CNN BASED APPROACH FOR HANDWRITING RECOGNITION USING TENSORFLOW

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF DEGREE
OF

MASTER OF TECHNOLOGY
IN
SOFTWARE ENGINEERING

Submitted by:

UTKARSH GUPTA
(2K16/SWE/19)

Under the supervision of:

Prof. RAJNI JINDAL
(HOD CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road , Delhi – 110042

JUNE 2018

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

CANDIDATE'S DECLARATION

I, UTKARSH GUPTA, 2K16/SWE/19 a student of M.TECH (Software Engineering) declare that the project Dissertation titled "CNN Based Approach For Handwriting Recognition Using Tensorflow" which is submitted by me to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Fellowship or other similar title or recognition.

Place: DTU, Delhi UTKARSH GUPTA

Date: (2K16/SWE/19)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

<u>CERTIFICATE</u>

I, hereby certify that the Project titled "CNN Based Approach For Handwriting Recognition Using Tensorflow" submitted By UTKARSH GUPTA, Roll number: 2K16/SWE/19, Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi **Prof. RAJNI JINDAL**

(HOD CSE)

Date: SUPERVISOR

<u>ACKNOWLEDGEMENT</u>

I am very thankful to Dr. Rajni Jindal (Head of the Department, Computer Science Eng. Dept.) and all the faculty members of the Computer Science Engineering Dept. of DTU. They all provided immense support and guidance for the completion of the project undertaken by me. It is with their supervision that this work came into existence.

I would also like to express my gratitude to the university for providing the laboratories, infrastructure, test facilities and environment which allowed me to work without any obstructions.

I would also like to appreciate the support provided by our lab assistants, seniors and peer group who aided me with all the knowledge they had regarding various topics.

UTKARSH GUPTA

M.TECH (SWE)

2K16/SWE/19

ABSTRACT

Handwriting Chinese Character Recognition (HCCR) has been area of research for over a decade. Because of its wide range of characters, similarities and complexities it is difficult to classify them as compared to other languages or numerical representations. It is very important to recognize handwriting as we are moving towards the automated world. It is a field of machine learning or more specifically the deep learning. Deep learning is showing great results in the fields of visual and speech recognitions. But still we are not able to match the accuracy of human vision and advancements will be continuing till we reach or cross that limit. It is also very significant today as we want to replace human beings with machines in every field one such example could be automatic cars. But for all this we need high accuracy because if there will be any error there could be a disaster.

In this thesis we are using deep learning techniques to improve the accuracy of HCCR. We are modifying the existing model to a more deeper and slimmer one i.e. having less number of overall parameters. We are evaluating our proposed model on ICDAR (International Conference on Document Analysis and Research) test dataset (HWDB1.1). There are many applications of reading human writing in the real world like it can be used in banks or post offices to read the cheques and envelops. We can directly submit them to machine and machine can do the specific tasks. This is our attempt to close the gap between machine and human vision.

CONTENTS

CANDIDATE'S DECLARATION **i**

CERTIFICATE **ii**

ACKNOWLEDGEMENT **iii**

ABSTRACT **iv**

CONTENTS **v**

LIST OF TABLES **vii**

LIST OF FIGURES **viii**

LIST OF ABBREVIATIONS **ix**

CHAPTER 1 INRODUCTION **1**

1.1 Literature Survey 3

1.2 Objective 5

1.3 Organization 5

CHAPTER 2 TECHNIQUES AND TERMINOLOGIES **6**

2.1 Convolutional Neural Networks 6

2.2 Convolution 8

2.3 Feature Detector 9

2.4 Rectified Linear Unit 10

2.5 Pooling Layer 13

2.6 Flattening 14

2.7 Full Connection 14

2.8 Softmax Function 15

2.9 Cross Entropy 16

2.10 KL Divergence 17

CHAPTER 3 IMPLEMENTATION DETAILS 18

3.1 Dataset Used 18

3.2 Backend Used 19

3.3 Model Fusion 20

3.4 IDE Used 21

3.5 Libraries Used 21

3.6 Mathematical Overview 22

3.7 Stochastic Gradient Descent 22

CHAPTER 4 RESULTS AND EVALUATION 25

4.1 Previous Models 25

4.1.1 GoogLeNet 25

4.1.2 VGG16 Net 26

4.1.3 AlexNet 27

4.2 Proposed Model 27

4.2.1 Parameterized Rectified Linear Unit 28

4.2.2 Batch Normalization 29

4.3.3 Model Summary 30

4.3 Result and Comparison 33

4.1.1 Result 33

4.1.2 Comparison 33

CHAPTER 5 CONCLUTION AND FUTURE WORK 35

5.1 Conclusion 35

5.2 Future Work 35

CHAPTER 6 REFERANCES 36

LIST OF TABLES

4.1 GoogLeNet Model	**26**
4.2 Summary of the Model	**33**
4.3 Comparison with other Models	**34**

LIST OF FIGURES

1.1 Parts of Human Brain	1
1.2 Similar Chinese Characters	2
2.1 Importance of Features in Classification	6
2.2 CNN Steps	7
2.3 CNN Example Model	7
2.4 Convolution Function	8
2.5 Feature Maps	9
2.6 ReLu Representation	10
2.7 Input Image Example	11
2.8 Image After Applying Feature Map	12
2.9 Image After Applying ReLu	12
2.10 Max Pooling	13
2.11 Flattening	14
2.12 Full Connection	15
3.1 TensorFlow Model	20
3.2 Gradient Descent Representation	24
4.1 VGG16 Net	27
4.2 AlexNet	28

4.3 Leaky ReLu Representation	29
4.4 Batch Normalization Algorithm	30
4.5 Accuracy Chart	35

LIST OF ABBREVIATIONS

HCR	Handwriting Character Recognition
CNN	Convolutional Neural Network
AHR	Automated Handwriting Recognition
HCCR	Handwritten Chinese Character Recognition
NLPR	National Laboratory of Pattern Recognision
ICDAR	International Conference on Document Analysis And Research
CASIA	Chinese Academy of Sciences Institute of Automation
MINST	Modified National Institute of Standards and Technology database
VGG	Visual Geometry Group
ReLu	Rectified Linear Unit
ANN	Artificial Neural Network
SGD	Stochastic Gradient Descent
PReLu	Parameterized Rectified Linear Unit
SVM	Support Vector Machine
TPU	Tensor Processing Unit

CHAPTER 1 INTRODUCTION

Handwriting Character Recognition (HCR) is a technique to correctly classify the written text into one of the classification classes. Complexity of classification increases as the number of classes increases. So, one of the ways to decrease the complexity is to reduce the domain but that is not feasible in real life scenarios. Nowadays no other technique is better than convolutional neural network (CNN) for image classification. And now more and more researches are going on in this field. As with the research accuracies are also improving.

We try to map the functionality of human brain through neural network and we need to do lot more to exactly mimic the human brain. Different parts of brain have different functionality. Similarly we have different types of networks for different functionalities. Artificial Neural Networks are like Temporal Lobe in human brain and Recurrent Neural Networks are like Frontal Lobe where as Convolutional Neural Networks are implemented to perform the similar functions as of Occipital Lobe which is visual processing center of mammalian brain.

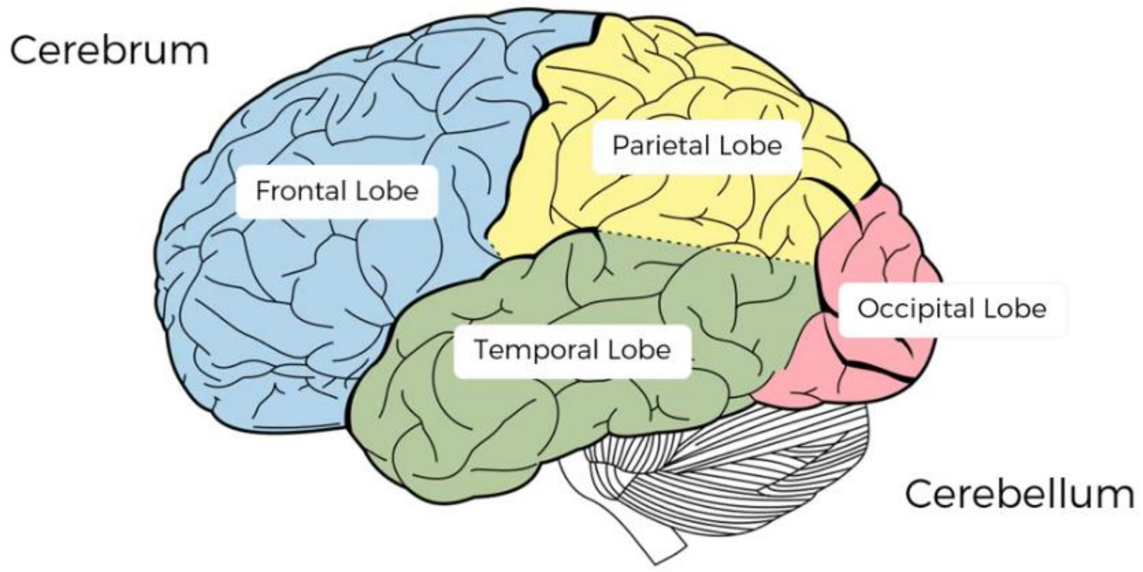


Figure 1.1 Parts of Human Brain

Image source: wikipedia.org

Automated Handwriting Recognition (AHR) is very important as we are moving very fast towards the automation of tasks. There will be more interaction between machines then the human and machine. In such environments it is very crucial that our machines can recognize handwritings. There are lots of scenarios in which we can use this technique like recognizing handwritten data on the bank checks or postal address on envelops. If we want machine to be fully automated then it must learn to recognize the handwritten data.

