PASHCHIMANCHAL CAMPUS

INSTITUE OF ENGINEERING

TRIBHUVAN UNIVERSITY

Lamachour, Pokhara

A FINAL YEAR REPORT

ON

"NEPALI HANDWRITING RECONGNITION"

SUBMITTED BY:

Alima Subedi [72BCT603]

Asim Sharma [72BCT606]

Jasmine Baral [72BCT624]

Rabinson Ghatani [72BCT635]

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

September, 2019

PASHCHIMANCHAL CAMPUS

INSTITUE OF ENGINEERING

TRIBHUVAN UNIVERSITY

Lamachour, Pokhara

A FINAL YEAR REPORT

ON

"NEPALI HANDWRITING RECONGNITION"

SUBMITTED BY:

Alima Subedi [72BCT603]

Asim Sharma [72BCT606]

Jasmine Baral [72BCT624]

Rabinson Ghatani [72BCT635]

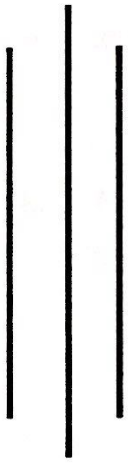
SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

September, 2019

NEPALI HANDWRITING RECOGNITION

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



SUBMITTED BY:

Alima Subedi [72BCT603]
Asim Sharma [72BCT606]
Jasmine Baral [72BCT624]
Rabinson Ghatani [72BCT635]

SUPERVISED BY:

Er. Bal Krishna Nyaupane

SUBMITTED TO:
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

September, 2019

त्रिभुवन विश्वविद्यालय
Tribhuvan University
इन्जिनियरिङ अध्ययन संस्थान
Institute of Engineering

पश्चिमाञ्चल क्याम्पस
PASHCHIMANCHAL CAMPUS

PO box- 46, Lamachaur, Pokhara
Tel: 977-061-440457, 440002, 440465,
Fax: 977-061-440158
info@wrc.edu.np, www.wrc.edu.np
info@ioepas.edu.np, www.ioepas.edu.np
पो.ब. न- ४६, लामाचौर, पोखरा
फोन- ९७७-०६१-४४०४५७, ४४००००२,४४०४६५,
फ्याक्स- ९७७-०६१-४४०१५८

LETTER OF APPROVAL

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance of the project report entitled **"Nepali Handwriting Recognition"** submitted by **Alima Subedi** [72BCT603], **Asim Sharma** [72BCT606], **Jasmine Baral** [72BCT624], **Rabinson Ghatani** [72BCT635] 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: 22nd September, 2019

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 bot been submitted to any other university or Institution for the award of any degree.

.....Alima.....

Alima Subedi [72BCT603]

.....

Asim Sharma [72BCT606]

.....Jasmine.....

Jasmine Baral [72BCT624]

.....

Rabinson Ghatani [72BCT635]

DATE: 22nd September, 2019

COPYRIGHT

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the major project undertaken during 4th year electronic and computer engineering. We owe special debt of gratitude to our supervisor, Er. Bal Krishna Nyaupane for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us.

Our deep sense of gratitude is accorded to HOD of electronic and computer department, Er. Hari Prasad Baral for his encouragement and motivation for our project work. We also express our gratitude and appreciation to all our juniors, school students, friends and family for their support and helping us out with dataset preparation.

We also don't like to miss the opportunity to acknowledge the contribution of all faculty member and lab-Instructors of the department for their kind assistance and cooperation during the development of our project. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in completing this project as well as in the preparation of this report.

Any suggestions for the further improvement of our project are hearty welcomed.

ABSTRACT

Handwriting recognition has been a research of interest among researchers because of possible applications in assisting technology for blinds and visually impaired, automatic data entry for business documents. However, very less effort has been made for the Devanagari script. In this work, we have proposed a deep learning architecture to recognize handwritten Devanagari characters using Convolution Neural Network
(CNN). We have also introduced new datasets of Devanagari script which consists of 48 different classes each having 1000 images. The classes are Barakhari letters (क, का, ... कः, ख, खा, ...खः, ग, गा, ... गः, घ, घा, ... घः, which are the compound letters with vowel diacritics of the very first four consonants of Devanagari script. We experimented our dataset along with the DHCD [1] with different CNN architecture with varying parameters for performance evaluation and improvements. By implementing these techniques in CNN, we are able to achieve highest test accuracy of 96.48 % and training accuracy of 99.58% on our dataset.

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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN: Convolution Neural Network

ANN: Artificial Neural Network

MLP: MultiLayer Perceptron

RBF: Radial Basis Function

PCA: Principle Component Analysis

GFF: Generalized FeedForward

SVM: Support Vector Machine

SGD: Stochastic Gradient Descent

DHCH: Devanagari Handwritten Character Dataset

CHAPTER 1: INTRODUCTION

1.1 Background

1.1.1 Handwriting Recognition

Handwriting Recognition is the mechanism for converting the handwritten text into a notational representation. It is a special problem in the domain of pattern recognition and machine intelligence. Automatic handwriting recognition have many application areas like postal addresses reading, bank check verification, ancient document digitalization, handwritten form verification, forensic and medical analysis, etc. The field of handwriting recognition can be split into two different categories: on-line recognition and off-line recognition. Online mode deals with the recognition of handwriting captured by a tablet or touch-screen device, and use the digitized trace of the pen to recognize the symbol. In the on-line case we have the captured trajectory, pan up and pen down time, stroke orders, etc. of the written characters. Off-line mode deals with the recognition of the character or word present in the digital image of written text. In this case, we just have the holistic image of the written text.

1.1.2 Devanagari Scripts

Nepali Language uses a script called The Devanagari to facilitate record keeping and conversation. This script is unique in the sense that it features a characteristic horizontal line above all the letter. In this script, there are vowels and consonants, which have separate characters. There are three possible sounds we can make: vowels sound, consonant sound, and consonant + vowel sound. The script is phonetic in nature; it is spoken as it is written. Furthermore, it is an alpha-syllabary script. This means, apart from having separate character for each vowel and consonant, when we express a consonant + vowel sound, the vowel is added directly to the consonant. This can be seen as diacritic on the consonant.

