Deep Learning based Handwritten Devanagari Character Recognition using Raspberry Pi 3

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Jasmine
(Roll No. 801632017)

Under the supervision of:
Dr. Seema Bawa
Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
PATIALA – 147004

June 2018

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Deep Learning based Handwritten Devanagari Character Recognition using Raspberry Pi 3*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Seema Bawa* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Jasmine

(Jasmine)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Seema Bawa

(Dr. Seema Bawa)
Professor,
CSED

Acknowledgement

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life. It is a great privilege to express my gratitude and admiration towards my respected supervisor **Dr. Seema Bawa**, Professor, Computer Science & Engineering Department. She has been an esteemed guide and great support behind achieving this task. This work would not have been possible without the encouragement and able guidance of her. I also thank my supervisor for her time, patience, discussions and valuable comments. Her enthusiasm and optimism made this experience both rewarding and enjoyable. I am truly grateful to her for extending her total co-operation and understanding whenever I needed help and guidance from her. I am also heartily thankful to **Dr. Maninder Singh**, Professor and Head, Computer Science & Engineering Department and **Dr. Ashutosh Mishra**, Assistant Professor and PG coordinator, for motivation and providing uncanny guidance and support throughout the preparation of the thesis report.

I will be failing in my duty if I do not express my gratitude to **Dr. S.S.Bhatia**, Senior Professor and Dean of Academic Affairs, for making provisions of infrastructure such as library facilities, computer labs equipped with net facilities, immensely useful for the learners to equip themselves with the latest in the field. I am also thankful to staff members of Computer Science and Engineering Department for their help, cooperation, love and affection, which made my stay at Thapar University memorable. Last but not least, I would like to thank my senior **Sukhandeep Kaur Shergill** and my friend **Gaurav Talwar** for their continous support and my family for their wonderful love and encouragement, without their blessings none of this would have been possible.

Jasmine

(801632017)

Abstract

A real time handwritten Devanagari character recognition framework using Raspberry Pi 3 is proposed. It has a Graphical User Interface application which takes image of character and trained deep convolutional neural network model as input and displays the output of character. The baseline deep convolutional neural network model is modified by changing its width i.e. number of filters and optimization function while keeping other hyperparameters like filter size, number of layers etc. constant. This is also helpful in understanding the impact of width of model on its accuracy that more width is not always better as training time increases with increase in width and eventually, accuracy starts decreasing. Large number of trainable parameters with increased width also make the model susceptiple to overfitting which is resolved by dropout technique. The highest accuracy achieved is 98.75%. By loading this model, our application recognizes the character.

Machine simulation and automatic recognition of Devanagari characters is useful in areas where the data present on paper has to be transferred to machine-readable format. It has a variety of practical and commercial applications in post offices, banks, libraries and publishing houses.Traditional methodologies used for Devanagari character recognition requires the extraction of statistical features and structural features of character. The accuracy of such classification was majorly dependent on feature descriptors. However, the nonlinear information processing used by deep learning models for feature extraction has been able to overcome these challenges. Among them, Convolutional Neural Networks (CNN) have become the leading architecture for image recognition and classification tasks.

Keywords: Devanagari Handwritten Character Dataset, Deep Convolutional Neural Network, Dropout, Raspberry Pi 3, Character Recognition.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reading text from images in real world without any human intervention is one of the difficult tasks in optical computer vision problems. Many character recognition and machine learning algorithms have been developed and implemented on various available datasets of significantly unsolved, harder, real world problems. The challenge in classifying handwritten characters is due to different layouts, fonts, scales, rotations of characters which result in their appearance variability. This work focuses on recognizing handwritten consonant and numeral characters of Devanagari Script using deep Convolutional Neural Networks. The deep learning has applications in fields like computer vision, audio and speech recognition and many more like natural language processing. Deep learning can help achieve Optical character recognition.

Deciding the hyperparameters (depth,width,optimization function etc.) of deep convolutional neural networks is a tough task as the accuracy mainly depends upon these parameters. Incorporating deep learning with low-end computating devices such as mobiles, Raspberry Pi, various parts of robotics is also a challenging task because of its high power consumption and high computation demands. But many ways have been introduced like offloading to the cloud or migrating tensor flow to devices [9].

One of the affordable low-end computing devices is Raspberry Pi. The small single-board computers are connected with its own Raspbian operating system which makes it an efficient system. It can be used as a platform for many applications including handwritten Devanagari character recognition.

1.1 Deep Learning Overview

Carrying out numerous tasks with the help of machines without any human intervention is the main concept behind artificial intelligence (AI). AI can be implemented in various ways. One of the implementations is situation-based coding which are hardcoded with responses as per few situations. But in real world, there are many variables with many values, according to which action must be taken but hardcoding is not possible or does not give good results. Hence, machine learning concept comes into picture. Machine learning helps machines to learn to solve problems by providing enough examples just like humans learn to solve further problems from few examples. There are several algorithms that are used for machine learning like random forests, decision trees, artificial neural networks etc. Deep Learning is also a way to implement machine learning.

The relation of artificial intelligence with machine learning and with deep learning is shown in Fig.1.1.

Deep learning belongs to broader family of machine learning methods to learn data representations which basically uses artificial neural networks. It combines more layers with artificial neural networks for better learning [Fig.1.2].

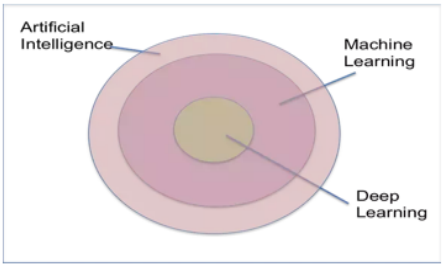


Figure 1.1: Relation of artificial intelligence with machine learning and with deep learning.

1.1.1 Artificial Neural Networks Overview

Artificial Neural Networks are inspired by biology of brains and all interconnections between neurons [Fig.1.4]. Neural networks are like graphs with nodes and

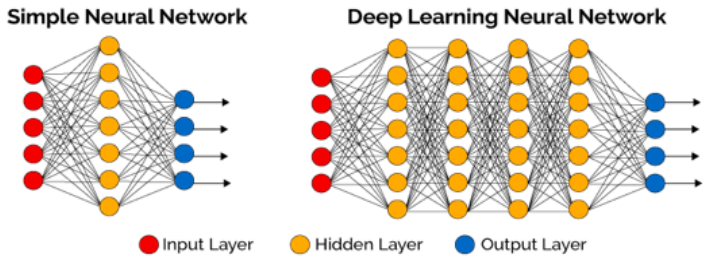


Figure 1.2: Deep learning neural network as an extension of simple neural network.

edges [Fig.1.3]. In the Fig.1.3, there is one layer of input nodes, one layer of output nodes and one layer of hidden nodes. Deep neural networks generally have one or more layer of hidden nodes [Fig.1.2]. The input is analogous to the sensory receptors like mechanoreceptors in finger tips or visual receptors in eyes. These input nodes get rescaled by a weight and then get fed into the hidden nodes using an activation function. These weights are like strength of synapse in brain. Stronger weights mean there is a stronger connection from one node to another. In terms of neuron, it means a signal gets transferred from one neuron to another without much attenuation [Fig.1.3].

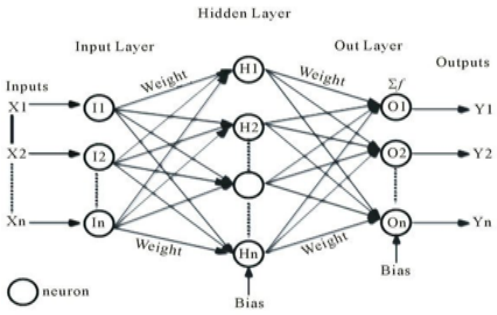


Figure 1.3: Simplest architecture of artificial neural networks.

