TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PASCHIMANCHAL CAMPUS
LAMACHAUR, POKHARA

[Subject Code: CT 755]
A MAJOR PROJECT REPORT
ON

“DEVANAGARI HANDWRITTEN CHARACTER RECOGNITION”

SUBMITTED BY:

Bikash Pandey	073/BCT/615
Bishwas Tripathi	073/BCT/617
Netra Prasad Neupane	073/BCT/626
Upendra Dhamala	073/BCT/648

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

July,2021

NEPALI HANDWRITING RECOGNITION

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

SUBMITTED BY:

Bikash Pandey	[073/BCT/615]
Bishwas Tripathi	[073/BCT/617]
Netra Prasad Neupane	[073/BCT/626]
Upendra Dhamala	[073/BCT/648]

SUPERVISED BY:

Er. Bal Krishna Nyaupane

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LETTER OF APPROVAL

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled "Devanagari Handwriting Character Recognition" submitted by Bikash Pandey, Bishwas Tripathi, Netra Prasad Neupane and Upendra Dhamala in partial fulfilment of the requirements for the Bachelor's degree in Computer Engineering.

____

Supervisor, Er. Bal Krishna Nyaupane
Assistant Professor
Department of Electronics and Computer Engineering

____

External Examiner, Er. Narkaji Gurung,
Telecommunication Engineer,
Nepal Telecom

____

Er. Hari Prasad Baral
Head of Department,
Department of Electronics and Computer Engineering
Date of Approval:

DECLARATION

We hereby declare that the final year project entitled “Nepali Handwriting Recognition” submitted to the Department of Electronics and Computer Engineering, Pashchimanchal Campus, Institute of Engineering, Tribhuvan University, Lamachour, Pokhara is an original piece of work under the supervision of Er. Bal Krishna Nyaupane, HOD Hari Prasad Baral and is submitted in partial fulfillment of the requirements for the Project Work as prescribed by the syllabus. This project work report has not been submitted to any other university or Institution for the award of any degree.

Bikash Pandey[073/BCT/615]

....

Bishwas Tripathi[073/BCT/617]

....

Netra Prasad Neupane[073/BCT/626]

....

Upendra Dhamala[073/BCT/648]

…………………………………...

DATE: July,2021

COPYRIGHT

ACKNOWLEDGMENT

Acknowledging the supported hands and guidance received from several individuals and organizations for the achievement and completion of the project is our great pleasure and responsibility. First of all, we would like to express our sincere gratitude to the Department of Electronics and Computer Engineering for giving us an opportunity to work on the project as it has helped us to expand the horizons of our knowledge. We are so humbled and grateful to Bal Krishna Neupane sir for his supervision, guidance, bona fide, integrity, obduracy and suggestions throughout the project. We are eternally grateful to Hari Prasad Baral sir, Head of Department, Computer and Electronics Engineering for his benevolent support for providing us a better working environment.We are much obliged with constant support and feedback from Rupesh Kumar Shah sir and Sangharsa Bidari sir. We would also like to admire the technical support from Deepak Raj Giri sir during our work. We are also highly indebted to Ashok Kumar Pant sir for his guidance at the starting of the project. And we have to give the deepest thanks to the schools, teachers and students from Pokhara valley for their huge support in our Devanagari handwritten character data collection phase. We would like to thank our friends for their endeavors in our project.

Any suggestions for the further improvement of this project are heartily welcomed.

ABSTRACT

Applications like number plate recognition, digitization of manuscripts, legal billing document OCR and data entry for business documents have carried out Handwriting recognition as a research of interest in the present time. However, no such significant effort has been made in the Devanagari script till date. So, we have tried to figure out a way to propose a deep learning architecture to recognize handwritten Devanagari characters using convolutional neural networks(CNN/ConvNet).

Due to the lack of a dataset for Devanagari scripts, we have prepared a new dataset for the Devanagari script of 456 different classes each having 1000 images. The classes are Bahrakhari (बाह्रखरी) letters- क,का,कि,की,..,कं,कः, ख,खा,खि,खी...,खं,खः, ... ज्ञ,ज्ञा,ज्ञि,ज्ञी,...,ज्ञं,ज्ञः which are the compound letters of vowel diacritics of consonants of Devanagari script, vowels- अ,आ इ ई...,अं,अः and Devanagari numerals-०,१,२,३,४....९. We have used this character dataset to experiment with different CNN architecture of varying parameters for performance evaluation and improvements. We are able to achieve high training accuracy and high validation accuracy on our dataset with the help of our experiments on CNN.

Keywords: Handwriting Recognition, Convolution Neural Network, Devanagari Script, Baahrakhari Dataset

TABLE OF CONTENTS

LETTER OF APPROVAL……………………………………………………………....iii
COPYRIGHT…………………………………………………………........................v
ACKNOWLEDGEMENT……………………………………………………….....vi
ABSTRACT…………………………………………………….........................vii
TABLE OF CONTENTS……………………………………………………......viii
LIST OF FIGURES……………………………………………….................x
LIST OF TABLES………………………………………………..................xi
LIST OF ABBREVIATIONS……………………………………………………xii
CHAPTER 1: INTRODUCTION……………………………………………...1
1.1 Background…………………………………………………………….......1
1.1.1 Handwriting Recognition…………………………………………………...1
1.1.2 Devanagari Scripts……………………………………………………………1
1.2 Motivation…………………………………………………………………….2
1.3 Problem Statement…………………………………………………………….2
1.4 Objectives……………………………………………………………………..3
1.5 Application……………………………………………………………........3
CHAPTER 2: LITERATURE REVIEW……………………………………...4
CHAPTER 3: METHODOLOGY……………………………………………..6
3.1 Introduction…………………………………………………………........6
3.2 Tools………………………………………………………………………...6
3.2.1 Python……………………………………………………………........6
3.2.2 Tensorflow………………………………………………………………6
3.2.3 Keras…………………………………………………………….........6
3.2.4 Scikit-Learn……………………………………………………………...7
3.2.5 OpenCV…………………………………………………………………7
3.2.6 Numpy……………………………………………………………..........7
3.2.7 Pandas……………………………………………………………...........7
3.2.8 Matplotlib……………………………………………………………........7
3.2.9 Seaborn……………………………………………………………...........8
3.2.10 Jupyter Notebook/Google Colaboratory…………………………………8
3.3 Dataset Generation………………………………………………………….....8
3.3.1 Data Collection……………………………………………………………….8
3.3.2 Preprocessing & Character Extraction…………………………………….9
3.3.2.1 Grayscale Conversion…………………………………………..10
3.3.2.2 Thresholding……………………………………………………...10
3.3.2.3 Inversion……………………………………………………………..10
3.3.2.4 Dilation………………………………………………………….....10
3.3.2.5 Contour Detection………………………………………………...10
3.3.2.6 Character Extraction………………………………………………10
3.4 Model Generation…………………………………………………………….....11
3.4.1Convolutional Neural Network......……………………………………………12
3.4.1.1 Input Layer……………………………………………………………12
3.4.1.2 Convolution Layer…………………………………………………12

3.4.1.2.1 Depth……..15
3.4.1.2.2 Stride……...15
3.4.1.2.3 Padding……15
3.4.1.3 Activation function……16
3.4.1.4 Pooling Layer……...17
3.4.1.5 Flatten Layer……..19
3.4.1.6 Fully Connected Layer……...19
3.4.1.7 Dropout……..20
3.4.1.8 Batch Normalization……..20
3.4.2 Optimization Algorithms……21
3.4.2.1 Adam Optimization Algorithm……21
3.4.2.1 Stochastic Gradient Descent……21
3.4.2 Proposed Model……..22
CHAPTER 4: EPILOGUE…….23
4.1 Experiment and discussion……23
4.2 Result Analysis……..............24
4.3 Conclusion……..............28
4.4 Further Enhancements……..29

LIST OF FIGURES

Figure 1 Vowels..1
Figure 2 Consonants……..1
Figure 3 Consonants with vowels diacritics..2
Figure 4 Numerals ……..2
Figure 5 Data Collection ……...9
Figure 6 Data Generation Steps…………………………..……………………………11
Figure 7 Convolutional Neural Network 12
Figure 8 Convolution Operation ....…………14
Figure 9 Size of Feature Map Calculation 14
Figure 10 Depth in Convolution Operation 15
Figure 11 Padding Operation……………. 16
Figure 12 ReLU………………..………...17
Figure 13 Max Pooling ....…………18
Figure 14 Pooling Applied To Rectified Feature Map...18
Figure 15 Flattening 19
Figure 16 Fully Connected Layer 20
Figure 17 Use of Dropout Layer 20
Figure 18 Proposed Model 22
Figure 19 Epoch vs Accuracy Plot 25
Figure 20 Epoch vs Loss Plot 26
Figure 21 Confusion Matrix 27

LIST OF TABLES

Table 1 Model Architecture with parameters……………………………………………………23
Table 2 Accuracy of different models…………………………………………………………...24

LIST OF ABBREVIATIONS

CNN: Convolutional Neural Network
ConvNet: Convolutional Neural Network
ReLU: Rectified Linear Unit
AI: Artificial Intelligence
SGD: Stochastic Gradient Descent
RMS: Root Mean Square
OpenCV: Open Computer Vision
JSON: JavaScript Object Notation
SQL: Structured Query Language
DHCR: Devanagari Handwritten Character Recognition
DHCH: Devanagari Handwritten Character Dataset
ML: Machine Learning
API: Application Programming Interface
GPU: Graphics Processing Unit
TPU: Tensor Processing Unit
MATLAB: Matrix Laboratory
RAM: Random Access Memory

CHAPTER 1: INTRODUCTION

1.1 Background

1.1.1 Handwriting Recognition

Handwriting recognition is the mechanism of recognition of handwritten characters with the help of deep neural networks and converting it into computerized form. It is used in applications which require entering data into the system with the analysis of handwritten text. The first real life application was by the American Bank to recognize the handwritten amounts in the chequebook. Since we are dealing with Devanagari, the system must recognize the Devanagari characters. It can be used for digitizing old scripts, in banks to auto-enter data from handwritten texts by the customers, in government offices and for many other purposes. Online and offline handwritten recognition are its types.