There are total of 12 vowels in the Nepali Language and 36 consonants on the script. Devanagari has no capital letters and is written left-to-right horizontally.

The consonant letters with the vowel diacritics in the Devanagari script are known as Bahrakhari letters.

अ आ इ ई उ ऊ ए ऐ ओ औ अं अः

Figure 1 Vowels (Sworvardna)

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 (Byanjanvarna)

क का कि की कु कू के कै को कौ कं कः
ख खा खि खी खु खू खे खै खो खौ खं खः
ग गा गि गी गु गू गे गै गो गौ गं गः
घ घा घि घी घु घू घे घै घो घौ घं घः

Figure 3 Consonants with vowels diacritics (Bahrakhari)

1.2 Problem Statement

Papers are replaced by digital documents for various reasons. However, we still see a lot of paper documentation in our daily life. Machines do not have the ability to understand what has been written on those physical papers. Converting handwritten characters to digital characters has been a tough problem in the past and continues to be. We cannot efficiently process those physical documents with computers unless we can convert them to digital documents.

1.3 Objectives

- To create new dataset for the Nepali consonant characters with vowel diacritics.
- To create the architecture/model for the recognition of off-line Nepali handwritten characters.

- To focus on the refinement and adaption of the model to acquire maximum accuracy for recognition.

1.4 Application

- Official or Personal writing document into typed format which is convenient to read or transfer.
- Conversion of old literature into digitized form.
- Proposed system served as guide and working in character recognition area.

CHAPTER 2: LITERATURE REVIEW

Although Computer Vision has recently been subject of attention to many researchers (the breakthrough moment happened in 2012 when AlexNet won ImageNet), it certainly isn't a new scientific field. It has nearly been 60 years that scientists been trying to find ways to make machines extract meaning from visual data. The most influential paper in Computer Vision was published by two neurophysiologists – David Hubel and Torsten Wiesel- in 1959, entitled "Receptive fields of single neurons in the cat's striate cortex", described core response properties of visual cortical neurons.

The neocognitron was introduced by Kunihiko Fukushima in 1980. It was inspired by above mention work of Hubel and Wiesel. It introduced two basic types of layers in CNNs: convolutional layer an down-sampling layers. The necognitron is the first CNN which require units located at multiple network positions to have shared weights [1].

LeNet was one of the very first convolutional neural networks proposed by Yann LeCun. It consists of two set of convolution and average pooling layers, followed by flattening convolutional layer, then two fully-connected layers and finally a softmax classifier. LeNet-5 was used on large scale to automatically classify hand-written digits on bank cheques in the United States. Character recognition had been done mostly by using feature engineering by hand, followed by a machine learning model to learn to classify hand engineered features. LeNet made hand engineering features redundant, because the network learns the best internal representation from raw images automatically [2].

One of the earliest attempts in character recognition was of Grimsdale et al. in 1958 where in a pioneering work they described a method where the input pattern is scanned by a flying spot scanner; subsequently it is analyzed for shape by a digital computer which extracts the basic features [4].

In 1966 Casey and Nagy at IBM presented one of the first attempts in Chinese characters recognition. In 1968 Casey and Nagy introduced an unconventional approach to character recognition, the so-called "autonomous reading machine",

since no previous training or a priori information about the character was needed, but instead the known letter-pair frequencies of the language were used for recognition [3].

K.Gaurav, Bhatia P.K. [5] Et al, this paper deals with the various preprocessing techniques involved in the character recognition with different kind of images ranges 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 possible to achieve the fully accuracy in a preprocessing system.

In [6], describes character recognition based on the neural networks. Two approach was experimented: Multilayer Perceptron (MLP) and Radial Basis Function (RBF) classifiers. The system was tested with three datasets for Nepali Handwritten numerals, vowels and consonants. The strength of that research was the efficient feature extraction and the comprehensive recognition techniques. This paper claims to have achieve the recognition accuracy of 94.44% for numeral dataset, 86.04% for vowel dataset and 80.25% for consonant dataset. It concluded that the RBF based recognition system outperforms the MLP based recognition system but RBF based recognition system takes little more time while training. However, the obtained accuracy is not enough for implementing the system for real life application and also the prevalent compound character was not even experimented with.

This [7] paper introduces new public image dataset of Devanagari Script: DHCD which consists of 92 thousand images of 46 different classes of characters of Devanagari script segmented from handwritten documents. It has also proposed a deep learning architecture for recognition of those character. With this the paper claimed to have achieve the highest accuracy of 98.47 %. However, this project doesn't address the character obtained from the combination of consonants and

vowels which we call Bahrakhari letters that are more prevalent in the Nepali Language

This [8] paper has developed the CNN based Optical Character Recognition System for Nepali Language, in python using Keras library on to of Theano and numpy. The system was trained using a set of real world and synthesized datasets 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 about the number of data used to train and also the dataset doesn't contain character obtained by consonants with vowels and the combination of consonants and vowels with special symbols.

This [9] paper proposes an approach for extracting features in context of Handwritten Devanagari Vowels recognition in which ANN is used for classification technique. The histogram oriented gradient features were extracted. The extracted features then selected for the classification. Different ANN models such as MLP, PCA, GFF and SVM were used for the classification process. The paper claims that the vowel recognition system using SVM classifier obtained 100% accuracy on training data, 85.74% on testing data and 87.60% on cross validation.

CHAPTER 3: METHODOLOGY

Starting from data collection to building a model that can effectively recognize the Nepali character there are various steps that we followed. First the data collection and then pre-processing the data that can be used in the CNN neural network then finally training the model various time to get optimum result.

