recognition problem becomes one of the most famous problems in computer vision. There are many papers which describes recognition of a single handwritten digit. In this paper we take multi handwritten digit and recognize each individual digit. In this paper convolution neural network is used with two convolution layers and two pooling layer followed by a dropout of 20%. The CNN model is trained on black and white image each of size 28 X 28 where black is background and white is foreground. The highest accuracy obtained from this model is 99.31% which far better than the traditional neural network approach. For segmenting the input image into individual image we use threshold technique. In this technique images are first enhanced by morphological dilation operations. Then a contour is applied on to the image to detect object from the image.

Table of contents

CHAPTER 1: INTRODUCTION



Chapter Gist: This chapter briefly explains the motivation behind and the focus of this thesis paper. It also gives the organization of this paper.

Problem Definition

Focus

Chapter Organizations

Problem Definition:
Reading is actually a difficult task. Humans can still read more accurately and more speed with a great accuracy than machines. Our goal here is to design machines that can be able to read document with the same accuracy and lesser time than humans are doing. We have developed algorithms that is able to detect any strings of handwritten digits (written by humans) using Convolution neural network. In recent years there has been a great increase in the area of documents reading on paper. The number of mails sending in post offices and handwritten checks in banks are increasing enormously.

Since these numbers are increasing drastically some techniques required that can process these documents more accurately and in lesser time than humans are doing. This field of automatic digits reading is a subfield of "Optical Character Recognition (OCR) ". In a typical robust system an image of a document is captured by camera or scanner. The system then locates the desired image on to the documents.

The key problem here is to recognize a particular digit. Because a digit can be written by different ways by different people due to an unlimited number sizes and styles for each digit. The next challenging part in this problem is to segment each individual digit from an image of string of digits. Because the images of each individual digit can overlap each other and also from the strings of digits one digit can be of large size while other digits can be of different sizes. In this problem we have taken dataset from MNIST [1]. This dataset contains each digit as 784 dimensions (28 X 28). The image dataset MNIST is a black and white image dataset with black as background and white in the image describes the digit features. In this paper a subfield of machine learning called deep learning is introduced. Deep learning plays a very important role in Optical character recognition problems. In that type of problems we need to first collect the required dataset. After collecting dataset from the different source data preprocessing is done because data preprocessing is another big challenge in OCR problems. In this paper we have taken the most popular MNIST [1] dataset which is first introduced by Yann LeCun.

Focus:

The main focus of this thesis is to make a robust system that can accept an input image of multiple handwritten digit and first segment the images into different parts then get individual digits as reshaped into 28 X 28 dimensions. After getting individual digits we give it as input to our CNN model and we can predict the desired digit as output with 99% accuracy.

Chapter Organizations:

Chapter 2: Provides an overview of literacy and survey done for handwritten digit recognition.

Chapter 3: This chapter describes the image segmentation technique using threshold function for a multi object recognition image.

Chapter 4: This chapter describes deep learning techniques used for the purpose of 10 class classification problem.

Chapter 5: In this chapter we discuss about the state of the art solution for the problem called multi handwritten digit recognition problem. In this chapter we use convolution neural network technique to train our model and recognize the desired digit. The model gives an accuracy of 99.31% on Intel i5 processor window 10 4GB RAM.

CHAPTER 2: LITERATURE SURVEY

Literature Survey and Related Work:

Handwritten digit has become one of the most important problems in optical character recognition and it has been also used as a test case for the theories of pattern recognition and machine learning algorithms.

Generally, it can be classified into two types:

Online recognition:

The online recognition uses the geometric features of input given by the users. It also uses the temporal features of the input. The methods which use online recognition require low resources and low processing requirement. They are effectively good user adaptation.

Offline recognition:

In case of offline recognition it mainly operates on the users input handwritten digit which are already scanned or the digital images. There are lots of methods which are proposed to solve offline recognition.

In this paper we mainly focus on offline recognition. There are many researchers working on offline recognition. One of the most famous researchers is Yann LeCun who first introduces the concept of convolution neural network for multiclass classification problems. Before CNN simply neural network is used. Simple neural network gives less accuracy as compared to CNN. CNN actually extracts the desired features from the image and do lots of operations on the extracted features like pooling, striding, and padding.

In [1] the author introduces some issues in designing high reliability system for hand-written digit recognition using SVM classifiers. In this paper they used two different types of feature families' structural and statistical features. In this paper the presented result shows that it is very difficult to gain the recognition rate of a single classifier which is applied on the feature set which also includes both the feature families. In this paper the rule based co-operation schemes enable an easy implementation of rejection criteria. Except this, the cooperation of different classifiers designed for different feature families reduces the classifier complexity and also offers better possibilities to understand the role of features in the recognition step. There are some other classifiers except SVM which can produce good recognition results like KNN, K-nearest neighbors.[5]

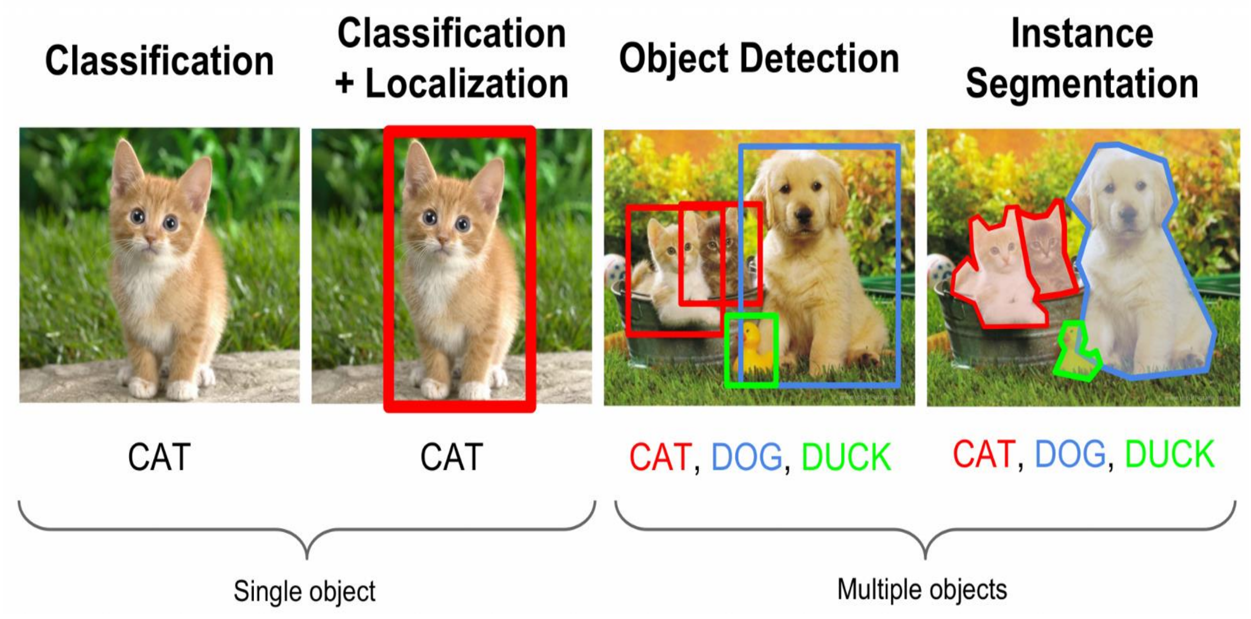
In [2] the paper is presented by Yaan LeCun and his team members at AT&T Bell Laboratories. They simply developed neural network architecture for the recognition of handwritten digits. This network has an error rate of 1% with rejection rate of 7% on handwritten digits provided by the U.S. Postal Service. In this paper they discuss an implementation of neural network architecture for a real world character recognition system. The main issue in this paper is cost related to problem while benefits are the accuracy and speed. Accuracy in this paper estimates the probability of correct recognition of a digit. Per-character probabilities give the necessary information for the per-field and multi-field probabilities.

The author also presented here two techniques for neural network classification and has also applied to a zip code digit recognition problem. The results presented here are very far from the systematic study. This work on classification describes us to view the propagation of a neural network from the first stage of the classification to the last stage of the classification. The output from the one layer is fed as the input to the next layer and so on. Thus it uses the knowledge about the neural network and about the problem we are trying to solve.[7]

In [3] the author focuses on three different neural network approaches. The most famous three approaches are deep neural network (DNN), deep belief network (DBN) and convolution neural network (CNN). In this paper the author compared and evaluated the three different NN approaches in terms of accuracy and performance. In this paper the author not only considers the recognition rate and performance but also he used execution time. All these experiments are conducted based upon random and standard dataset. The results of the algorithm show that DNN is the most accurate algorithm which has a 98.08% accuracy rate. The execution time of the DNN is comparable with the two other algorithms.

In this paper Wu et al. have applied deep learning to the real world problem of handwritten digit recognition and also obtained good performance for image recognition. By doing experiment they differentiate between DBN and DNN. Through its nonlinear function DNN can approximates the complex function. It can avoid the large workload of manually extracting the features and describe the potential information from the data. It can extract the desired features from the image and then it fed it to the neural network that way it can do a better job of extracting features from the image. Kaensar et al. have recognized that different classifier affects the recognition rate for the hand written digit recognition. They used open source tool kit Weka for the training and testing dataset which was obtained from the UCI repository. The presented result here shows that SVM is the best classifier among all of them. However, the main problem of SVM is it is mainly time consuming for the training of the dataset. Conversely, other techniques like neural networks give worse results but their training much quicker.[19]

CHAPTER 3: IMAGE SEGMENTATION



Chapter Gist: This chapter provides an overview of the various details and techniques involved in Image segmentation using thresholding.

Overview

Image segmentation is a fundamental process in many image, video, and computer vision applications. It is often used to partition an image into separate regions, which ideally correspond to different real-world objects. It is a critical step towards content analysis and image understanding. The gray level of pixels belonging to the object is entirely different from the image pixel in the background of the image. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image.

Importance of Image Segmentation

Segmentation subdivides an image into its basic regions or objects. The level of detail to which the subdivision is carried depends on the problem being solved. That is interest in an application have been detected [4]. The goal of image segmentation is to cluster pixels into salient image regions, i.e., regions corresponding to individual surfaces, objects, or natural parts of objects [5]. Image is formed in the eye and in the camera by the amount of illumination reflected by an object [6].In computer vision; image processing is any form of signal processing for which the input is an image, such as photographs or frames of videos. The output of image processing can be either an image or a set of characteristics or parameters related to image. The image processing techniques like image restoration, image enhancement, image segmentation etc. [7].Image segmentation is a important yet still challenging problem in computer vision and image processing. In specific, it is an essential process for many applications such as object recognition, target tracking, content-based image retrieval and medical image processing, etc. Generally speaking, the goal of image segmentation is to partition an image into a certain number of pieces which have coherent features (color, texture, etc.) and in the meanwhile to group the meaningful pieces together for the convenience of perceiving.

Image Thresholding

In many vision applications we want to separate some specific objects from the image from its background image. Image thresholding is an easy and convenient way to perform this type of segmentation. In addition to this image thresholding is also useful where we want to extract some pixels from the image where the pixels lie within a given specific range of values. thresholding is one of the widely used segmentation algorithm. It is very useful in finding objects from the foreground of the image. By using thresholding we can convert a grey scale image into binary image. This binary image contains all the essential information about the objects of the interest from the foreground of the image. By using binary image we can decrease complexities of the image data and simplifies the process of recognition and segmentation. The simplest way to convert a grey scale image into binary image is to select a threshold value. After that all the pixel values of the image is compared with this threshold value and value which less than threshold value is set to black(0) and value which is higher that this threshold value is set to white(1). Now the segmentation problem now boils down to the problem of selecting a threshold value which is correct. One way to select a correct threshold value is analyzing the histograms for the image we use.[4]

How Image Thresholding Actually Works:

Input given to a thresholding function is simply a grayscale or color image. In its simplest form it outputs a binary black and white image representing the segmentation where black pixels represent background image while white pixels represent foreground of the image. Actually segmentation is determined by a single parameter called threshold intensity. In first pass each pixel of the image is compared to the threshold intensity, if pixel’s intensity is higher than the threshold intensity, pixel is set to white and if pixel’s intensity is less than the threshold intensity, pixel is set to black. In addition to this image thresholding is also useful where we want to extract some pixels from the image where the pixels lie within a given specific range of values. Thresholding is one of the widely used segmentation algorithm. It is very useful in finding objects from the foreground of the image. By using thresholding we can convert a grey scale image into binary image.[6]

Example of Thresholding

An image is nothing but 2-D or 3-D array.

2-D array= [[10, 24]

[34,67]]

Threshold_intensity=50

If Threshold_intensity>50

Pixel=255

Else

Pixel=0

For above 2-D array output will be

[[0,0]

[0,255]]

Thresholding techniques are classified as follows

Thresholding algorithm is divided into parts Global and Local Thresholding. Global Thresholding is further classified into three parts namely Tradition (Otsu's), Iterative and Multistage. The image processing techniques like image restoration, image enhancement, image segmentation etc. [7].Image segmentation is a important yet still challenging problem in computer vision and image processing. In specific, it is an essential process for many applications such as object recognition, target tracking, content-based image retrieval and medical image processing, etc. Generally speaking, the goal of image segmentation is to partition an image into a certain number of pieces which have coherent features (color, texture, etc.) and in the meanwhile to group the meaningful pieces together for the convenience of perceiving.

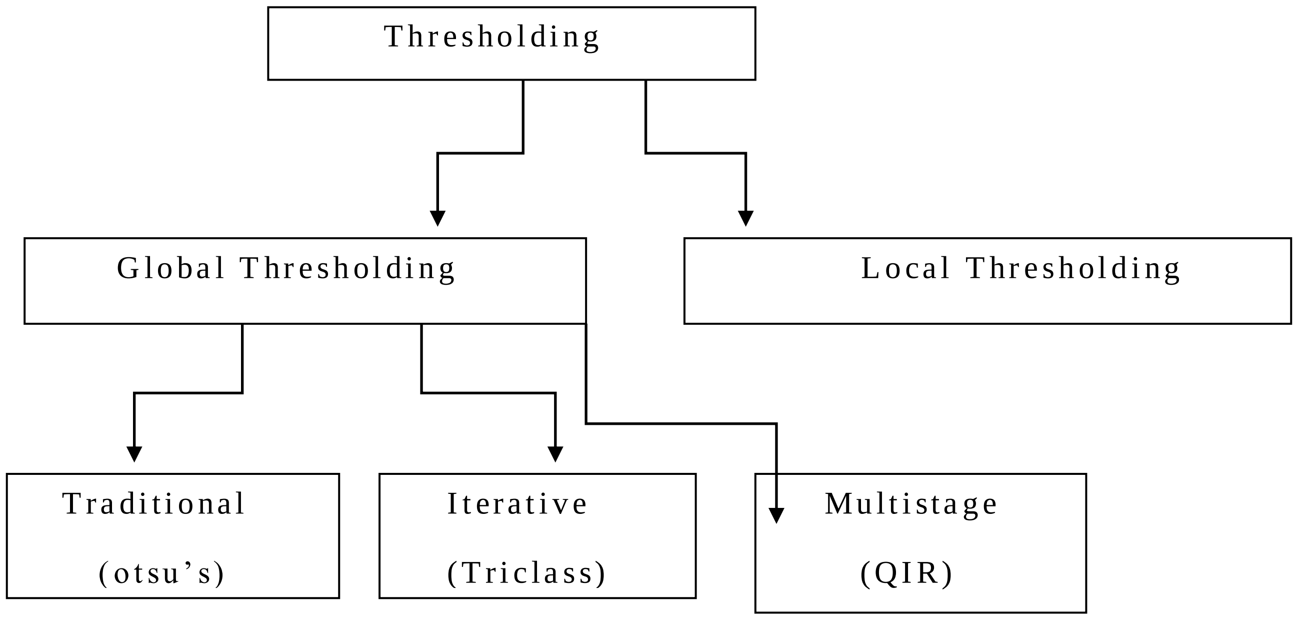


Figure 3.1: Thresholding Techniques

Global Thresholding

Global thresholding is used when the intensity distribution between the objects of foreground and background are very different. A histogram of the input image intensity should reveal two peaks, corresponding respectively to the signals from the background and the object. (Note: It is a core assumption of the current version of the 3DMA software that the input data set consists of 2 phases, a phase comprising the object of interest and a single other background phase.) Global thresholding consists of setting an intensity value (threshold) such that all voxels having intensity value below the threshold belong to one phase, the remainder belongs to the other. Global thresholding is as good as the degree of intensity separation between the two peaks in the image. It is an unsophisticated segmentation choice. . When the difference between the foreground and background are very different to each other, a single threshold intensity value can be used to differentiate. This type of thresholding the threshold intensity is only dependent on the pixel value of the image. Some of the most used global thresholding is Otsu's method.