The Fig.1.4 explains the relation between biological neuron artificial neuron in artificial neural network. Neuron receives one or more inputs, i^{th} input is denoted as x_i. There are weights (w_i) which express the importance of respective inputs to the output. Bias (b) value for each neuron is added to the weighted input [Fig.1.4].

After the weighted input is received, operations performed by neuron are:

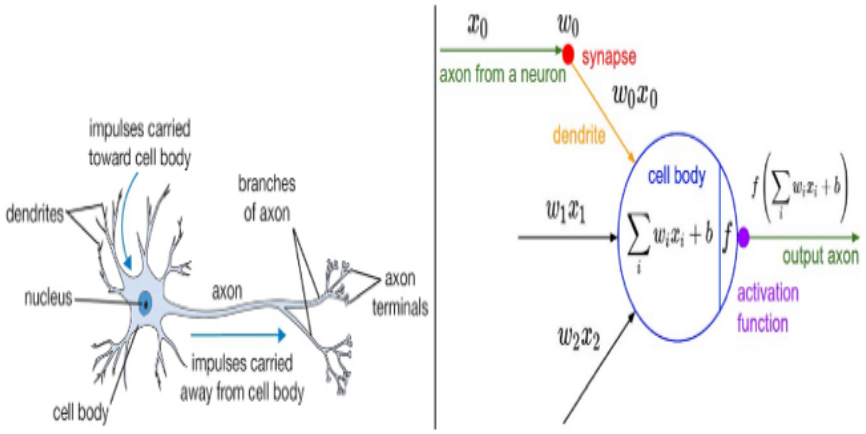


Figure 1.4: Analogy of biological neuron (left) and its mathematical model (right).

(i) Summation of weighted inputs received:

$$\sum_{i=1}^{n} w_i x_i = w.x \tag{1.1}$$

where n is the number of inputs and w.x is the dot product of the weights and inputs.

(ii) Addition of the bias:

$$\sum_{i=1}^{n} w_i x_i + b = w.x + b \tag{1.2}$$

(iii) Application of nonlinear activation function:

$$f(\sum_{i=1}^{n} w_i x_i + b) = f(w.x + b) \tag{1.3}$$

where f is the activation function.

The artificial neural network has two phases:

(i) Forward Phase: It is the procedure of prediction. Innput layer is fed with input data, propagating through the weighted connections into hidden layers by using activation function. The node activation is a function of the weighted sum of the connected node from the previous layer. This process continues until the output layer. [Fig.1.3].

(ii) Backpropagation Phase: It is the procedure of training. Training includes learning weights on each edge. The training is called backpropagation that is errors get propagated backwards one layer at a time. The second set of weights will change depending on the errors on output layer and first set of weights will change depending on the errors on middle layer [Fig.1.3].

1.1.2 Challenges in training Deep Neural Networks

There are various challenges posed by deep neural networks [Table 1.1]:

(i) Trainable parameters in Deep neural networks are very large because they consist of several nonlinear hidden layers.

(ii) Deep Neural Network also requires a very huge dataset to avoid overfitting.

(iii) Large amount of memory is needed to store large trainable parameters as an input is passed through the network. In training, activation from a forward pass must be stored until they can be used to calculate the error gradient in the backward pass.

(iv) Before the start of the learning process, hyper parameters need to be defined, Changing the value of these parameters by small amount can bring a large change in the performance of model.

(v) The hardware for training the deep neural networks need to be equipped with adequate processing power.

(vi) Neural networks are hard to understand how the input fed into the network arrives at particular solution. This makes it difficult to implement high-level cognitive functions.

Table 1.1: Challenges in deep neural networks

Large trainable parameters
Requirement of large dataset
Large amount of memory
Hyperparameters Optimization
Enough processing power

1.1.3 Convolutional Neural Networks Overview

Images are special types of inputs that can be seen with our own eyes so it will be known that what type of noise a neural network should be robust to like translational invariance [Fig.1.5]. If we are trying to classify objects and one of the inputs is a picture of car and a traditional feedforward neural network is trained to classify it. But, if the car is shifted to right, a neural network with all different sorts of weights will not be able to classify this picture as car and may give the wrong answer because it was not trained to see the car in this exact position. Therefore, for neural network to be invariant to translations of an image. Translation can also be up or down or left or right.

Figure 1.5: Translational invariance of a picture of car.

In convolution, a box of weights which is small relative to the whole image is taken, then this small box of weights is multiplied by every part of the image [Fig.1.6]. This small box of weights is a feature finder which outputs a very positive number when it sees a car and very negative number when it sees something else. So, this small box of weights can find car in the image no matter where it is because the convolution uses the same weight everywhere. This the main motivation behind using convolution in neural networks. Feedforward neural networks that contain multiple layers allow each layer to find successively complex features. In the same way, the first layer of convolutions find very basic features like line, the next layer of convolutions might more complex features and so on. Table 1.2 discusses the benefits of convolutional neural networks.

Convolutional neural networks are similar to visual cortex. The original inspiration of neural networks is to model the brain which is made up of neurons. So, to make image processing system, it makes sense to model the features of biological visual

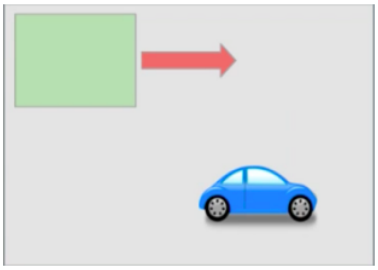


Figure 1.6: Small box of weights finding features of image

Table 1.2: Advantages of convolutional neural networks

Fewer Weights
Able to handle translational invariance, rotational invariance and object distortions.
Learn feature representations among input pixels

system. All the inputs or sensors in body are made up of special structures called receptors. There are different types of receptors like mechanoreceptors that can sense pressure or when one is physically in contact with something. The receptors in eye are light receptors that means they can convert light into an electrical signal which then travels through neurons and into brain. One attribute of receptor is its receptive field which is a region of space that the receptor responds to which means each neuron is responsible for transmitting data from just a small region of space. So, in basic fully connected feed forward neural network [Fig.1.3], it doesnt make much sense to connect all the inputs to all the neurons in the next layer. It makes more sense that some small area would affect one neuron in the next layer and that neuron in this layer would combine with the neighboring neurons in the next layer and so on.

The Fig.1.7 shows the basic layout of convolutional neural network. In first layer, image is taken as input with all the colors and its original shape that is without flattening. Convolution is performed on this image. Next, max pooling is done to decrease size of the features. Then, another convolution and another max pooling is done. Finally, the features learned are flattened into a vector and put into fully connected neural network.

The basic components of Convolutional Neural Network are:

(i) Filters: These are the small boxes of weights that maps feature maps of one

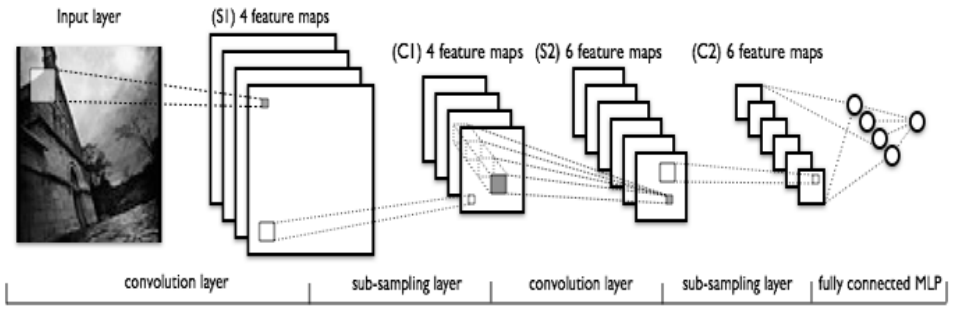


Figure 1.7: Layout of convolutional neural network.

layer to feature maps of next layer [Fig.1.6]. These small boxes move across the whole image. Stride is the number of pixels filters move at a time. Stride of 1 means the box moves one pixel at a time. Once the filter is moved across the whole image, it would map to a feature map in the next layer.