3.1 Data collection

Data collection was the very first step we carried out where we gathered data from various sources. This step of data collection was carried out in two methods: a) Collect handwritten data from friends, juniors, school students, etc. and b) Synthesize new data from the already prepared data sets. For the first method, the school students, friends, juniors, and family members were asked to provide us their handwritten Nepali Bahrakhari characters in the provided A4 sized papers. For the second method, the dataset available in the internet, DHCD [7], which consists of the Nepali consonants characters only was used in order to synthesize new data i.e. the consonants with the vowel diacritics.

3.2 Data Preprocessing

The data collected in A4 sized paper were scanned with CamScanner (mobile application) for implementing following preprocessing steps. Noise removal was also done by the application. Then the scanned images were fed into the system for further processing and extraction.

. The preprocessing steps are:

1. Grayscale Conversion
2. Thresholding
3. Dilation
4. Contour detection
5. Resizing

3.2.1 Grayscale Conversion

Grayscale is the collection or the range of monochromic (gray) shades, ranging from pure white on the lightest end to pure black on the opposite end. Grayscale only contains luminance (brightness) information and no color information; that is why maximum luminance is white and zero luminance is black; everything in between is a shade of gray. Hence grayscale images contain only shades of gray and no color.

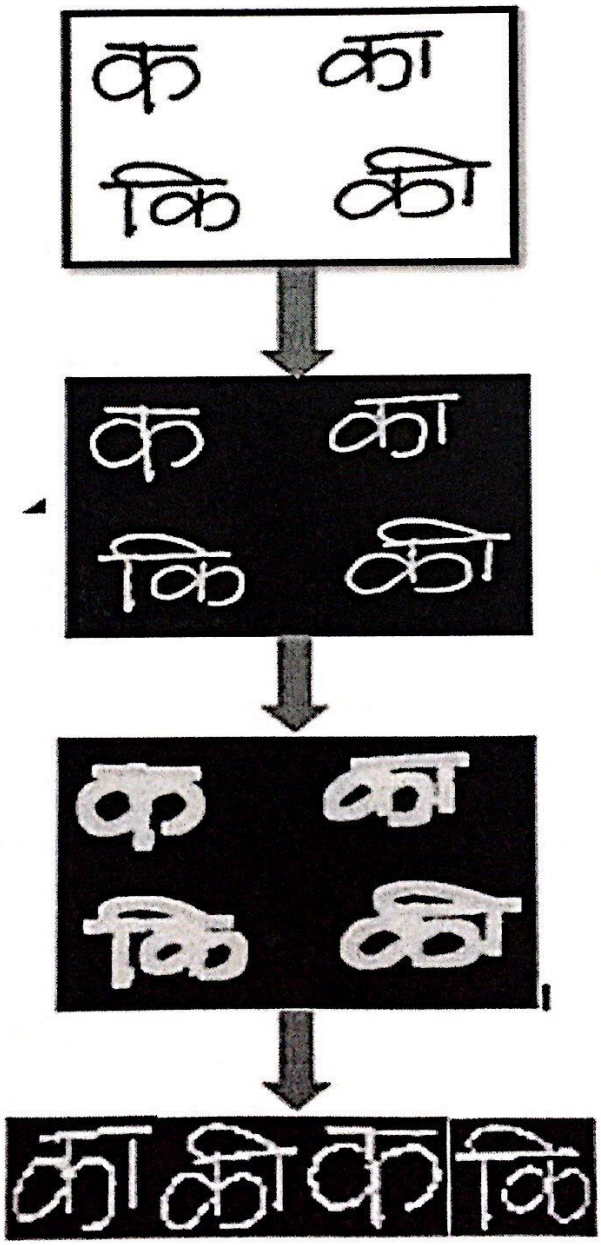


Figure 4 Data Collection Steps

3.2.2 Thresholding

Thresholding is a non-linear operation that converts a gray-scale image into a binary image where the two levels are assigned to pixels that are below or above the specified threshold value. In other words, if pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). In our system we vary the threshold parameter depending upon the methods of data collection. The process of inversion (i.e. interchanging the black and white pixel) was also carried out in this step.

3.2.3 Dilation

Dilation is a morphological operation typically applied to binary images that adds pixel to the boundaries of objects 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 to process the image. In our system, we varied the dilation parameter according to our requirement so as to separate individual characters from one another.

3.2.4 Contour Detection

Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity. Contours come handy in shape analysis, finding the size of the object of interest, and object detection.

3.2.5 Resizing

After contour detection, each character is resized to 28 x 28 and saved to specific folder of every class. Then these images are used for training of CNN architectures.

3.3 Training model

3.3.1 Convolution Neural Network

A CNN is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a

ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.

Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

Layers used to build Convolution Neural Network

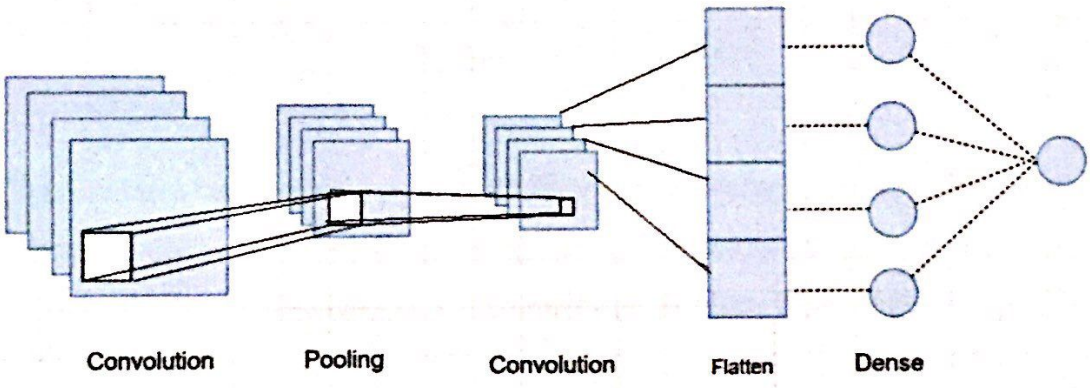


Figure 5 Convolution Neural Network

Convolution Layer

The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel. A convolution is how the input is modified by a filter. In convolutional networks, multiple filters are taken to slice through the image and map them one by one and learn different portions of an input image.