We can generalize HCR into two categories one is online handwriting recognition and other one is of offline. Online generally contains the recognition of handwriting when it is wrriten on digitizers, where sensor can detect the movement of pen or stylus tip as we seen in note series of Samsung galaxy phones. Offline recognition is more tedious job as it is conversion of text in an image into a letter code. The complexity and accuracy is highly dependent on language and image quality.

As compared to digits and letters in latin alphabets, Handwritten Chinese Character Recognition (HCCR) is more complex and challenging because of the two main reasons. First, Chinese language has thousands of characters of wide complexities where as English has only 26 characters. Second, most of the characters have more complex structure and consists of much more strokes then Arabic or English characters. Moreover, degree of similarities between characters is also high in Chinese language as illustrated in figure 1.2.

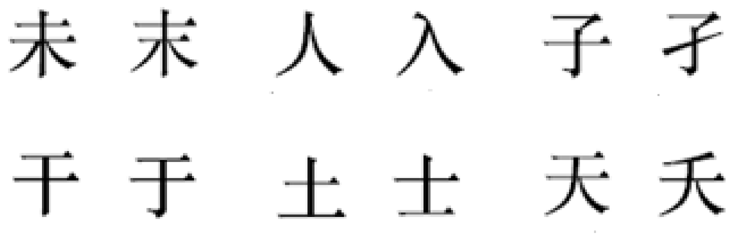


Figure 1.2 Similar Chinese Characters

1.1 LITERATURE SURVEY

According to the paper by Geoffery Hinton [1] on deep learning, Deep learning is a method that allows computational models which contains many processing layers to learn the different features of data with multiple levels of abstraction. These methods have been significantly improved the state-of-the-art in various domains such as speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning identifies complex structures in large amount of processing data by utilizing the backpropagation calculation to demonstrate how a machine should update its internal parameters that are utilized to register the representations in each layer from the representations in the past layer. Deep convolutional systems have achieved leaps forward in processing images, video, speech and audio, while recurrent systems have bought breakthrough in successive information.

Handwriting recognition is an undertaking of mapping a dialect spoke to in its spatial form of matrix into its emblematic representation. Throughout the most recent five decades [2], handwritten Chinese character recognition (HCCR) has pulled in impressive consideration from scientists and has widely been examined in view of its expansive number of character classes, likeness amongst characters, and variety in composing style. Handwriting recognition can mainly be specialized into online and offline handwriting recognition. In comparison to offline HCCR, in which gray-scale images are analyzed and classified into different groups, for online HCCR, movement of pen is the main source of information to recognize different characters [3]. Moreover, online HCCR finds numerous applications in pen input devices, personal digital assistants, smart phones, touch-screen devices, etc.

CNNs were proposed in the 1990s by Yan Lecunet [6]. The experiments in handwritten digit recognition have gained great success. About 99% classification accuracy rate was achieved on the MINST (Modified National Institute of Standards and Technology database) dataset. Convolutional Neural Networks also outperform the traditional machine learning methodology in handwritten Chinese characters recognition. Multi-scale gradient and deep neural network was proposed in 2015 by Weishen Pan, which could effectively recognize offline Chinese characters

[7]. And the CNN trained with the combination of traditional eight-direction features could effectively recognize online Chinese characters, proposed by Xin Liu [8].

HCCR has been a theme of talk in design acknowledgment before convolutional systems turned out to be generally well known, in spite of the fact that there were instances of CNN utilization in HCCR dating back to 1993. In [4], the authors used a LeNet to group a subset of Chinese vocabulary, accomplishing test accuracy of 94.2%. Actually, the constrained size of the character set was because of restricted computational assets, as even today training and testing a convolutional neural network is require much processing power.

The first use of CNN for HCCR in a large dataset was shown by Ciresan [9], with a multi-column deep neural network to classify the characters from the CASIA dataset, among other tasks in image recognition (*e.g.* traffic signs recognition). On Offline HCCR, the authors obtained an error rate of 6.5%.

The NLPR (National Library of pattern recognition) additionally held many competitions for Chinese Handwriting Recognition, the most noticeable being from ICDAR-2013 (International Conference on Document Analysis And Research). In disconnected acknowledgment, the group from Fujitsu Focus took the prize with a 4-CNN voting framework. The CNN themselves were made out of ten convolutional layers and two completely associated layers, prepared in the HWDB1.1 dataset. The ICDAR 2013 [11] dataset is utilized these days to prepare and assess HCCR arrangements.

Cheng.[10] proposed a deep triplet network (DTN) method, which learns a CNN model using both classification and similarity of the characters as supervision. It maximizes the inter-class variations, minimizes the intra-class variations, and which in turn minimizes the cross-entropy loss. The work by Zhong [11] uses directional feature maps along with an ensemble of AlexNet and GoogLeNet [15] architectures. The paper reports classification rate of 96.29% accuracy in the ICDAR 2013 offline test set.

1.2 OBJECTIVE

As its breakthrough in resolving many computer vision problems, the convolutional neural networks (CNN) provided new end-to-end method to classify handwritten Chinese character recognition (HCCR) with very good results in recent years. We are also using a very deep model but with very less number of parameters as compared to its predecessor i.e. it is more slim. We have shown in thesis work that, adding more layers to the model can profit HCCR a great deal to accomplish higher performance and furthermore can be outlined with less number of parameters. We likewise demonstrate that the customary feature extraction techniques are as yet helpful for improving the execution of CNN. We proposed a modified version of VGG net (Visual Geometry Group), which was initially proposed for image classification lately with very deep architecture as compared to previous ones. The VGG network we used is 16 layers deep but involves with only 13 million parameters. We also tried fusion of our model with SVM (Support Vector Machine). Testing on model is conducted using the ICDAR 2013 offline HCCR competition dataset. It has been shown in our work that with the proper incorporation of layers, the proposed single and ensemble VGG16 model can achieve new state of the art recognition accuracy, outperforming previous best result with significant gap.

1.3 ORGANISATION

This thesis is divided into various chapters and each chapter is described accordingly.

- The starting and this chapter of the thesis give the general overview and describe the problem and related goals.
- The second chapter presents the brief review of the research terminologies and various techniques which are used in our model of HCCR.
- In third chapter whole process used in the implementation of the HCCR and dataset used is described.
- The results of our model of HCCR have been shown in fourth chapter of this thesis and also the result is compared with other existing models.
- The fifth part concludes the model and explains the future work which can be done to improve the system further.
- The last and sixth chapter contains all the references which are helpful in our work.

CHAPTER 2 TECHNIQUES AND TERMINOLOGIES

This chapter will comprise of the explanation of techniques and terms which are used as a building blocks for our recognition model and other related concepts.

2.1 CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional Neural Networks (CNNs) are like other artificial neural networks have a structure of neurons which are more focused on visual recognition. In images we have features and according to features we classify them. Like in below example.

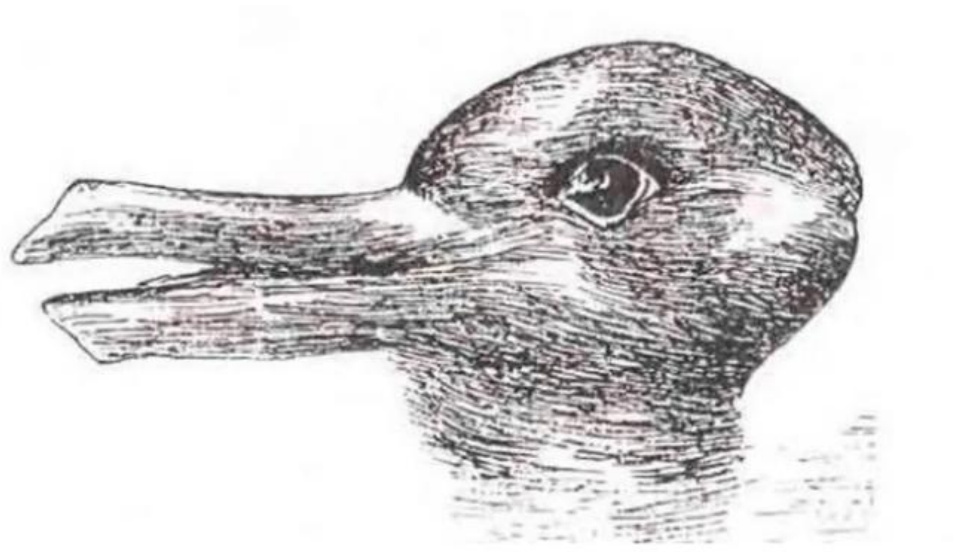


Figure 2.1 Importance of Features in Classification

Image source: mathworld.wolfram.com

In the picture above, if we focus on the features of left side then our brain will classify it as a duck and if we focus more on the features that are on right side then it will be classified as a rabbit. So it's all depends on the features on which we are focusing on in our test image.