1.1.2 Devanagari Scripts

Devanagari language uses the Devanagari Script. Sanskrit, Hindi, Nepali, Marathi and other languages prevalent in Indian subcontinent use Devanagari Scripts. Devanagari language has 36 consonants and 13 vowels. Consonants and vowel sounds(except ऋ) are directly mixed to form a sound. It has a unique characteristic unlike other popular languages: a line above the word called dika (डिका) in Devanagari. Every character must have dika otherwise it won't be accepted in Devanagari script.

There are 13 vowels:

अ आ इ ई उ ऊ ऋ ए ऐ ओ औ अं अः

Figure 1: Vowels

There are 36 consonants:

क, ख, ग, घ, ङ, च, छ, ज, झ, ञ, ट, ठ, ड, ढ,
ण, त, थ, द, ध, न, प, फ, ब, भ, म, य, र, ल, व,
श, ष, स, ह, क्ष, त्र, ज्ञ

Figure 2: Consonants

Every consonant can have 12 variations (बाह्रखरी) formed by combining consonants with the different vowels.

क, का, कि, की, कु, कू, के, कै, को, कौ, कं, कः

Figure 3: Consonant with vowel diacritics

There are 10 numerals:

०, १, २, ३, ४, ५, ६, ७, ८, ९

Figure 4: Numerals

1.2 Motivation

We have come across many software and applications that do the task of scanning and detecting the texts in images and handwritten texts in English language. However, there are not any significant tasks done in the Devanagari scripts for Optical Character Recognition(OCR).

The task of manually entering the handwritten texts into the system is quite slow and cumbersome. In many places, especially the administrative ones, we see that the manpowers is assigned for entering tasks specially for Devanagari texts. A lot of time is wasted when it comes to typing the texts, especially the Devanagari texts. So, there is the utmost need of a software or an application which can automatically scan and detect the texts written in an image as well as the handwritten texts. Our project focuses on the development of a software tool that can efficiently do the task of Devanagari script recognition with high accuracy.

1.3 Problem Statement

Nepal has not been able to use the blessing offered by deep learning in the field of Devanagari handwritten recognition. The system can be used in government offices and banks but due to low accuracy offered by the system, the society's time has been wasted. We dug deep in this line of work and we found that the main hindrance was that there was no proper dataset for the application in real life work. So, we focused our energy in solving this problem so that handwritten characters would be easily converted into digital form and the efficiency of the organizations of the Nepali society would improve a lot.

1.4 Objectives

The main objectives of the project are:

A. To prepare a brand new dataset of Devanagari handwritten Bahrakhari characters and numerals.
B. To create and compare different convolution network architecture for recognition of Devanagari handwritten characters.

1.5 Application

A. Converting official or personal handwritten document into typed format which is easier to read or transfer.
B. Conversion of old literature and manuscripts into digital form.
C. To make the proposed system serve as a guide and work in the character recognition area.

CHAPTER 2: LITERATURE REVIEW

Machine replication of human functions, like reading, is an ancient dream. However, over the last five decades, machine reading has grown from a dream to reality. Optical character recognition has become one of the most successful applications of technology in the field of pattern recognition and artificial intelligence. Many commercial systems for performing OCR exist for a variety of applications, although the machines are still not able to compete with human reading capabilities. The traditional way of entering data into a computer is through the keyboard. However, this is not always the best nor the most efficient solution. In many cases automatic identification may be an alternative. Various technologies for automatic identification exist, and they cover needs for different areas of application.

To imitate human functions by machines, making the machine able to perform tasks like reading, is the earliest nightmare. The origins of character recognition can actually be found back in 1870. This was the year that C.R.Carey of Boston Massachusetts invented the retina scanner which was an image transmission system using a mosaic of photocells. Two decades later the Polish P. Nipkow invented the sequential scanner which was a major breakthrough both for modern television and reading machines. During the first decades of the 19th century several attempts were made to develop devices to aid the blind through experiments with OCR. However, the modern version of OCR did not appear until the middle of the 1940's with the development of the digital computer. The motivation for development from then on, was the possible applications within the business world [2].

There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters [3].

- Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation".
- Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. The extraction features reduces the dimensionality of the representation and makes the recognition process computationally efficient. [4].

In 1974, Kurzweil Computer Products developed the first omni-font software. Much of the first work in this space was in English, using the Latin alphabet. In 1992, the Cyrillic alphabet got some love and Russian text could be digitized [5].
K.Gaurav, Bhatia P.K. [6] Et al, this paper deals with the various preprocessing techniques involved in the character recognition with different kinds of images ranging from a simple handwritten form based documents and documents containing color end complex background and varied intensities. In this, different preprocessing techniques like skew detection and correction, image enhancement techniques of contrast stretching, binarization, noise removal techniques,normalization and segmentation, morphological processing techniques are discussed.However, even after applying all the said techniques might not be possible to achieve full accuracy in a preprocessing system.
In [7], Devanagari character recognition experiment with 20 different writers with each writer writing 5 samples of each character in a totally unconstrained way, has been conducted. An

accuracy of 86.5% with no rejects is achieved through the combination of multiple classifiers that focus on either local online properties, or global off-line properties.
S.Acharya, A.Pant [8]Et al, they introduced a new public image dataset for Devanagari script: Devanagari Handwritten Character Dataset(DHCD). The dataset consisted of 92,000 images of 46 different classes of characters of Devanagari script segmented from handwritten documents. Along with the dataset, they have also proposed a deep learning architecture for recognition of those characters. The proposed architecture scored the highest test accuracy of 98.47%. However, this project did not address the characters formed by the combination of vowels and consonants.

This [9] paper has developed the CNN based Optical Character Recognition System for Devanagari language, in python using Keras library on top of Theano and numpy. The system was trained using a set of real world and synthesized dataset considering various noise conditions. This paper claims to have achieved 99.31% accuracy for previously untrained data and 96.5% for data that had been used for training, upon running the proposed system for 32 Epochs. But the paper doesn’t mention the number of data used to train and also the dataset doesn’t contain characters obtained by consonants with vowels and the combination of consonants and vowels with special symbols.

CHAPTER 3: METHODOLOGY

3.1 Introduction

Advancement in the field of artificial intelligence, machine learning and deep learning has accomplished the research that is based on how to analyze and train various neural network models.

The key things that we have to consider while carrying out Devanagari Handwritten Character Recognition (DHCR) include the steps and algorithms that we have to carry out from the initial step of data collection, dataset generation, setting the environment requirements and importing libraries for the final step of model development and algorithm. Since only limited dataset for Devanagari characters (क..... ज्ञ) were available, we took an initiation to collect the whole Devanagari dataset in handwritten format from many students and friends of various schools and colleges in Pokhara. For Devanagari character recognition, we choose to use convolutional neural networks (CNN) for extracting characteristics of handwritten characters.

Machine learning techniques need a large set of data to support the model for convolution operations so that it can give the model with approved accuracy and minimum loss. Among the various deep learning techniques, CNN proves to be the best option for us for training, validation and testing the dataset.

3.2 Tools

We have used following programming language, libraries, frameworks and development environment for developing our system:

3.2.1 Python

Python is an interpreted, high-level, easy-to-use programming language developed by Guido van Rossum in 1991. Python has a great ecosystem for data science and machine learning. Python has a large number of excellent libraries that boost the development capabilities and experienced community to solve particular problems. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large projects.

3.2.2 Tensorflow

Tensorflow is a machine learning library developed by the Google Brain Team to accelerate machine learning and deep neural network research and development. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state of the art in ML and developers easily build and deploy ML powered applications. Tensorflow enables developers to quickly and easily get started with deep learning in the cloud. It helps to build and train ML models easily using intuitive high-level APIs like Keras with eager execution and also helps to train and deploy models easily in the cloud.

3.2.3 Keras

Keras is an open-source library that provides a python interface for artificial neural networks. Keras acts as an interface for the Tensorflow library (version 2.4). It is designed to enable fast experimentation with deep neural networks. It focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers and a host tool to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. In addition to standard neural networks, keras has support for convolutional neural networks, recurrent neural networks and common utility layers(dropout, normalization, pooling) of networks.

3.2.4 Scikit-Learn
Scikit-learn(Sklearn) is a free software machine learning library for the python programming language. It is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and is designed to interoperate with the python numerical and scientific libraries like NumPy and SciPy.

3.2.5 OpenCV
OpenCV is an open source,cross-platform library using which we can develop real-time computer vision applications. It was originally developed by intel and later supported by Willow Garage and then itseez. It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. It has more than 1500 algorithms, extensive documentation and sample code for real-time computer vision. It supports windows, Linux, Mac OS X, Android, FreeBSD etc operating systems.

3.2.6 Numpy
Numpy(Numerical Python) is an open-source library for the Python programming language. It is used for scientific computing and working with arrays. It was created by Travis Oliphant in 2005. It supports large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogenous data types, with many operations being performed in compiled code for performance.

3.2.7 Pandas
Pandas is an open-source Python library providing high-performance data manipulation and analysis tools using its powerful data structure called Dataframe. Pandas is built on top of the numerical library of Python named numpy. It offers data structures and operations for manipulating numerical tables and time series. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. It allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning and data wrangling (transformation) features.

3.2.8 Matplotlib
Matplotlib is a multi-platform data visualization library built on NumPy arrays. It was introduced by John Hunter in 2002. It is a comprehensive library for creating static,animated, and interactive visualizations in Python. It makes easy things easy and hard things possible. Pyplot is a Matplotlib

module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advantage of being free and open-source.

3.2.9 Seaborn

Seaborn is a Python data visualization library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. It provides a high interface for drawing attractive and informative statistical graphics. It provides beautiful default styles and color palettes to make statistical plots more attractive.