(ii) Feature Maps: It is the output of filter applied to previous layer. It is equal to the number of filter applied to previous layer. If n filters are applied, then there are n feature maps in the following layer. The process of mapping the inputs to the feature maps in the next layer is called convolution.

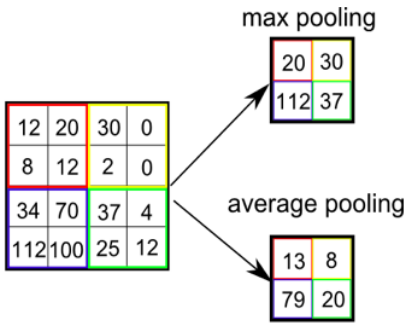


Figure 1.8: Max pooling and average pooling of image.

(iii) Pooling Layer: After convolution, it is of interest to know if there was feature present in the certain area of image. This is done by down sampling or sub-sampling the image. It means changing the resolution of image. It is done by converting an image from 32×32 to 16×16 that is down sampling by a factor of two in both horizontal and vertical directions. The different ways of doing this are max pooling and average pooling [Fig.1.8]. In max pooling,

block of any size is taken and maximum value in that block is the output. In average pooling, the average on the values of block is the output.

(iv) Fully Connected Layer: This is a basic feed forward fully connected neural network which takes the output from the last pooling layer and flatten it into a vector which will be the input of this fully connected layer. Finally, this will predict the final output that is the class to which the input belongs.

Chapter 2

Literature Review

Recognition of Devanagari characters fall in the category of handwritten text recognition system. It can also be recognized as a problem of image classification. Image can be categorized into any predefined class in image classification. Reading text from images in real world without any human intervention is one of the difficult tasks in optical computer vision problems. Convolutional Neural Networks (CNNs) can be applied to visual tasks.

2.1 Devanagari Character Recognition

Devanagari optical character recognition has been a topic of research since 1970s. Devanagari script is being used by more than 300 million people use for documentation in India. Machine simulation and automatic recognition of Devanagari characters on documents is useful in areas where hardcopy of document is need to be converted to e-document. It has a variety of commercial and practical applications in post offices, libraries, publishing houses, banks [1].

Devanagari has 34 consonants [Fig. 2.1(b)], 13 vowels and 14 modifiers [Fig. 2.1(a)]. Apart from these, composite characters can be formed using two or more basic characters resulting in more complex structures than that of the basic characters. Moreover, there are no lowercase or uppercase characters. It is a phonetic script i.e. words are written same as they are pronounced and syllabic script which means word is written using vowels and consonants that together form the syllables [1].

Many methodologies have been used for the handwritten and machine printed Devanagari character recognition.

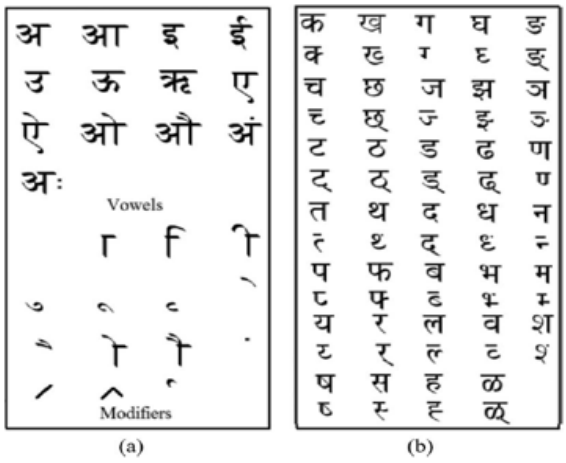


Figure 2.1: (a) Devanagari vowels and modifiers; (b) Devanagari consonants [1].

2.1.1 Review of traditional methodologies used for Devanagari character recognition

In this development of Optical Character Recognition System, the most important step is feature extraction which can difference in patterns in the characters, symbols or words followed by recognition or classification process. The features are divided into two kinds, one is statistical and other is structural. A statistical feature is derived from Statistical distribution of points like moments, histograms, zoning of the projection features whereas structural feature is based on geometrical properties like loops, endpoints, direction and intersection of strokes. The approaches used for classification are usually feature based or template based. Feature based approach uses features extracted from the test pattern in classification models like Artificial Neural Network, Hidden Markov Model, Support Vector Machine etc. Some traditional approaches such as fuzzy rules, rough sets, Mahalanobis and Hausdorff distances and evolutionary algorithm is also used for Devanagari character recognition [1]. Devanagari character classification in the scene images is difficult because methods performing well are for English only [2].

The results of classification techniques used till 2010 for printed [Fig.2.2] and handwritten [Fig.2.3] Devanagari character recognition are addressed. The few observations are reported from the survey. A different shapes and sizes of hand-

written Devanagari characters is a challenge in their classification. Segmenting words in characters is another difficult problem in automatic processing of text documents. An extracting text information from images have gained importance due to availability of cameras attached with handheld devices. There is a need to find the ideal combination of classifiers to achieve more effectiveness [1].

Method	Feature	Classifier	Data Set (Size)	Accuracy (%)
Govindaraju et al.[16]	Gradient	Neural networks	4,506	84
Kompalli et al.[22]	GSC	Neural networks	32,413	84.77
Bansal et al.[20]	Statistical & Structural	Statistical Knowledge Sources	Unspecified	87
Huanfeng Ma et al.[1]	Structural & Statistical	Hausdorff image comparison	2,727	88.24
Sinha et al.[10]	Structural	Syntactic pattern analysis	Unspecified	90
Natarajan et al.[94]	Derivatives	HMM	21,982	91.3
Bansal et al.[29]	Filters	Five filters	Unspecified	93
Dhurandhar et al.[27]	Contours	Interpolation	546	93.03
Kompalli et al.[26]	GSC	K-nearest neighbor	9,297	95
Jayanthi et al.[25]	Statistical	Binary tree	4863	95.08
Chaudhuri et al.[5]	Statistical	Tree classifier and template matching	10,000	95.42
Kompalli et al.[19]	SFSA	Stochastic finite state automaton	10,606	96
Jawahar et al.[23]	PCA	Support Vector Machine	2,00,000	96.7
Dhingra et al.[31]	Gabour	MCE	30,000	98.5

Figure 2.2: Details of the printed Devanagari character recognition systems [1].

Method	Feature	Classifier	Data Set (Size)	Accuracy (%)
Sharma et al.[50]	Chain code	Quadratic	11,270	80.36
Deshpande et al.[63]	Chain-code	RE & MED	5,000	82
Arora et al.[72]	Structural	FFNN	50,000	89.12
Arora et al.[77]	Combined	MLP	1,500	89.58
Hanmandlu et al.[68]	Vector distance	Fuzzy sets	4,750	90.65
Arora et al.[61]	Shadow & CH	MLP & MED	7,154	90.74
Kumar et al.[48]	Gradient	SVM	25,000	94.1
Pal et al.[51]	Gradient & Gaussian filter	Quadratic	36,172	94.24
Mane et al.[66]	Eigen deformation	Elastic Matching	3,600	94.91
Pal et al.[55]	Gradient	SVM & MQDF	36,172	95.13
Pal et al.[35]	Gradient	MIL	36,172	95.19

Figure 2.3: Details of the handwritten Devanagari character recognition systems [1].

2.1.2 Part-based technique for Devanagari character recognition in scene images

Character recognition of Devanagari scripts in scene images is challenging because methods performing well are for English only. A novel-part based techniques are proposed by classifying any given test sample of Devanagari characters from scene images as an instance of 40 basic classes. The proposed technique can either use the machine printed or handwritten dataset for training so as to overcome the drawback of unavailability of large dataset of scene images. Corner points of

characters that can act as parts are considered for part-based models. Lowest score computed between corner point of the test character and the parts of the model defines a class. Since a K-means clustering is used, this stage is computationally demanding [2].