Local Thresholding

A threshold g(x,y) is a value such that

$B(x, y)=\{ 1$ if $f(x, y) < g(x, y)$, 0 otherwise

Where b(x, y) is the required binarized image and f(x, y) be intensity of the pixel at location (x, y) of the given image. In local adaptive thresholding techniques a threshold is calculated for each pixel based on some range, variance and others parameters of neighborhood pixel. It can be approached in different ways such as background subtraction, water flow model, means and standard derivation of pixel values, and local image contrast. There are some drawbacks of local thresholding too. It depends upon region size and it is also very time consuming. That's why some people use hybrid of global and local thresholding techniques.

Iterative Thresholding

This is a new method that is based on Otsu's but it differs from the actual method. For the first iterations of the algorithm we apply Otsu's method onto the image to find out the Otsu's threshold value. Our method in this approach divides the image into three parts instead of dividing into two classes which separated by Otsu's threshold. These three classes are defined as first which is higher than the larger mean called as foreground, the background with pixel values are less than the smaller mean and more importantly a third class we call the TBD(to-be-determined) region with pixel values fall between the foreground and background mean class. Then at the next iteration, the method keeps the previous foreground and background regions unchanged and re-applies Otsu's method on the TBD region only to, again, separate it into three classes in the similar manner. When the iteration stops after meeting a preset criterion, the last TBD region is then separated into two classes, foreground and background, instead of three regions. The final foreground is the logical union of all the previously determined foreground regions and the final background is determined similarly.

Multistage Thresholding

Multistage Thresholding is also called Quadratic Ratio Technique. It is found very useful in case of handwritten character. In this type of Thresholding techniques all the details of the handwriting are to be retained. The handwriting may be written by variety of media such as fountain pen, ballpoint pen and also a fuzzy image where it is very hard to read whether an object belongs to foreground or background of the image. Two important parameters that separate the sub images are A, which separates the foreground and the fuzzy sub image, and C, which separate the fuzzy and the background sub image. If a pixel's intensity is less than or equal to A, the pixel belongs to the foreground. If a pixel's intensity is greater than or equal to C, the pixel belongs to the background. If a pixel has an intensity value between A and C, it belongs to the fuzzy sub image and more information is needed from the image to decide whether it actually belongs to the foreground or the background.[11]

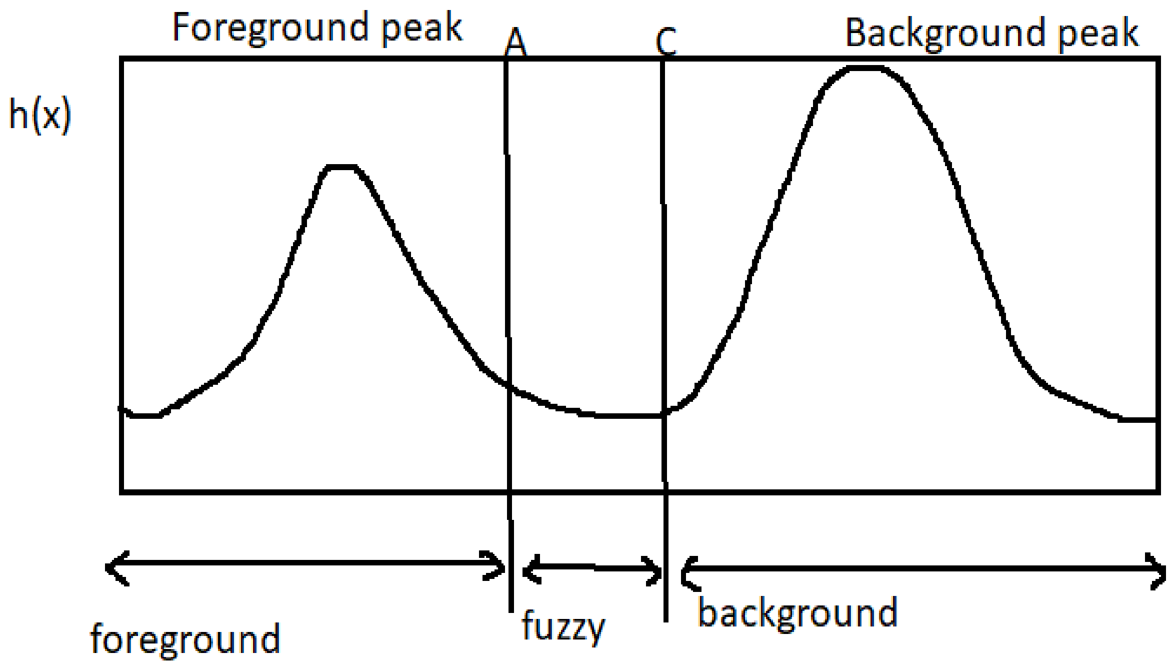


Figure 3.2: Foreground and background peak of Threshold function.

Image Dilation

In image processing there are some operations called Morphological operations that process images based on shapes of the images. They apply some structuring element on to the image and generate an output image. There are mainly two basics morphological operations. Erosion and Dilation.

Erosion:

1. It is generally used to erode away the boundaries of foreground object from the background of the image.
2. It can be mainly used to decrease the features of an image.

How Erosion Works:

1. First of all original image is taken for considerations.

2. A kernel of size (3 or 5 or 7) is convolved with the above given image.

3. In the original image pixel value 1 is considered if and only if all the pixels around the kernel is 1, otherwise it is set to 0.

4. Thus all boundary pixels of the image is discarded by using desired size of kernel.

5. As a result the thickness of the foreground object decreases.

Example of Erosion: Before Erosion:

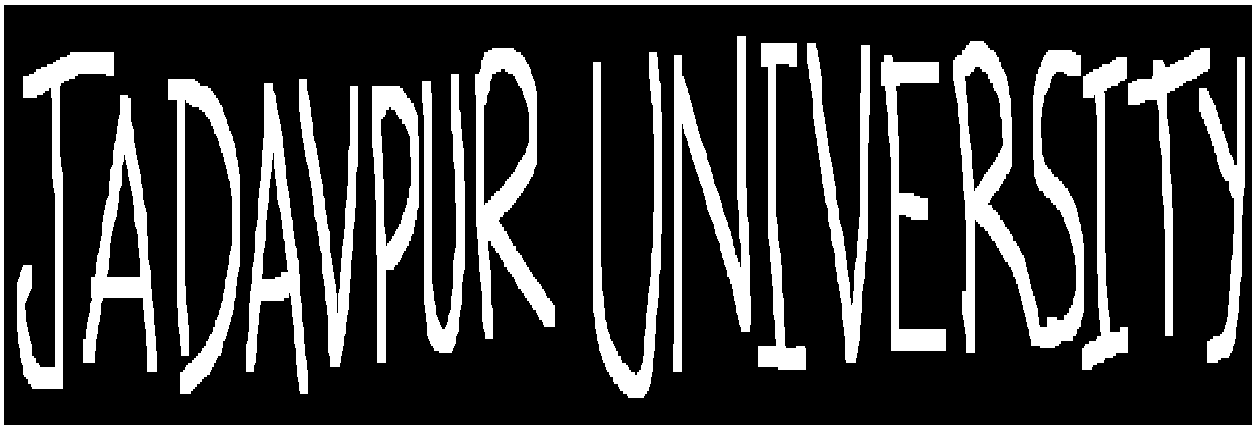


Figure 3.3: Before applying erosion on the image

After Erosion:

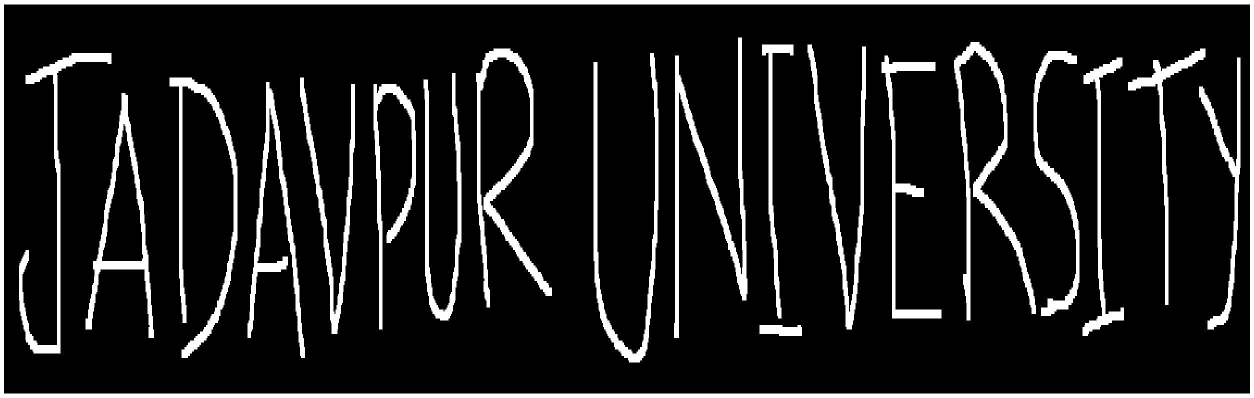


Figure 3.4: After applying erosion on the image

Dilation:

Dilation is another technique of morphological operations. It plays a very important role in image segmentation. It works as opposite to the Erosion. It increases the object area. It is also used to accentuate features.

How Dilation Works:

1. First of all an original black and white image is taken.
2. A kernel of size (3 or 5 or 7) is convolved with the above given image.
3. In whole matrix of original image a pixel element is 1 if at at least one pixel under the kernel is 1.
4. It also increases size of foreground object (white region) from the background of the image.

Example of Dilation:

Before Dilation:

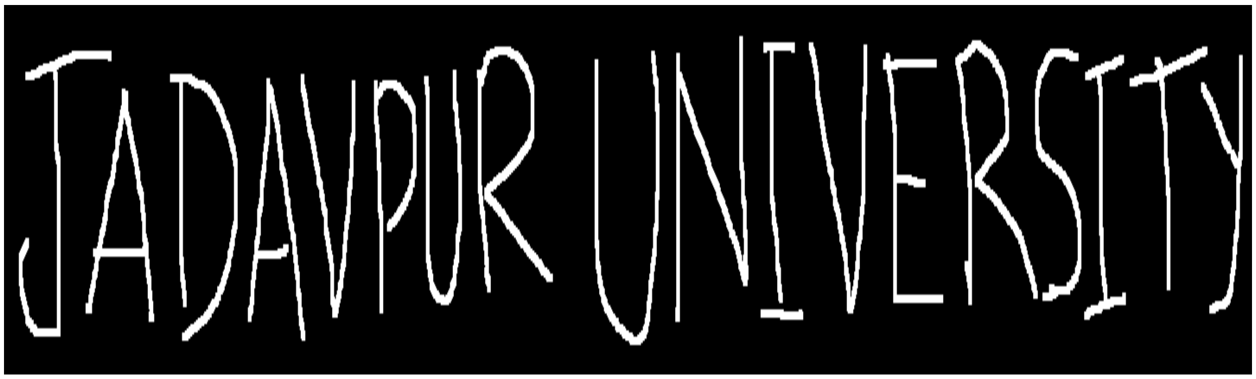


Figure 3.5: Before applying dilation on the image

After Dilation:

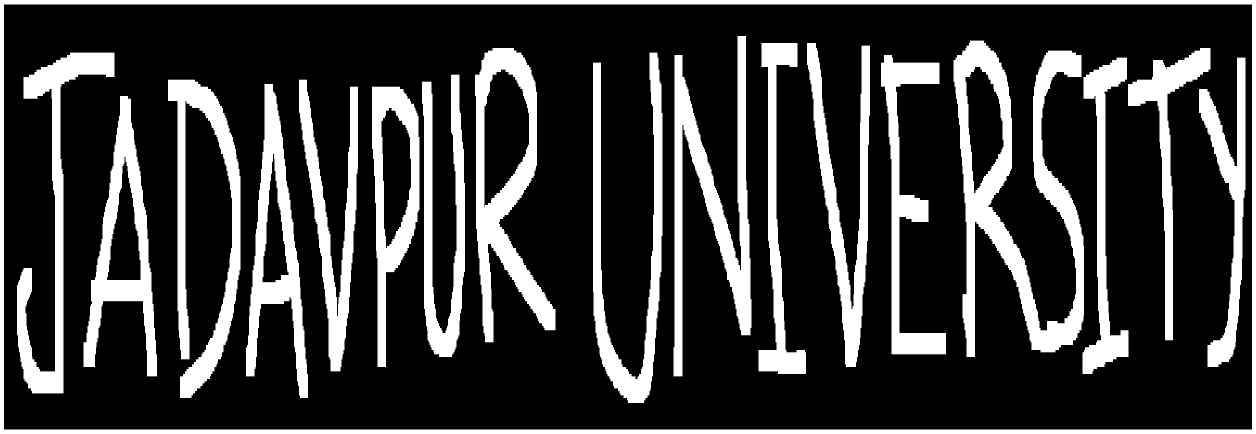


Figure 3.6: After applying dilation on the image

Contours based Image Segmentation Algorithm:

What are Contours?

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. Theoretically contours means detecting curves in the image. There is definitely a difference between edges and contours.

Difference between Edges and Contours:

Edges:

Edges are computed as points that are extreme of the image gradient in the direction of the gradient. if it helps, you can think of them as the min and max points in a 1D function. The point is, edge pixels are a local notion: they just point out a significant difference between neighboring pixels.

Contours:

Contours are often obtained from edges, but they are aimed at being object contours. Thus, they need to be closed curves. You can think of them *as* boundaries (some Image Processing algorithms & libraries call them like that). When they are obtained from edges, you need to connect the edges in order to obtain a closed contour.

Mathematical Morphological Operators:
Mathematical morphology theory is first introduced by Matheron to analyze the geometrical structure of metallic geological samples. Based on this theory, it is based on set like of operations. It also provides an approach that incorporate shape information of a signal. There are basically two types of morphological operations: erosion and dilation.

Erosion:

Erosion of a grey scale image A by another structuring element B, denoted by AϴB is defined as follows,

$$A\Theta B(m,n)=\min\{A(m+s,n+t)-B(s,t)\}$$

Erosion is simply a operator having values less than or equal to the values of A.

When the structuring element *B* has a center (e.g., *B* is a disk or a square), and this center is located on the origin of *E*, then the erosion of *A* by *B* can be understood as the locus of points reached by the center of *B* when *B* moves inside *A*. For example, the erosion of a square of side 10, centered at the origin, by a disc of radius 2, also centered at the origin, is a square of side 6 centered at the origin.

How Erosion Works:
1. First of all original image is taken for considerations.

2. A kernel of size (3 or 5 or 7) is convolved with the above given image.

3. In the original image pixel value 1 is considered if and only if all the pixels around the kernel is 1, otherwise it is set to 0.

4. Thus all boundary pixels of the image is discarded by using desired size of kernel.

5. As a result the thickness of the foreground object decreases.

Dilation:
Mathematical morphology theory is first introduced by Matheron to analyze the geometrical structure of metallic geological samples. Based on this theory, it is based on set like of operations. It also provides an

approach that incorporate shape information of a signal. There are basically two types of morphological operations: erosion and dilation.