In purely mathematical terms, convolution is a function derived from two given functions by integration which expresses how the shape of one is modified by the other. Mathematically,

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

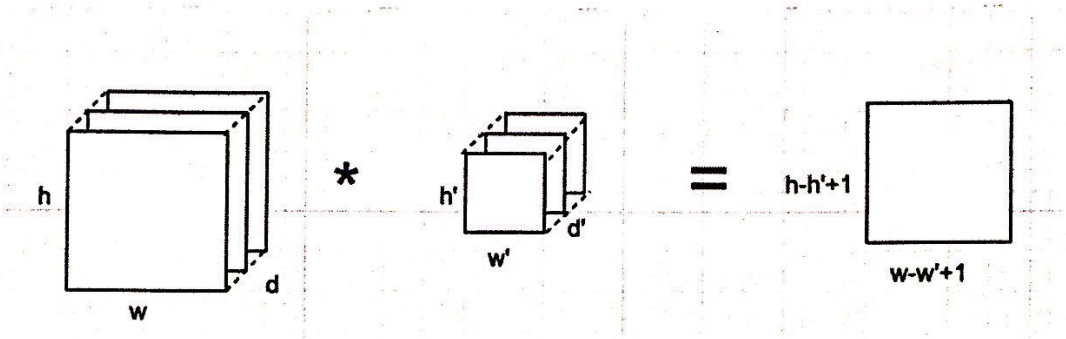


Figure 6 Operation in Convolution Layer

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution image after applying different types of filters (Kernels).

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

Activation functions

Non-Linearity (ReLU):

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = max(0, x)$. ReLU purpose is to introduce non-linearity in our ConvNet. Since, the real-world data would want our ConvNet to learn would be non-negative linear values.

There are other nonlinear functions such as tanh or sigmoid can also be used instead of ReLU. Most of the data scientists uses ReLU since performance wise ReLU is better than other two.

SoftMax:

SoftMax activation function turns numbers, also known as logits, into probabilities that sum to one. SoftMax function outputs a vector that represents the probability distributions of a list of potential outcomes. SoftMax activation functions helps best when we deal with the classification problems. This function squashes the outputs of each unit to be between 0 and 1, and also divides each output such that the total sum of the outputs is equal to 1.

The output of the SoftMax function is equivalent to a categorical probability distribution, it tells you the probability that any of the classes are true.

Mathematically the SoftMax function is shown below, where z is a vector of the inputs to the output layer (if you have 10 output units, then there are 10 elements in z). And again, j indexes the output units, so j = 1, 2, ..., K.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}}$$

Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces

the dimensionality of each map but retains the important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

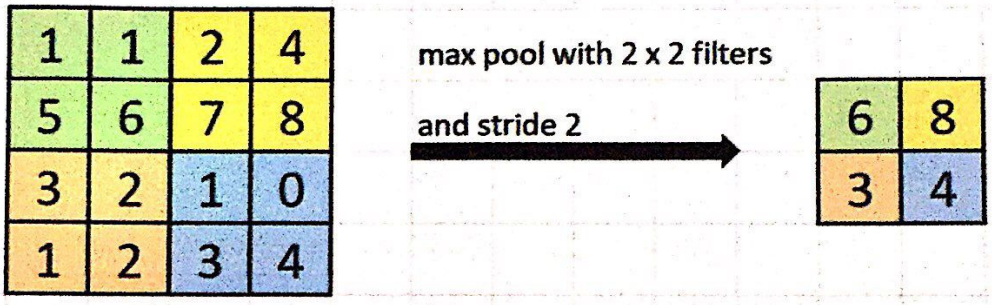


Figure 7 Max Pooling

Fully Connected Layer

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network. In the below diagram, feature map matrix will be converted as vector (x1, x2, x3, ...). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as SoftMax or sigmoid to classify the outputs as क, का, ख, खा, etc.

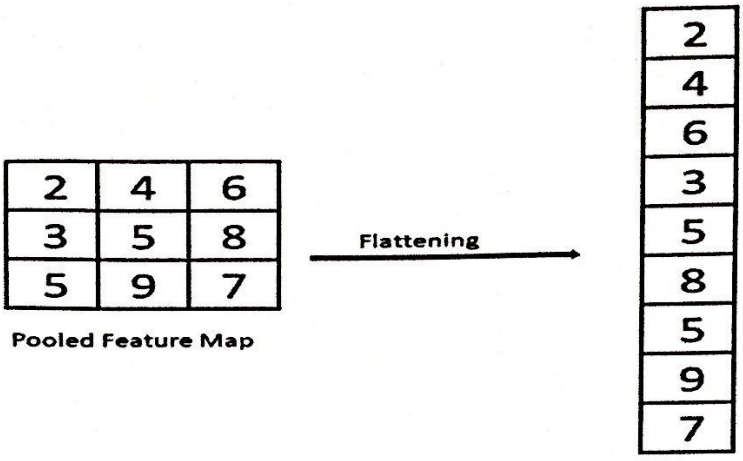


Figure 8 Flatten

Dropout:

Deep learning neural networks are likely to quickly overfit a training dataset. To avoid overfitting, single model can be used to simulate having a large number of different network architectures by randomly dropping out nodes during training. This is called dropout and offers a very computationally cheap and remarkably effective regularization method to reduce overfitting and improve generalization in deep neural networks of all kinds. Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