3.2.10 Jupyter Notebook/Google Colaboratory

Jupyter notebook is an open-source web application that allows researchers to create and share documents that contain live code, equations,visualizations and narrative text. It is also known as IPython Notebook. Jupyter notebook can be executed on a local computer or remote servers. Google Colaboratory is a hosted Jupyter notebook service. Colab notebooks execute code on Google's cloud servers, meaning we can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of the local machine. Colaboratory is integrated with Google Drive and Github. It allows anybody to write and execute arbitrary python code through the browser, and is especially well suited for machine learning, data analysis and education. It also allows users to share, comment, and collaborate on the same document with multiple people. Colaboratory provides 12GB RAM along with GPU and TPU. It gives 12 hours of continuous execution time in a session. After that, the whole virtual machine is cleared and needs to start again.

3.3 Dataset Generation

3.3.1 Data Collection

At first we searched for datasets on the internet. We consulted our supervisor and other experts who have worked on the Devanagari Handwritten Character Recognition project. To our dismay, We did not find the Baahrakhari dataset needed for our project. The good one was uploaded on Kaggle by Rishi Anand [1]. It consists of 92 thousand images of Devanagari Characters including 36 consonants and 10 numerals. After consulting our supervisor and other researchers who have worked on Devanagari Handwritten Character Recognition we planned for our own dataset preparation.

To make the task of dataset collection possible, we bought around 7,000 A4 size paper sheets and printed those papers with 104 rectangular boxes to make the task of collection of handwritten characters feasible and convenient. We distributed the same brand of pen with the same ink to students to fill up our boxes with the Devanagari characters.

In the first phase, we went to various schools in the Pokhara valley. We requested the school principals to manage time for our data collection. At first we shared the purpose of the task and then we instructed students on how to write the character inside the boxes in the proper way. A student had to fill up 52 characters (i.e half of the A4 paper to increase the variance in character

dataset). The students of grade four to grade twelve were used for writing the characters on the A4 paper.
Following is a sample of an A4 size paper before and after characters were written:

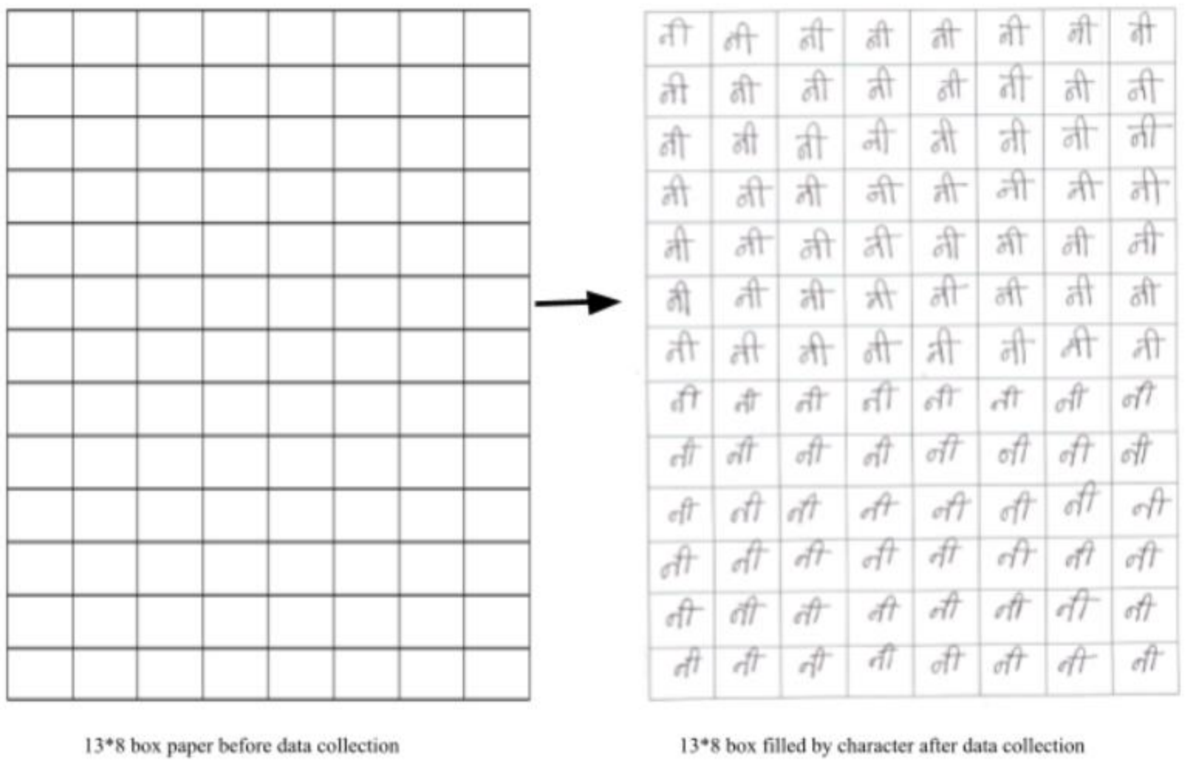

13*8 box paper before data collection
13*8 box filled by character after data collection

Figure 5: Data Collection

3.3.2 Preprocessing & Character Extraction
Since we had 104 characters in a single A4 paper, we had to extract 104 characters from the paper in a proper format by removing all the noises contained in the paper. Thus, the data collected in A4 size papers were scanned by the scanner. The scanned images were sorted by respective characters name in corresponding directory in our local system. For each character, we had 12 scanned images filled up with characters stored in .tif image format. We wrote a simple program using OpenCV Python library to generate characters(12*104) automatically and put those generated characters into the corresponding character folder. Following are the general steps taken during pre-processing of the image and extraction of characters from the image-

1. Grayscale Conversion
2. Thresholding
3. Inversion
4. Dilution
5. Contour Detection
6. Character Extraction

3.3.2.1 Grayscale Conversion

Grayscale is a black and white monochrome that uses different shades of gray. It is a group/collection of gray (monochromatic) shades, ranging from pure white on the lightest end to pure black on the opposite end without any visible color. It only contains luminance(brightness) information and no color information. White has maximum luminance and black has minimum luminance in grayscale images. It helps in dimension reduction because it converts RGB images having three dimensions (three color channels) into a single dimension. It reduces the model complexity by reducing the number of input neurons required.

3.3.2.2 Thresholding

Thresholding is a non-linear operation that converts a gray-scale image into a binary image. In thresholding pixel values are assigned into two levels in relation to the specified threshold value. Each pixel value is compared with the threshold value, if the pixel value is greater than a threshold value, it is assigned to maximum value(generally 255 or white) otherwise, it is set to minimum(0 or black) value. Threshold value varies according to the structure and quality of input image and expected output image.

3.3.2.3 Inversion

Inversion is a process of interchanging the white and black pixel. In OpenCV image inversion can be performed in two ways. One way is by subtracting the threshed image from 255. Another approach is by setting the pixel to minimum value (zero) if pixel value is greater than threshold value and setting pixel to maximum value (255) if pixel value is smaller than threshold value during image thresholding.

3.3.2.4 Dilation

Dilation is a morphological operation that typically takes a binary image as an input (expanded to grayscale image) and adds pixels to the boundaries of objects in the image. The dilation operation usually uses a structuring element for probing and expanding the shapes contained in the input image. In morphological operation, a structuring element is a shape, used to probe or interact with a given image, with the purpose of drawing conclusions on how this shape fits or misses the shapes in the image. The number of pixels added in the objects in an image depends on the size and shape of the structuring element used. In our system, we are using varied dilation parameters for boxes and character separation.

3.3.2.5 Contour Detection

Contours are the lines or points that join the continuous points of the boundary of an object in an image that are having the same intensity. Finding contours is like finding white object from black background i.e. objects to be found should be white and the background should be black. Contours come handy in shape analysis, finding the size of the object of interest, and object detection. OpenCV has a cv2.findContours function that helps in extracting the contours from the image.

3.3.2.6 Character Extraction

After contour detection, we eliminated all contours except box-contours by comparing height and width of the extracted contour with minimal box width and height and stored them in an array. Then, we considered only the box-contour(stored) as the character image and we extracted the original image using box-contour height and width. Extracted image is resized to 128*128 pixels and processed using step two to step four to generate the final character image. Resized 104*12 characters from 1*12 images which belong to a single character directory are extracted and saved to a specific folder(similar to character name) inside every character directory. Then noisy character images are eliminated manually by checking all 12*104 images. Then the images produced after filtering are considered as the final images for the particular character and are used for training using the CNN architecture.