Dilation of a grey scale image A by another structuring element B. is given by as follows,

$$A\Theta B(m,n)=\max\{A(m+s,n+t)+B(s,t)\}$$

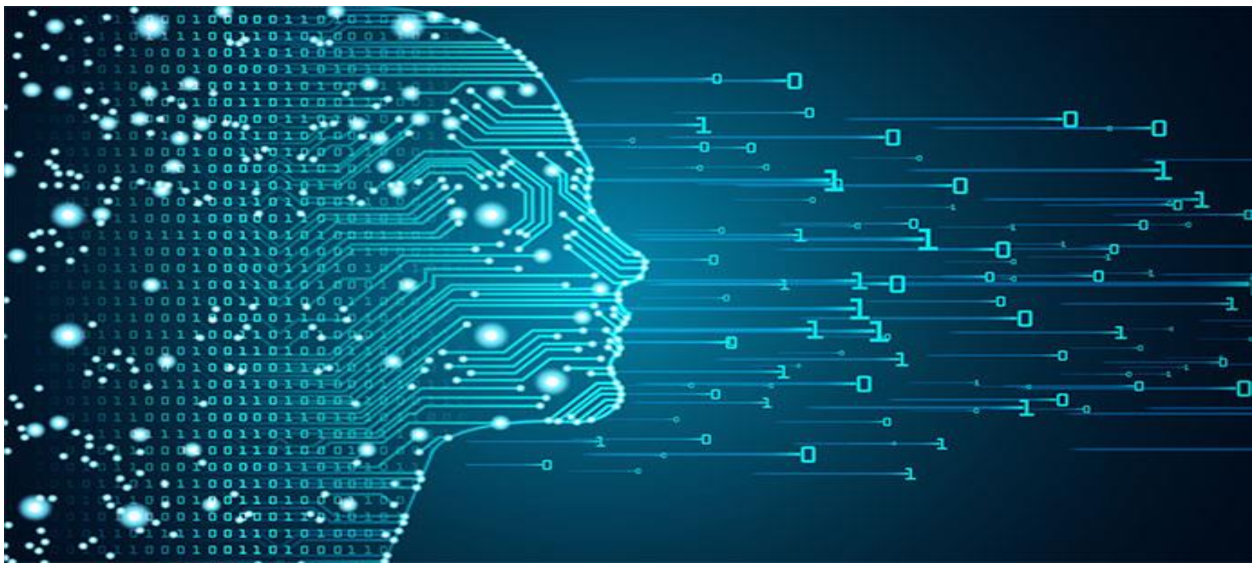
Dilation is a expansion operator in the values of $A\Theta B$ are always larger than or equal to the values of A. If *B* has a center on the origin, as before, then the dilation of *A* by *B* can be understood as the locus of the points covered by *B* when the center of *B* moves inside *A*. In the above example, the dilation of the square of side 10 by the disk of radius 2 is a square of side 14, with rounded corners, centered at the origin. The radius of the rounded corners is 2.

How Dilation Works:

1. First of all an original black and white image is taken.
2. A kernel of size (3 or 5 or 7) is convolved with the above given image.
3. In whole matrix of original image a pixel element is 1 if at at least one pixel under the kernel is 1.
4. It also increases size of foreground object (white region) from the background of the image.

By the Erosion and dilation operators above, we are able to detect the edge of the given image A which can also be denoted by $E_e(A)$, is also defined as the difference set of original image A and the result of the erosion A.

CHAPTER 4: DEEP LEARNING FOR HANDWRITTEN DIGIT



Chapter Gist: This chapter briefly explains the deep learning and its architecture, their usefulness and role of deep learning in very complex problems.

Overview

Deep Learning is the subfield of Machine Learning which is inspired by human brain also called artificial neural network. Deep Learning makes learning algorithm much better and easier to use in problems which are very complex. Deep learning is also known as deep structural learning or hierarchical learning. The deep in deep learning is all about how many layers are into the neural networks. It can be used to solve many complex problems like automatic colorization of black and white image, object detection from the image, automatic handwriting generation, automatic text generations. [12]

Why Deep Learning?

Deep learning always outperforms the older learning algorithms. Deep learning requires high computing operations as it is very complex in functioning. Deep learning is all about computing derivatives and multiplying them.

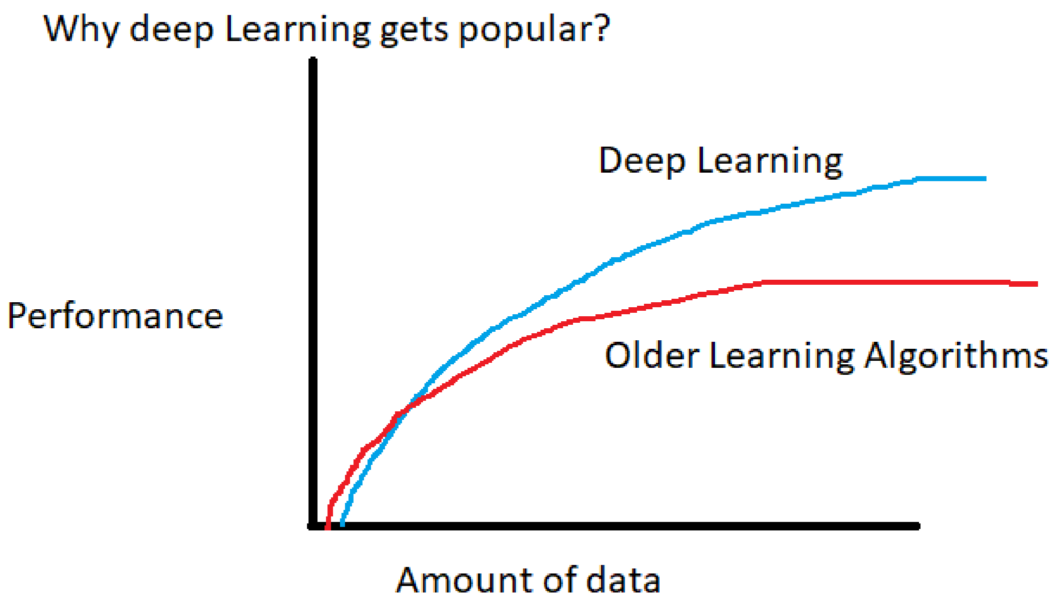


Figure 4.1: Why deep learning gets popular?

The major difference between deep learning and machine learning is approach to solve a problem. Deep learning solves a problem using end to end approach while machine learning requires dividing the original problems into many smaller problems and solving each individual problem and combining results of each individual problem into one big one. The execution time is comparatively very high for deep learning algorithms. The major drawbacks of this technique are it requires lots of data to be processed on. In this technique our algorithms learn as a realistic human do.

Neural Networks

A neural network in a simple term is the representation of human brain in computer. A neural is consists of lots of neurons where each neuron compute some mathematical operations. These operations can be also computed parallel with the other neurons. A simple neural network works as same human brain can do actually. [16]

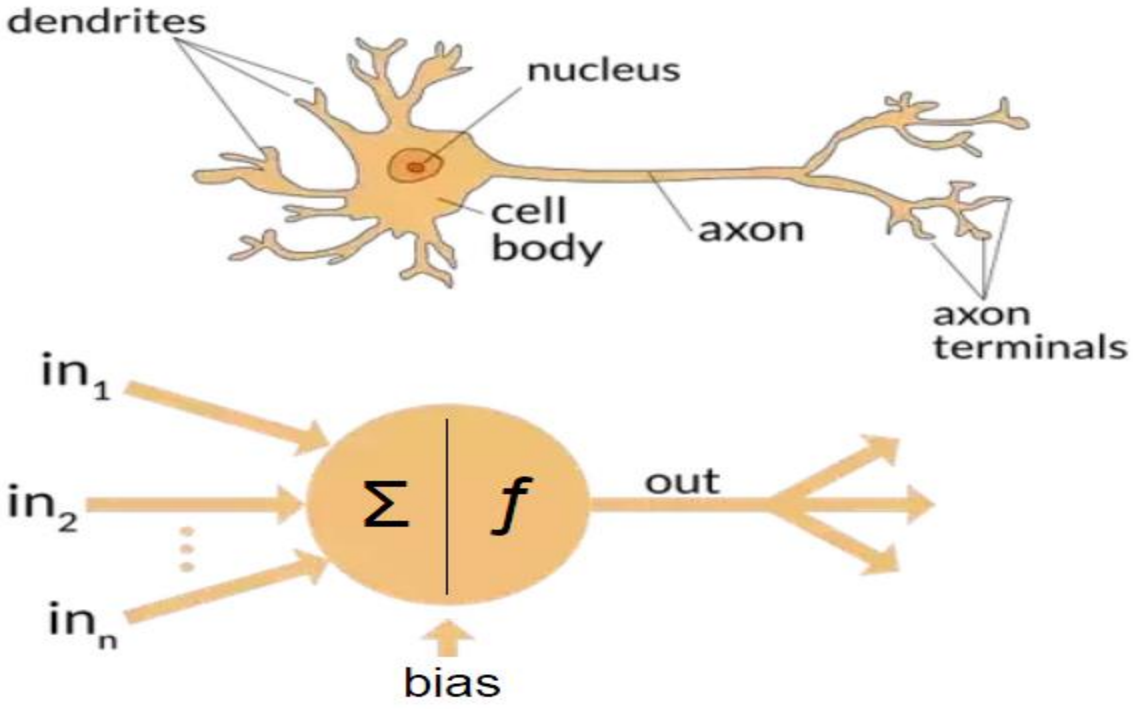


Figure 4.2: Similarity between biological neuron and artificial neuron

In the above figure it shows the similarity between human brain and artificial neural network. This neuron has three inputs and three outputs. First of all three inputs are multiplied with the weights of the each input layer and then it is added and given as input to a function named activation function. After applying these functions we get output.

Single Layer Perceptron

In machine learning, the perceptron is an algorithm that is used for supervised learning of binary classifier. A binary classifier is simply a function which can take input and decide which class the input belongs to. The perceptron is simply separating the input into 2 categories, those that cause a fire, and those that don't. It does this by looking at (in the 2-

dimensional case). In single layer perceptron it simply has two layer one is input layer and another is output layer.

Inputs are multiplied with weights and summed all and this is given to activation function and this function decides the output of the inputs. In the phase of perceptron training output is compared with the original output and error which get is compensated with calculating derivatives of the output.

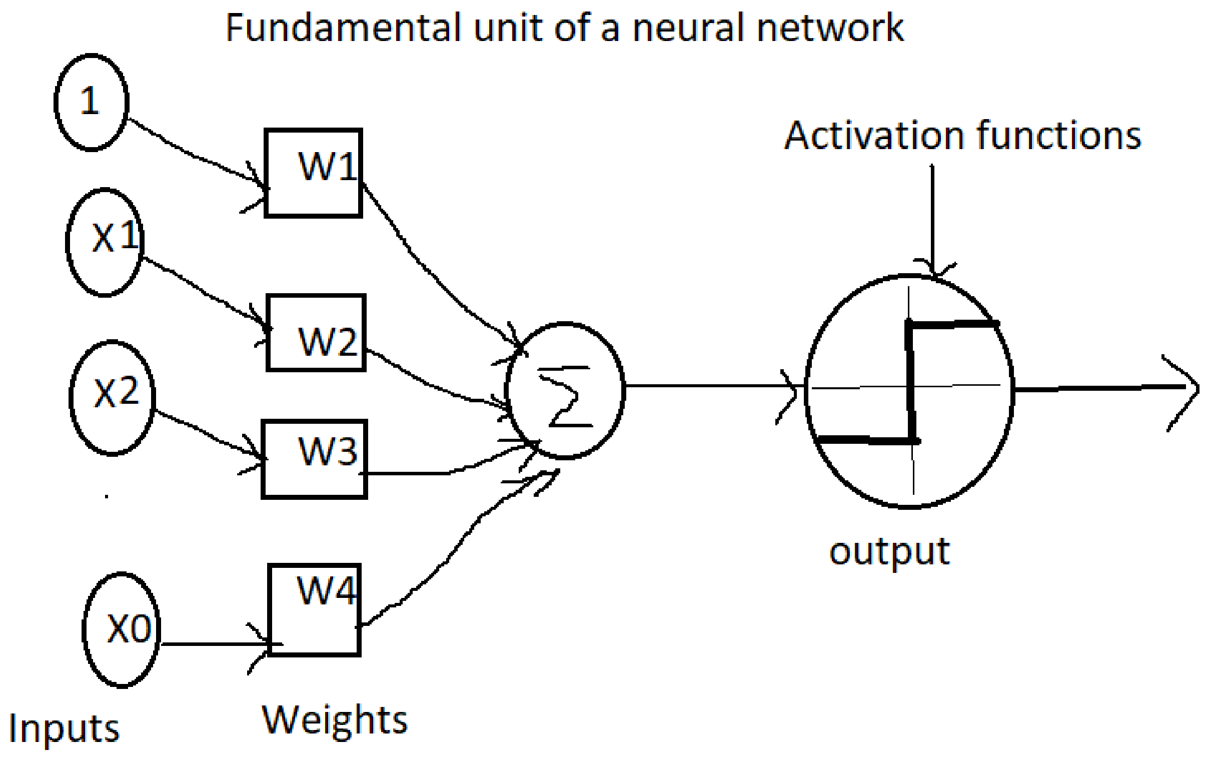


Figure 4.3: Single Layer Perceptron

Multilayer Perceptron

A multilayer perceptron is a combination of many single layer percetron. Multilayer perceptron are often applied to supervised algorithms. They train on input and output pairs and learn the correlation between them.

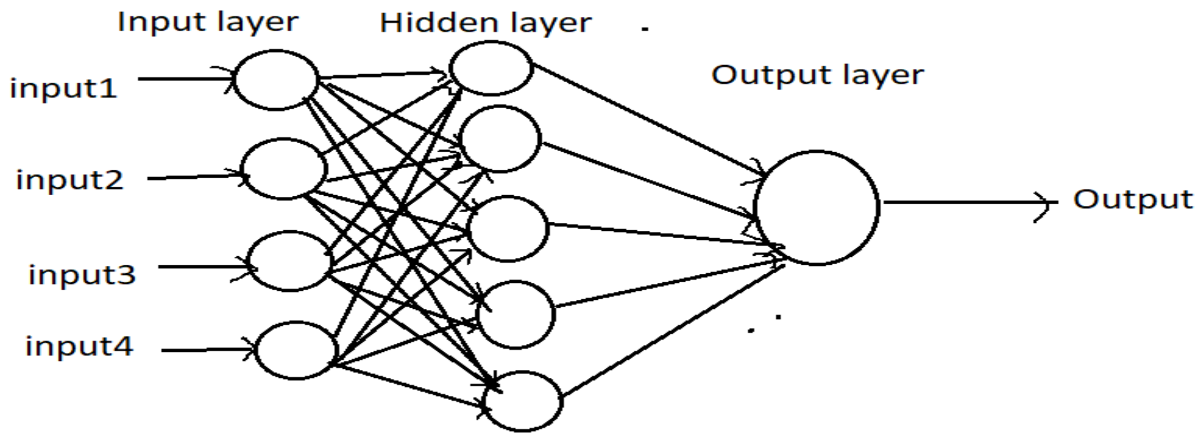


Figure: 4.4 Multi-Layer Perceptron

In multilayer perceptron it consists of more than layer. In the above figure it has three layers one is input layer, second is hidden layer and third is output layer. Here we can increase the number of hidden layers depending upon the complexity of the problems. Depending upon the problem output of the multilayer perceptron can be linear or binary or multiclass. For linear model we can use mean squared error as loss function. For binary class model we use logistic loss function while in case of multi class we use softmax as loss function.[18]

Forward and Backward Propagation

In neural networks we forward propagate to calculate the output and compared this output with the original output and calculate the error. To minimize this error we propagate backward by calculating the derivatives of error with respect to the weight. In neural network basic training we train our neuron when to get activated. Each neuron is trained such that it gets activated when a particular input is given to the neuron. Therefore by doing forward propagating we can see how well our network is performing and also finding the error. After finding errors into the network we back propagate and gradient descent algorithm to update the weights by subtracting it from the previous value. Forward pass of neural network depends upon multiplication and addition which is nothing but matrix operations and matrices are differentiable. In case of backward propagation it is all depending upon the partial differentiation of the loss function. So the function which is to be used for the purpose of loss calculating must be differentiable otherwise the back propagation algorithms does not work. [19]

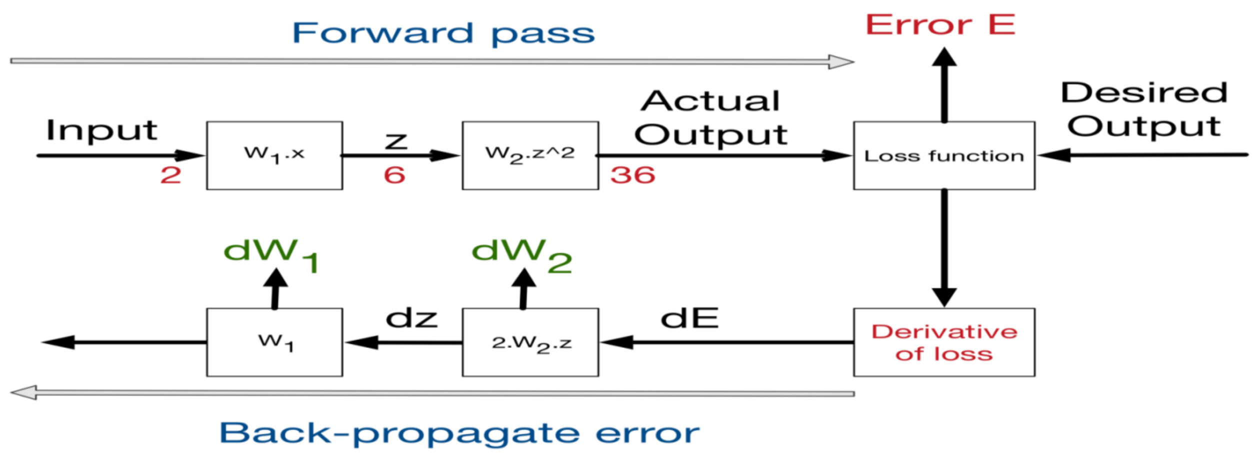


Figure: 4.5: Forward and backward propagation through the network

Activation Functions

Activation functions are functions which can decide whether to activate a function or not by calculating weight sum with adding bias. The purpose of this type of function is to introduce non- linearity into the output of neuron. We know, neural network has neurons that work in correspondence of weight, bias and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

Why do we need Activation Functions

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks. If we do not apply an Activation function then the output signal would simply be a simple linear function. A linear function is just a polynomial of one degree. Now, a linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data. A Neural Network without Activation function would simply be a linear regression Model, which has limited power and does not, performs good most of the times. We want our Neural Network to not just learn and compute a linear function but something more complicated than that. Also without activation function our neural network would not be able to learn and model other complicated kinds of data such as images, videos, audio, speech etc. [10]

Different types of Activation Functions

There are different types of activations functions for different types of problems.