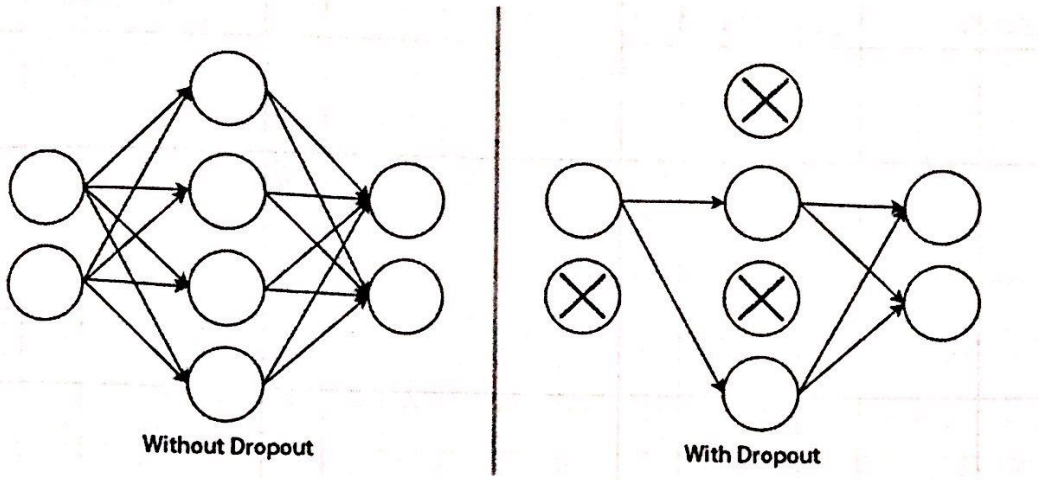


Figure 9 Concept of Dropout

3.4 Optimization Algorithm

The choice of optimization algorithm for a deep learning model can mean the difference between good results in minutes, hours, and days. After experimenting with various optimization algorithms such as Adam, SGD (Stochastic Gradient Decent), AdaGrad (Adaptive Gradient) and RMSProp (Root Mean Square Propagation), we finally used the Adam optimization algorithm since it provided the best accuracy for our model.

Adam Optimization Algorithm:

Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based on training data. The list of the attractive benefits of using Adam optimizer is:

- Straightforward to implement
- Computationally efficient
- Little memory requirements
- Appropriate for problems with very noisy/or sparse gradients.

Adam is different to classical stochastic gradient descent. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

Adam optimizer can be described combining the advantages of two other extensions of stochastic gradient descent. Specifically:

- **Adaptive Gradient Algorithm** (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).

- **Root Mean Square Propagation** (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

Adam realizes the benefits of both AdaGrad and RMSProp. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). Adam is a popular algorithm in the field of deep learning because it achieves good results fast.

3.5 Proposed Model

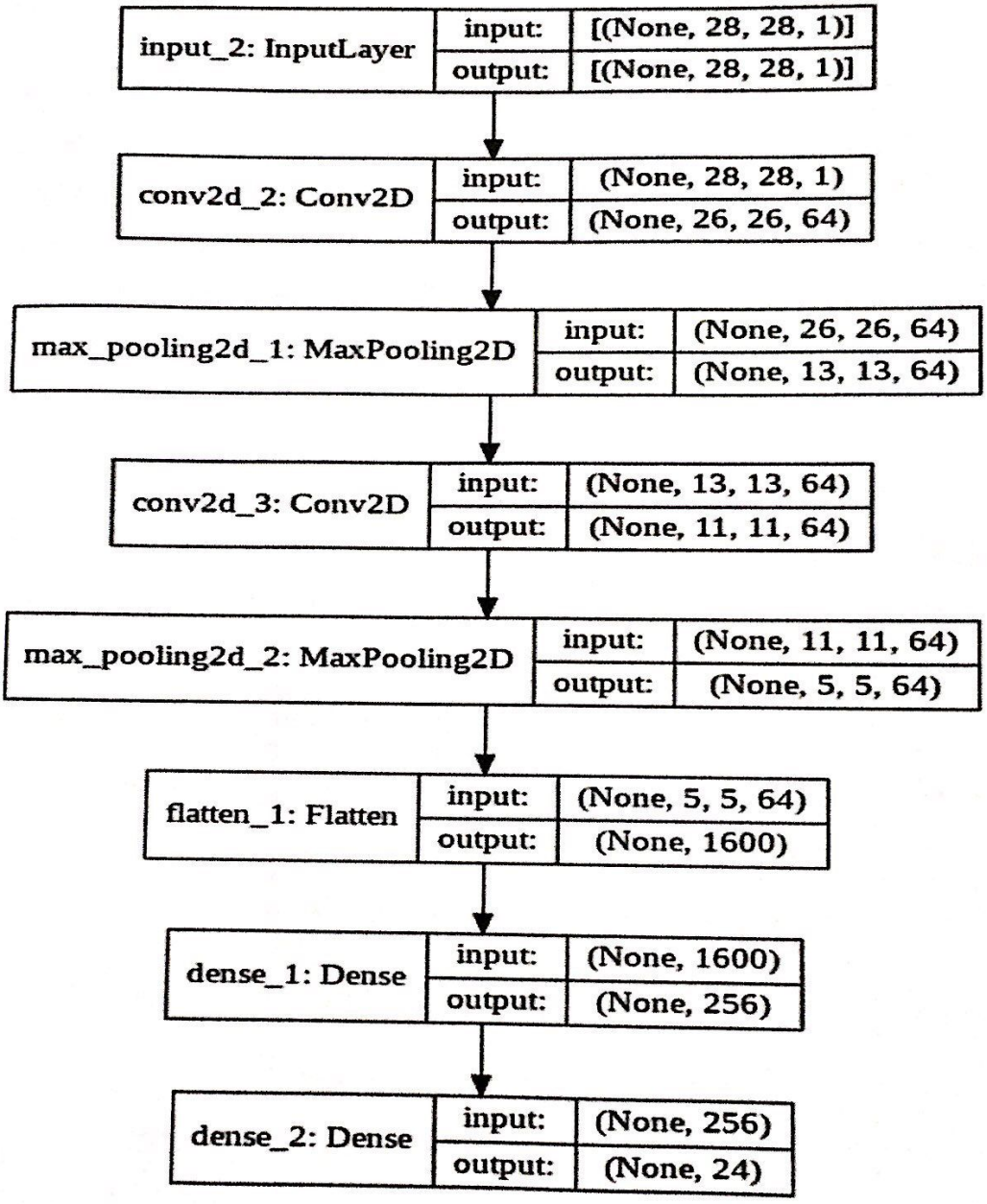


Figure 10 Proposed Model

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. The results are presented in the result and analysis section below. The sequence of the layers is shown in the table below, where I is Input Layer, C is the Convolution Layer, R is Rectified Linear Unit Layer, P is pooling layer implementing Max Pooling, D is the Dropout Layer, FC is Fully Connected Layer, O is the Output Layer.