The overall data generation steps are shown in following figure:

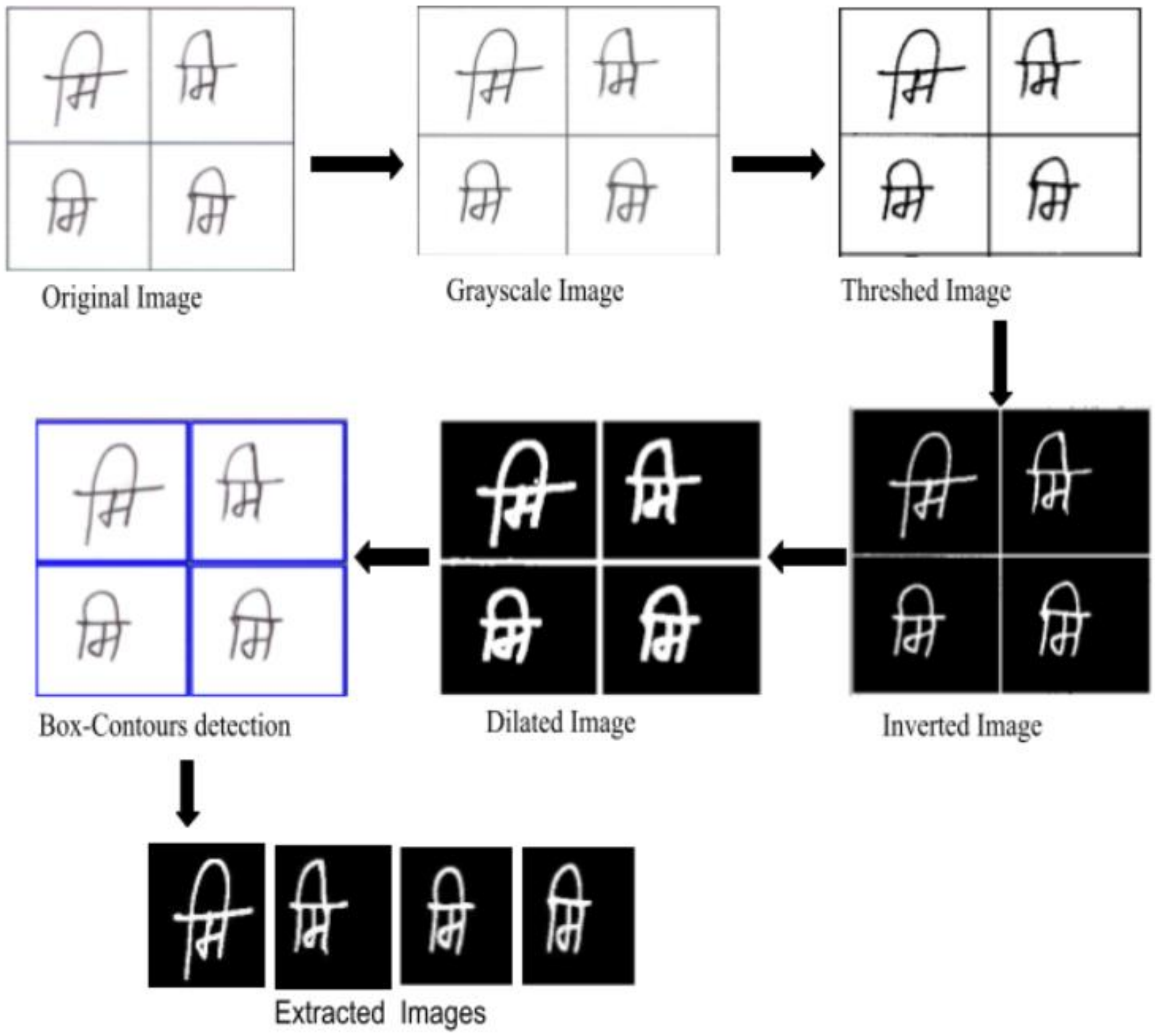


Figure 6: Data Generation Steps

3.4 Model Generation

Deep Learning is the advanced field of Machine Learning which has been widely used in the field of speech ,vision and image processing. Though the Deep learning process is much more complicated, it has been successfully applied to computer vision and image processing. It is a neural network architecture consisting of multiple layers that can learn from an image dataset. It follows the concept that the larger data supplied to the neural network the better the model can be generated with high generalization capability. So, we have tried to give large volumes of dataset in our project. We are using Convolutional Neural Network(CNN) to train our handwritten Devanagari Character dataset.

3.4.1 Convolution Neural Network (CNN)

Convolution neural network (ConvNet/CNN) was inspired by the Visual cortex of the human brain as its architecture is analogous to the connectivity patterns of neurons in the human brain. It is a Deep Learning algorithm which takes an input image, assigns weights(learnable) and biases to various objects in the images and is able to differentiate each from another one. CNN consists of one input layer, one to many hidden layers, and one output layer.Each neuron having different inputs gives a weighted sum which then passes through the activation function and responds with output. ConvNet has the ability to learn the characteristics of the image with the help of different types of filters.

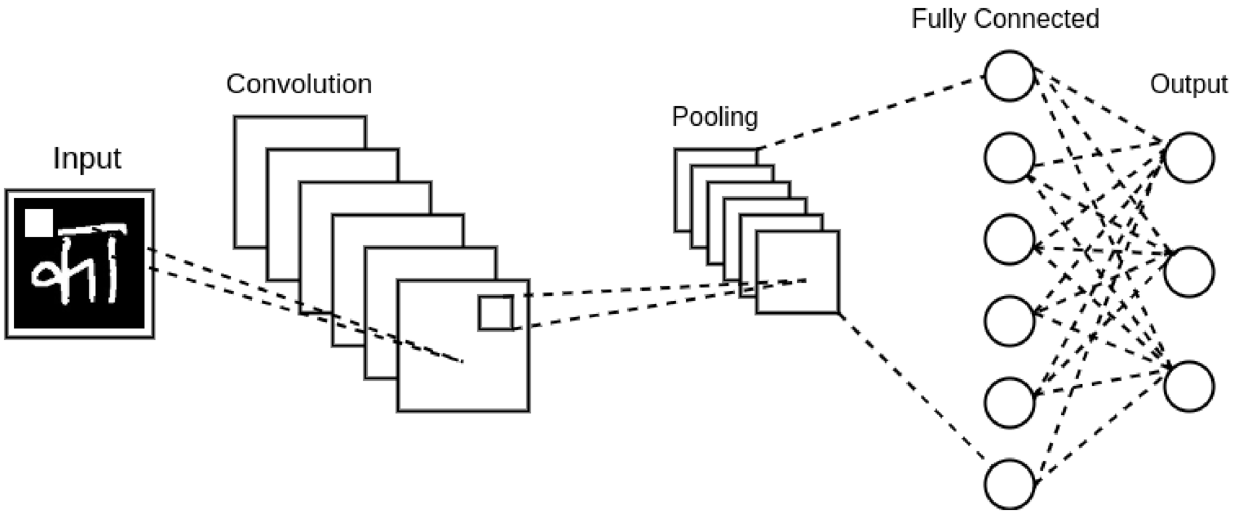


Figure 7: Convolutional Neural Network

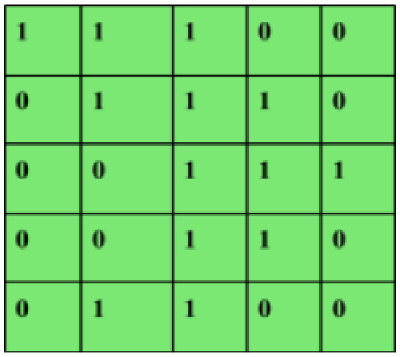
3.4.1.1 Input Layer

Input layer is the first layer of CNN model which represents the pixel matrix of input image. The input image must have fixed dimensions and size as it has to be preprocessed before feeding into the convolution layer. In our work, binary images of size 128*128 pixels are used for training, validation and testing as input.

3.4.1.2 Convolution Layer

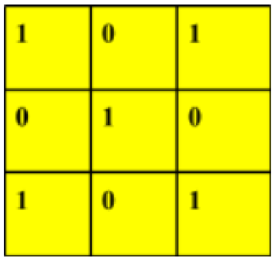
Inputs from the input layer are fed into the convolution layer which does most of the computational work. This layer is the building block of the CNN model. The primary purpose of the Convolution is to extract features from the input image. Image features of input data are preserved by it according to the relationship between pixels. It preserves the spatial relationship between pixels by learning image features using small squares of input data.It consists of two inputs: original image which is represented by a matrix of pixel values and a filter(kernel or feature detector) which is also represented by a matrix. Filter(filter matrix) will produce different Feature Maps(also called convolved feature) for the same input image by performing operations like edge detection(vertical, horizontal), blurring, sharpening etc.

As we discussed above, every image can be considered as a matrix of pixel values. Consider a 5 * 5 image size whose pixel values are only 0 and 1.



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

Also, consider a 3 * 3 matrix (filter) as shown below:



1	0	1
0	1	0
1	0	1

Then, the convolution of the 5 * 5 image and 3 *3 filters can be computed as shown in figure below.

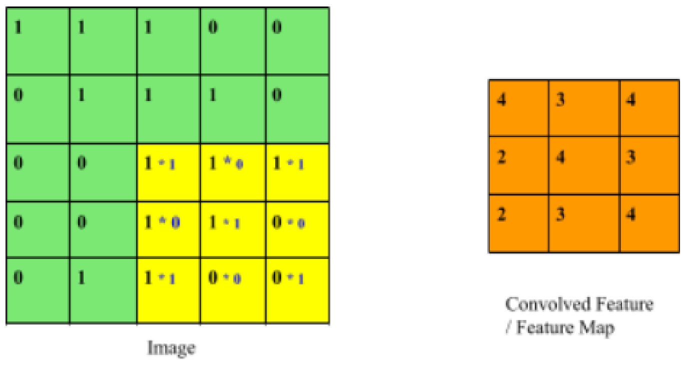


Figure 8: Convolution Operation

In the above figure, the orange matrix is slided by 1 pixel and for every position in the input green image. Then element wise multiplication is computed between two matrices and multiplication outputs are added to get a final integer which forms a single element of the output matrix which is called Feature Map. Thus, the matrix formed by sliding the filter over the image and computing the dot product is called the 'Feature Map' or 'Convolved Feature'. It is evident from the above operation that different values of the filter matrix will produce different Feature Maps for the same input image. The size of Convolved feature/Feature Map is calculated as follows:

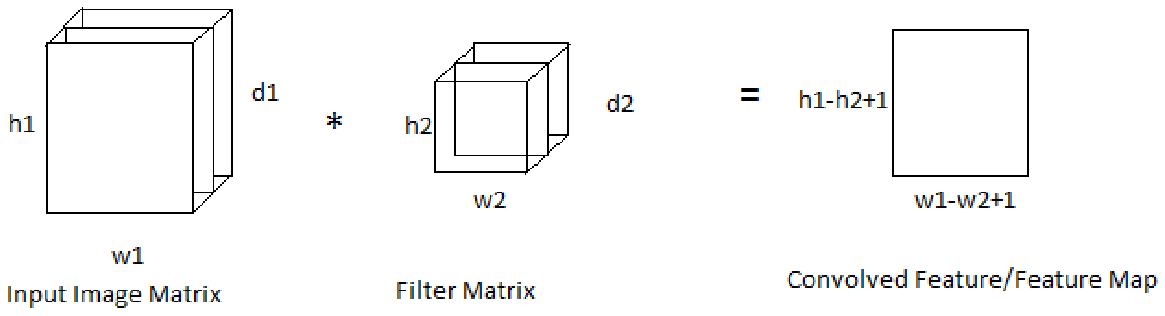


Figure 9: Size Feature Map Calculation

In practice, CNN learns the values of filters on its own during the training process (but needs to specify parameters such as number of filters, filter size, architecture of network etc. before the training process). The more the number of filters we have, the more images feature gets extracted and better our network becomes at recognizing patterns in unseen images.