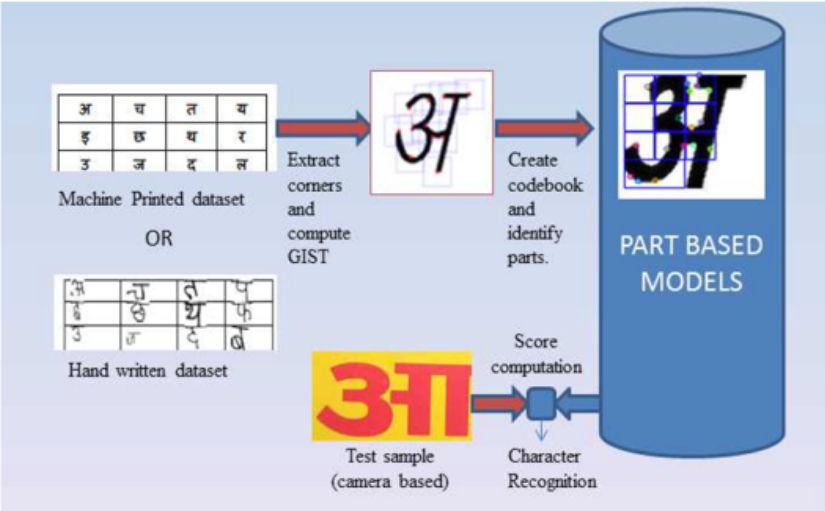


Figure 2.4: Part-based model technique for Devanagari crharacter recognition [2].

2.1.3 Neural network approach for Devanagari character recognition

Neural network approach combined with optimization before classification, extracting feature and recognition level has given better performance in recognition [Fig 2.6]. Before classification of character is done into different class using structural feature like enclosed regions in the character, present vertical line and position of endpoints. An optimized pixels density features extraction technique is used to train neural networks like Feed-forward and back-propagation network (FFBPN), Cascade-forward and back-propagation network (CFBPN), Elman and back-propagation network (EBPN) and then tested by applying similar feature from the tests character. EPBN has the highest recognition rate of recognizing each character in approximately 0.25 sec on Intel Core i3 having 2 GB RAM [3].

A multilayer perceptron neural network is also considered for classification of Devanagari numeral and characters [4]. Each image is zoned into 9 blocks and 8

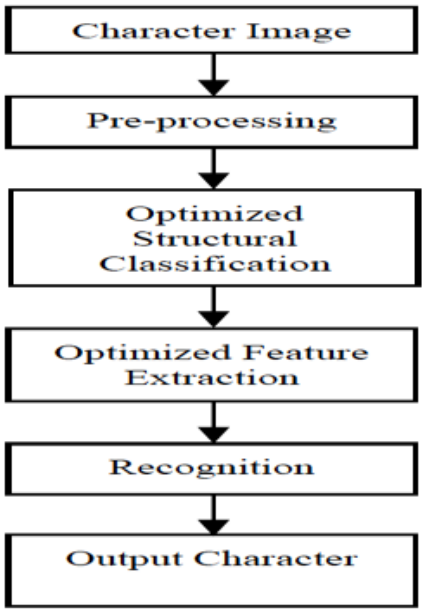


Figure 2.5: Proposed model for recognition incorporated with optimization [3].

Neural Network	Before optimization	After optimization
	Recognition Rate (%)	*Recognition Rate (%)*
FFBPN	96.95	97.20
CFBPN	97.32	97.46
EBPN	97.60	98.10

Figure 2.6: Comparison of different neural networks performance [3].

structural features and 9 global geometric features are extracted from each block which are used for representing numerals and characters [Fig. 2.7].40 hidden neurons for numerals and 60 hidden neurons for characters are used for recognition. The result shows 93.17% accuracy for numeral recognition [Fig.2.9] and 82.7% accuracy for character recognition [Fig.2.10]. Fivefold cross validation technique is used to verify the results [4].

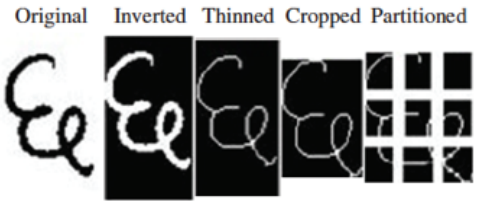


Figure 2.7: Pre-processing of Devanagari image [4].

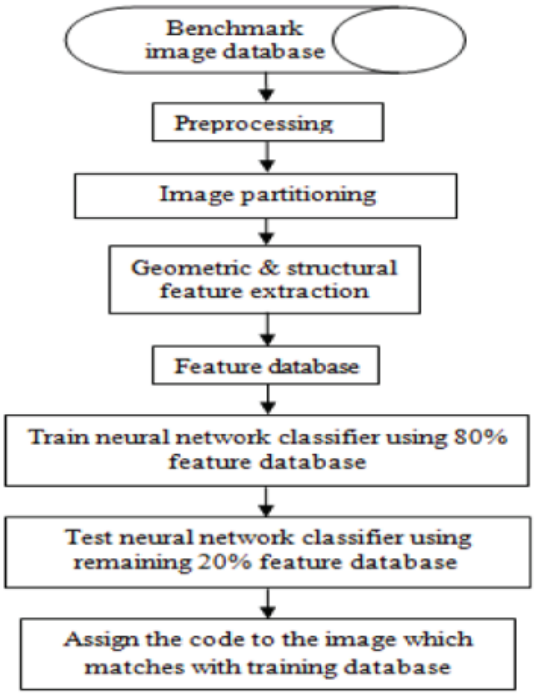


Figure 2.8: Recognition process of Devanagari image [4].

2.1.4 Handwritten Devanagari compound character recognition

Compound characters are formed by combining more than one character. Their occurrences in the script are upto 12 to 15%. Devanagari compound character recognition system uses Legendre moment feature descriptor for structural and statistical features extraction. After pre-classification, the images are normalized to 30×30 pixel size which is further divided into zones. Moment based feature is extracted from every zone [Fig. 2.11]. Then, Feed forward neural networks are trained and tested. The average classification accuracy obtained is 98.25%. Feature Set 2 [Fig.2.11] have given the better result [Fig.2.12] [5].

Numeral	०	१	२	३	४	५	६	७	८	९	Accuracy
०	59	0	0	0	0	0	0	2	0	0	98.33
१	0	53	1	0	0	0	0	0	0	1	88.33
२	0	3	53	4	0	0	0	0	0	0	88.33
३	0	1	3	56	0	0	3	0	0	1	93.33
४	0	0	0	0	55	0	1	0	0	1	91.67
५	0	0	0	0	0	58	0	0	0	0	96.67
६	0	1	0	0	0	1	55	2	0	2	91.67
७	1	0	0	0	3	0	0	55	0	0	91.67
८	0	0	1	0	1	0	0	2	60	0	100
९	0	2	2	0	1	1	1	1	0	55	91.67
										Overall accuracy%	93.17

Figure 2.9: Confusion matrix for 40 hidden neurons and 80:20 train : test ratio of numerals [4].

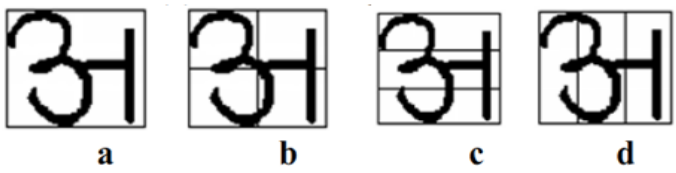


Figure 2.10: (a) Feature Set 1: Image considered as a whole; (b) Feature Set 2: Image divided into four equal zones; (c) Feature Set 3: Image divided into three horizontal equal zones; (c) Feature Set 3: Image divided into three vertical equal zones [5].