Table 1 Model Architecture with parameters

Network Architecture	Version	Layer Type	Layer Size	Trainable parameters	Total parameters
NA -1	1.1 Filter:3	Conv2D Flatten Output	26 x 26 x 32 21632 48	320 0 1038384	1038704
	1.2 Filter:5	Conv2D Flatten Output	24 x 24 x 64 36864 48	640 0 1769520	177118
NA -2	2.2	Conv2D Max Flatten Dropout Dense	24 x 24 x 64 13 x 13 x 64 36864 10816 48	640 0 0 0 519216	519856

	2.2 Filter: 5	Conv2D Max Flatten Dense Output	26 x 26 x 32 13 x 13 x 32 5408 5408 48	320 0 0 0 259632	259952
NA-3	3.1 Filter:5	Conv2D Max Conv2D Flatten Dropout Dense	24 x 24 x 64 12 x 12 x 64 8 x 8 x 64 4096 4096 48	1664 0 102464 0 0 196656	300784
	3.2 Filter:5	Conv2D Max Conv2D Flatten Dense Dense	26 x 26 x 64 13 x 13 x 64 11 x 11 x64 7744 256 48	640 0 36928 0 1982720 12336	2032624
NA-4	4.1 Filter:5	Conv2D Max Conv2D Max Flatten	24 x 24 x64 12 x 12 x64 8 x 8x64 4 x 4 x 64 1024	1664 0 102464 0 0	372696

		Dense	256	262400	
		Dense	24	6168	

4.2 Result analysis

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

Table 2 Accuracy of different models

Architecture	Version	Accuracy	Optimizer	
			Adam	SGD
NA-1	1.1	Train	0.9998	0.9877
		Validation	0.9108	0.8732
	1.2	Train	0.9940	0.9982
		Validation	0.924	0.9333
NA -2	2.1	Train	0.9788	0.9864
		Validation	0.9288	0.9343
	2.2	Train	0.9787	0.9632
		Validation	0.9370	0.9260
NA -3	3.1	Train	0.9924	0.9872
		Validation	0.9524	0.9353
	3.2	Train	0.9912	0.9901

		Validation	0.9665	0.9577
NA -4	**4.1**	**Train**	**0.9968**	0.9677
		Validation	**0.9648**	0.9219

After experimenting with all these models and architectures the follow model achieved the best performance for our dataset.

The architecture NA-4 includes Convolution layer followed by max pooling and again a set of Convolution and max pooling then followed Flatten and two Dense layer as specified in table 1.