The size of the Feature Map is controlled by following three parameters that we need to decide before the convolution step is performed.

3.4.1.2.1 Depth
Depth refers to the number of filters we use for the convolution operation. In the network shown in figure, we are performing convolution of the original character image using ‘n’ distinct filters, thus producing n different feature maps as shown. Here, ‘n’ varies according to the architecture of the CNN. These ‘n’ feature maps can be considered as stacked 2D matrices, so depth of the feature map would be ‘n’.

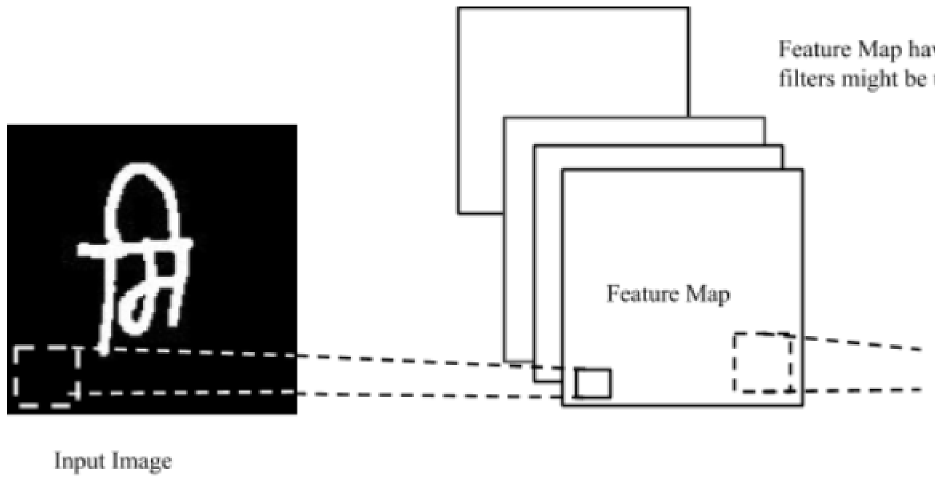


Figure 10: Depth in Convolution Operation

3.4.1.2.2 Stride
Stride is the number of pixels by which we slide the filter (kernel) matrix over the input matrix. When the Stride is 1, then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide around the input matrix. Having a smaller stride will produce larger feature maps.

3.4.1.2.3 Padding
Padding is important to preserve the size of the matrix passing through one layer to another. It is the process of fitting the filters perfectly over the input image. Adding zero to the outer layer(around the border) of the image matrix is called zero-padding(wide convolution) and dropping out the layers of the image matrix where the filter doesn’t fit perfectly is valid padding(narrow convolution)[10]. Generally it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. Zero padding allows us to control the size of the feature maps.

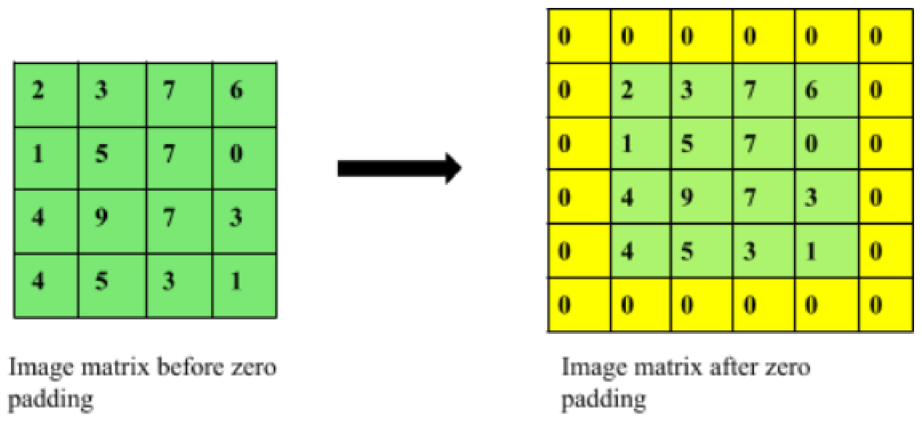


Figure 11: Padding Operation

3.4.1.3 Activation Function

Activation function is the deciding end which decides what is to be fired to the next neuron i.e, it decides whether the neuron should be activated or not. It is a node that you add to the output end of any neural network. It is also known as transfer function. It can also be attached in between two neural networks. It is used to determine the output of a neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. depending upon the function. It is a function used in artificial neural networks which outputs a small value for small inputs, and a larger value if its inputs exceed a threshold.

Activation functions are useful because they add non-linearities into neural networks, allowing the neural networks to learn powerful operations. If the activation functions were to be removed from a feedforward neural network, the entire network could be refactored to a simpler linear operation or matrix transformation on its input, and it would no longer be capable of performing complex tasks such as image recognition. Following are few widely used activation functions:

3.4.1.3.1 ReLU

ReLU stands for rectified linear unit and is a non-linear operation. The purpose of ReLU is to introduce non-linearity in ConvNet, since most of the real-word data we would want our ConvNet to learn would be non-negative linear values. Its output f(x) is given by

Output $f(x) = \text{Max}(0, x)$,

Where, x is the input pixel in the feature map.

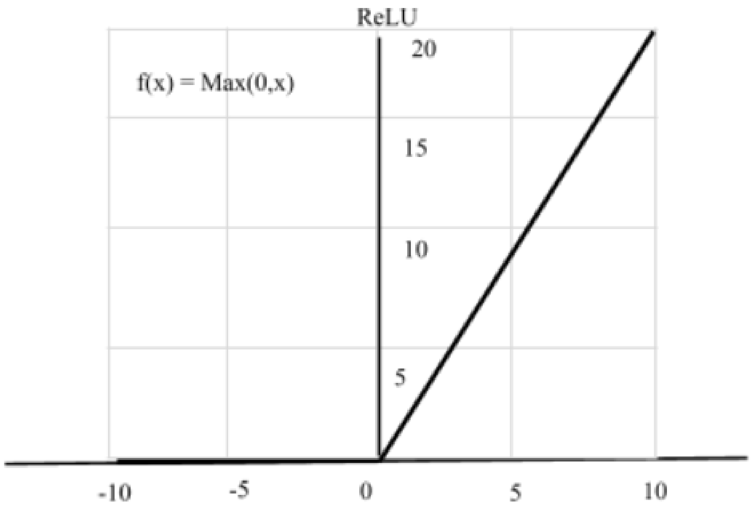


Figure 12: ReLU

ReLU is an element-wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero and positive value remains unchanged. The output of the feature map on which ReLU operation is applied is called Rectified feature map.

There are other nonlinear functions such as tanh,sigmoid etc which can be used instead of ReLU. In General, ReLU is used because it’s performance is better than others.

3.4.1.3.2 Softmax

The softmax function, also called a normalized exponential function, turns numbers(logits) into probabilities that sum to one. It takes a vector of K real numbers as input and normalizes it into a probability distribution consisting of K probabilities proportional to the exponential of the input numbers, i.e, prior to applying softmax, some vector components could be negative,or greater than one,and might not sum to 1,but after applying softmax ,each component will be in the range zero to one and sum will be 1. It is normally applied to the very last layer in the neural network and is equivalent to a categorical probability distribution. Mathematically, the softmax function is given by

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1,\ldots,K$$

Where z is a vector of the inputs to the output layer (eg. If you have 5 output units, then there are 5 elements in z).

3.4.1.4 Pooling Layer

Pooling is the process of reducing the dimensionality of each feature map with the consideration of retainment of important information. It is used when the size of an image is very large and to reduce the number of parameters. A pooling layer is added after the convolution layer, specifically, after a nonlinearity (e.g. ReLU) has been applied to the feature maps output by a convolution layer. Pooling may be of different types like max-pooling, average pooling and sum pooling. Max-pooling returns the largest element value from a rectified feature map. Average pooling returns the average of all the elements from a rectified feature map. Sum pooling returns the sum of all the elements from a rectified feature map. Among all, max-pooling is the most popular one and widely used.

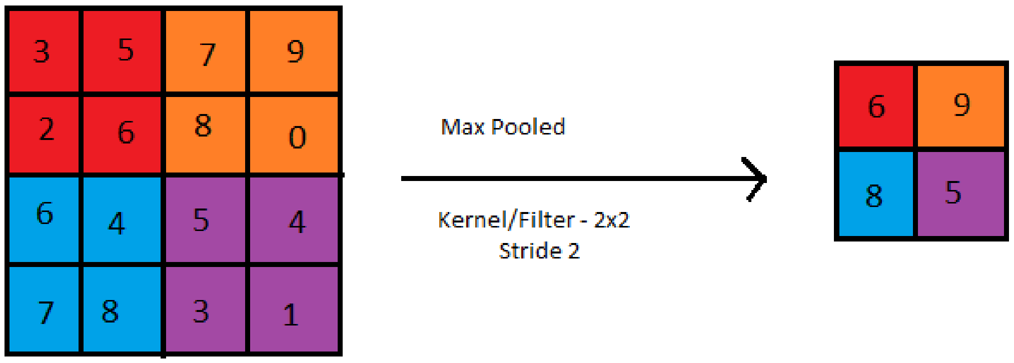


Figure 13: Max Pooling

In the network shown in the following figure, pooling operation is applied separately to each feature map.