STEPS INVOLED IN CONVOLUTIONAL NEURAL NETWORK

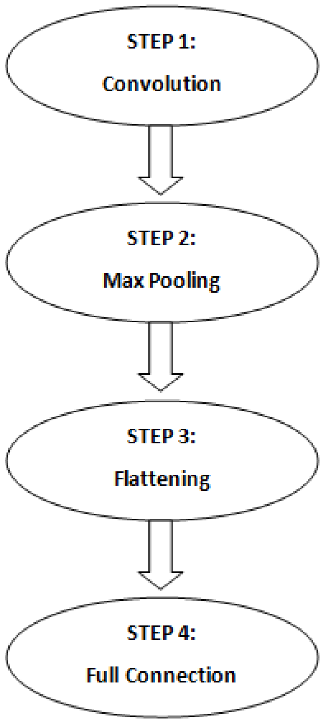


Figure 2.2 CNN Steps

Below figure shows the step by step process of classification of input image by CNN. And the following subsections present the detailed overview of the steps mentioned in above figure.

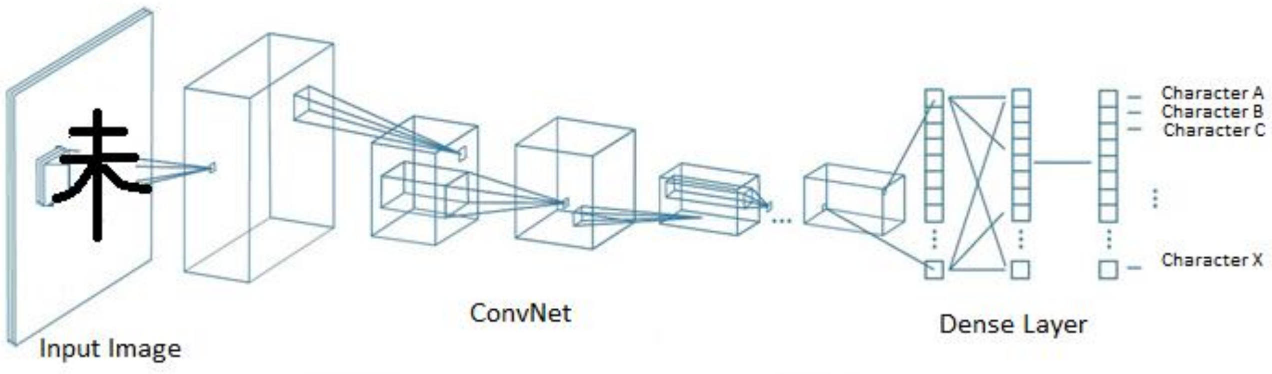


Figure 2.3 CNN Example Model

2.2 CONVOLUTION

In arithmetic, convolution is a scientific task on two functions (f and g) to deliver a third function, that is commonly seen as a changed rendition of one of the first functions, giving the fundamental of the point wise increase of the two functions as an element of the sum that one of the first functions is deciphered. Mathematical expression of convolution is

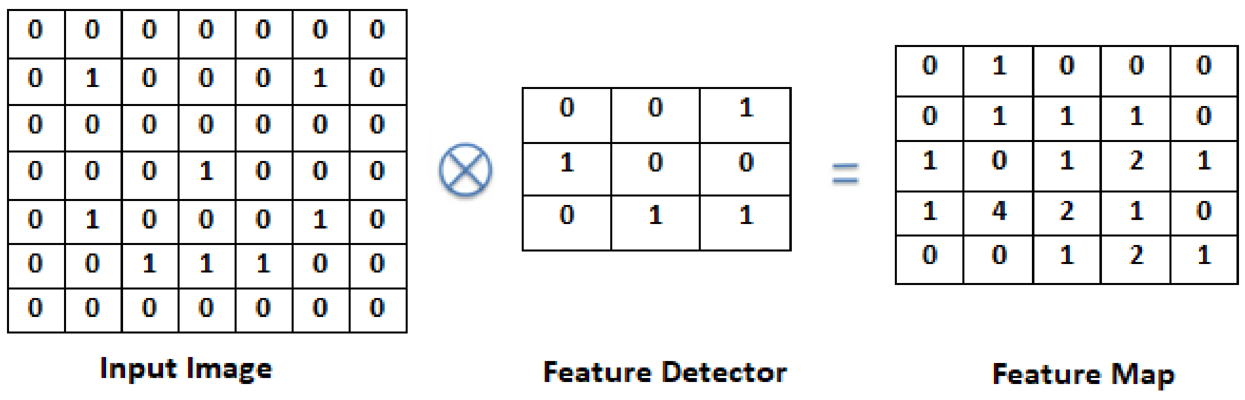
$$(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau \qquad (1)$$

It is basically a combined integration of two functions and represents how one function modifies the shape of the other function. This will be more clear in the following steps. In the area of matrices and arrays it is more like the multiplication of two matrices. We can imagine the input image as a function f which is changed by convolution with the other matrix. That matrix is called feature detector or kernel which is described in the next section. The visualization of convolution in matrix is given in figure 2.4 in which we are using feature detector of size 3 X 3.

Convolution is the very first and most crucial step in CNN. As CNN comes under the field of deep learning, we generally have more than one convolution layer of different number of neurons. Generally every convolution layer is succeeded by or associated with other layers like pooling or rectified linear function. These associated layers further enhance the model capability of classification with the help of functionality they are adding. Other important techniques and layers are discussed in the coming sections.

2.3 FEATURE DETECTOR OR KERNAL OR FILTER

Feature Detector is the function in convolution equation according to which main function changes its shape. In convolutional neural networks we have images as the main function and we apply feature detector upon them. We get feature map as an output of convolution between image matix and feature map. Generally after applying the feature map the size of image metrics reduces. They are of varying sizes but in square matrices form like alexnet uses filter of 7 X 7.



$$\begin{bmatrix} 0&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&1&0&0&0&1&0 \\ 0&0&1&1&1&0&0 \\ 0&0&0&0&0&0&0 \end{bmatrix} \otimes \begin{bmatrix} 0&0&1 \\ 1&0&0 \\ 0&1&1 \end{bmatrix} = \begin{bmatrix} 0&1&0&0&0 \\ 0&1&1&1&0 \\ 1&0&1&2&1 \\ 1&4&2&1&0 \\ 0&0&1&2&1 \end{bmatrix}$$

Figure 2.4 Convolution Function

In the above figure as described in the previous section we are using 3 X 3 square matrix as feature detector. Feature Map which is the output of convolution preserved the features of input image. As we can see in row and column [2, 2] the value is four because all the values of feature detector are matched.

The purpose of the feature detector to detect features in image or the parts of the images that is integral. So we have lots of feature detectors for detecting the features in our image. And all those outputs of convolution will form the very first layer of out convolutional neural network. Like described in the following figure.

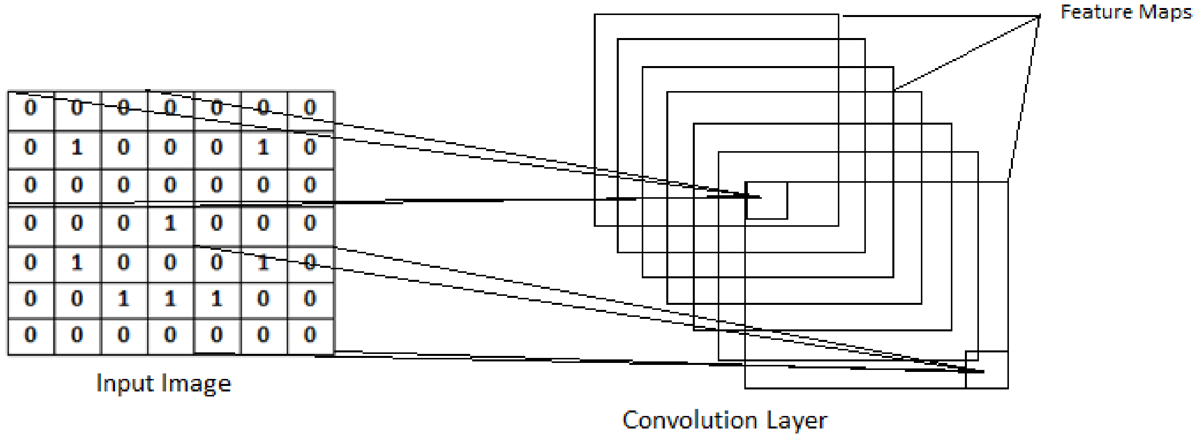


Figure 2.5 Feature Maps

2.4 ReLu (Rectified linear unit) layer

ReLu function is applied to truncate each and every input cell. So it does not change the size of input. We use this additional layer to increase the non linearity in our model. ReLu function is also not linear which is shown in figure 2.4. There are many other functions also used for increasing the non linearity like sigmoid or leaky ReLu (modified version of ReLu).

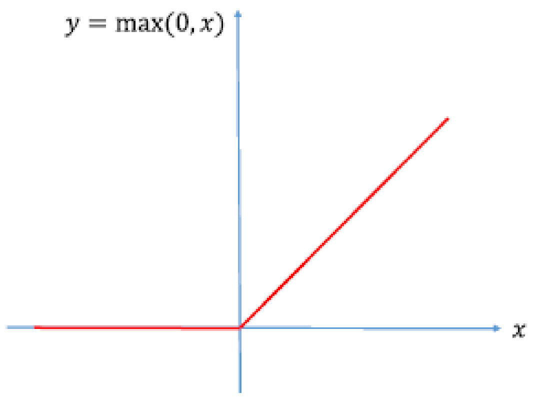


Figure 2.6 ReLu Reprasentation

Let us take an image to describe this step in detail. Below is the input image.

Figure 2.7 Input Image Example

Image source: mlss.tuebingen.mpg.de

Now we give this image as an input to the CNN model then there will be several feature detectors applied on this image. Let us have an output of this image after applying a feature detector. As we can see in figure 2.6 there are certain linearities of white grey and black areas. We need to remove these for better system performance.