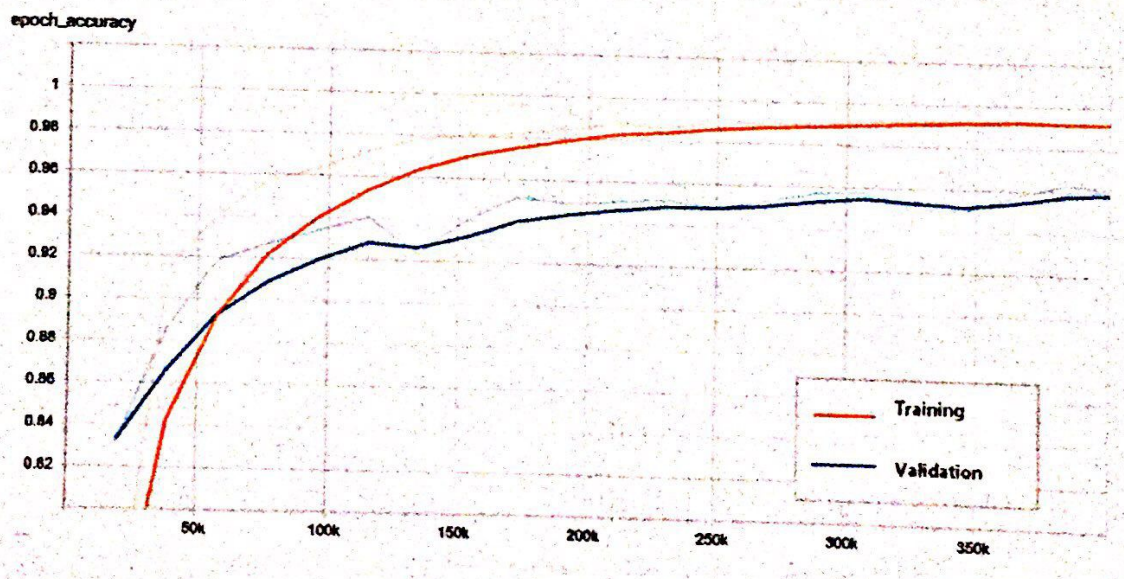


Figure 11 Epoch Accuracy Plot

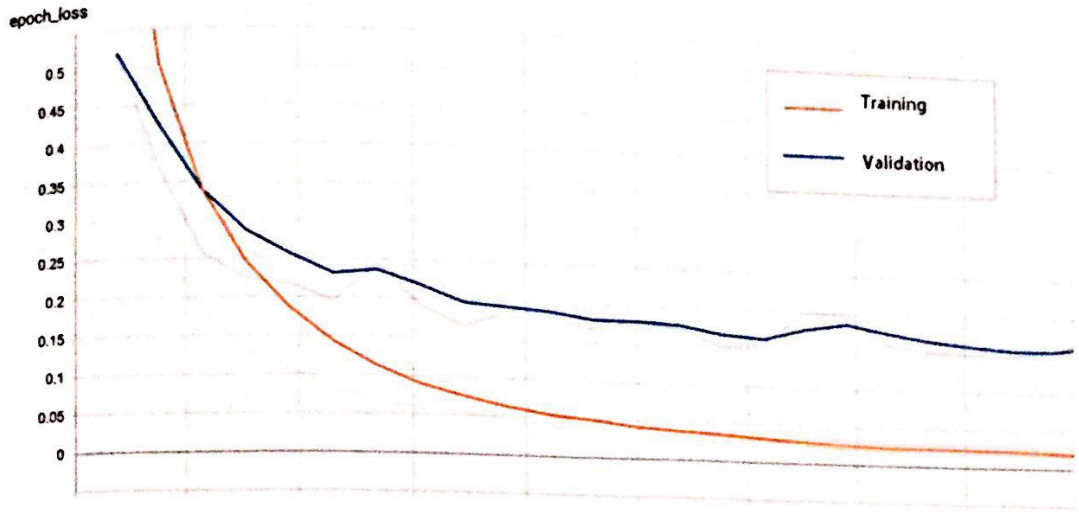


Figure 12 Epoch Loss Plot

4.3 Conclusion

With the project we conclude that the proposed architecture achieves the training accuracy of 99.68 % and testing accuracy of 96.48%. And the classification report of our proposed model (NA-4) is given below:

Table 3 Classification Report

Classes	Precision	Recall	F1-score	Support
Character_01_ka	1.00	1.00	1.00	20
Character_02_kaa	1.00	1.00	1.00	20
Character_03_ki	0.87	1.00	0.93	20
Character_04_kee	0.79	0.95	0.86	20
Character_05_ku	0.95	1.00	0.98	20
Character_06_kuu	0.79	0.95	0.86	20
Character_07_kay	1.00	0.45	0.62	20
Character_08_kai	0.62	1.00	0.77	20

Character_09_ko	1.00	0.60	0.75	20
Character_10_kau	0.67	1.00	0.80	20
Character_11_kam	1.00	1.00	1.00	20
Character_12_kaha	1.00	1.00	1.00	20
Character_13_kha	0.94	0.75	0.83	20
Character_14_khaa	0.90	0.90	0.90	20
Character_15_khi	1.00	0.85	0.92	20
Character_16_khee	1.00	0.80	0.89	20
Character_17_khu	0.91	0.50	0.65	20
Character_18_khuu	0.95	1.00	0.98	20
Character_19_khay	1.00	0.75	0.86	20
Character_20_khai	0.80	1.00	0.89	20
Character_21_kho	1.00	0.50	0.67	20
Character_22_khau	0.67	1.00	0.8	20
Character_23_kham	0.83	1.00	0.91	20
Character_24_khaha	1.00	1.00	1.00	20

Confusion matrix of the proposed architecture is:

Figure 13 Confusion Matrix

4.4 Further enhancements

Our project is limited to only few data at the moment due to the time limit. The following enhancements in our project are possible if further time could be allocated for the project.

- To build our own complete dataset of Nepali characters.
- To enhance our project as the Nepali Handwriting recognition and not just the Nepali handwritten characters recognition.

REFERENCES

[1] "Convolutional neural network", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network#History. [Accessed: 03- Jun- 2019].

[2] "Key Deep Learning Architectures: LeNet-5", Medium, 2019. [Online]. Available: https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-56fc3c59e6f4. [Accessed: 03- Jun- 2019].

[3] R. Casey and G. Nagy, "Recognition of Printed Chinese Characters", IEEE Transactions on Electronic Computers, vol. -15, no. 1, pp. 91-101, 1966. Available: 10.1109/pgec.1966.264379.

[4]J. Mantas, "An overview of character recognition methodologies", Pattern Recognition, vol. 19, no. 6, pp. 425-430, 1986. Available: 10.1016/00313203(86)90040-3.

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

[6] Ashok K. Pant, Sanjeeb P. Panday and S.R. Joshi, "Off-line Nepali handwritten character recognition using Multilayer Perceptron and Radial Basis Function neural networks" , Third Asian Himalayas International Conference on Internet, 2012.

[7] S. Acharya, P.K. Gyawali and Ashok K. Pant, "Deep learning based large scale handwritten Devanagari character recognition", 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), 2015

[8] Manish K. Sharma and Bidan Bhattrai, "Optical Character Recognition System for Nepali Language Using ConvNet, 9th International Conference, 2017

[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] "Understanding of Convolutional Neural Network (CNN) — Deep Learning", *Medium*, 2019. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. [Accessed: 03- Jun- 2019].

APPENDICES

Pre-Processing

```
import cv2
import numpy as np

#import image
image = cv2.imread("test1.png")

#grayscale
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
cv2.imshow('gray',gray)
cv2.waitKey(0)

#binary
ret,thresh = cv2.threshold(gray,200,255,cv2.THRESH_BINARY_INV)
cv2.imshow('second',thresh)
cv2.waitKey(0)

#dilation
kernel = np.ones((20,12), np.uint8)

img_dilation = cv2.dilate(thresh, kernel, iterations=1)
cv2.imshow('dilated',img_dilation)
cv2.waitKey(0)

#find contours
ctrs, hier = cv2.findContours(img_dilation.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

#sort contours
```

```
sorted_ctrs = sorted(ctrs, key=lambda ctr: cv2.boundingRect(ctr)[0])
i = 0
for i, ctr in enumerate(sorted_ctrs):
    # Get bounding box
    x, y, w, h = cv2.boundingRect(ctr)

    # Getting ROI
    roi = thresh[y:y+h, x:x+w]

    #resizing
    re_size = cv2.resize(roi,(28,28))
      #saving ROI
    cv2.imshow('contour',re_size)
    cv2.waitKey(0)
    cv2.imwrite(str(i)+'.png', re_size)

    # show ROI
    cv2.imshow('segment no:'+str(i),re_size);
    cv2.rectangle(image,(x,y),( x + w, y + h ),(90,0,255),2);
```