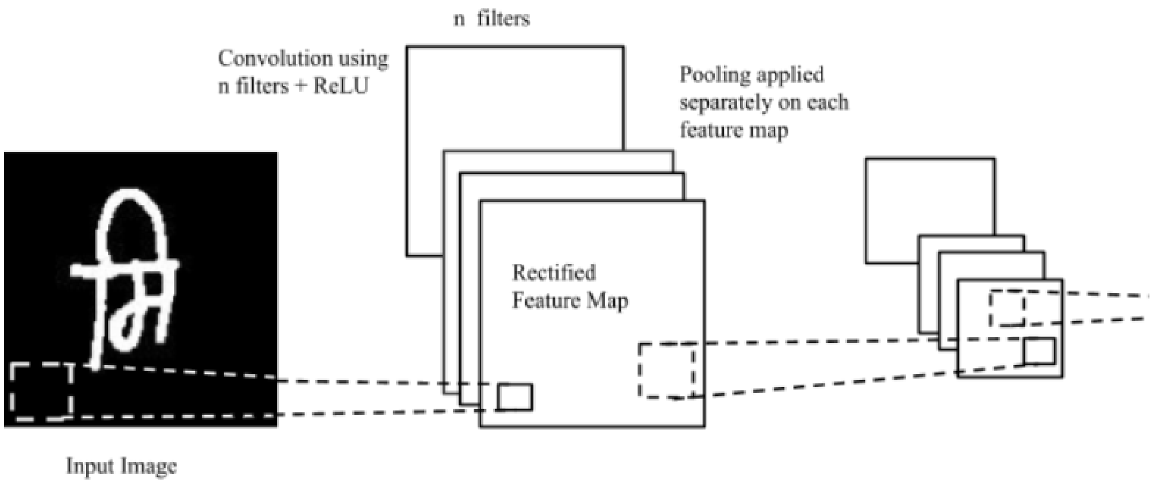


Figure 14: Pooling Applied To Rectified Feature Maps

3.4.1.5 Flatten Layer

Flattening is the process of converting the data into the one dimensional array for inputting it to the next layer. It is used for creating a single long feature vector. Then, such a vector is passed to the fully connected layer for further processing.

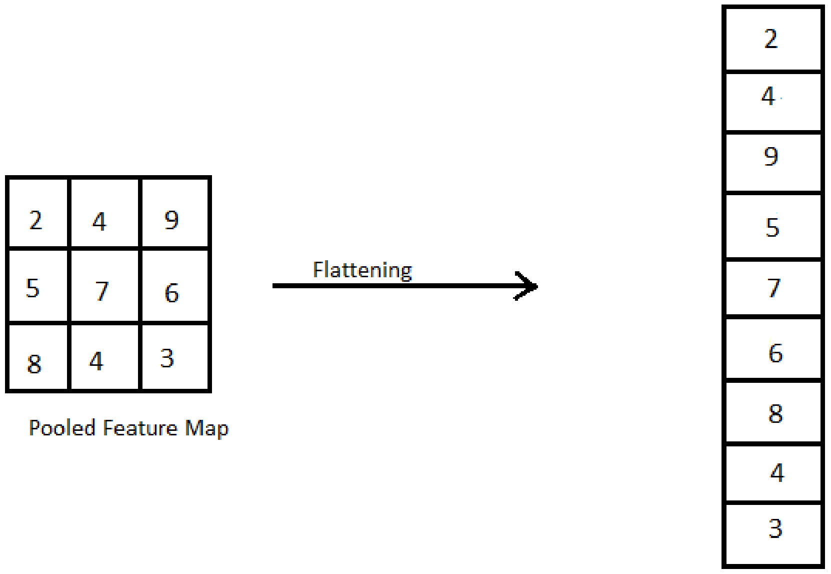


Figure 15: Flattening

3.4.1.6 Fully Connected Layer

Fully Connected Layer is simply a feed forward neural network. The objective of a fully connected layer is to take the results of the convolution/pooling process and use them to classify the image into a label. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer. Later, we have activation functions such as softmax or sigmoid for classifying the outputs in the different class labels such as क, का, कि, की.... (softmax activation)

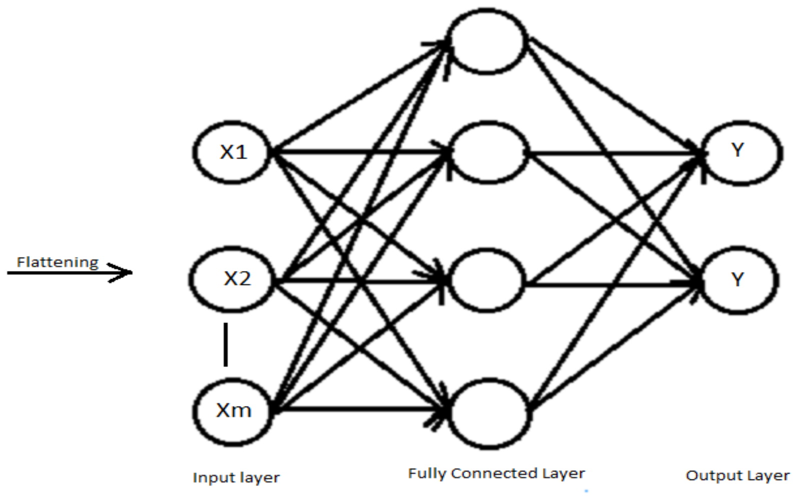


Figure 16: Fully Connected Layer

3.4.1.7 Dropout

The reason that deep learning neural networks are likely to overfit a training dataset is the necessity of dropout which avoids overfitting from randomly taken neurons by dropping out nodes during training. Overfitting means an increase in test loss as we keep improving in training loss. So, dropout improves generalization in deep neural networks by preventing complex co-adaptations on training data. It is the way of model averaging with neural networks which drop out both hidden and visible nodes.

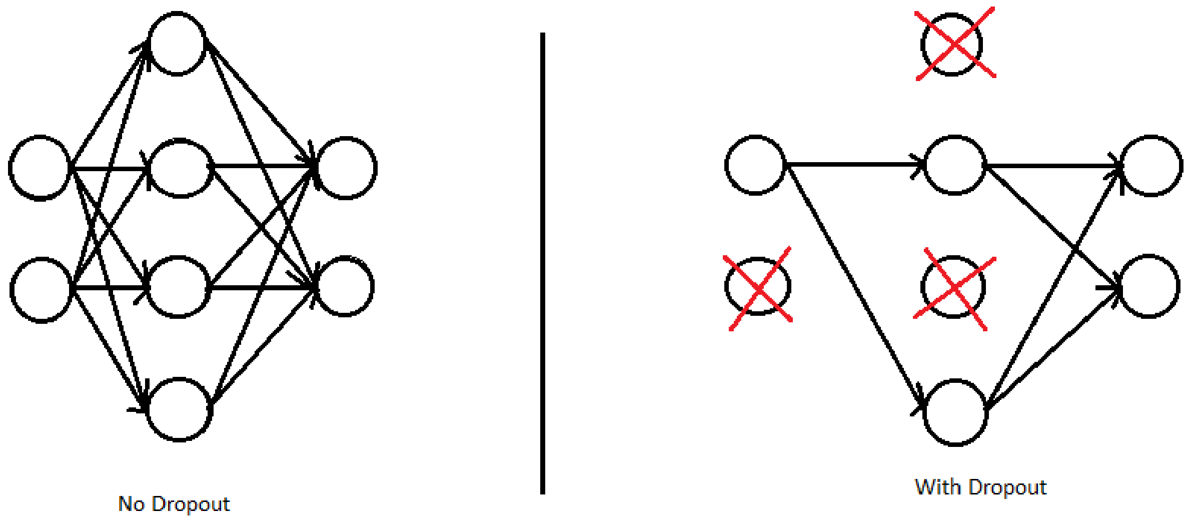


Figure 17: Use of Dropout Layer

3.4.1.8 Batch Normalization

Normalization comes in the process of neural networking because of the wider range between data points. So, that it is applied to overcome the problem of exploding gradient descent and to outline large weights in the neural network. Batch normalization makes CNN faster and more stable by rescaling between new data points and solves the issue of low training speed due to non-normalized data. It normalizes output from activation function and uses mean, standard deviation and arbitrary parameters which all are trainable to rescale the data points.

3.4.2 Optimization Algorithms

Optimization refers to a procedure for finding the input parameters or arguments to a function that results in the minimum or maximum output of a function. Training a complex deep learning model can take hours, days or even weeks. The performance of the optimization algorithm directly affects the model's training efficiency. After experimenting with various optimizers such as gradient descent, Stochastic Gradient Descent (SGD), RMSProp and Adam, we found out that Adam optimizer provided us with the best accuracy for our model.

3.4.2.1 Adam Optimization Algorithm

Adam [11] is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks. Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using the moving average of the gradient instead of the gradient itself like SGD with momentum. Adam computes individual learning rates for different parameters and the learning rate does not change during training.

The list of the attractive benefits of using Adam optimizer are:
• Straightforward to implement
• Computationally efficient
• Little memory requirements
• Appropriate for problems with very noisy/or sparse gradients.

3.4.2.2 Stochastic Gradient Descent

Stochastic gradient descent is variation on gradient descent, also called batch gradient descent.It is an optimization algorithm used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs.[12]In Stochastic gradient descent(SGD), the user initializes the weights and the process updates the weight vector using one data point. The gradient descent continuously updates it incrementally when an error calculation is completed to improve convergence. The method seeks to determine the steepest descent and it reduces the number of iterations and time taken to search large quantities of data points.It decreases machine computation time while increasing complexity and performance for large-scale problems.