Linear Activation Functions

Linear equation works as same as no activation function is used with the input. It has same equation as straight line i.e. Y=mX+C. we can make a neuron to use linear function depending upon the input and output. The range of linear function is from negative infinitive to positive infinitive. This function is used only as output layer. Linear function just tells about the nature of input is same as its output.

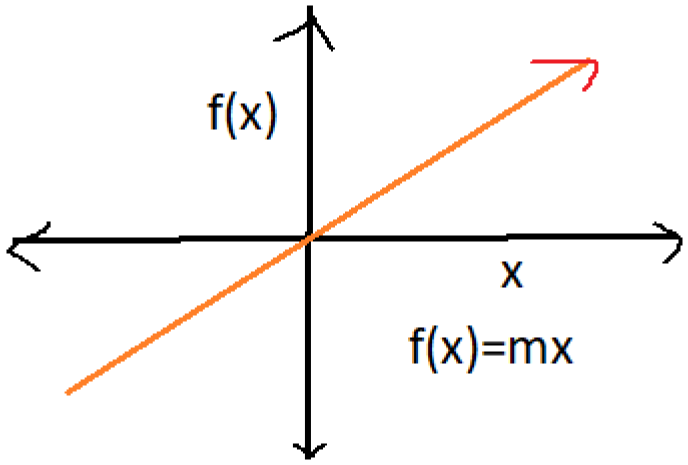


Figure 4.6: Linear Activation functions

A = cx, derivative with respect to x is c. That means, the gradient has no relationship with X. It is a constant gradient and the descent is going to be on constant gradient. If there is an error in prediction, the changes made by back propagation is constant and not depending on the change in input delta(x).

Sigmoid Activation Function

Sigmoid function has shaped like 'S'. The equation of the sigmoid function is $Y=1/\ (1+e^{-X})$. It is a non-linear function where X lies between -2 to 2 while Y is very step. This means very small changes in x would result large changes in Y. The range of this function is 0 to 1. Looks like its good for a classifier considering its property? Yes! It indeed is. It tends to bring the activations to either side of the curve (above x = 2 and below x = -2 for example). Another advantage of this activation function is, unlike linear function, the output of the activation function is always going to be in range (0, 1) compared to (-inf, inf) of linear function. So we have our activations bound in a range. Nice.

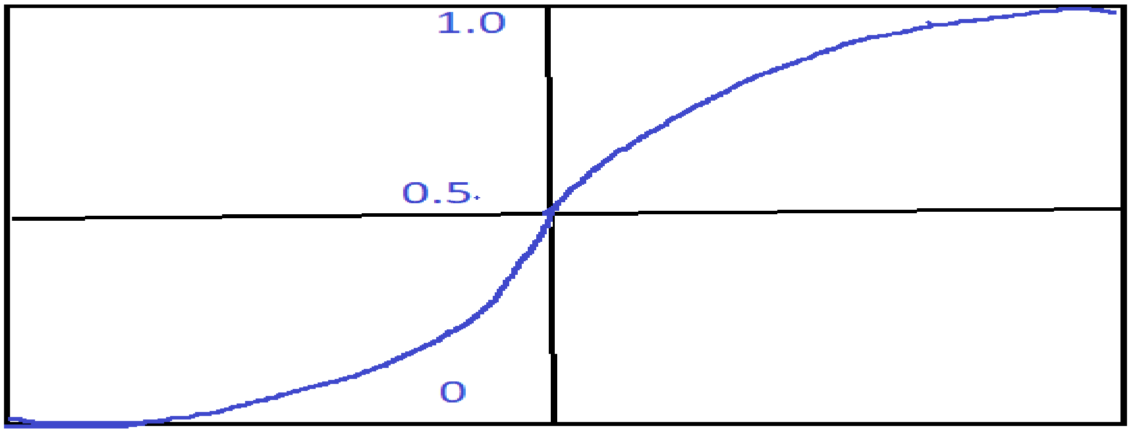


Figure 4.7: Sigmoid function

Tanh Function

This function is also called hyperbolic tangent function. This activation function always works better than sigmoid function. This function is also called hyperbolic tangent activation function. This activation function is actually shifted version of sigmoid function.[13]

```
tanh(x) = 2 * sigmoid(2x) - 1
```

This activation is ranging from -1 to 1. The nature of this function is non-linear. It is basically used in hidden layers of a neural network since it's values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in centering the data by bringing mean close to 0. This makes learning for the next layer much easier.

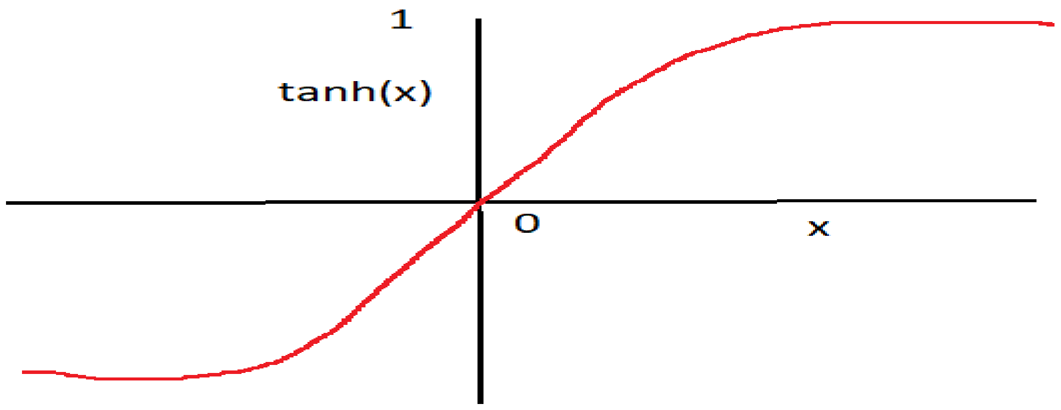


Figure 4.8: Tanh Activation Function

ReLU(Rectified Activation Function)

ReLU is the most used activation in the world till now. For many people this is the default activation function because of its simplicity. It is used in all Convolution neural networks and deep learning. But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

Advantages of ReLU over all Activation Functions:

1. ReLU has no problems of exploding gradient.
2. It has also no problem of vanishing gradient.
3. ReLU has a problem of dead activation state because of its zero nature.

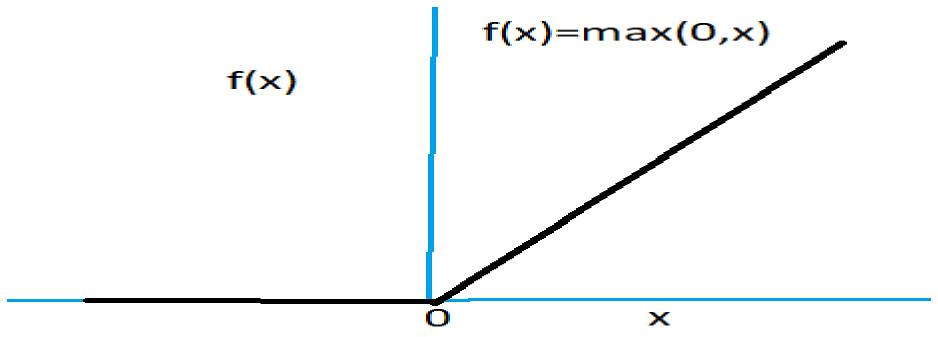


Figure 4.9: ReLU Activation Function

Optimization Algorithms

In deep learning optimization algorithms play an important role. Through the use of optimization algorithms we can reduce losses in our algorithms as much as possible. These algorithms actually tie the loss function and model parameters by updating the model with respect to output function. There are lots of optimizations in deep learning but gradient descent is the most popular among all of them.

Gradient Descent

Gradient descent is an algorithm that finds best fit line for given training data set.

For example:

X= [1, 2, 3, 4, 5]

Y= [12, 13, 14, 15, 16]

Given this data set X and Y we have to find out an equation (Y=mX+C) which best fit this data set. There are different variants of gradient descent algorithms.

1. Batch Gradient Descent
2. Stochastic Gradient Descent
3. Mini Batch Gradient Descent

In batch gradient descent we use full dataset to train our model while in case of batch gradient descent algorithms we use a mini batch of the dataset to train our model. In all of these algorithms stochastic gradient descent algorithms is the best algorithms.

Mean Squared Error and Batch Gradient Descent

$$MSE=1/n * \sum_{i=1}^{n} (y_i - (mx_i+b))^2$$

MSE is mean squared error. It is also called as cost function which we have to minimize. Gradient descent is the algorithm which helps to find such values of m and b for which the cost function is minimum.

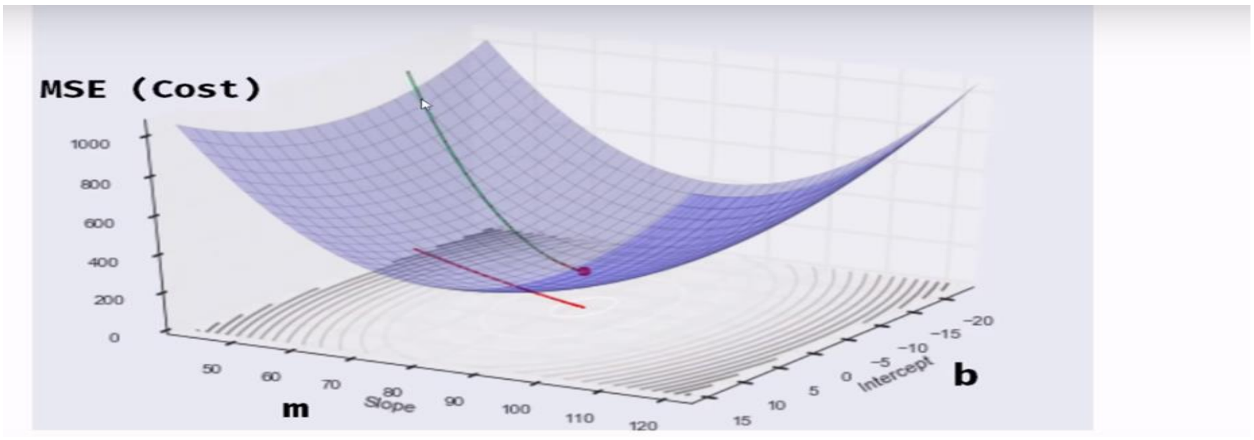


Figure 4.10: Cost function verses parameter m and b

Gradient Descent Algorithms:

```
Gradinet_Descent(X,Y):
        M_curr=B_curr=0
        Iterations=1000
        N=len(X)
        Learning_rate=0.001
        For i in range(iterations):
                Y_predicted=M_curr * X +B_curr
                Cost=(1/N) * sum([val ** 2 for val in (y-y_predicted)])
                Md=-(2/N) * sum(X * (y – y_predicted))
                Bd=-(2/N) * sum(y- y_predicted)
                M_curr=M_curr – Learning_rate * Md
                B_curr= B_curr – Learning_rate * Bd
```

Stochastic Gradient Descent

The term ‘stochastic’ suggests a system or a process that is engaged with a random probability. Hence in stochastic gradient descent we choose a random number every time we call the algorithms. In Gradient descent also there is term called “batch” which indicates a total number of dataset which is used for calculating the gradient descent for each iterations. In typical normal gradient descent algorithms we choose a our whole dataset as a batch size. Although by the use of whole dataset it is really useful for getting the minima value for the dataset but there are some problems when the dataset is really very large in dimensions. Suppose you have ten millions of dataset from which we want to apply batch gradient descent we need to calculate the whole dataset every time we need to updates our weights.

Difference between Gradient descent and Stochastic Gradient descent:

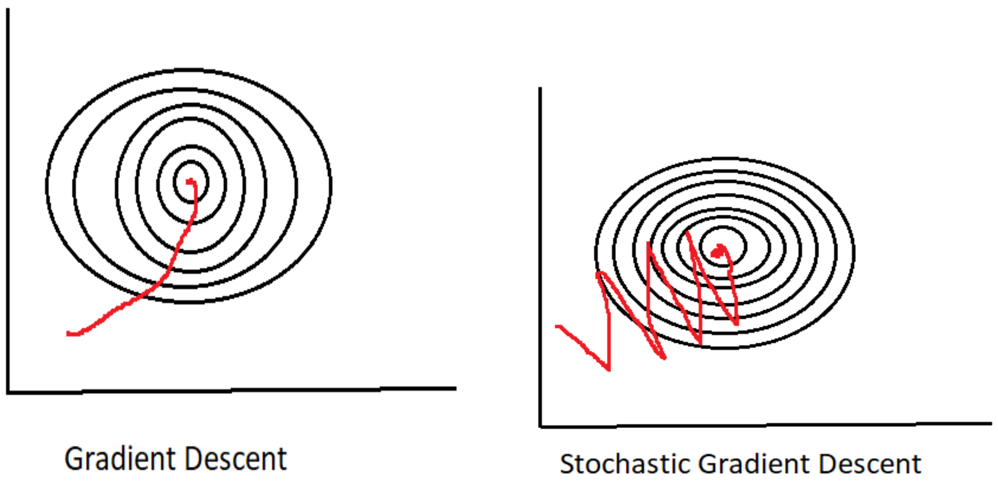


Figure 4.11: Paths taken by Gradient Descent and Stochastic Gradient Descent

Adam (Adaptive Moment Estimation)

This is the most popular optimization algorithm till 2019. Since this is the most popular algorithm it is modified version of the Gradient Descent algorithm.

Benefits of Adam over Stochastic Gradient Descent:

1. Adam is very straightforward to implement.
2. It is computationally very efficient.
3. It requires very little memory.
4. This algorithm is very appropriate for non- stationary objects.