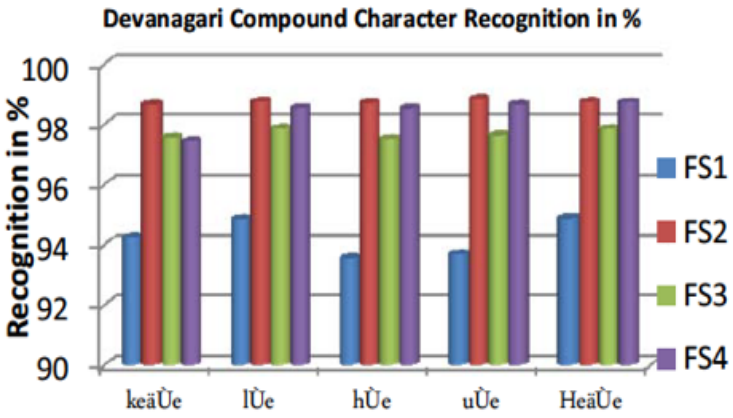


Figure 2.11: Comparison of Devanagari compound character recognition using feature set [5].

2.2 Deep learning approach for character recognition

Deep convolutional Neural Networks have achieved excellent results due to availability of enough amount of computational power required to train large dataset for better learning [6]. Deep neural networks do not require explicit character of extraction of features [8].

Deep learning model for Farsi recognition designed only for machine-generated character [Fig.2.13] is proposed. This model consist of convolution with pooling and fully connected layer. Main goal of convolutional layer is to extract features from the inputs. The fully connected layers use the outputs from these layers to perform inferences and a score is assigned to every possible output. This model is implemented using Torch framework. Other layers used to reach maximum generalization as well as performance is dropout, spatial max-pooling and batch normalization. Not only the results of machine generated character recognition are good but also the accuracy obtained by handwritten character recognition is admirable [Fig.2.14] [6]. The shared-hidden layer deep convolutional neural network

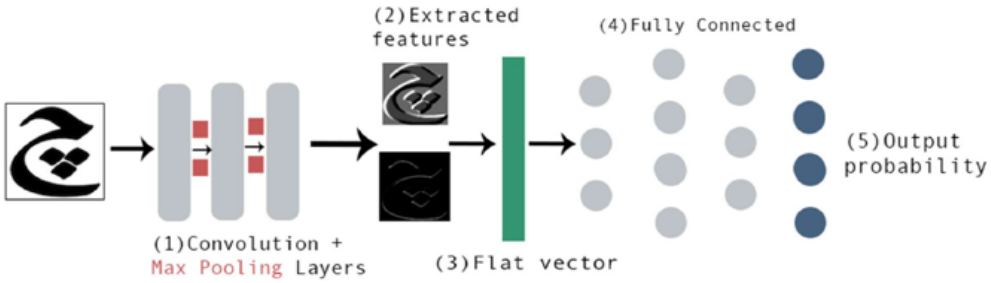


Figure 2.12: Flow of data inside the deep donvolutional neural network model [6].

MODEL	Training Dataset	Validation Dataset (count)	Validation Accuracy
Plain CNN	Machine Generated	Machine Generated (8000)	89%
Plain CNN	Machine Generated	Handwritten (9000)	2%
Optimized CNN	Machine Generated	Machine Generated (8000)	97%
Optimized CNN	Machine Generated + Modification	Handwritten (9000)	68%

Figure 2.13: Summary of the results obtained [6].

(SHL-CNN) for image character recognition by learning common features among

different languages using hidden layers but making final softmax layer available for destination language only [Fig.2.15] is proposed. This model is verified on image recognition of English and Chinese characters. Compared with the conventional CNN model which is trained using the characters of only one language, this model reduces the errors by 16-30%. Also, this model is useful for unseen images classification tasks [7].

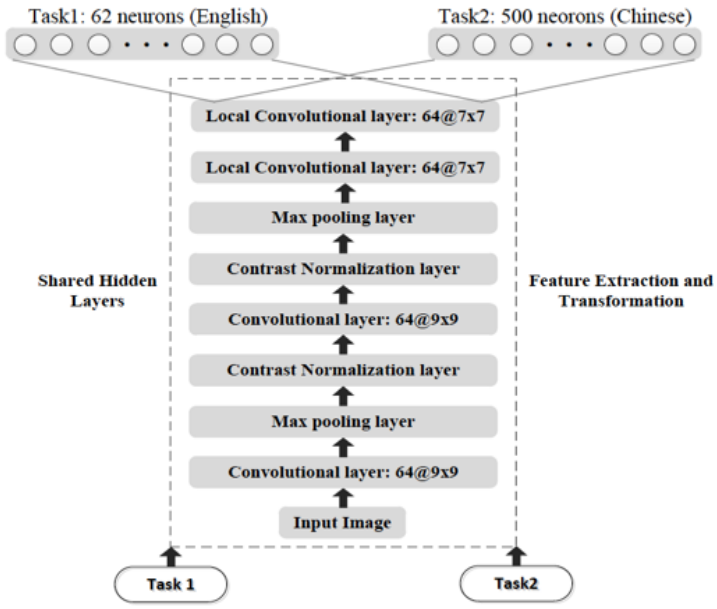


Figure 2.14: Architecture of the shared-hidden-layer CNN [7].

The latest dataset of Devanagari script: Devanagari Handwritten Character Dataset (DHCD) consisting of 92000 image of 46 different basic class of Devanagari script which are cropped from handwritten document and deep learning architecture of classification of these characters is introduced. The input layer consist of raw pixel value with no trainable parameters. Since the available dataset is not sufficient to train the deep CNN, techniques like dataset increment and dropout are used to prevent the overfitting problem. Depth, width and number of parameter of network are varied in order to find the architecture with maximum accuracy. The results of two models are presented resulted in highest testing accuracy of 0.98472 for the model combined with dataset increment and 0.982681 for the model combined with dropout technique [8].

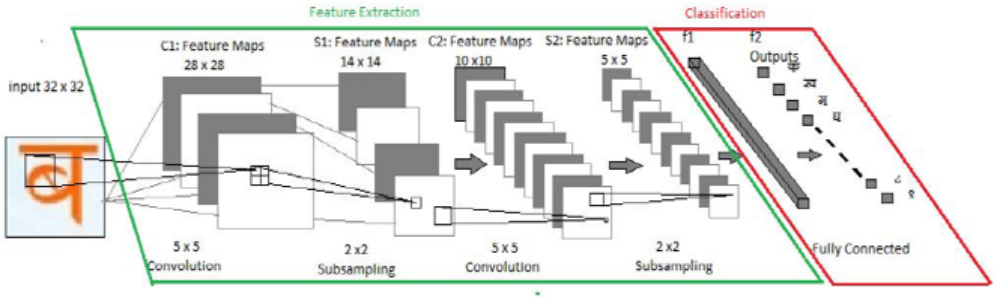


Figure 2.15: Convolutional Neural Network [8].

2.2.1 Review of Deep Convolutional Neural Networks for Image Classification

Review of over 300 publications has been done which focuses on how CNNs have evolved after the development of computing power around mid-2000s and their applications. It also traces the path of CNNs development from their early successes to recent implementation of deep learning systems. In the early stages of completing computer vision tasks, feature descriptors were used to extract handcrafted features and these features were input to trainable classifier. The major drawback in this dual-stage approach was that the accuracy of classification was majorly dependent on feature descriptors. But these days, nonlinear information processing used by deep learning models for feature extraction has been able to overcome these challenges. Among them, CNNs have become the leading architecture for most image recognition and classification tasks. Discussion on how CNN is inspired by neuroscience is also done. The main discovery was that neurons in different stages respond differently to specific stimulus, ignoring others that resulted in strong responses from neurons in later stages. Backpropagation algorithm was first introduced as a chain-rule algorithm which was inefficient. The modern efficient form of backpropagation algorithm was first used in ANN and later was successfully applied to CNN on a large scale. Though the performance of CNN has surpassed human-level performance on several single label image classification, it is still a challenge for deeper understanding of multilabled images. Rise of DCNN has inspired researchers to find out their robustness and computational characteristics, their trends and challenges [14].