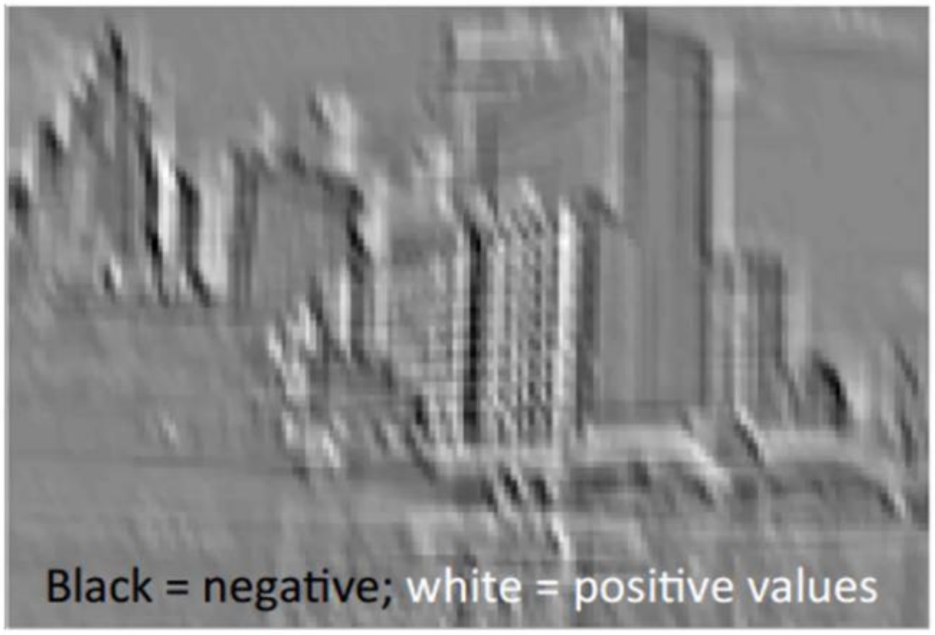


Figure 2.8 Image After Applying Feature Map

We will apply our non-linear activation function (ReLU) on the output image of feature detector to clip the non linearity in the input image. The output image after applying ReLU layer is shown in figure 2.7.

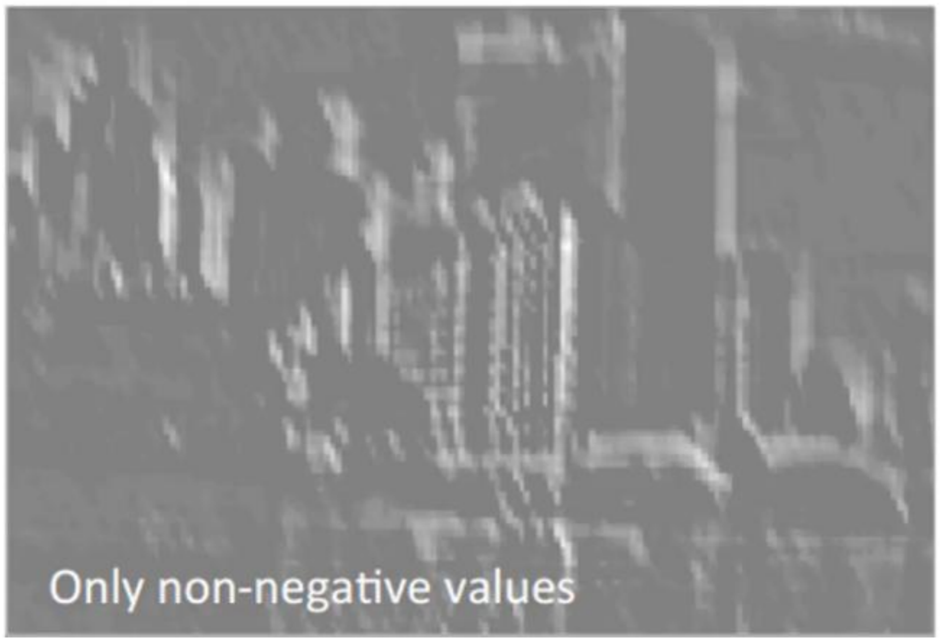


Figure 2.9 Image After Applying ReLu

2.5 POOLING LAYER

Pooling is a technique in which we preserve the features and reduce the size of feature map. Let suppose x^l (be the input layer to the l-th layer, which suppose pooling layer) belongs to

$$\mathbb{R}^{H^l \times W^l \times D^l} \tag{2}$$

then output of pooling will be size with

$$H^{l+1} = \frac{H^l}{H}, \quad W^{l+1} = \frac{W^l}{W}, \quad D^{l+1} = D^l . \tag{3}$$

and the output depends on the pooling operator we are using. Two types of pooling operators are widely used, which are

1. Max Pooling: Maps the region to the maximum value.
2. Average Pooling: Maps the region with average value.

Example of max pooling is in below figure, we can similarly perform the average pooling just replace the region with average value instead of maximum value.

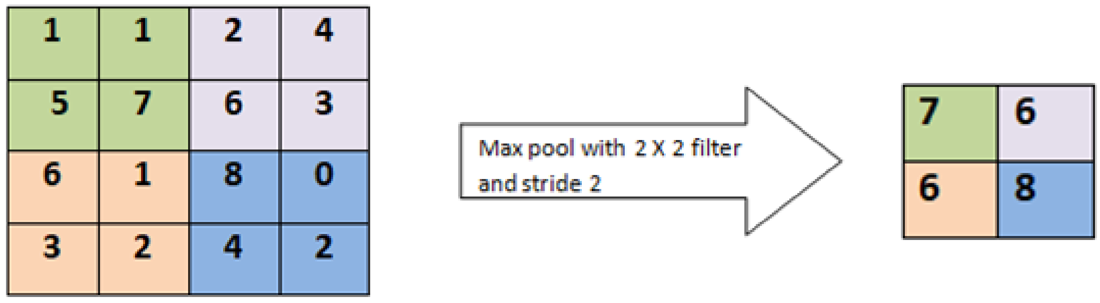


Figure 2.10 Max Pooling

2.6 FLATTENING

The last part of convolutional neural network (CNN) is a classifier which is simply the artificial neural network (ANN). And for ANN we need input layer as a one dimensional feature vector. This process of conversion to 1D feature vector is called flattening. It gets the output of the convolutional layers, flattens all its structure to create a single long feature vector to be used by the dense layer for the final classification.

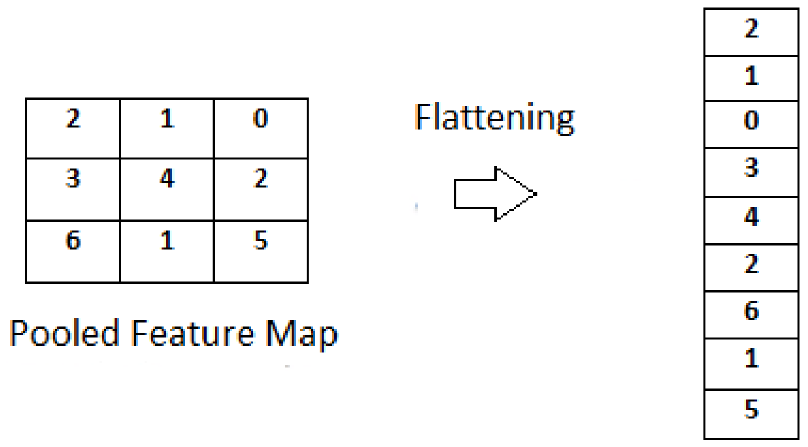


Figure 2.11 Flattening

2.7 FULL CONNECTION

The output of flattening works as an input layer of fully connected dense neural network. And the output layer contains the number of neurons equal to the number of classification classes. In between we can have number of hidden layers. As in the below image we have two classification classes and one hidden layer. Output neurons can classify themselves based on the features reflected by previous layers. One way is to use softmax function which we are discussing next.

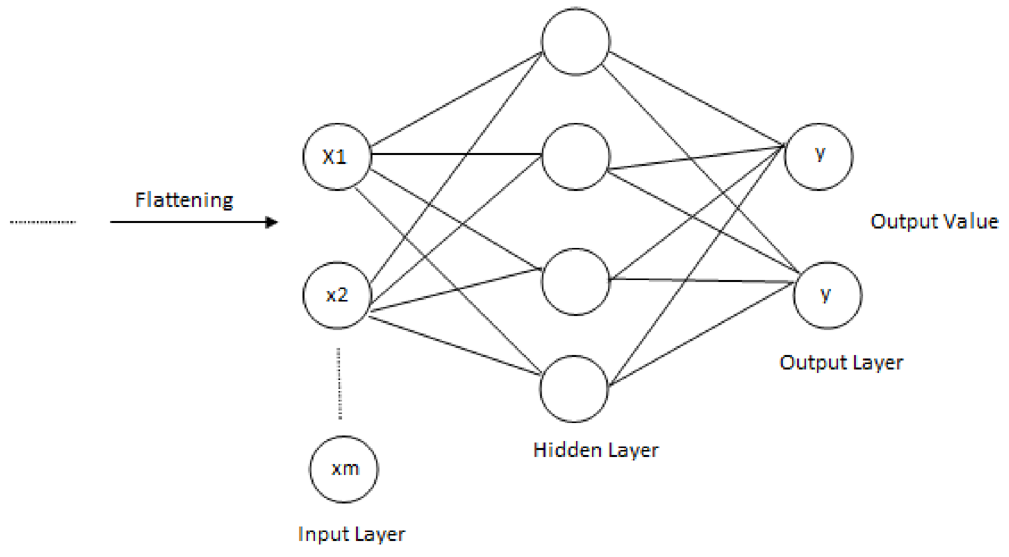


Figure 2.12 Full Connection

2.8 SOFTMAX FUNCTION

In probability theory, we apply softmax function to the categorical distributions. It gives the probability distribution over K different classification classes. So, as it is a probability distribution function the total probability of K classes will be equal to one. We can classify the input as the highest probability class. The mathematical expression for softmax function is

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^{\mathsf{T}}\mathbf{w}_k}} \tag{4}$$

j is the class for which we are calculating the probability and as we can see in denominator there are K different classes. X is the vector and w signifies the weight. We can see in above expression if we add all the probabilities for K classes then the expression became same for numerator as well as for denominator and that will give one. And every probability will range between 0 and 1 because the exponentiation is a positive function.