Comparisons of Different Optimization Algorithms:

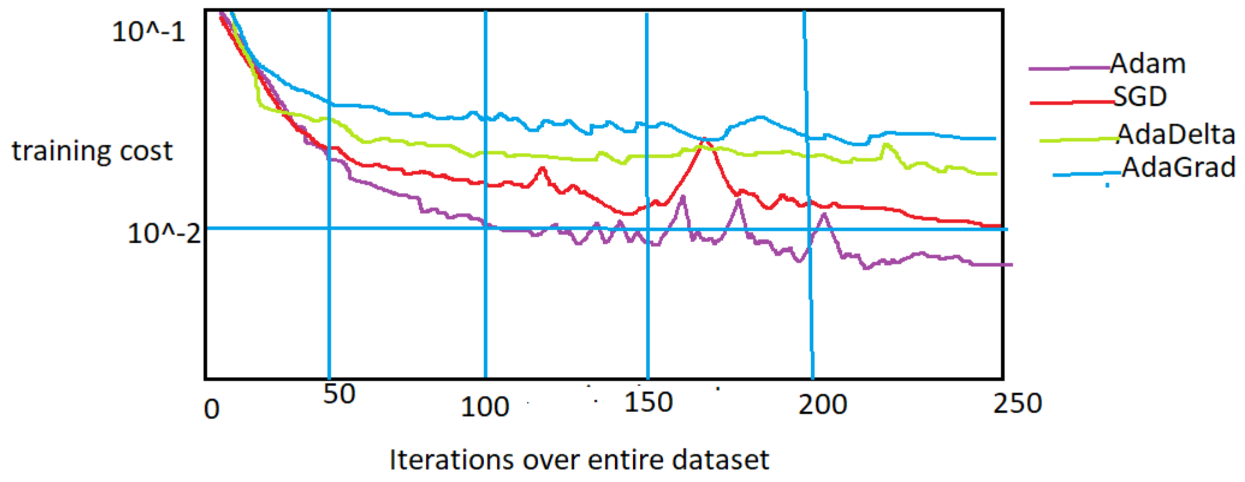


Figure 4.12: Comparisons between all optimization algorithms

Dropout and Regularizations:

Dropout in deep learning means we have to drop some of the unnecessary layers which are not important. Simply putting during the training phase of the neural networks some neurons are kept ignored which is chosen randomly. By ignoring we mean during particular instant that neuron is not considered. More technically speaking at each training individual nodes are either dropped or kept ignored given with the probability p. Dropout simulates a sparse activation from a given layer, sparse representation as a side-effect. As such, it may be used as an alternative to activity regularization for encouraging sparse representations in auto encoder models.[15]

Why do we need dropout?

We need dropout due to the problem of over fitting. Since over fitting can affect a lot a model so we need to take care of this problem. Since fully connected layers occupy most of the parameters, hence neurons require avoiding some of the precalculated values which are actually unnecessary to develop all the over fitting. In machine learning the only way to solve this type of problems of over fitting is regularizations. Regularization reduces the problem of over fitting by adding some penalty to the loss functions.[15]

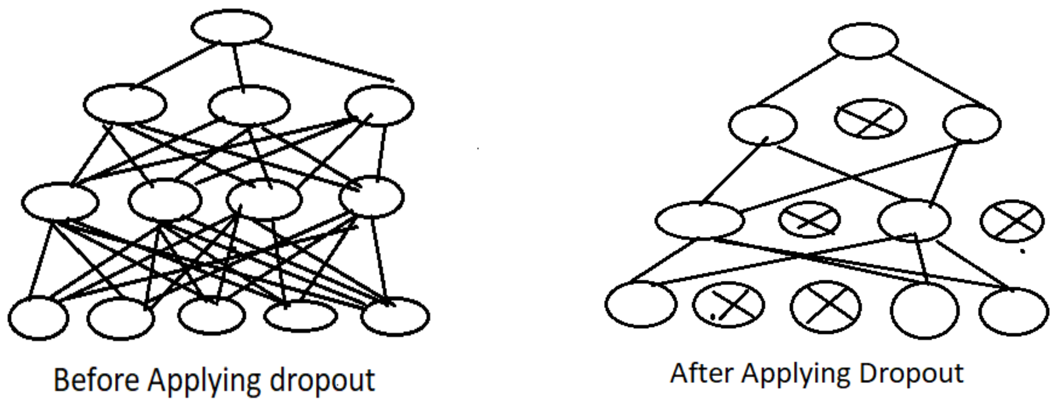


Figure 4.13: Logic behind using dropout

Multi-Class Classifier:

Those types of problems whose output belongs to more than one classes are called Multi-class classification problems. In scikitlearn there exists lots of algorithms which can be able to classify multi class problems. For the problems like handwritten digit which is ten class classification problem there exists a classifier called softmax is the best multi class classification algorithm.

Softmax Classifier

Softamx is the most popular multi class classification algorithm for multi class classification problems. Softmax is the extended version of the logistic regression which is based upon binary class classification problems. Softmax is just the generalization of the logistic regression to the multi class classification problem.

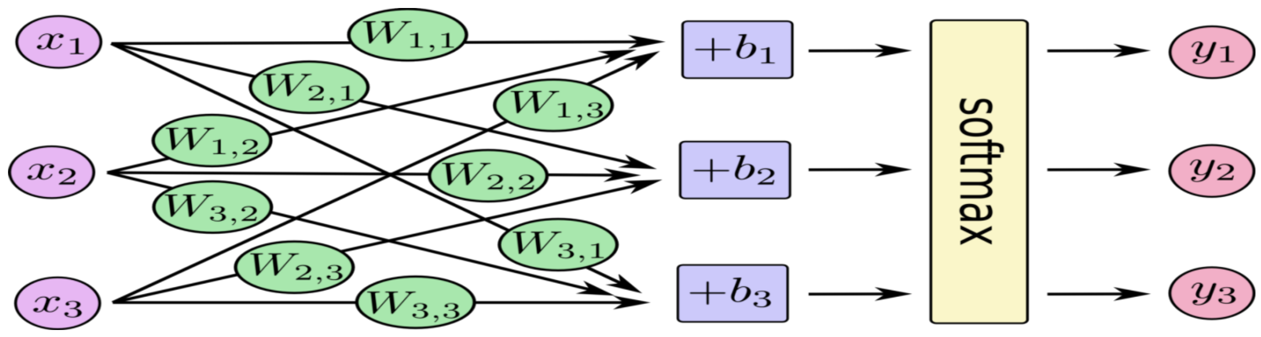


Figure 4.14: Example of a softmax classifier

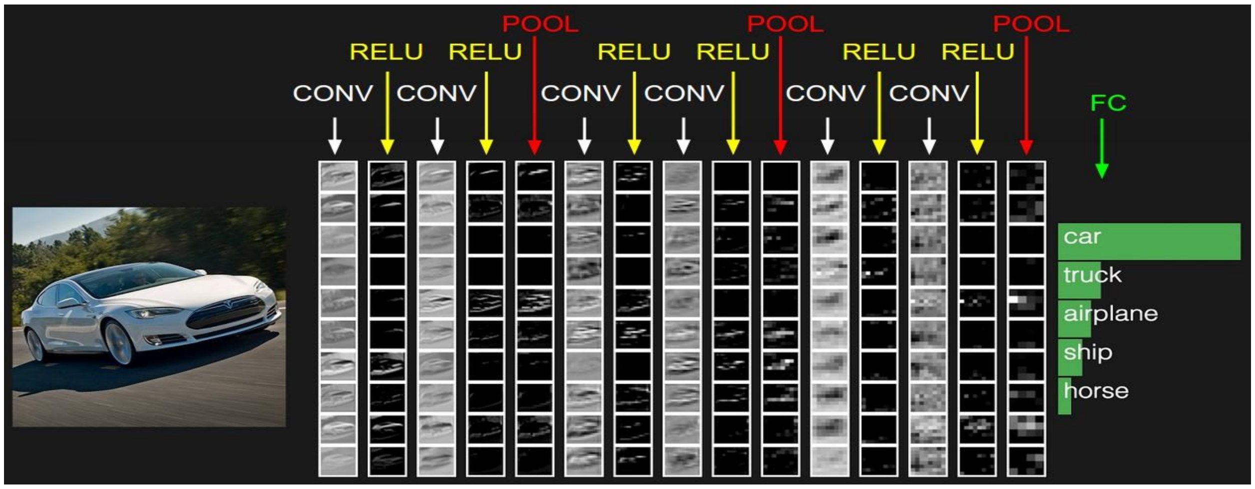
Batch Normalization:

We normalize the input layer by adjusting and scaling the activations. For example, when we have features from 0 to 1 and some from 1 to 1000, we should normalize them to speed up learning. If the input layer is benefiting from it, why not do the same thing also for the values in the hidden layers, that are changing all the time, and get 10 times or more improvement in the training speed. Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). To explain covariance shift, let’s have a deep network on cat detection. We train our data on only black cats’ images. So, if we now try to apply this network to data with colored cats, it is obvious; we’re not going to do well. The training set and the prediction set are both cats’ images but they differ a little bit. In other words, if an algorithm learned some X to Y mapping, and if the distribution of X changes, then we might need to retrain the learning algorithm by trying to align the distribution of X with the distribution of Y. [17]

Why Do we need?

Usually, in order to train a neural network, we do some preprocessing to the input data. For example, we could normalize all data so that it resembles a normal distribution (that means, zero mean and a unitary variance). Why do we do this preprocessing? Well, there are many reasons for that, some of them being: preventing the early saturation of non-linear activation functions like the sigmoid function, assuring that all input data is in the same range of values, etc.

CHAPTER 5: HANDWRITTEN DIGIT RECOGNITION USING CNN



Chapter Gist: This chapter provides an overview of the various details and techniques involved in Edge Detection, giving special emphasis on the Canny Edge Detection Technique.

Overview

Convolution Neural Network is modified version of traditional Neural Network. They are good at classifying image based things. Traditional approach is transferring all image pixels to neural networks. In CNN, we will detect some pattern first and feeding these patterns to neural networks. In this way we process images in less complex way whereas we can get more successful results.

Is CNN Perfect?

Convolution neural networks are not perfect. They only look for some patterns. They do not interest in about that the locations of these pattern are meaningful or not. For example changing ear and lip of image of a girl does not change the result. CNN are powerful deep learning method for pixel based things. They produce successful results even on single CPU.

The most successful approach for detecting handwritten digits is still CNN. The primary purpose of convolution is to find features in image using feature detector put them into feature map which keeps the original relation with image.

Convolution Architecture

A Convolution Neural network is a popular deep learning algorithm which takes input as image either color or grey scale image and some kernels are also learned from the image dataset. CNN can be able to differentiate one image from the other. In this algorithm preprocessing required is very low compared to the others algorithms. The architecture of the CNN is equivalent to the connectivity patterns of the human brains and it was also inspired from the visual cortex. Each individual neuron can send and receive information through the restricted regions. Convolution Networks are not much efficient deep learning models that can classify image data. The multi stage architecture to CNN is inspired from the human brain. Since in these models features are learned automatically, it becomes the most demanding algorithm for image dataset [17]. These types of different feature come from different layers of the networks. In CNN each layer has distinct number of neurons and is presented in three dimensions: height, width, depth. To understand the architecture of convolution neural networks we can observe it into two parts. Input images are presented as a matrix of pixels. By convention a grey scale can be represented by two dimension while a color image is represented by 3 dimension.

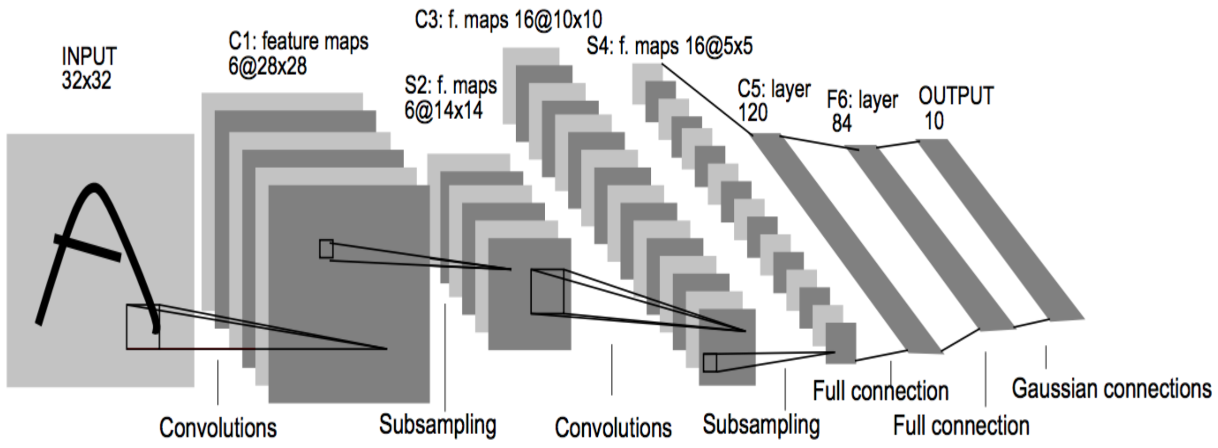


Figure 5.1: CNN architecture step by step

Image Representation in CNN:

An image can be represented in CNN either by a two dimensions matrix or three dimension matrixes. A grey scale image is represented by two dimensions while RGB image is represented by three dimension matrixes. The values of the matrix lie between 0 to 255. By convention 0 is considered as black while 255 is considered as white and the values between 0 to 255 are considered as grey. A grey scale representation of image in 2-D is very easy but an RGB representation of the image in 3-D is very difficult. Because we have to consider all the three red, green and blue colors for the consideration of the three dimension matrix. The values of the pixels need to be stored in the computer’s memory; this means that in the end the data ultimately need to end up in a binary representation, the spatial continuity of the image is approximated by the spacing of the samples in the sample grid. The values we can represent for each pixel is determined by the sample format chosen.[2]

Convolution Operations:

Convolution can do edge detection within a image. Edge in a image is a border between black and white in gray scale image. In convolution operation a kernel or filter is convolved with the input image to get a feature map of desired size. This desired size of feature map represents the edge within the input image. Corresponding to our input image we get a horizontal edge as a two or three pixel wide image as output.

How to perform convolution operation?

In order to perform the convolution operation we need to go through a set of operations,

1. First of all we have to flip the mask of the image.

2. Then we have to side the mask into the image.

3. There after we have to multiply the corresponding elements and then add them.

4. In the next step we have to repeat this procedure until and unless all the values of the image have been calculated.

What is mask?
A mask is defined as a signal which can be represented by a two dimensional matrix. Actually, the masks are of usually different sizes like 1x1, 3x3, 5x5, and 7x7. A mask should always be in odd number otherwise we can't be able to find out the middle of the mask. Convolution plays an important role while doing the masking operation in order to achieve the goal of the procedure. [5]

Convolution Layers:
In Convolution neural network we learn the kernel or sobel from the dataset given that is our only goal. Kernel or sobel in CNN works like weights. Let layer l be a convolution layer. Then, the input of layer l comprises m (l-1) 1 feature maps from the previous layer, each of size m (l-1) 2 ×m (l-1) 3. In the case where l = 1, the input is a single image I consisting of one or more channels. This way, a convolution neural network directly accepts raw images as input. The output of layer l consists of m (l) 1 feature maps of size m (l) 2 ×m (l) 3. A convolution neural network is a type of deep learning algorithm which takes input as binary or color images and gives output as kernels which can detect special features in the image. The outputs of the CNN can also assign weights and biases to objects in the image and can be able to differentiate one image from the other. The preprocessing required in CNN is much lower than any other classification algorithms. The architecture of the CNN is very much similar to the human brain and it was definitely inspired from the visual cortex of the human.[16]

Feature detector*:* These are the actual kernels which are learned as a part of the back propagation. For each and every feature in an image there is also a feature detector**.**

Feature Map*:* Feature maps are the results of the convolution between the input image and feature detector.

Data Augmentation:

Data augmentation is technique through which we can be able to discard the problem of acquiring more data for the vision problems. In spite of data availability we need to acquire some sort of data where we need right type of data. Moreover the data has to be in good diversity because we do not need to create a simple copy of our data we need to just get data which are very similar in looking like the data but the shapes pf the data should be different. If we think of our data generalizes well during the testing phases of the development we might get an actual generalize result. To overcome this type of problems of limited quantity and diversity of data, we need to generate our own data with the help of existing data which we have already been given to us. This method of generating our own data is known as data augmentation.

In this paper, let us explore few of the most commonly used data augmentation techniques with the help of visualization of the image. Here are the different techniques that we are using in our data augmentation techniques.

- Scaling
- Flipping
- Translation
- Rotation (by 90 degrees)
- Perspective transform
- Adding salt and paper noise
- Lighting condition
- Rotation(at finer angles)
- Cropping
- Horizontal shifting
- Zooming
- Shearing
- Vertical shifting
- Image Resizing

But before applying any techniques we need to resize our image because image is gathered from the various sources which needs to be resized into proper shapes.