Method for fulfiling the needs of segmentation for multi-digits number recognition in natural scene images. CNNs require fixed dimensional inputs while number image contains unknown number of digits. The model integrates segmentation and recognition in a probabilistic framework. Hidden Markov Model (HMM) is used to model the images and CNN to model digit appearance. CNN improved the performance compared with using Gaussian Mixture model as the digit model. Competitive results on SVHN dataset are observed. Single digit recognition accuracy improved as compared to traditional K-means. The accuracy obtained by the model is 91.4% [20].

Neural Network is designed to increase training speed and decrease implementation complexity. It uses convolutional filters, randomly-valued classifier-stage inputs weights, use of least square regressions to train the classifiers output weights in a single batch and linear classifier stage-output units. The network is performed on MNIST, NORB-small and Google SVHN dataset. The networks performance on SVHN database is competitive when compared to traditional methods giving the 4% error [21].

2.3 Architecture of Deep Convolutional Neural Networks (CNNs)

Convolutional neural networks are feed-forward networks i.e. the information flow takes place in one direction only [14].

2.3.1 Correlations between depth and width in CNNs

Though DCNNs have gained popularity in recent years, but to improve the accuracy of image classification, they are becoming more complex and time consuming. In practical applications, developers, researchers and engineers face the problem of constrained time budget while implementing DCNNs. Under this constraint, model architecture should be modified with trade-offs among the factors like depth, number and size and strides. An architecture by modifying baseline model while keeping its time complexity is presented. All the comparisons are

made by training the model on ImageNet dataset for 75 epochs. Time complexity of all convolutional layers is calculated as follows:

$$O(\sum_{i=1}^{n} n_{l-1}.s_l^2.n_l.m_l) \tag{2.1}$$

The time cost of full-connected layer and pooling layer is not considered in complexity and they take approximately 5-10% of computational time. Their input and output dimensions are also fixed. [Fig. 2.17] shows the configurations of model under constrained time complexity.

	top-1	top-5	d	stage 1	pool	stage 2	pool	stage 3	pool	stage 4	comp.
A	37.4	15.9	5	$(7,64)_{/2}$	$3_{/3}$	(5, 128)	$2_{/2}$	(3, 256)×3			1
B	35.7	14.9	8	$(7,64)_{/2}$	$3_{/3}$	(5, 128)	$2_{/2}$	(2, 256)×6			0.96
C	35.0	14.3	6	$(7,64)_{/2}$	$3_{/3}$	(3, 128)×2	$2_{/2}$	(3, 256)×3			1.02
D	34.5	13.9	9	$(7,64)_{/2}$	$3_{/3}$	(3, 128)×2	$2_{/2}$	(2, 256)×6			0.98
E	33.8	13.3	11	$(7,64)_{/2}$	$3_{/3}$	(2, 128)×4	$2_{/2}$	(2, 256)×6			0.99
F	35.5	14.8	8	$(7,64)_{/2}$	$3_{/3}$	(5, 128)	$2_{/2}$	(3, 160)×5+(3, 256)			1
G	35.5	14.7	11	$(7,64)_{/2}$	$3_{/3}$	(5, 128)	$2_{/2}$	(3, 128)×8+(3, 256)			1
H	34.7	14.0	8	$(7,64)_{/2}$	$3_{/3}$	(3, 64)×3+(3, 128)	$2_{/2}$	(3, 256)×3			0.97
I	33.9	13.5	11	$(7,64)_{/2}$	$3_{/3}$	(3, 64)×3+(3, 128)	$2_{/2}$	(2, 256)×6			0.93
J	**32.9**	**12.5**	11	$(7,64)_{/2}$	$3_{/3}$	(2, 128)×4	$2_{/2}$	(2, 256)×4	$3_{/3}$	(2, 2304)+(2, 256)	0.98

Figure 2.16: Architecture of the model with constrained time complexity [15].

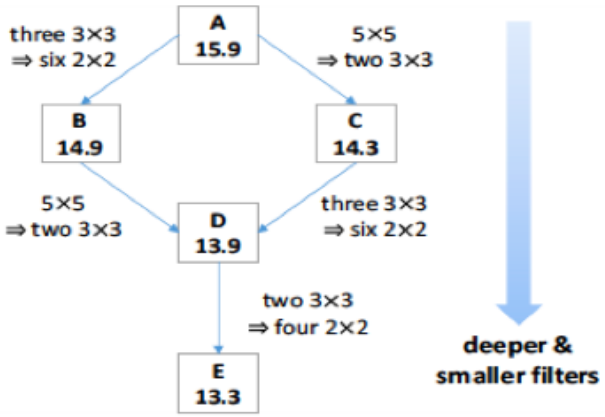


Figure 2.17: The relation of models between depth and filter sizes [15].

[Fig. 2.18-2.20] concludes that though the depth of network is of high priority for improving accuracy but the accuracy remains same if depth is overly increased

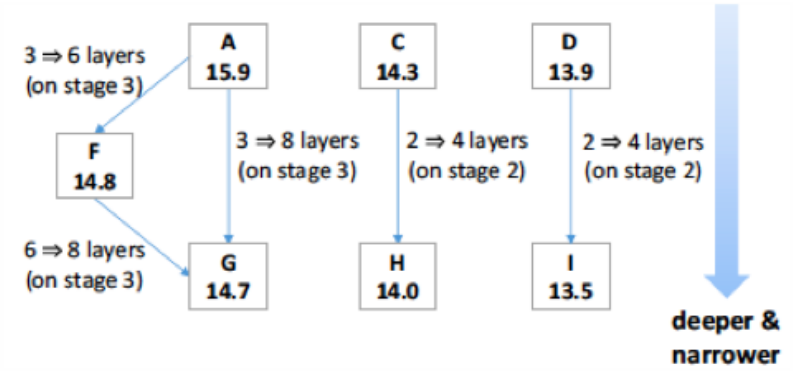


Figure 2.18: The relation of models between depth and width [15].

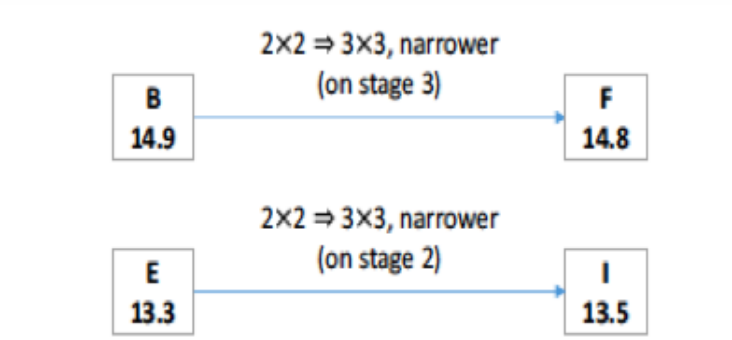


Figure 2.19: The relation of models between width and filter sizes [15].

even when the width is not changed. Also, the model is obtained which achieves 40% less complexity and 20% faster than AlexNet [15].

Though the filters attributes have impact on overall accuracy of model but their weights is separately trained by its own residual error and the relation between them is not explored for better training of network. An approach is proposed which uses the explicitly defined logical relation between filters in convolutional layers. This approach is verified with five benchmark datasets. Its performance is observed to be better than number of traditional methods of CNN. Opposite relation between filters after the input layer, rotational associated filter in shallow level and correlation like translation and scaling in deeper layers on CNN are observed [Fig. 2.21].