Training Model

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input,Dense,Dropout
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten
import cv2
import numpy as np

inputs = Input(shape=(28,28,1))
x = Conv2D(filters = 64,
        kernel_size = 5,
```

```
        activation = 'relu')(inputs)
x = MaxPooling2D()(x)
x = Conv2D(filters = 32,
        kernel_size = 3,
        activation = 'relu')(x)
x = MaxPooling2D()(x)
x = Flatten()(x)
x=Dense(256,activation='relu')(x)
outputs = Dense(24, activation = 'softmax')(x)

model = Model(inputs = inputs, outputs = outputs)

model.summary()

# Convolutional Neural Network
from keras.utils import plot_model
plot_model(model, to_file='convolution_neural_network.png',show_shapes =
True)

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])


model.fit_generator(train_generator,epochs = 35,
steps_per_epoch=19200//64,validation_steps=4800//64,validation_data=valid_ge
nerator)
```

```
300/300 [==============================] - 14s 48ms/step - loss: 0.0461 - accuracy: 0.9845 - val_loss: 0.1576 - val_accuracy: 0.9552
Epoch 10/35
300/300 [==============================] - 15s 48ms/step - loss: 0.0408 - accuracy: 0.9860 - val_loss: 0.1164 - val_accuracy: 0.9663
Epoch 11/35
300/300 [==============================] - 15s 49ms/step - loss: 0.0331 - accuracy: 0.9878 - val_loss: 0.1256 - val_accuracy: 0.9617
Epoch 12/35
300/300 [==============================] - 14s 48ms/step - loss: 0.0377 - accuracy: 0.9873 - val_loss: 0.1237 - val_accuracy: 0.9627
Epoch 13/35
300/300 [==============================] - 15s 48ms/step - loss: 0.0266 - accuracy: 0.9916 - val_loss: 0.1248 - val_accuracy: 0.9623
Epoch 14/35
300/300 [==============================] - 14s 48ms/step - loss: 0.0290 - accuracy: 0.9909 - val_loss: 0.1315 - val_accuracy: 0.9613
Epoch 15/35
300/300 [==============================] - 14s 48ms/step - loss: 0.0276 - accuracy: 0.9914 - val_loss: 0.1583 - val_accuracy: 0.9594
Epoch 16/35
300/300 [==============================] - 14s 48ms/step - loss: 0.0245 - accuracy: 0.9926 - val_loss: 0.1502 - val_accuracy: 0.9567
Epoch 17/35
300/300 [==============================] - 14s 47ms/step - loss: 0.0266 - accuracy: 0.9908 - val_loss: 0.1213 - val_accuracy: 0.9617
Epoch 18/35
300/300 [==============================] - 14s 48ms/step - loss: 0.0207 - accuracy: 0.9939 - val_loss: 0.1283 - val_accuracy: 0.9621
Epoch 19/35
300/300 [==============================] - 15s 48ms/step - loss: 0.0199 - accuracy: 0.9933 - val_loss: 0.1143 - val_accuracy: 0.9694
Epoch 20/35
300/300 [==============================] - 15s 50ms/step - loss: 0.0180 - accuracy: 0.9934 - val_loss: 0.1290 - val_accuracy: 0.9694
Epoch 21/35
300/300 [==============================] - 15s 50ms/step - loss: 0.0245 - accuracy: 0.9921 - val_loss: 0.1084 - val_accuracy: 0.9698
Epoch 22/35
300/300 [==============================] - 15s 49ms/step - loss: 0.0153 - accuracy: 0.9948 - val_loss: 0.1141 - val_accuracy: 0.9715
Epoch 23/35
300/300 [==============================] - 15s 50ms/step - loss: 0.0164 - accuracy: 0.9940 - val_loss: 0.1078 - val_accuracy: 0.9729
Epoch 24/35
300/300 [==============================] - 15s 50ms/step - loss: 0.0155 - accuracy: 0.9952 - val_loss: 0.1415 - val_accuracy: 0.9654
```

Figure 14 Training the model

Prediction

```
def prediction(x):

  switcher = {

      0:'ka',

      1:'kaa',

      2:'ke',

      3:'kii',

      4:'ku',

      5:'kuu',

      6:'kay',

      7:'kai',

      8:'ko',
```

```
    9:'kau',
    10:'kam',
    11:'kaha',
    12:'kha',
    13:'khaa',
    14:'khi',
    15:'khii',
    16:'khu',
    17:'khuu',
    18:'khay',
    19:'khai',
    20:'kho',
    21:'khau',
    22:'kham',
    23:'khaha'
  }
  return switcher.get(x)
import matplotlib.pyplot as plt
import numpy as np
import cv2
#Image to predict to out model
```

```
img = cv2.imread('/content/1.png',0)

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7ff945954d30>
```

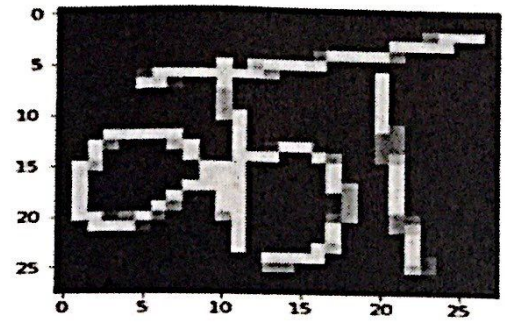


Figure 15 Image to predict

```
#Expand the dimension of image as required for the model

img = np.expand_dims(img, axis= 0)

img = np.expand_dims(img, axis = -1)

pred = model.predict(img)

print('The predicted character is:' prediction(np.argmax(pred)))
```

OUTPUT:

The predicted character is: kaa