Some Examples of data augmentation techniques:

Scaling:

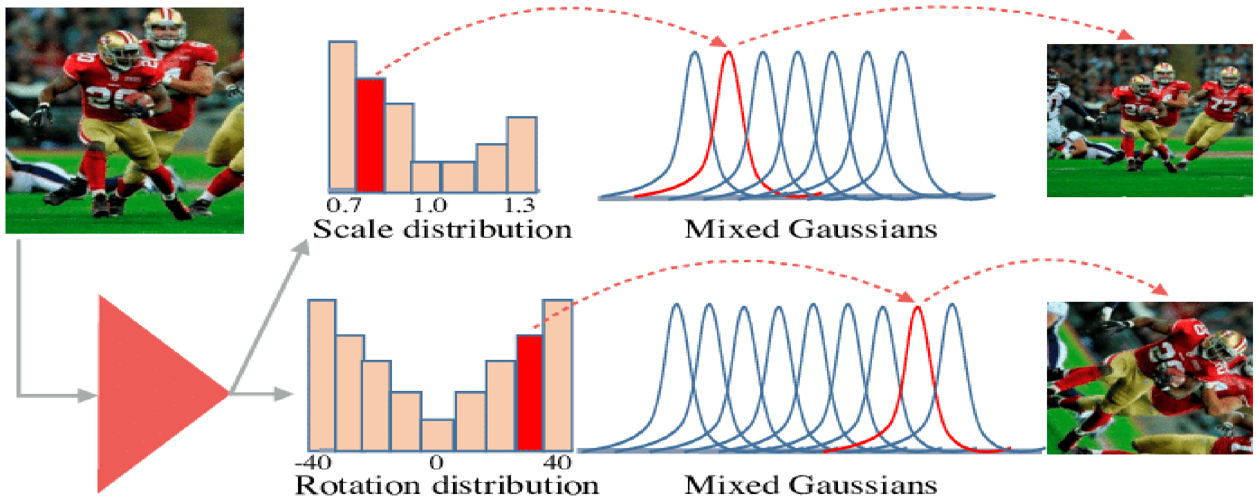


Figure 5.2: Adversarial scaling and rotating.

Shearing:

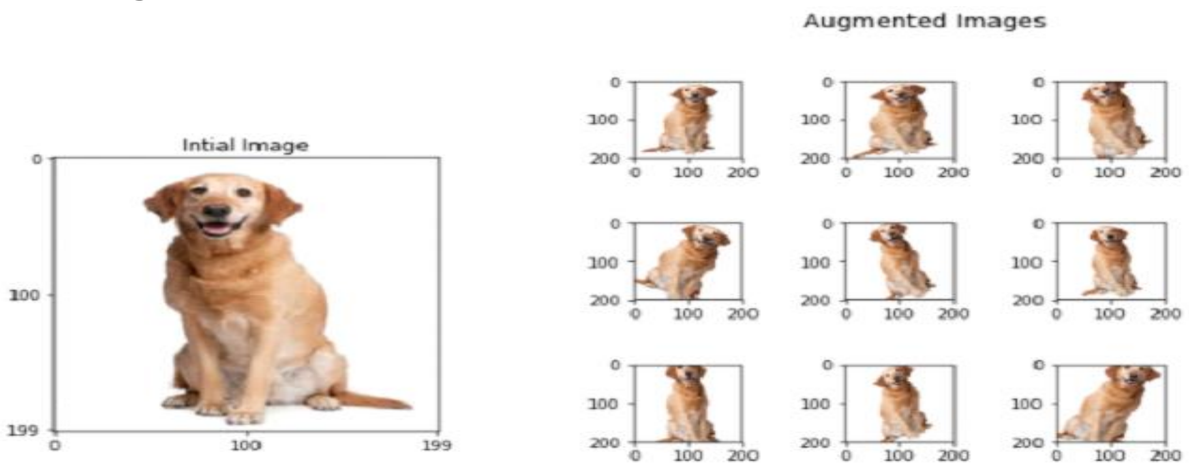


Figure 5.3: Example of shearing

Strides:

Stridng is used to reduce the size of the matrix.

Using stride=1

1	0	0	1
1	0	0	1
0	0	1	1
1	1	0	0

We get a matrix of size 3x3.

1	**0**	**1**
0	**1**	**0**
0	**1**	**1**

Padding:
Padding plays a very important role in convolution layer. Because of doing convolution and striding together we actually loses our image sizes. In case we want our output as the same size of original image we need to do padding. There are various types of padding which is used in convolution among them two given are the most important.

1. Zero padding
2. Same Padding

Suppose the given matrix is 3x3.

0	0	0
255	255	255
0	0	0

After Applying Zero padding we get a matrix of size 5 x 5.

0	0	0	0	0
0	0	0	0	0
0	255	255	255	0
0	0	0	0	0
0	0	0	0	0

Why need padding?
In padding every time we are using filter or sobel to scan the image the size of the image go smaller and smaller every time. In our case we do not want to lose the original size of the image that's why we are doing padding of the original image.

Example:
Suppose we want my output to be 6 X 6 then I have to pad my input image.

1. For a normal convolution where we do not use padding the result we will be as follows. If the kernel size is of K x K and input image size is n x n, then the resulting image will be as follows.

   $$N \times N \longrightarrow (N-K+1) \times (N-K+1)$$

2. For another case where we use padding techniques to preserve the original size of the image. If the kernel size is of K x K and input image size is n x n and padding with p, then the resulting image will be as follows.

   $$N \times N \longrightarrow (N-2K+1) \times (N-2K+1)$$

3. For another case where we use padding with strides. If the kernel size is of K x K and input image size is n x n and padding with p and stride s, then the resulting image will be as follows.

   $$N \times N \longrightarrow ((N-K+2P)/S +1) \times ((N-K+2P/S +1))$$

Pooling Layer:

Normally it is very convenient to insert a pooling layer between two successive convolution layers. Its primary function is to reduce the size of the image and to draw special features from the image and also to avoid over fitting. The pooling layers are so design that it can operate on each slice of the input and resize it as it is required using its most popular MAX operation. The most common use of the pooling layer is the pooling layer of size 2 x 2 with a stride of size 2 along both size width and height thus it discarding 75% of the activations. More generally pooling layer can be described as follows:

- Pooling accepts a volume of size W x H x d
- It also requires two hyper parameters

    1. Their spatial extent P,
    2. The stride S,

- As a result it produces a volume of size $W_1 \times H_1 \times D_1$

W_1 x H_1 x D_1 where,

$W_1= (W_1-P)/S+1$

$H_1= (H_1-P)/S+1$

$D_1=D$

- It introduces zero parameters because it can compute a fixed function of the input.
- For pooling layers also it is very uncommon to pad the input using zero-padding techniques.

Here it is worth noting that there exist only two variations of pooling techniques which are always found in the practice:

- A pooling layer with p=3 and s=2
- A pooling layer with p=2 and s=2

General Pooling Techniques:

- In addition to max pooling techniques there are also many techniques which operate the same way as the max pooling techniques. There are average pooling and L2 norm pooling.

Example of max pooling:

1	1	8	1
5	6	6	4
4	3	6	9
2	5	5	8

Pool=2 Stride=2

4 x 4

After applying max pooling:

6	**8**
5	**9**

Pooling layer can easily reduces the size of matrix as shown in the above figure. The size of a 4 x 4 matrix is reduced to a size of 2 x 2. In the given example the input is of size [224x224x64] is pooled with a pool size of 2 and a stride of 2 as a result we get the output size as [112x112x64].

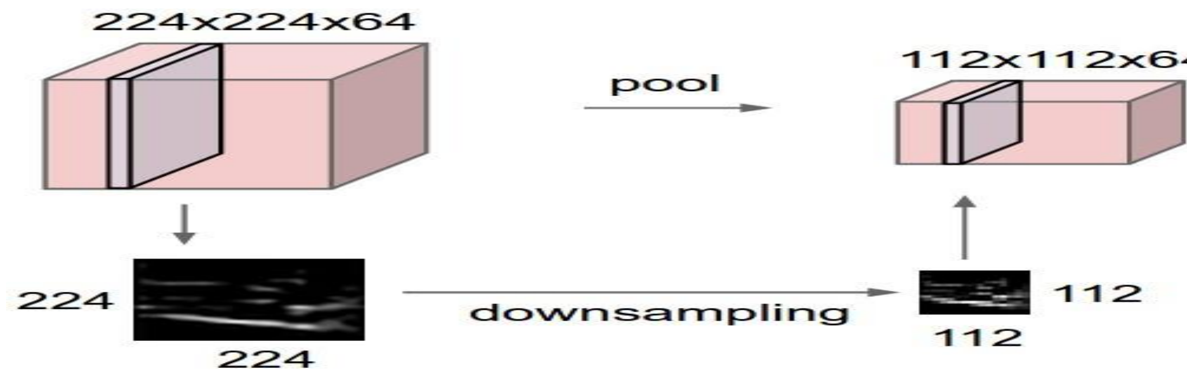


Figure 5.4: Example of pooling and down sampling

Getting rid of pooling:

In our examples we use pooling layer bacause we do not want our to lose our size of the image so there is another methos which we can use to just avoid using of pooling layer and the effect will be remain as same without using pooling layer. To discard the pooling layer in favour of architecture which consits of only convolution layers. To reduce the size of the image another method is introduced . Discarding pooling layers has also been found to be very important in training of good models.

Disadvantages of pooling:

Max-pooling outperforms the others alternatives because is the only one which is invariant to the special pad tokens that are appending to the shorter sentences known as padding. Actually, the combination of max-pooling and attentive pooling does not improve the performance as compared with the single max-pooling technique. Pooling is designed to forget about spatial structure, for example to recognize whether or not there is a bird an image, you don't care where the bird is. How a pixel is drawn on the screen? The process by which a computer transforms shapes stored in memory into actual objects to be drawn in the screen is known as rendering. And most well-known and used technique during rendering is called rasterization. The basic concept can be described as shown in the figure 4.1 where each pixel is drawn on the screen if there exists a trace of light (the arrow) that intersect one of the triangle shapes stored in the memory. This simplistic view also shows that if a triangle is further back than other surrounding triangles, then this should be reflected on the display and if a triangle is on top of another triangle, then the latter will be partly obscured by the former on the display. It is this approach what makes the foundation for 3D graphic support on computer screens. Stride convolution can be a useful alternative if positional information is important for the task at hand. Additionally, it's easier to reverse stride convolutions for things like convolution auto-encoders -- although it doesn't tend to matter that much if you do pooling one way and uniform up sampling in practice.

Finally, after several convolution and max pooling layers, the high level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. Pooling layer actually downs samples the volumes easily.

Convolution with GPU:

The GPU acronym stands for Graphics Processing Unit (or graphics processor), a term which was coined by NVIDIA in 1999 when it released its GeForce 256 “GPU”. Such a processing unit often describes a chip or a sub-unit of a larger chip called “system on chip” (SoC). GPUs were originally designed to accelerate graphics operations, thanks to specialized hardware which operates at a much faster rate than the general purpose processors called Central Processing Unit, or CPU. [5]

Graphics History:
How a pixel is drawn on the screen? The process by which a computer transforms shapes stored in memory into actual objects to be drawn in the screen is known as rendering. And most well-known and used technique during rendering is called rasterization. The basic concept can be described as shown in the figure 4.1 where each pixel is drawn on the screen if there exists a trace of light (the arrow) that intersect one of the triangle shapes stored in the memory. This simplistic view also shows that if a triangle is further back than other surrounding triangles, then this should be reflected on the display and if a triangle is on top of another triangle, then the latter will be partly obscured by the former on the

display. It is this approach what makes the foundation for 3D graphic support on computer screens.[8]

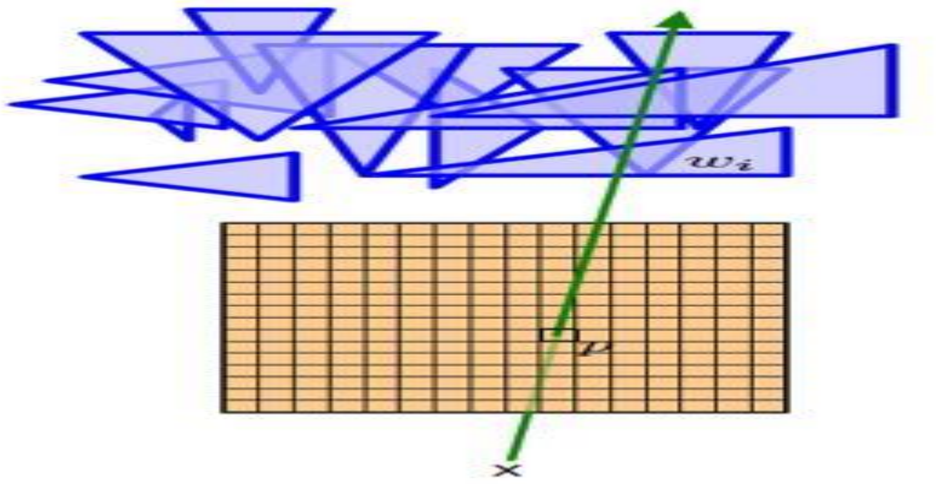


Fig 5.5: How pixels are drawn on display

Earlier GPUs in 1990s were offered with a fixed functional pipeline that was implemented directly via mainly two APIs: OpenGL and DirectX. This meant that GPUs were programmable with a low level set of operations that interacted directly with the hardware similar to assembly languages. At that time almost all computer graphics looked similar because they used either one of these two available APIs. OpenGL in particular was not a programming language but rather a vendor specification of how the hardware interface should be implemented. On the other hand, DirectX offered by Microsoft was essentially a library that allowed development of graphics in computer games. The pipeline then evolved into a more advanced level and shading capabilities were added into the APIs. The two main vendors who were manufacturing GPUs were NVIDIA and ATI. **[15]**

Reasons for GPU being faster:

On the outside, GPU looks just like another chip, but inside there is an array of dozens, hundreds or thousands of small computing units (or “GPU cores”) that work in parallel, and this is essentially why GPUs are so much faster than CPUs to compute images: hundreds of computing cores gets the job done faster than a handful of big cores. Computer graphics is fundamentally an “embarrassingly” parallel-program, which means that the workload can easily be spread across multiple compute units. Since each pixel on the screen can (mostly) be worked-on independently from one another, it is therefore possible to scale performance by adding more compute units. This is achieved by adding more compute units to a single chip, or by adding many chips into a computer with a multi-GPU setup. **[13].**

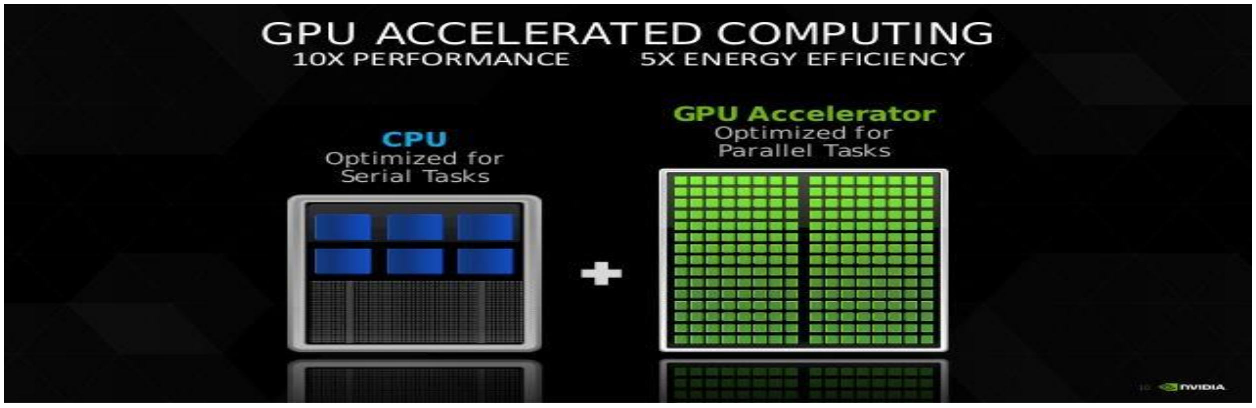

GPU ACCELERATED COMPUTING
10X PERFORMANCE
5X ENERGY EFFICIENCY
CPU
Optimized for Serial Tasks
GPU Accelerator
Optimized for Parallel Tasks
NVIDIA