Two approaches Static Correlative Filters(SCF) and Parametric Correlative Fil-

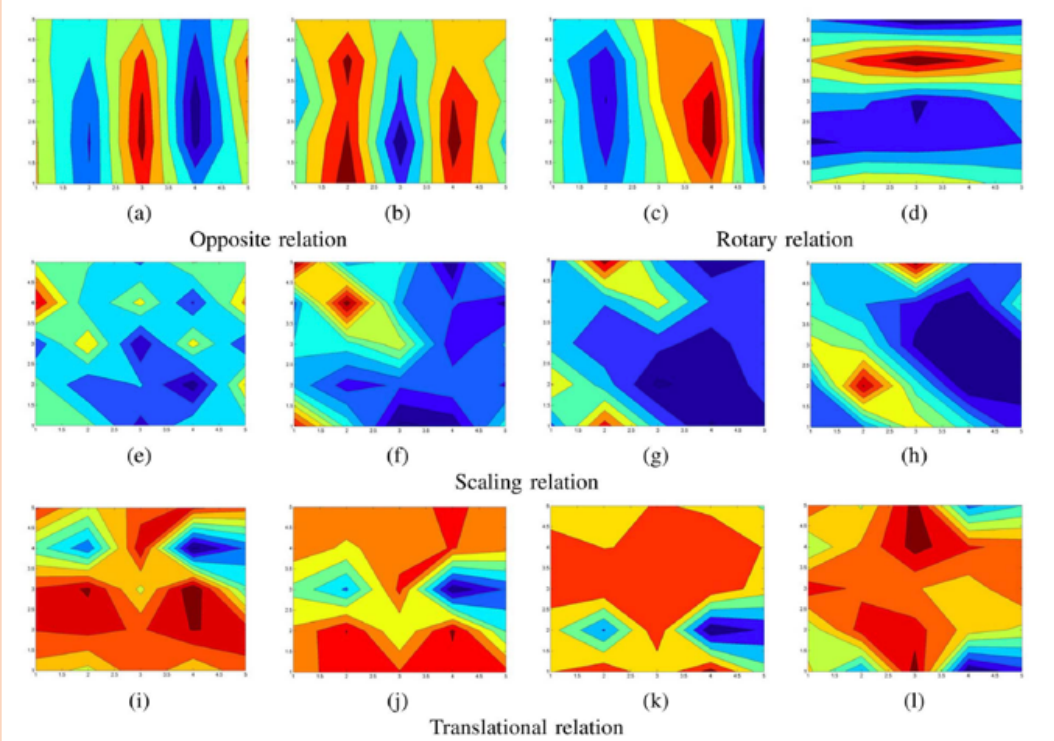


Figure 2.20: Correlations between filters of CNN [16].

ters(PCF), have been used for training purpose. SCF takes the advantage of freely trained filters and predefine some filters to be master filters such that other dependent filters are used for transformation of their attached masters. PCF is an extension of SCF. It takes the master filter as input and exports its correlated dependent filter so as there are no freely trained weights. With this approach, CIFAR-10 has achieved error rate of 9.27 (SCF+PCF) which is better than earlier approaches having error rate of more than 9.78. Similarly error rate of CIFAR-100 , STL-10, MNIST and SVHN are 30.460.13 , 22.940.19, 0.290.007 and 1.900.010 respectively [16].

2.3.2 Dropout techniques to prevent overfitting of deep learning model

Deep convolutional neural networks due to their large architecture with large number of trainable parameters are susceptible to overfitting. Overfitting arises when model performs on training data too well that it negatively impacts the perfor-

mance on testing data or any unseen data. It can be seen if validation accuracy is less than training accuracy. Dropout is one of the techniques to prevent overfitting [8].

Conventional dropout technique results in increased training time. Different approach of applying dropout so as to improve training speed is proposed. This is a controlled dropout which drops the hidden units of layer in column-wise or row-wise manner on the matrices whereas conventional dropout drops the units randomly. This approach is experimented on feed-forward neural network for MNIST dataset and CNNs for CIFAR-10 and SVHN datasets. Training data is trained with compressed parameters whereas testing data is tested with original parameters. Dropout is implemented in the first step of iterative training before the forward propagation. Results [Fig. 2.22] shows the better performance of CNN with controlled dropout in both training speed and accuracy as compared to conventional dropout technique [17].

	Without Dropout	**Dropout**	**Controlled Dropout**
MNIST on CPU			
Error (%)	1.77	1.11	1.12
Time (sec)	68,619.29	69,159.77	31,523.33
MNIST on GPU			
Error (%)	1.75	1.12	1.10
Time (sec)	2,841.05	3,116.84	2,494.44
CIFAR-10 on GPU			
Error (%)	17.53	15.23	15.27
Time (sec)	15,659.11	15,661.19	14,678.80
SVHN on GPU			
Error (%)	5.31	3.35	3.37
Time (sec)	16,623.96	17,079.10	15,760.21

Figure 2.21: Test Error and Training Time [17].

Standard dropout uses fixed dropout probability while training the network. Time scheduling for dropout training is proposed so as to achieve the adaptive regularization and provide the smooth initialization for weights optimization. It is achieved by the notion of curriculum learning which divides training data examples based on its difficulty level. The model is configured so that learning starts

with the easier examples. It allows to train better models. Experiments are done on seven image classification datasets. Results are better and if not, then same as standard dropout method. It is proved to be effective. Future scope of this work includes implementing the same on other computer vision tasks like recurrent neural networks, inter-neural connection inhibitions [18].

2.4 Deployment and Development of Deep Learning based Applications

Many products of Internet of Things are available in market which learn from their environmental data by using traditional machine learning techniques like Googles Nest Learning Thermostat. Emerging IoT devices are using deep learning to capture, understand and respond to their environments like Amazon Echo which understands and converts human voice commands into text and uses it for searching relevant information. Also, there is Microsofts IoT based face-recognition security system that uses deep learning to unlock door when it recognizes its users face. But deep learning requires high power consumption and high performance computing which is very difficult on low power IoT device for real time implementation. Offloading the deep learning workloads to cloud has shown better performance. But offloading requires very good quality network connection which is not available at all places. Tensor-Flow an existing and an open source deep learning framework is migrated to IoT devices. There are other ways also to enable deep learning on IoT devices that is to build the inference engine from scratch so as to avoid the difficult task of migrating. Also, optimizing the deep learning model into executable code for target platform is an option [9].

A framework [Fig.2.18] to infer the situation from the low-level contexts of event that occur in given space and time of IoT sensor data like persons, objects, scenes which are derived from multiple deep neural networks learning results is suggested. This framework consists of two modules: Low-level context learning module which implies the multiple deep neural network learning results in order to improve the accuracy and reliability of situation reasoning and Situation learning module which

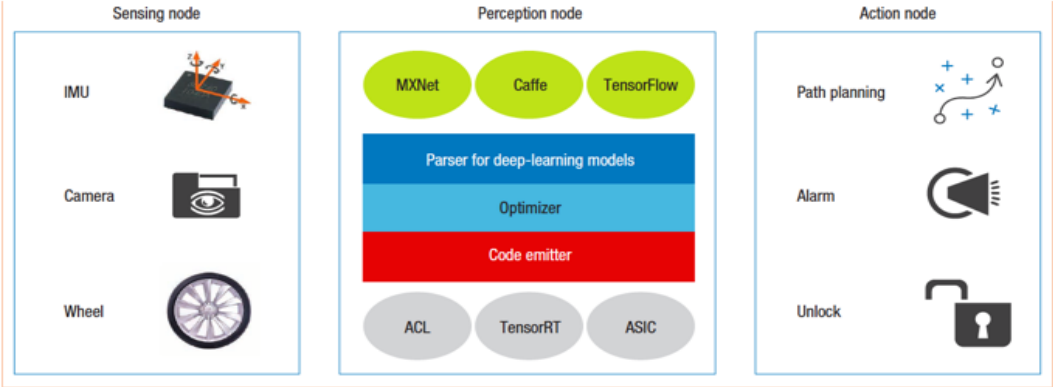


Figure 2.22: Basic services of IoT devices [9].

reflects the spatio-temporality. The experiment was done on real-time IoT environment using 87 cases. The result of reasoning stored can be used for learning a case base. This framework reflects the main characteristics of IoT sensor data especially spatio-temporality which means the context depends on the occurrence time and location of IoT sensor data. As per first attempt, the results are satisfactory and method is suitable in terms of hit-ratio which is 76% of this framework [10].