3.4.3 Proposed Model

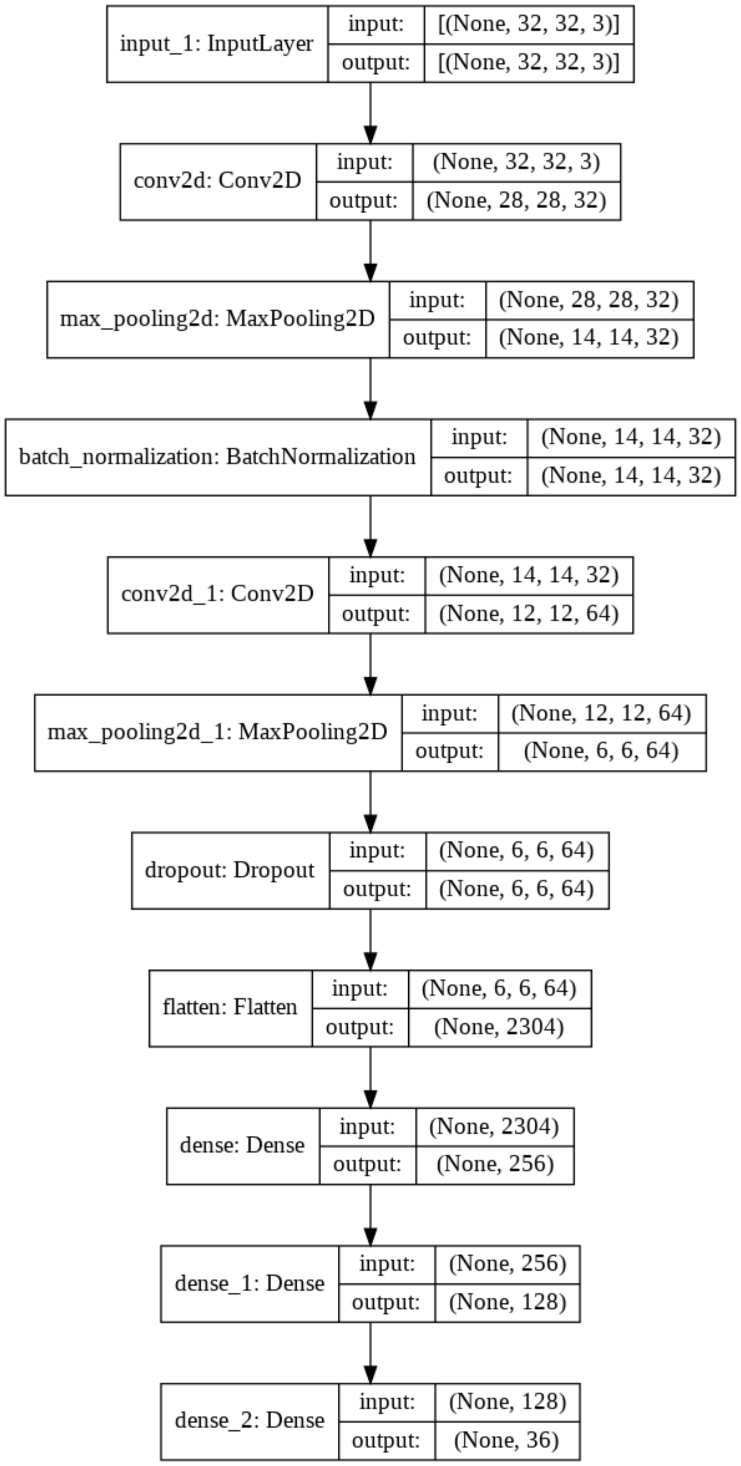

input_1: InputLayer | input: [(None, 32, 32, 3)] | output: [(None, 32, 32, 3)]
conv2d: Conv2D | input: (None, 32, 32, 3) | output: (None, 28, 28, 32)
max_pooling2d: MaxPooling2D | input: (None, 28, 28, 32) | output: (None, 14, 14, 32)
batch_normalization: BatchNormalization | input: (None, 14, 14, 32) | output: (None, 14, 14, 32)
conv2d_1: Conv2D | input: (None, 14, 14, 32) | output: (None, 12, 12, 64)
max_pooling2d_1: MaxPooling2D | input: (None, 12, 12, 64) | output: (None, 6, 6, 64)
dropout: Dropout | input: (None, 6, 6, 64) | output: (None, 6, 6, 64)
flatten: Flatten | input: (None, 6, 6, 64) | output: (None, 2304)
dense: Dense | input: (None, 2304) | output: (None, 256)
dense_1: Dense | input: (None, 256) | output: (None, 128)
dense_2: Dense | input: (None, 128) | output: (None, 36)

Figure 18: Proposed Model

CHAPTER 4: EPILOGUE

4.1 Experiment and discussion

We tested the dataset with different architectures by varying depth and number of parameters of the network. Following table shows the architecture of the tested models with parameters.

Network Architecture	**Version**	**Layer Type**	**Layer Size**	**Trainable Parameters**	**Total Parameters**
NA 1	1.0	Conv2D	28x28x64	4864	356,292
		Max_Pool.	14x14x64	0	
		Batch_Norm.	14x14x64	256	
		Conv2D	12x12x32	18464	
		Max_Pool.	6x6x32	0	
		Dropout	6x6x32	0	
		Flatten	1152	0	
		Dense	256	295168	
		Dense	128	32896	
		Dense	36	4644	
NA 2	1.0	Conv2D	28*28*64	4868	328,132 (trainable:327,940)
		Batch_Norm.	28*28*64	256	
		Activation	28*28*64	0	
		Max_Pooling	14*14*64	0	
		Dropout	14*14*64	0	
		Conv2D	12*12*32	18464	
		Batch_Norm.	12*12*32	128	
		Activation	12*12*32	0	

		Max_Pooling	6*6*32	0	
		Dropout	6*6*32	0	
		Flatten	1152	0	
		Dense	256	295168	
		Activation	256	0	
		Dense	36	9252	
		Activation	36	0	

4.2 Result Analysis

With experimenting with the various CNN architecture, we observed the following training and validation accuracy in respective network architecture and parameters.

Architecture	**Version**	**Accuracy**	**Optimizers** (Adam)
NA1	1.0	Train Validation	99.15 % 91.94 %
NA2	1.0	Train Validation	99.75 91.68

After experimenting with the above CNN architecture, we observed an training accuracy of 99.15% and validation accuracy of 91.94 %.