2.9 CROSS ENTROPY

When we build up a model for probabilistic classification, we mean to delineate model's input to probabilistic forecasts, and we frequently train our model by incrementally changing the model's parameters with the goal that our expectations draw nearer and closer to the ground-truth probabilities.

In our model we have 3755 different mutually exclusive classes and we know the ground truth probabilities i.e. one for matching class and zero for others. As in supervised learning we know the label of images. Let’s say we have three classes viz. house, car and cat. And we have a model to classify them. For the image of car we have ground truth probabilities as [0, 1, 0]. Let’s say the output probability distribution by softmax function is [0.5, 0.4, 0.1]. Now we need to modify our model such that the output probabilities get close to the ground truth probabilities. But the problem is how to measure the difference between two? And the answer is cross entropy.

$$H(y) = \sum_i y_i \log \frac{1}{y_i} = -\sum_i y_i \log y_i \qquad (5)$$

the above expression represents the entropy which defines the optimal number of bits required for the transmission. We encode the higher frequency symbol with less number of bits to minimize the total transmission bits. It turns out that if you have access to the underlying distribution y, then to use the smallest number of bits on average, you should assign log $1/y_i$ bits to the i-th symbol.

If we consider distribution as the tool we use to encode symbols, then entropy measures the number of bits we'll need if we use the correct tool y. This is best possible case, in that we cannot encode the symbols using smaller number of bits on average.

On the other hand, cross entropy is the number of bits we'll need if we encode symbols from y using the wrong tool y^. This consists of encoding the i-th symbol using log

1/y^ibits instead of log 1/y_i bits. We of course still take the expected value to the true distribution y, since it's the distribution that truly generates the symbols:

$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i \qquad (6)$$

Cross entropy is always greater than entropy as encoding symbols according to the wrong distribution y^ will always make us use more bits. The only exemption is the trivial case where y and y^ are equal, and hence the entropy and cross entropy are also qual.

2.10 KL DIVERGENCE

The KL divergence from the calculated to actual value is basically the difference between cross entropy and entropy:

$$\mathrm{KL}(y \,||\, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} - \sum_i y_i \log \frac{1}{y_i} = \sum_i y_i \log \frac{y_i}{\hat{y}_i} \qquad (7)$$

It’s always positive and equal to 0 only when y and y^ are same. Minimizing the entropy is same as minimizing the KL divergence from y^ to y.

CHAPTER 3 IMPLEMENTATION DETAILS

Handwriting recognition is a very crucial task as it aims to replace the human beings from lots of field. As in other areas for the same goal lots of research is going on in this area. Especially on Chinese language because of its large number of characters and the complexity of these characters set. In this work we are providing a better approach for improving the accuracy of the model using modified VGG 16 net with softmax as a classification function. We also tried a fusion approach in which we fuse model with SVM.

3.1 DATASET USED

The dataset is taken from twelfth International Conference on Document Analysis And Recognition (ICDAR13) [9]. This is the third major competition in arrangement utilized the CAISA-HWDB/OLHWDB (Chinese Academy of Sciences Institute of Automation) databases as the preparation set. The best outcome (revise rates) in this gathering for disconnected character acknowledgment was 94.77%. The databases CASIA-HWDB and CASIA-OLHWDB contain offline/online transcribed characters and ceaseless content composed by 1,020 people utilizing stylus pen on paper, with the end goal that the on the web and disconnected information were delivered simultaneously. For evaluating the separated characters the ICDAR13 contains the Chinese character set of 3755 classification classes which is partitioned in the 8:2 amongst training and test sets respectively. The handwritten information was produced by hand-replicating characteristic dialect message on unformatted pages. After the competition the samples are released for the research purposes. We used the same dataset to train and evaluate the proposed model for calculating the accuracy matrix. Fusion model is also tested on the same test samples.

3.2 BACKEND USED (TENSORFLOW)

TensorFlow is an open source programming library for numerical calculation utilizing information stream diagrams. It was initially created by the Google Brain Team inside Google's Machine Intelligence investigate association for machine learning and deep neural network explore, however the framework is sufficiently general to be pertinent in a wide assortment of different areas too. It achieved rendition 1.0 in February 2017, and has proceeded with fast improvement, with 21,000+ submits up to this point, numerous from outside contributors. This segment presents TensorFlow, its open source network and environment.

TensorFlow is a cross-platform. It runs on nearly everything: GPUs and CPUs—including mobile and embedded platforms—and even tensor processing units (TPUs), which are specialized hardware to do tensor math on.

The TensorFlow distributed execution engine abstract away the numerous upheld gadgets and gives an elite center actualized in C++ for the TensorFlow stage. Over that sit the Python and C++ frontends (with additional to come).

The layers API gives an easier interface to normally utilized layers in deep learning models. Over that there is a more elevated level APIs, including Keras and the Estimator API, which makes preparing and assessing distributed models less demanding.

Lastly, various usually utilized models are ready to use out of the box, with more to come. Many of the TensorFlow models include trained weights and examples that show how you can use them for transfer learning, e.g. to learn your own classifications. For example, you can use transfer learning with the Inception image classification model to train an image classifier that uses your specialized image data. [12]

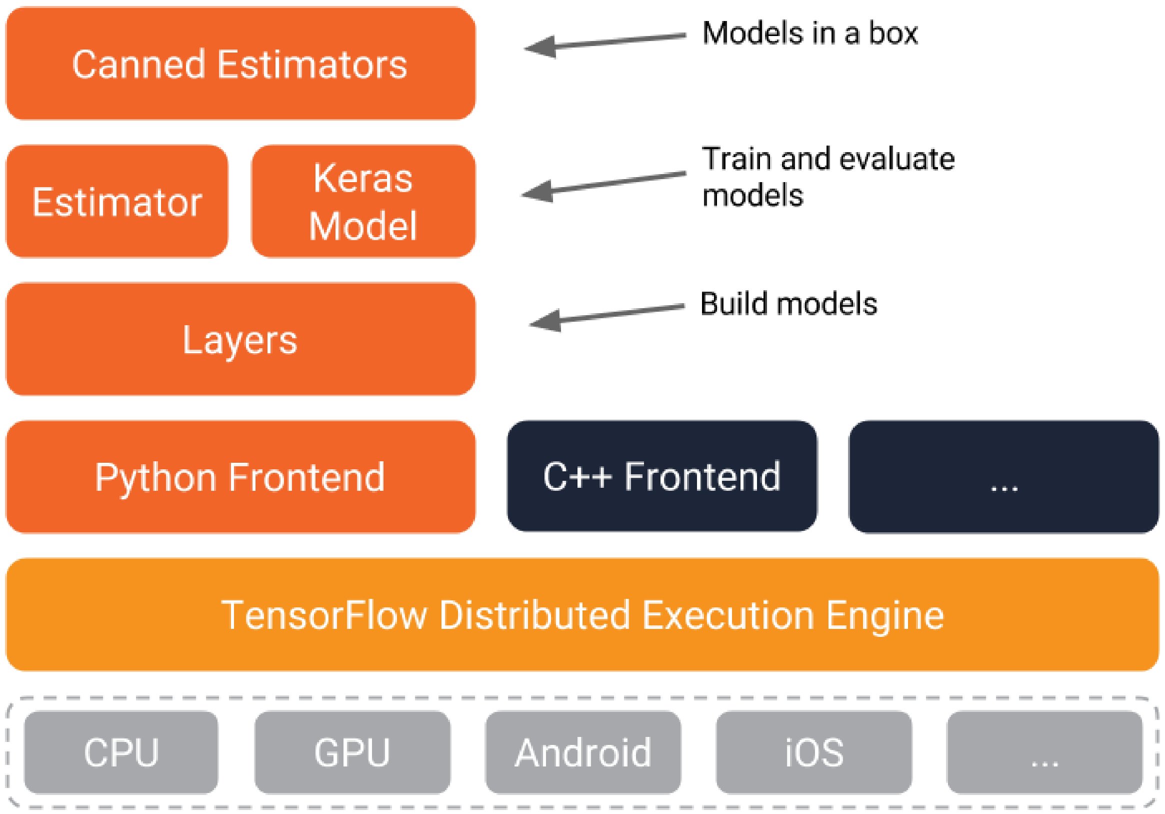


Figure 3.1 TensorFlow Model

Image source: tensorflow.org

3.3 MODEL FUSION

Model fusion with SVM (Support Vector Machine) of our proposed model is also done in this thesis work. SVM is a technique which is used to separate classes in multi dimensionality using hyper planes of mathematics geometry. Hyper planes are adjusted in a way such that the distance between the classes could be maximized. And the separating vectors which represent the separated region are called support vectors. Only these vectors contribute in the forming of hyper planes and that's why they are called support vectors. There are various kernels of SVM which are included in sklearn library. In our model we are fusing our model with the Gaussian RBF kernel.