Fig 5.6: Difference between the working of CPU and the GPU

Architecturally, the CPU is composed of just few cores with lots of cache memory that can handle a few software threads at a time. In contrast, a GPU is composed of hundreds of cores that can handle thousands of threads simultaneously. The ability of a GPU with 100+ cores to process thousands of threads can accelerate some software by 100x over a CPU alone. What’s more, the GPU achieves this acceleration while being more power and cost-efficient than a CPU. The figure 4.2 gives the difference between the working of CPU and the GPU. **[12]**

Convolution on color images (RGB images):

Convolution can be applied the same way as it is applied on to the gray scale image. An RGB or color image can be represented in computer memory as 3- dimensional matrix. Here in this the kernel should also be of size 3- dimensional matrix which is used to convolve with the input RGB image.[13]

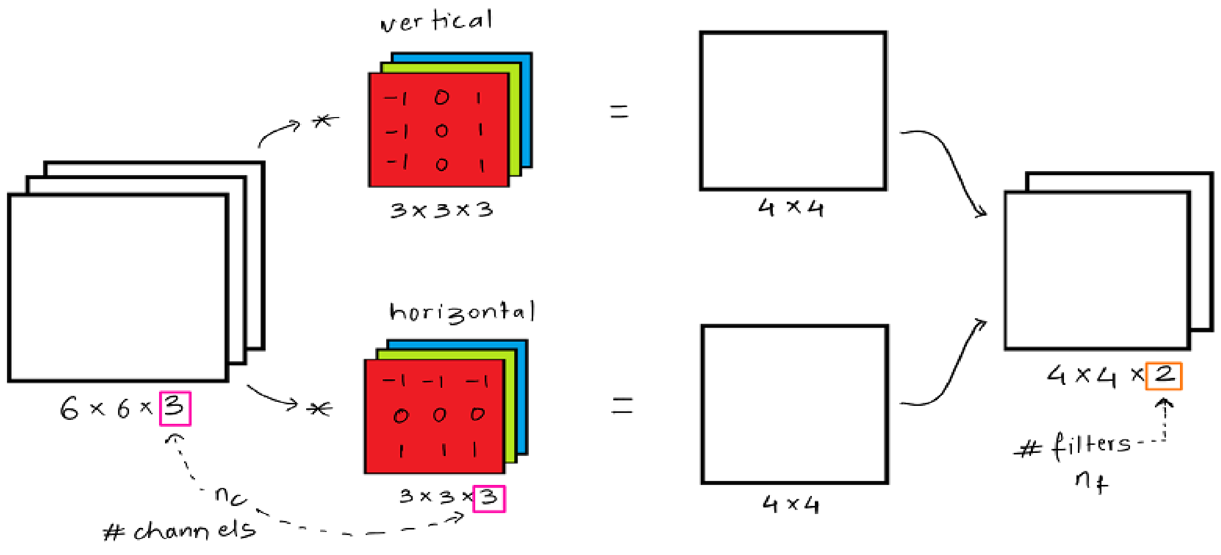


Figure 5.9: Convolution on RGB image

In the given figure the input image size is 6x6x3 where 6x6 is the size of input image while 3 is the number of channels denoted by n_c in the figure. The kernel or sobel should be of any number but their size should be k x k x n_c. After applying say m numbers of kernels each of size say k x k x n_c on input image of size say n x n x n_c, the resulting size of the image will be n x n x m. Here the hyper-parameters are kernels of size k x k x n_c , padding p and stride s. After getting the image as output we have to apply on to the image element wise RELU so that we get the final matrix as output. In CNN we are trying to learn the kernel/sobel matrix from the given dataset that is our main goal into the problem.[11]

CONVOLUTION NEURAL NETWORK ALGORITHM

CNN Model:

A simple CNN model can be seen in the given below figure. The first layer is input layer which contains the input size of 56 x 56. The second layer is also a convolution layer which we will get by applying another kernel on to the image. As we know we can apply as many as filters or kernel we can as per required. The next layer is pooling layer which also plays an important role to reduce the size of the input image. It also computes average or maximum or minimum of the input features. The convolution layer and pooling layer operate the same way. They only differ by the size and number of kernels. The output is fully connected with the pooling layer by using flattening operation.[4]

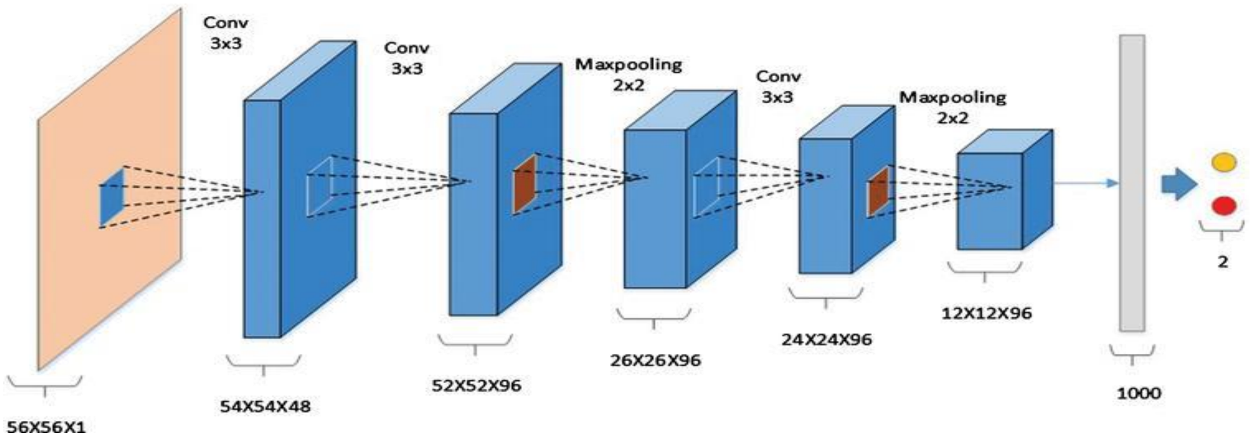
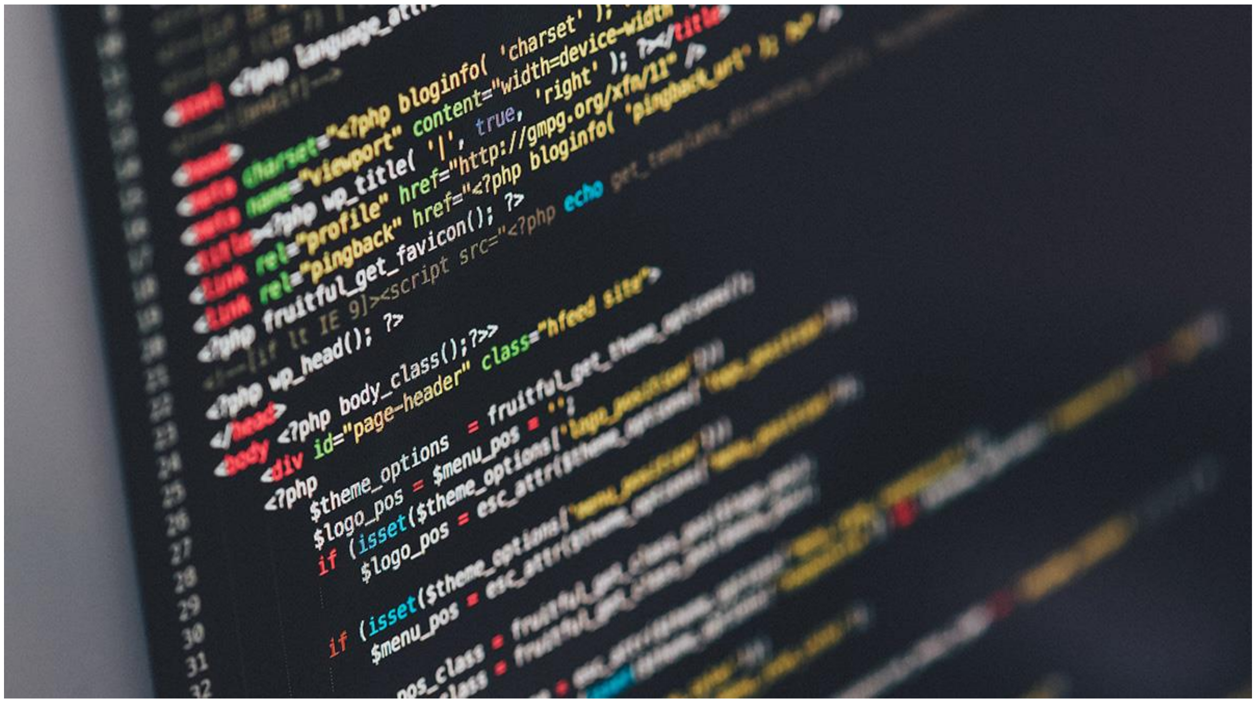


Figure 5.10: Convolution layer example

CHAPTER 6: IMPLEMENTATION DETAIL AND ANALYSIS



Chapter Grist: This chapter deals with the implementation of the thesis and result analysis.

Overview

In the implementation part of the multi-handwritten digit recognition we used keras library from the python which is very popular. Keras works same as tensorflow library but the implementation of the tensorflow is very complex while using keras library is very easy. Using keras for CNN is just importing some essential libraries. In our methods we have used keras mnist dataset which very popular and described clearly by[1].

Import dataset:

We have used keras library to import mnist dataset. This dataset is a black and white image of each digit where white as foreground and black as background. Some examples of the datasets are as follows:

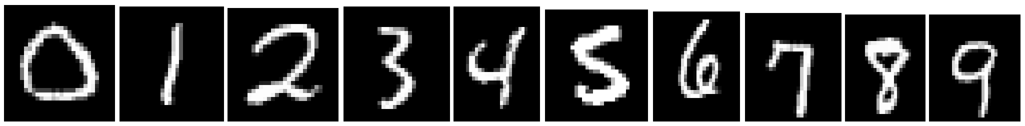


Figure 6.1: MNIST [1] dataset some examples

This dataset is used for the purpose of training of the model. Each image having size of 28 X 28 when this image is flattened into one dimensional array we get a one dimensional array of size 784. Below code is used for the purpose of loading mnist dataset from keras.[1]

```
from keras.datasets import mnist
```

Train and test dataset:

Since in our example we have used mnist dataset for train and test, the train dataset have 60,000 image each image of 784 dimensions (28 X 28). While test dataset have 10,000 examples each of 784 dimensions (28 X 28).[5]

Reshaping of images:

The next step is to reshape the image which we loaded earlier so that we can be sure that the image is in the size of 28 X 28. This is the size which our CNN algorithm accepts for input image. Finally we convert each value as float. By reshaping of our input image we can be sure that our input image and CNN input will match properly. Thereafter each value is divvied by a value 255 so that we can be sure that our input image values lie between zeros to one.[7]

One Hot Encoding of class-labels:

One hot encoding is very important when the output is categorical. It means we have ten class labels so we have here ten categorical data. Each categorical data is represented by 10 digits.[10]

Example: 0 in categorical is represented by as follows:

0	[0 0 0 0 0 0 0 0 0 0]
1	[0 1 0 0 0 0 0 0 0 0]
2	[0 0 1 0 0 0 0 0 0 0]
3	[0 0 0 1 0 0 0 0 0 0]
4	[0 0 0 0 1 0 0 0 0 0]
5	[0 0 0 0 0 1 0 0 0 0]
6	[0 0 0 0 0 0 1 0 0 0]
7	[0 0 0 0 0 0 0 1 0 0]
8	[0 0 0 0 0 0 0 0 1 0]
9	[0 0 0 0 0 0 0 0 0 1]

Create Model:

In keras creating our new model is very easy we have to just create the object of sequential class and try to add as many layers as we want. To solve multiclass classification problem we have used the popular VGG net architecture which is very popular for solving multiclass classification problem.

VGG neural network:

VGG neural network is very popular for detecting objects in the image. It can work for 1000 class classification problems very easily that’s why it is very popular nowadays.[13]

The model is created by using these criteria,

1. First 32 convolution layer each of size 5 X 5 .
2. Second a pooling layer of size 2 X 2.
3. Again 32 convolutions layer each of size 3 X 3 followed by relu as activation.
4. Again we use a pooling layer of size 2 X 2.
5. Next we use 20% connections as dropouts.
6. Next we do flattening of the whole data.
7. In next step we apply a dense layer of size 128 followed by relu activation.
8. In next step we use 10 as number of classes followed by softmax activation.

Compile the model:
In compiling phase we use our loss function as categorical cross entropy which gives best result in case of multiclass classification problems. For optimizing the loss we use adam as optimizer which is the modified version of the Gradient Descent algorithm. For the purpose for accuracy we use metrics as accuracy.

Fitting the model:
In this step we fit our model. Fitting is all about giving training data to the model and after getting 200 images we calculate the error get and back propagate this error to the entire network so that our network can update its weights very accurately. Here the number 200 is called batch size and this all process is repeated 10 times which is called epochs. From the first epoch we get a validation accuracy of almost 97.91% which is very good model. Finally we get an accuracy of 99.31% on test (unseen) data.

Image segmentation:
In this module we have taken as input a black and white multi handwritten digit image which has black as background and white as foreground. We have identified each digit from the image and cropped that image and then resized it into 28 X 28 and then it is finally fed into our model which can detect the desired digit with an accuracy of 99.31%.[15]

Steps of segmenting the multi handwritten digit image:

Loading image and remove noise

In this an image is taken as input which is created on paints and looks like it has black as background and white as foreground.

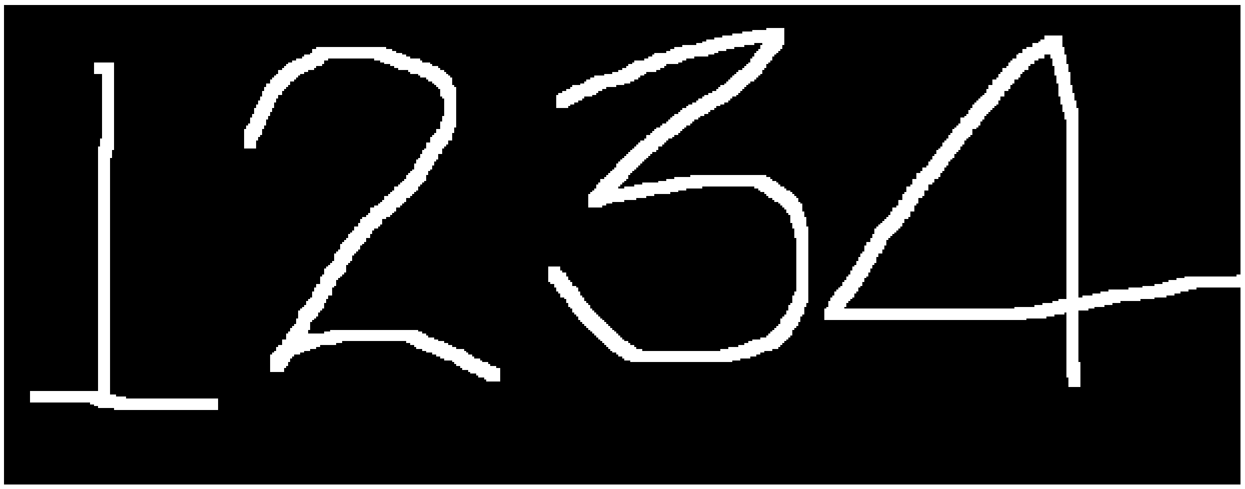


Figure 6.2: This image is input to our segmentation algorithms.

Threshold the image

In the next step we threshold the input image using the threshold function of opencv class in python. We use here THRESH_BINARY function which considers the white part of the image and we also use a threshold value of 127.

Dilate the white portions

In the next step we use dilation to increase the white portion of the image to clearly identify the digits. [18]

Find contours in the image and take ROI

In this step we find contours in the image and take ROI (Region of interest) within the image and taking a rectangular shape and crop the image and save it. After this we can apply reshaping to each image to get a size of 28 x 28 because our CNN accepts only these dimensions as input.