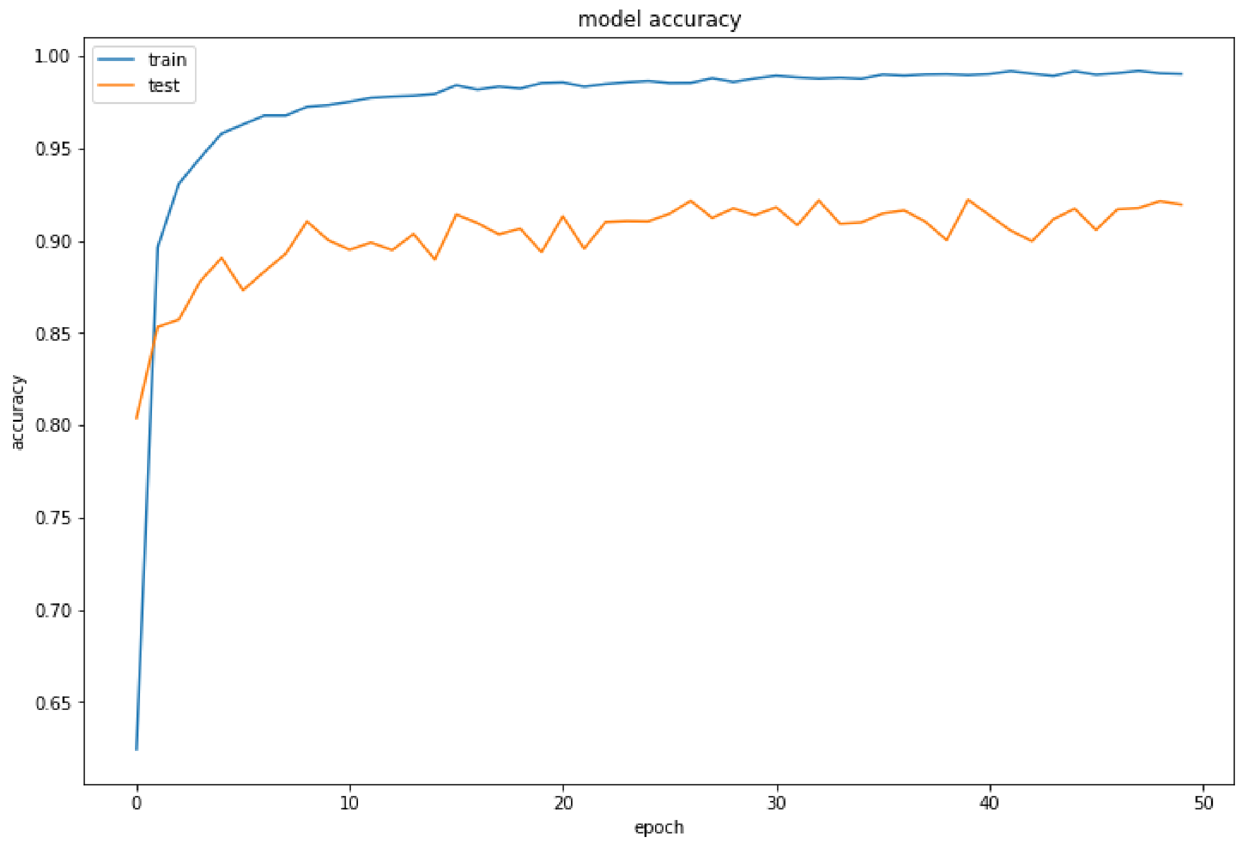


Figure 19: Epoch vs Accuracy Plot

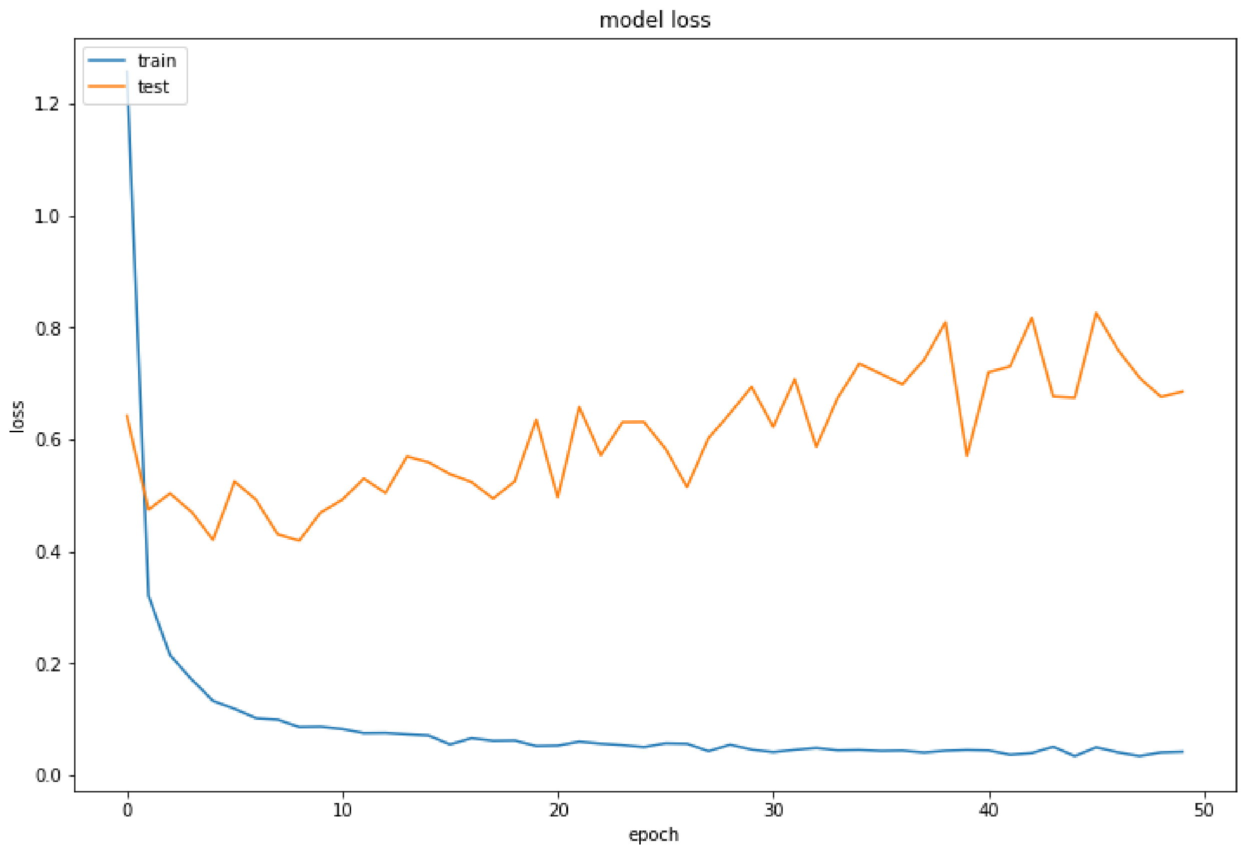


Figure 20: Epoch vs Loss Plot

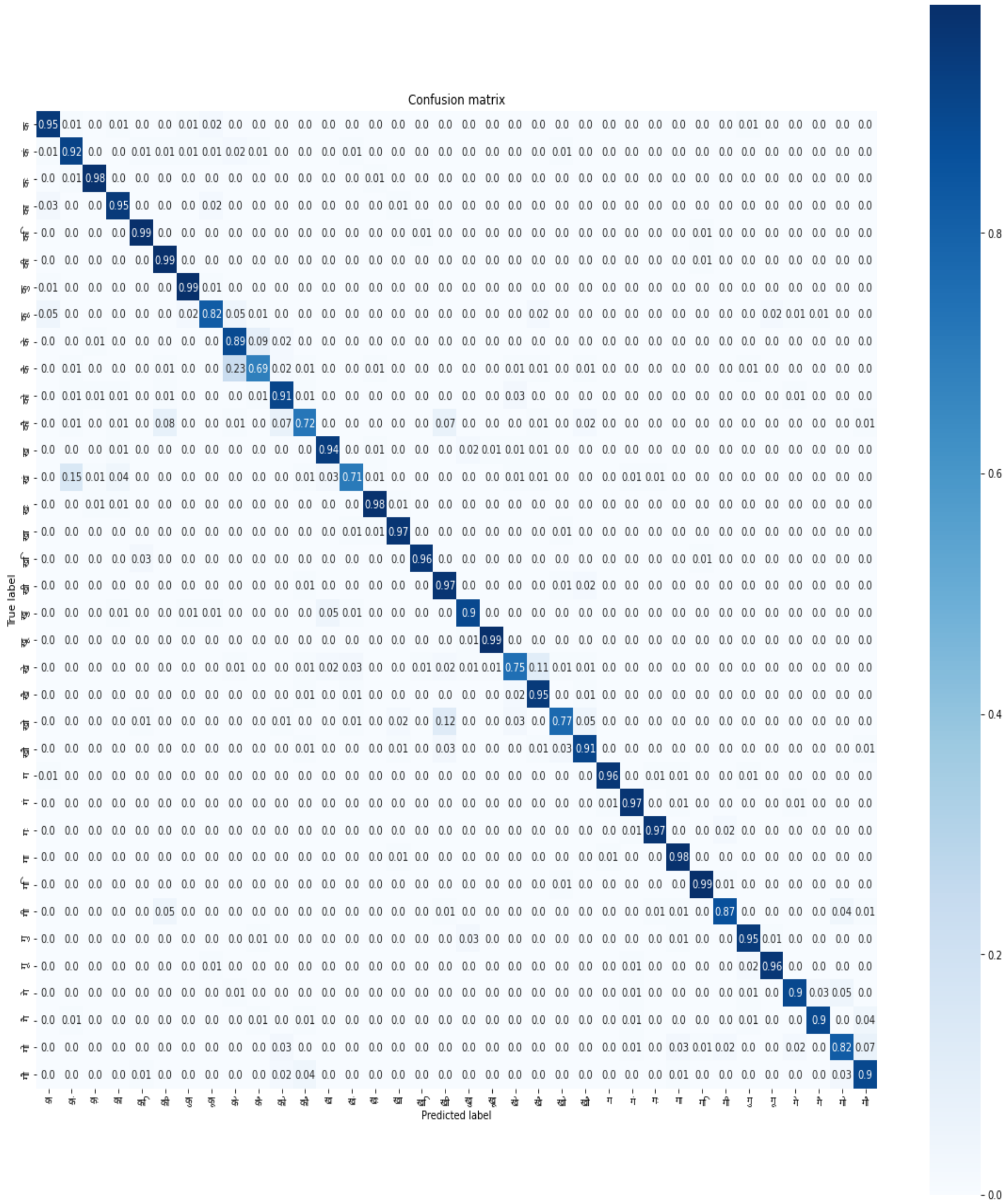


Figure 21: Confusion Matrix

4.3 Conclusion

The main problem in our project was to collect the large amount of Devanagari character dataset since there was no efficient dataset in Nepali Barhakhari available till now.So, we sorted out this problem by collecting the handwritten nepali character datasets with the help of different students of different schools and colleges.We had tried to make out variations in dataset as much as possible so that we could get abundance different data. After preprocessing of data, we got the data required for training our model. We trained our data in Google Colab as a cloud service.With the project we conclude that the proposed architecture achieves the training accuracy of 99.75% and validation accuracy of 91.68%.

4.4 Further enhancements

- To work on the Devanagari word recognition and not just the Devanagari character recognition.
- To develop a mobile application and a web application for recognizing the Devanagari Script.

REFERENCES

[1] https://www.kaggle.com/rishianand/Devanagari-character-set
[2] Optical Character Recognition. (1993, December). Line Eikvil.

[3] "OCR Introduction". Dataid.com. Retrieved June 16, 2013.

[4] Wikipedia contributors. (2021, February 27). Optical character recognition. Wikipedia. https://en.wikipedia.org/wiki/Optical_character_recognition#cite_note-23

[5] Liedle, D. (2018, November 9). A Brief History of Optical Character Recognition. Filestack Blog.https://blog.filestack.com/thoughts-and-knowledge/history-of-ocr/#:%7E:text=In%201974%2C%20Kurzweil%20Computer%20Products,English%2C%20using%20the%20Latin%20alphabet.

[6] K. Gaurav and Bhatia P. K., "Analytical Review of Preprocessing Techniques International for Offline Conference Handwritten on Emerging Character Trends Recognition",in 2nd Engineering & Management, ICETEM, 2013.

[7] S. D. Connell, R. M. K. Sinha and A. K. Jain, "Recognition of unconstrained online Devanagari characters," Proceedings 15th International Conference on Pattern Recognition. ICPR-2000, Barcelona, Spain, 2000, pp. 368-371 vol.2, doi: 10.1109/ICPR.2000.906089.

[8] S. Acharya, A. K. Pant and P. K. Gyawali, "Deep learning based large scale handwritten Devanagari character recognition," 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Kathmandu, Nepal, 2015, pp. 1-6, doi: 10.1109/SKIMA.2015.7400041.

[9] P. Ajmire, "Handwritten Devanagari Vowel Recognition using Artificial Neural Network", International Journal of Advanced Research in Computer Science, vol.8, no. 7, pp. 1059-1062, 2017. Available: 10.26483/ijarcs.v8i7.4560.

[10] https://cs231n.github.io/convolutional-networks/

[11] Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. 2014. arXiv:1412.6980v9

[12] https://optimization.cbe.cornell.edu/index.php?title=Stochastic_gradient_descent