3.4 IDE USED (SPYDER)

Spyder is an intense logical environment written in Python, for Python, and planned by and for researchers, architects and information experts. It offers a remarkable blend of the advanced editing, analysis, troubleshooting, and profiling functionality usefulness of a thorough improvement instrument with the information investigation, intelligent execution, deep examination, and excellent representation abilities of a scientific bundle.

Beyond its numerous built-in highlights, its capacities can be broadened significantly advance by means of its module framework and API. Besides, Spyder can likewise be utilized as an expansion library, enabling you to build upon its functionality and implant its segments, for example, the intuitive console, in your own software.

3.5 LIBRARIES USED

- **NumPy**

  NumPy is a Python programming language library which includes the help for vast, multidimensional arrays and metrics alongside substantial accumulation of high level mathematical operations.

- **SciPy**

  SciPy is a free and open-source Python library utilized for logical figuring and specialized registering. It contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing. It expands on the NumPy array object and is a piece of the NumPy stack which incorporates devices like Matplotlib, pandas and SymPy, and a growing arrangement of logical computing libraries.

- **Keras**

  Keras is a high level neural netwok liberary which is composed in python and fit for running over TensorFlow or Theano. It permits simple and quick prototyping. It underpins convolution and recurrent networks and in addition combination of both.

3.6 MATHEMATICAL OVERVIEW

As we are familiar with the matrices and vectors, let us take a representation for a vector as $x \in R^{D}$ which is a column vector with D number of elements. For matrix lets use representation $X \in R^{H \times W}$ where H is the number of rows and W is the number of columns. Vector x can also be viewed as a matrix with one column and D rows. This concept can be further extended to higher order tensors like colored images which are can be imagined as three 2-D matrices each containing R , G and B values. Scalar values are just 0 order tensors and vectors are order 1 where as matrices are order 2 tensors.

Tensors are very important in CNN models. The inputs to the CNN model and all the parameters are tensors in CNN. Given a tensor we can generate a long single vector in the desired order. Below is the example.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad A(:) = (1,3,2,4)^T = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}$$

In mathematics this concept is called vectorization. For vectorization of higher order tensors first we need to vectorize its first channel, then its second channel and so on until the last one.

The input tensor goes through series of steps of processing. One step represents a layer, which would be any layer like a convolution layer, a pooling layer, a normalization layer, a fully connected layer, a loss layer, etc.

Let us take an abstract description of forward pass.

$$\boldsymbol{x}^1 \longrightarrow \boxed{\boldsymbol{w}^1} \longrightarrow \boldsymbol{x}^2 \longrightarrow \cdots \longrightarrow \boldsymbol{x}^{L-1} \longrightarrow \boxed{\boldsymbol{w}^{L-1}} \longrightarrow \boldsymbol{x}^L \longrightarrow \boxed{\boldsymbol{w}^L} \longrightarrow z$$

The input is x^1 which is an image, it goes through processing in first layer which is first box. We denoted the parameters involved in first layer collectively as w^1. The output of first layer is x^2 which also act as a input to the second layer processing. This forward run will processed unlit the last layer. To have probability distribution in last layer we can use softmax function in the last layer as a processing step.

After all the layers finished processing we have probability distribution which further goes through the loss layer which is used to measure the discrepancy between the CNN prediction and ground truth values. For which we can use cross entropy function which we discussed in the previous chapter.

3.7 STOCHASTIC GRADIENT DESCENT (SGD)

Let us take an illustration to show why we really need the stochastic gradient descent (SGD). Suppose we have a small network of 25 weights and this weights can have 1000 values then using brute force method total time taken would be,

Total Combinations = 1000 X 1000 X 1000 ………..25 times

= 1000^{25}

= 10^{75} combinations

The world's fastest computer Sunway TaihuLight which has speed of 93 PFLOP. Total Time on this computer,

Total Time = 10^{75} / (93 X 10^{15})

= 1.08 X 10^{58} seconds

= 3.42 X 10^{50} years.

This is longer than the universe has existed and that too for only twenty five weights. So, clearly this method is not feasible.

The solution of above problem is gradient descent. As in many other systems here also we try to minimize the loss z, i.e., we want our predictions to match the ground truth values. We achieve the loss z in previous section. The loss z is like a supervision signal, which guides the system how to modify weights. The equation for this concept is

$$\left(\boldsymbol{w}^i\right)^{t+1} = \left(\boldsymbol{w}^i\right)^t - \eta \frac{\partial z}{\partial \left(\boldsymbol{w}^i\right)^t} \, . \tag{8}$$

Partial differentiation in above equation represents the rate of increase of z with respect to change in different dimensions of w^i (point of tangency). This partial derivative vector is called the gradient in mathematical optimization. Graphical representation of the gradient g is depicted in figure below.

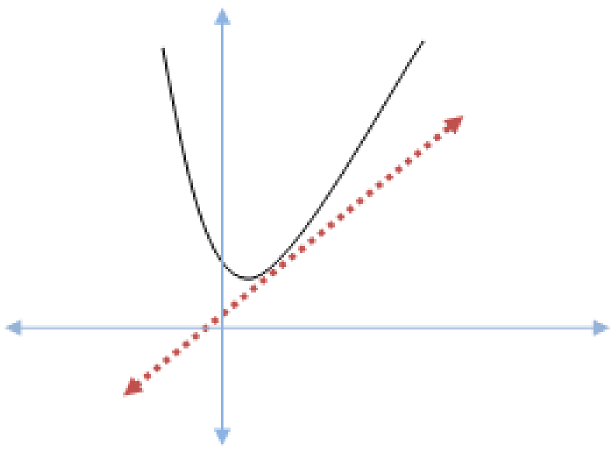


Figure 3.2 Gradient Descent Representation

Gradient descent may seem to be simple in mathematical notations but it is not in training system. For example, if we update the parameters using only one training set we will get the unstable loss function: the average loss of all training set will bounce up and down with large differences. This is because the gradient is calculated using only one training set. So, instead of this we use small subsets or batches of training examples which is called the stochastic gradient descent.

CHAPTER 4 RESULTS AND EVALUATION

The model we are proposing for HCCR is a modified version of VGG 16 net and we also try to fuse our model with SVM. The proposed model for HCCR is trained on ICDAR dataset and then the accuracy of the model also evaluated on the test samples provided by the ICDAR for research purpose. There are total 3755 classification classes of Chinese characters present in ICDAR dataset. There are various models proposed prior to this study but all those models have less accuracy as compared to our proposed model.

So, in this section first we briefly introduce other models and then we explain our proposed model. After that we present the result and comparison between models.

4.1 PREVIOUS MODELS

There are many deep CNN models which are proposed or applied on HCCR. Among them some are relatively better in terms of accuracy then the others. Some models are even very close to human level performance of image recognition. Some of those models are described in the coming sub sections.

4.1.1 GoogleNet/Inception

This model was proposed by google in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC2014) [13]. The primary idea driving this model was better usage of processing resources inside the model. The system utilized a CNN propelled by LeNet yet actualized a novel component which is named an inception module. It utilized batch normalization, image distortions and RMSprop. This module depends on a few little convolutions with a specific end goal to radically decrease the quantity of parameters. This deep network architecture comprises of aggregate 22 layers however it diminished the quantity of parameters when contrasted with its antecedents (4 million). Points of interest of GoogleNet is given in underneath table.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 4.1 GoogLeNet Model

4.1.2 VGG 16

As the name suggest it has 16 conv layers and 5 max pooling of order 2 X 2. It can classify image among thousand other images. The size of the feature detectors used is 3 X 3 because of which the number of parameters is very large (138 million). [15]

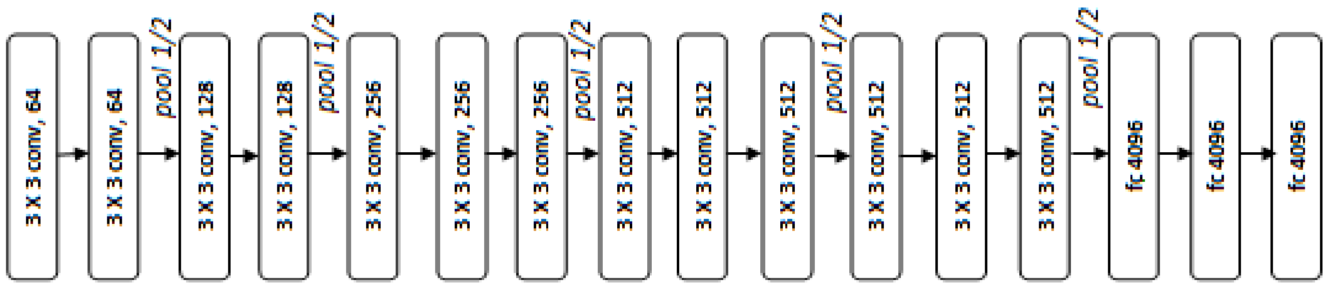


Figure 4.1 VGG16 Net

4.1.3 AlexNet

AlexNet [14] consist of five convolution layer and three fully connected layers. ReLu layer is associated with every convolution layer. Dropout is applied at beginning and second fully connected layer of dense network. The network contains total 62.3 million parameters.