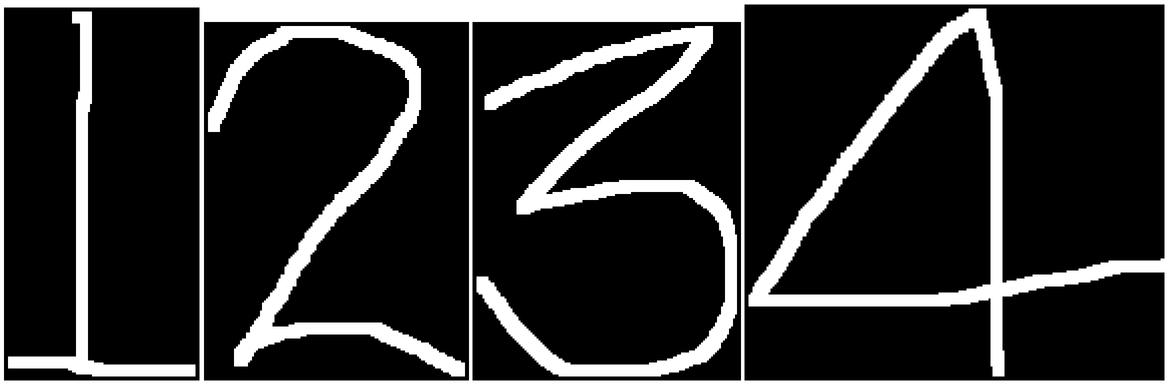


Figure 6.3: After segmenting of the image

Recognition of the digits:

This is the final step of recognizing the each individual digit. We first take input of the image then convert this image into grey scale. In next step we have to reshape into size 28 X 28 and convert into one dimensional array and finally it is fed to our model to predict the actual digit. It is basically a digit recognition task where we need to recognize an individual digit. As such there are 10 classes are there to predict. From this result we can predict error of prediction of a particular digit.

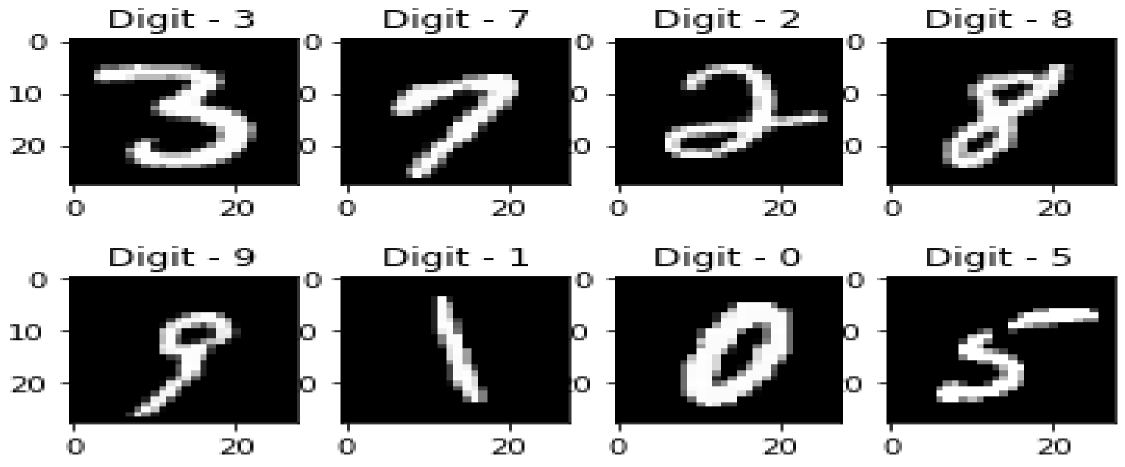


Figure 6.4: Recognizing the handwritten digits

Experimental Results:

Some examples of multi handwritten digit have taken here from the different users and segmentation results are shown in the examples. The images are made from Microsoft paint using black as the background of the image and white as the foreground of the image. The images can be created either as jpg or png file. We here created png file of the image and results will be shown after the segmentation of the image. After doing segmentation of the image we need to resize the image into 28 X 28 which our CNN accept.[16]

Example 1:

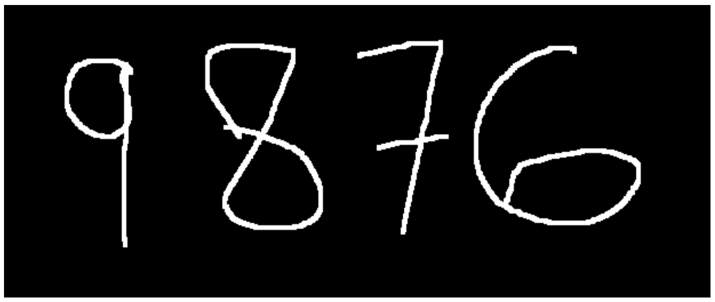


Figure: 6.5

After applying segmentation:

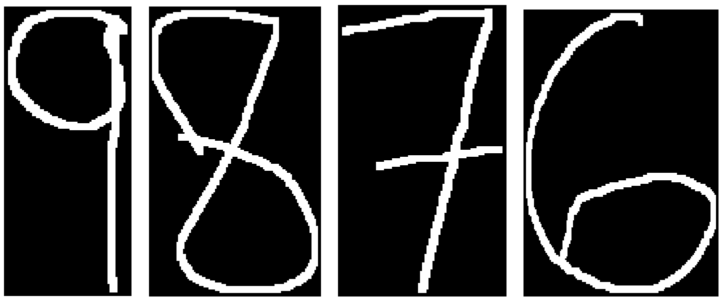


Figure: 6.6

Example 2:

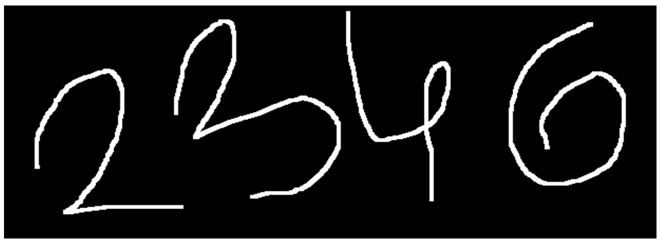


Figure: 6.7

After applying segmentation:

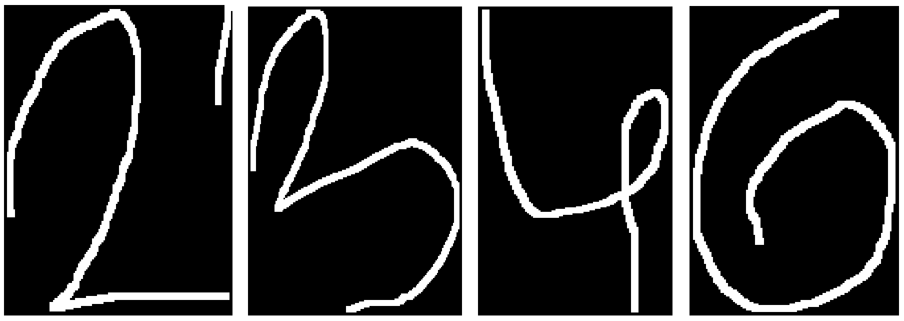


Figure: 6.8

Example 3:

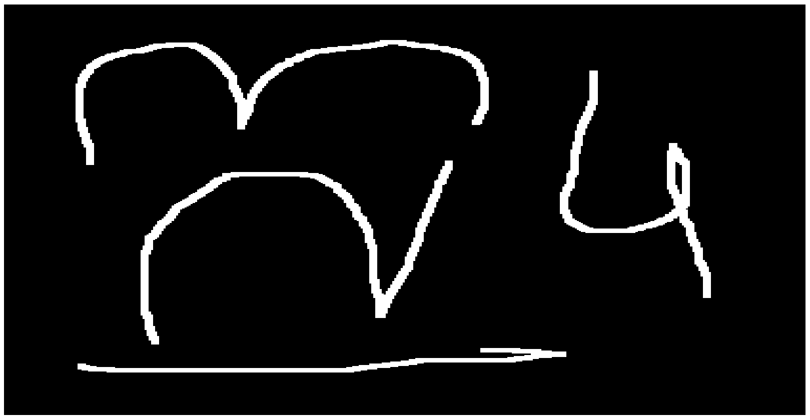


Figure: 6.9

After applying segmentation:

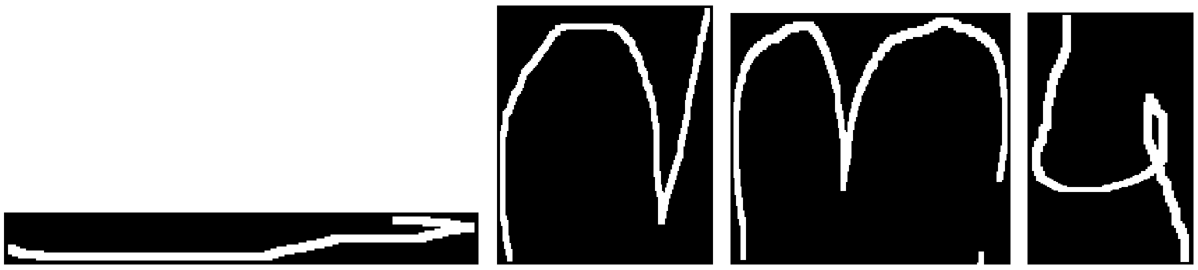


Figure: 6.10

Example 4:

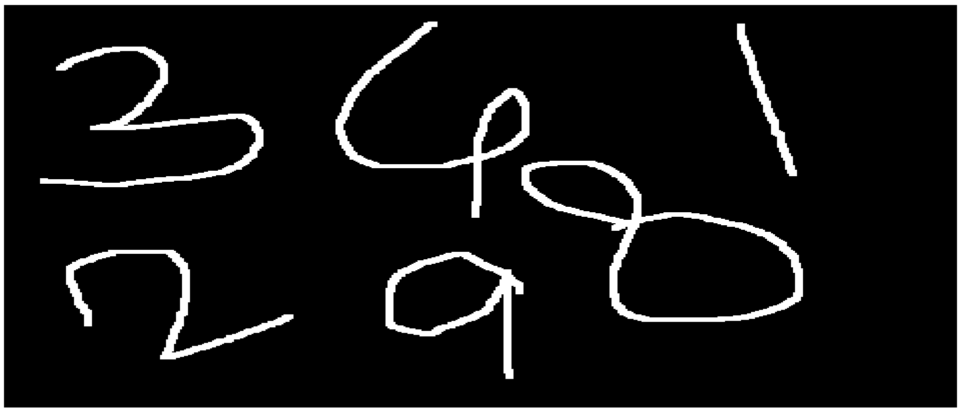


Figure: 6.11

After applying segmentation:

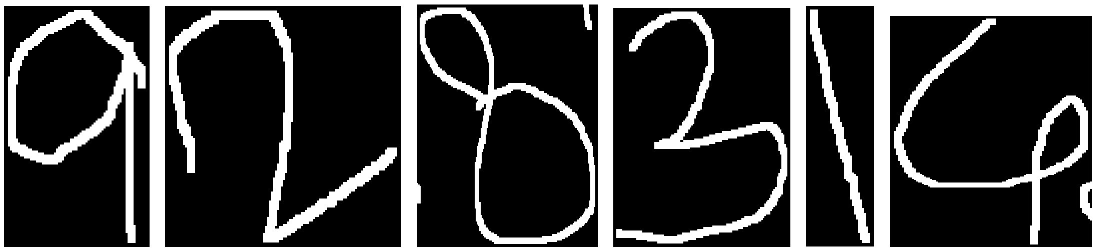


Figure: 6.12

Image Recognition of segmented digits:

After segmenting these images we need to resize these images using online image resize without losing the image's original feature. For the purpose of resize we have used here [10] which is an online site for doing resize of the image. There are the following basic steps to recognize the segmented digits.

1. Convert image into gray scale functions.
2. Convert image into Numpy array and resize into 28 by 28.
3. Again reshape this image.
4. Now the digit is predicted using predict classes function.

Performance analysis:

For the performance analysis, we have taken a set of pre-processed image of the different multiple digits containing 9876, 2346, 1234, 324981. Then we feed these images to our segmentation algorithms and result has been shown in the above figures 6.5 illustrates the segmentation of multi digits for simple orientation. Also figure 6.12 illustrates a very difficult orientation to observe but algorithm detects it more accurately.

We measure here the pre-processed processing time for our CNN algorithm because of the fact that neural networks can do better in GPU environment. The CNN stage runtime is directly proportional to the number of digits in the frame. Figure 6.10 also shows the performance of the segmentation for the digit 1234 which is very difficult to segment individually.

CONCLUSIONS AND FUTURE WORK

In this paper we developed a simple convolution neural network which based upon multiclass classification problems. We have found that softmax is the multiclass classifier which works better and the best optimizer algorithm is adam. Our simple convolution neural network has low computational cost. For the purpose of training of our CNN model we use ReLU and 20% dropout. Convolution neural network has developed in such way that it can extract the desired features from the image and try to learn from those important features. We also observed that shallow network has a good recognition effect in comparison with the complex network. We have developed a system which can segment a multi handwritten digit and recognize each individual digit. The model is trained on MNIST [1] dataset which has 60,000 as training data. The proposed model is a sub part of the most complex system where our main aim is to analyze images of checks to extract and recognize important information [20]. The system can also be used in traffic system where we want to recognize the number plate of a vehicle. In this paper we applied deep learning algorithm to the real world problem of recognizing handwritten digits. Each individual human has different methods to write a single digit so it becomes very important in case of handwritten digit recognition since preprocessed images of digits are very easy to recognize by the model [18].

The future scope of includes a complex model which can detect a string of characters which comprises of alphabets and numeric digits. This can also be extended to system where text strings are used like bank checks. Another future work related to this is read characters from the book and translated to audio. In self-driving car this is very useful in reading optical characters from the street signal board which can tell the car to understand the roadside very efficiently. In future it is also modified to improve the feature extraction from the images and represent efficiently. In future we will study the optimization techniques of deep learning like improving the adam algorithm and apply it on more complex recognition problems [13]. It can also be used for a real time classifier where the user can take input and the application immediately recognizes the digit efficiently [3].

REFERENCES:

[1] LeCun, Y., C. Cortes, and C.J. Burges, The MNIST database of handwritten digits. 1998.

[2] LeCun, Y., Y. Bengio, and G. Hinton, Deep learning. Nature, 2015. 521(7553): p. 436-444

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradientbased learning applied to document recognition. In Proceedings of the IEEE., 1998.

[4]P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In Proceedings of the Seventh International Conference on Document Analysis and Recognition, volume 2, pages 958–962. Institute of Electrical and Electronics Engineers, Inc., August 2003.

[5]M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In Computer Vision and Pattern Recognition Conference (CVPR07), San Mateo, CA., 2007.

[6] Y. Tang, “Deep learning using linear support vector machines,” arXiv prprint arXiv:1306.0239, 2013.

[7] Ciregan,D.,U.Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. 2012. IEEE.

[8] R. Saabni. Ada-boosting extreme learning machines for handwritten digit and digit strings recognition. In Digital Information Processing and Communications (ICDIPC), 2015 Fifth International Conference on, pages 231–236, Oct 2015.

[9] Lee, C.-Y., P.W. Gallagher, and Z. Tu. Generalizing pooling functions in Convolutionalal neural networks: Mixed, gated, and tree. In International Conference on Artificial Intelligence and Statistics. 2016.

[10] Krizhevsky A, Sutskever I, Hinton G E. ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in Neural Information Processing Systems, 2012, 25(2):2012.

[11] https://en.wikipedia.org/wiki/MNIST_database

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

[13] https://www.kaggle.com/c/digit-recognizer

[10] http://yann.lecun.com/exdb/mnist/

[14]https://world4jason.gitbooks.io/researchlog/content/deepLearning/CNN/Model%20&%20ImgNet/lenet.html

[15] http://cs231n.github.io/convolutional-networks/#pool

[16]https://stackoverflow.com/questions/47273372/charactersegmentation-and-recognition-for-unevenly-spaced-digits

[17] Lee, C.-Y., P.W. Gallagher, and Z. Tu. Generalizing pooling functions in Convolutionalal neural networks: Mixed, gated, and tree. in International Conference on Artificial Intelligence and Statistics. 2016.

[18] MDig: Multi-digit Recognition using Convolution neural Network on Mobile by Xuan Yang nad Jing Pu Stanford University.

[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014

[20] Recognizing handwritten single digits and digits strings using deep architecture of neural networks. Raid Saabni ISBN: 978-1-4673-9187-0 ©2016 IEEE.