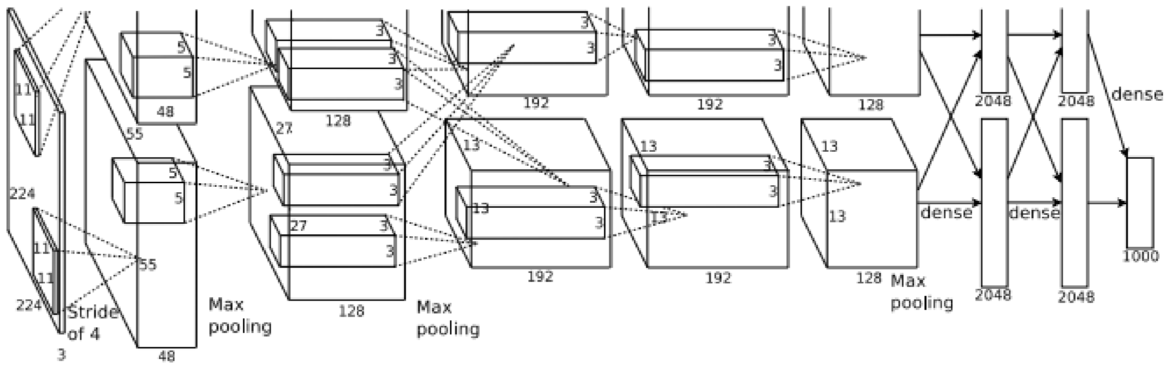


Figure 4.2 AlexNet

4.2 PROPOSED MODEL

We proposed modified version of VGG16 [15] network which is better in accuracy as compared to the models discussed in previous subsections. This model contains total 11 convolutional layers and every layer is augmented with leakyReLu layer which is described in coming subsection. There are total five max pooling layers. With leakyReLu, batch normalization is also associated. First two layers contains 32 neurons each, next two contains 64 each, next two 128 each, next two 256 each and the last three contains 512 neuron. Each of this five sections contain max pooling layer at the end of them. After this three fully connected layers are present first layer contains 1024 input neurons and then second (hidden layer) contains 256 neurons and last output layer contains 3755 neurons representing the classes for classification. In the very last layer of model softmax function is used.

4.2.1 PReLU (Parameterized Rectified Linear Unit)

Leaky ReLU [16] also known as Parameterized ReLU is a non-linear activation function. Activation functions act like clipping process. Like sigmoid function that clips the value between 0 and 1 where as ReLU eliminates all negative values which we discussed in chapter 2. Similarly, PReLU decrease the range of negative values. It maps the large negative values to the smaller one by reducing the slop of mapping function. When we apply kernel or feature detector on our image then image may have linearity and to eliminate this we use activation functions. These non-linear functions largely improves the performance of the system if they are removed then system performance will drop.

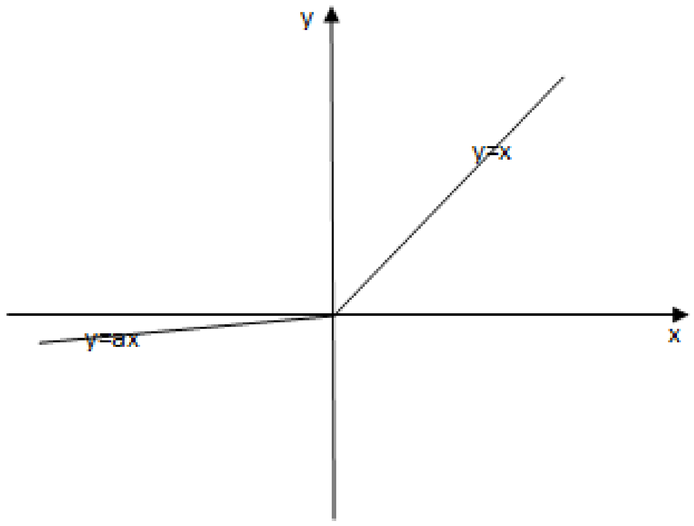


Figure 4.3 Leaky ReLu Representation

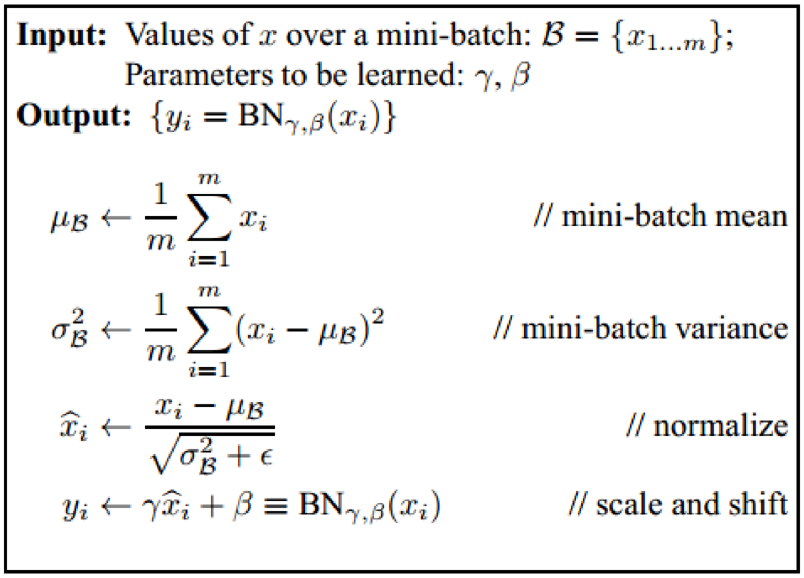
As shown in above figure,

Y = X for X >= 0 and

Y = aX for X < 0

4.2.2 Batch Normalization

In CNN we use batch normalization to normalize the different elements of the same feature map, at different locations similarly. We apply Batch normalization at the output of every convolutional layer. Below is the figure which shows the steps performed in batch normalization.



```
Input:  Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
        Parameters to be learned: $\gamma$, $\beta$
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

    $\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$    // mini-batch mean
    $\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$    // mini-batch variance
    $\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$    // normalize
    $y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$    // scale and shift
```

Figure 4.3 Batch Normalization Algorithm

The first two steps are simple mean and variance over the input x_i. The next two equations have β and ϒ as two hyper parameters which can be learned by neural networks. The use β and ϒ is to keep the mean and variance fixed.

After normalization we don’t have to worry about the scale of input features would be extremely different. So, the gradient descent can reduce the iterations when approaching the zero slope tangent and converge faster. Also the batch normalization reduces the dependency of earlier layers by keeping the mean and variance fixed which in some way makes layers independent with each other.

4.2.3 Model Summary

The following is the output of model.summary() command of compiled model in python.

```
Layer (type)                              Output Shape                  #Param
==============================================================================
input_1 (Input Layer)                     (None, 96, 96, 1)             0
______________________________________________________________________________
zero_padding2d_1 (Zero Padding))          (None, 98, 98, 1)             0
______________________________________________________________________________
conv2d_1 (Conv2D)                         (None, 96, 96, 32)            320
______________________________________________________________________________
leaky_re_lu_1 (LeakyReLU)                 (None, 96, 96, 32)            0
______________________________________________________________________________
batch_normalization_1 (Batch Norm)        (None, 96, 96, 32)            128
______________________________________________________________________________
zero_padding2d_2 (Zero Padding)           (None, 98, 98, 32)            0
______________________________________________________________________________
conv2d_2 (Conv2D)                         (None, 96, 96, 32)            9248
______________________________________________________________________________
leaky_re_lu_2 (LeakyReLU)                 (None, 96, 96, 32)            0
______________________________________________________________________________
batch_normalization_2 (Batch Norm)        (None, 96, 96, 32)            128
______________________________________________________________________________
max_pooling2d_1 (MaxPooling2)             (None, 48, 48, 32)            0
______________________________________________________________________________
zero_padding2d_3 (Zero Padding)           (None, 50, 50, 32)            0
______________________________________________________________________________
conv2d_3 (Conv2D)                         (None, 48, 48, 64)            18496
______________________________________________________________________________
leaky_re_lu_3 (LeakyReLU)                 (None, 48, 48, 64)            0
______________________________________________________________________________
batch_normalization_3 (Batch Norm)        (None, 48, 48, 64)            256
______________________________________________________________________________
zero_padding2d_4 (Zero Padding)           (None, 50, 50, 64)            0
______________________________________________________________________________
```

conv2d_4 (Conv2D)	(None, 48, 48, 64)	36928
leaky_re_lu_4 (LeakyReLU)	(None, 48, 48, 64)	0
batch_normalization_4 (Batch Norm)	(None, 48, 48, 64)	256
max_pooling2d_2 (MaxPooling2)	(None, 24, 24, 64)	0
zero_padding2d_5 (Zero Padding)	(None, 26, 26, 64)	0
conv2d_5 (Conv2D)	(None, 24, 24, 128)	73856
leaky_re_lu_5 (LeakyReLU)	(None, 24, 24, 128)	0
batch_normalization_5 (Batch Norm)	(None, 24, 24, 128)	512
zero_padding2d_6 (Zero Padding)	(None, 26, 26, 128)	0
conv2d_6 (Conv2D)	(None, 24, 24, 128)	147584
leaky_re_lu_6 (LeakyReLU)	(None, 24, 24, 128)	0
batch_normalization_6 (Batch Norm)	(None, 24, 24, 128)	512
max_pooling2d_3 (MaxPooling2)	(None, 12, 12, 128)	0
zero_padding2d_7 (Zero Padding)	(None, 14, 14, 128)	0
conv2d_7 (Conv2D)	(None, 12, 12, 256)	295168
leaky_re_lu_7 (LeakyReLU)	(None, 12, 12, 256)	0
batch_normalization_7 (Batch Norm)	(None, 12, 12, 256)	1024
zero_padding2d_8 (Zero Padding)	(None, 14, 14, 256)	0

conv2d_8 (Conv2D)	(None, 12, 12, 256)	590080
leaky_re_lu_8 (LeakyReLU)	(None, 12, 12, 256)	0
batch_normalization_8 (Batch Norm)	(None, 12, 12, 256)	1024
max_pooling2d_4 (MaxPooling2)	(None, 6, 6, 256)	0
zero_padding2d_9 (Zero Padding)	(None, 8, 8, 256)	0
conv2d_9 (Conv2D)	(None, 6, 6, 512)	1180160
leaky_re_lu_9 (LeakyReLU)	(None, 6, 6, 512)	0
batch_normalization_9 (Batch Norm)	(None, 6, 6, 512)	2048
zero_padding2d_10 (Zero Padding)	(None, 8, 8, 512)	0
conv2d_10 (Conv2D)	(None, 6, 6, 512)	2359808
leaky_re_lu_10 (LeakyReLU)	(None, 6, 6, 512)	0
batch_normalization_10 (Batch Norm)	(None, 6, 6, 512)	2048
zero_padding2d_11 (Zero Padding)	(None, 8, 8, 512)	0
conv2d_11 (Conv2D)	(None, 6, 6, 512)	2359808
leaky_re_lu_11 (LeakyReLU)	(None, 6, 6, 512)	0
batch_normalization_11 (Batch Norm)	(None, 6, 6, 512)	2048
max_pooling2d_5 (MaxPooling2)	(None, 3, 3, 512)	0
flatten_1 (Flatten)	(None, 4608)	0

dense_1 (Dense)	(None, 1024)	4719616
leaky_re_lu_12 (LeakyReLU)	(None, 1024)	0
dense_2 (Dense)	(None, 256)	262400
leaky_re_lu_13 (LeakyReLU)	(None, 256)	0
batch_normalization_12 (Batch Norm)	(None, 256)	1024
dense_3 (Dense)	(None, 3755)	965035
activation_1 (Activation)	(None, 3755)	0

Total parameters: 13,029,515

Trainable parameters: 13,024,011

Non-trainable parameters: 5,504

Table 4.2 Summary of the Model

4.3 RESULT AND COMPARISON

4.3.1 Result

We already described the ICDAR 2013 dataset in the previous chapter. The model which is proposed in this thesis is trained and tested on HWDB1.1 of ICDAR 2013 competition which is offline handwritten character dataset for Chinese characters. The model is evaluated against the 3755 test classes in the batch size of 64. The accuracy and loss of the model up to 5 decimal points after evaluation are 0.97838 and 0.11710 respectively. We also tried to fuse the model with SVM but the results remain same.

4.3.2 Comparison

Below table compares the proposed model with other existing ones. The references to other models are [9], [11], and [17].

S. No.	Approach	Accuracy	#Parameters
1.	AlexNet	95.49%	60 million
2.	GoogLeNet	96.29%	4 million
3.	Fujitsu	94.77%	NA
4.	VGG16	94.54%	100 million
5.	Proposed	**97.84%**	13 million
6.	Proposed+SVM	97.84%	13 million

Table 4.3 Comparison with other Models

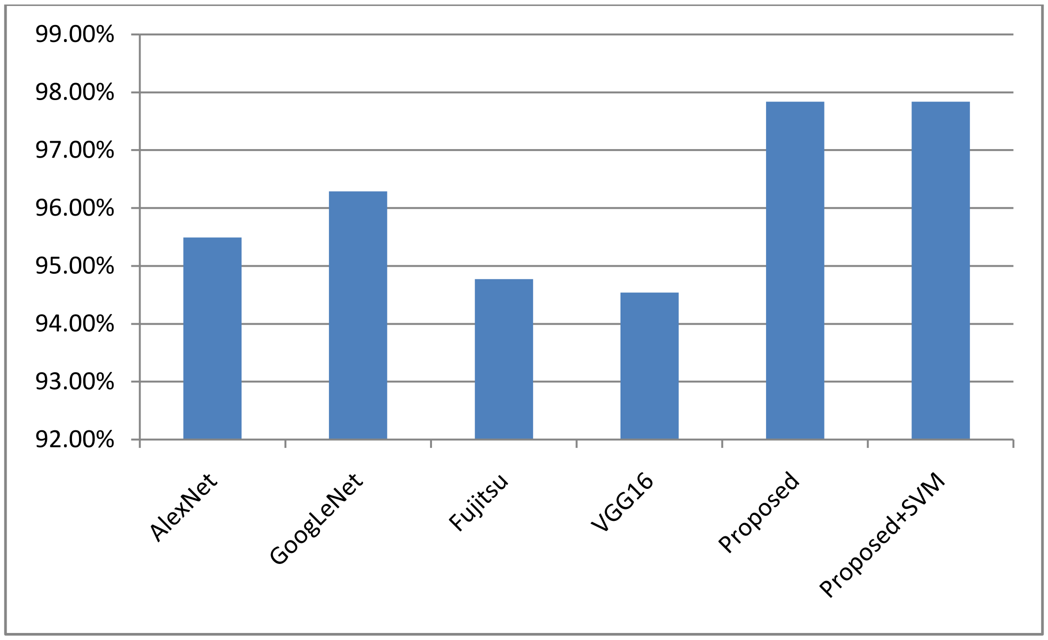


Figure 4.4 Accuracy chart

CHAPTER 5 CONCLUION AND FUTURE WORK

5.1 Conclusion

Deep learning is quite new and emerging technology in the field of machine learning. As now data is increasing exponentially day by day we need a technique which is deep enough to generate good results using this data. And solution seems to be deep learning models. But still there are challenges as we are competing against the human mind. And as the name suggested they are very deep so the time taken by models to execute is very large. So, still there is lot to do in this field to replicate the vision of human being. These models also should be accurate enough to not have a huge consequence at the time of error. We also tried to do some advances in the area of HCCR using deep learning concepts.

We modified the VGG16 net to make it more slimmer and added few more layers to make it deeper. This new model has more layers and less number of parameters as compared to VGG16. We also achieve better accuracy as compared with the other models. We also tried to fuse the model with SVM but it didn’t affect the accuracy.

5.2 Future work

In future we can add functionality from more different models as layers to extract more features in dataset. More features might help in better recognition of images. This in turn could increase the accuracy of model.

6. REFERANCES

[1] Yann LeCun, Yoshua Bengio and Geoffrey Hinton, “Deep Learning,” *Nature International journal of science*, 521, pp. 436-444 (28-May-2015).

[2] R. Dai, C. Liu, and B. Xiao, “Chinese character recognition: history, status and prospects,” *Frontiers of Computer Science in China*, vol. 1, no. 2, pp. 126–136, 2007.

[3] C. Liu, S. J¨ager, and M. Nakagawa, “Online recognition of Chinese characters: The state-of the-art,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 198–213, 2004.

[4] LeCun Y, Jackel L D, and Bottou L, “ Learning Algorithms for Classification: A Comparison on Hand written Digit recognition.” *Neural Networks: the Statistical Mechanics Perspective*, pp. 261- 276, 1995.

[5] W Pan, L Jin, and Z Feng, “ Recognition of Chinese characters based on multi-scale gradient and deep neural network.” *Journal of Beijing University of Aeronautics and Astronautics*, 41(4): 751-756,2015.

[6] Xin Liu and Shen Zhen. “Online Chinese Handwriting Character Recognition System Based on Convolutional Neural Network.” *Harbin Institute of Technology*, pp. 1-67, 2015.

[7] Q.Z. Wu, Y. LeCun, L. D. Jackel, and B.S. Jeng, “On-line recognition of limited-vocabulary chinese character using multiple convolutional neural networks.” *IEEE International Symposium on Circuits and Systems*, pp. 2435–2438, 1993.

[8] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *Computing Research Repository,* vol. abs/1202.2745, 2012.

[9] F. Yin, Q. Wang, X. Zhang, and C. Liu, "ICDAR 2013 chinese handwriting recognition competition," *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1464–1470, 2013.

[10] C. Cheng, X.-Y. Zhang, X.-H. Shao, and X.-D. Zhou, "Handwritten chinese character recognition by joint classification and similarity ranking," *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 507–511, 2016.

[11] Z. Zhong, L. Jin, and Z. Xie, "High performance offline handwritten chinese character recognition using googlenet and directional feature maps," *Document Analysis and Recognition (ICDAR)*, pp. 846–850, 2015.

[12] https://www.tensorflow.org/tutorials/, (Accessed on: 27/March/2018).

[13] C. Szegedy, W. Liu, and Y. Jia. "Going deeper with convolutions." *Computing Research Repository*, vol. abs/1409.4842, 2014.

[14] A. Krizhevsky, 1. Sutskever and G.E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*, pp. 1097-1105, 2012.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

[16] C.-C. Jay Kuo, “Understanding Convolutional Neural Networks with A Mathematical Model” *Ming-Hsieh Department of Electrical Engineering University of Southern California, Los Angeles, CA 90089-2564, USA*,2-Nov-2016.

[17] Xiangsheng Zeng, Donglai Xiang, Liangrui Peng, Changsong Liu, and Xiaoqing Ding, “Local Discriminant Training and Global Optimization for Convolutional Neural Network based Handwritten Chinese Character Recognition” *IAPR International Conference on Document Analysis and Recognition*, 2017.