A fully functional parking management system [Fig.2.20] analysis by server to GUI by using the detection algorithm of deep convolutional neural networks is developed. Visual methods are used to recognize vacant stalls in parking lot using already installed cameras. One camera is used to monitor many stalls thereby reducing the cost of one stall. The network is trained for 2000 iteration with the batch size of 256.The training is limited to top three fully connected layers. The detection method shows better results when compared to the present approach of Area Under the Curve metric by 8.13% [11].

A method for real-time emotion recognition from facial images using Raspberry Pi II is proposed. This method is in alignment with the requirement for understanding the human emotions by understanding the facial expressions. The method is divided into three steps: face detection using Haar cascade, feature extraction using Active Shape Model and Adaboost classifier. The classifier classifies the images taken through webcam into five emotions: anger, disgust, happiness, neutral

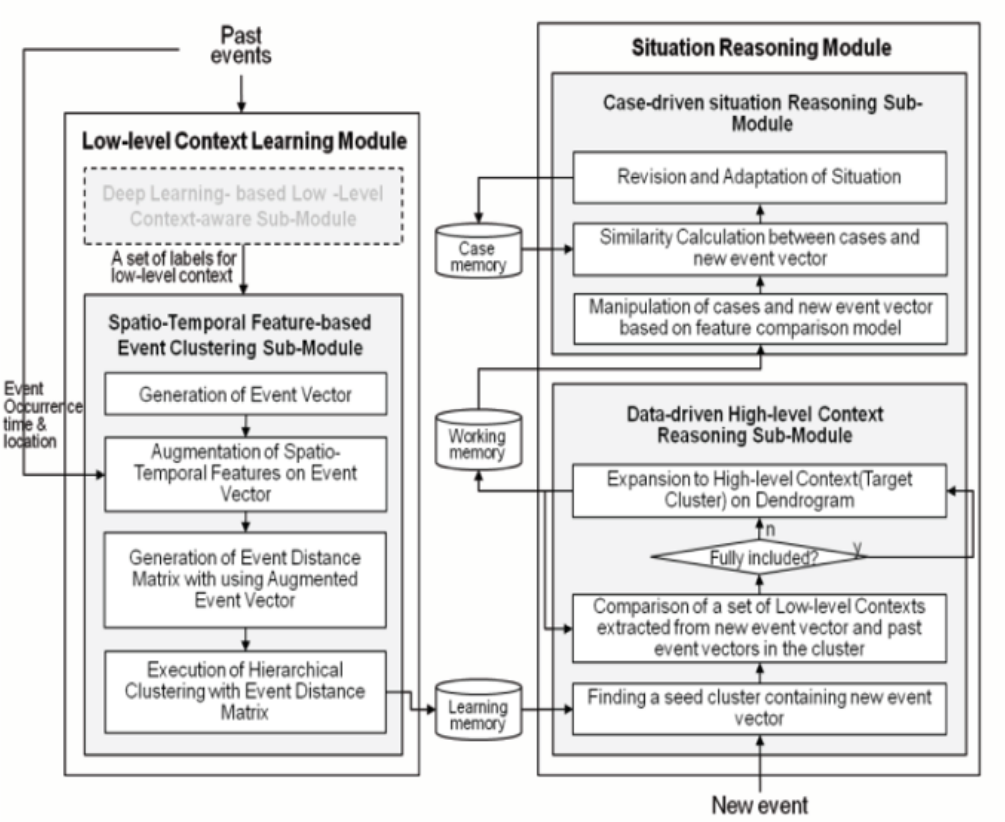


Figure 2.23: Architecture of the framework for situation identification [10].

Figure 2.24: Screenshots of the application showing parking lot and its status [11].

and surprise. The image pre-processing is done by cropping according to required size and converted into gray image. 26 facial points are extracted by using Active

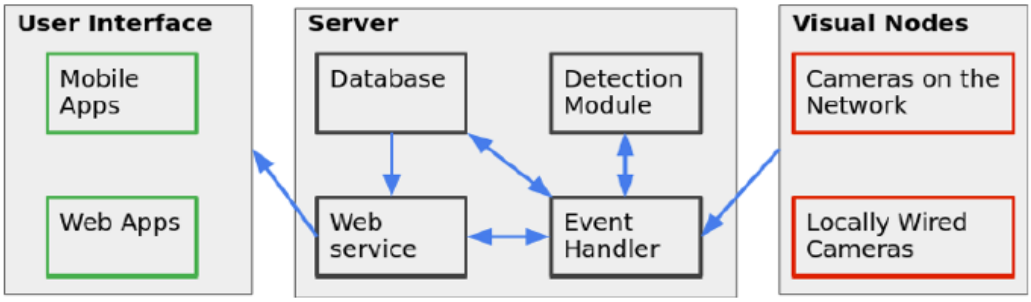


Figure 2.25: Fully functional system architecture from server side image analysis to GUI [11].

Shape Model(ASM) feature extraction technique and Euclidean distance is calculated between points. The software is developed, deployed and tested for real time implementation on Raspberry Pi II. The result shows 94% recognition accuracy with processing time of 120 ms on Linux platform [12].

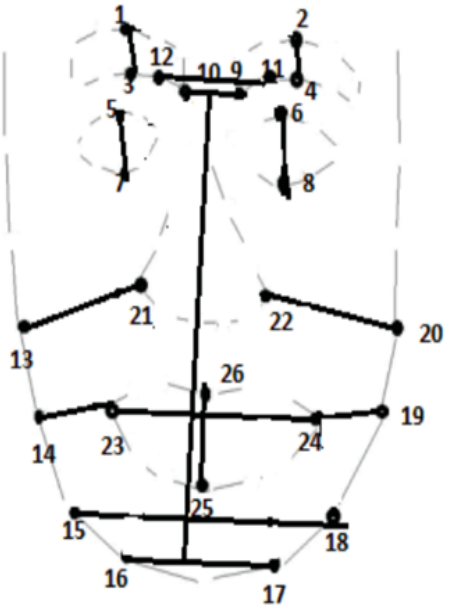


Figure 2.26: 26 facial points extracted using ASM [12].

The framework for IoT based data processing for Automated Industrial Meter Reader using Raspberry Pi is presented. This application also includes digital image processing. The four-step process followed by device Image Acquisition using Raspberry Pi Camera Module, Optical Character Recognition using feature

	Anger	Disgust	Happy	Neutral	Surprise
Anger	90%	10%			
Disgust		100%			
Happy			80%	10%	10%
Neutral				100%	
Surprise					100%
Recognition Accuracy	90%	100%	80%	100%	100%

Figure 2.27: Confusion matrix showing the accuracy of proposed model for emotion recognition [12].

extraction technique, Internet Upload Mechanism using Google Forms and Online Data Processing using Google Spreadsheet. This device uploads collected data to cloud storage for further data processing. The total time required by the hardware is upto 35 s. Test sample is of 100, out of which 62 images were processed without any errors [Fig.2.24] [13].

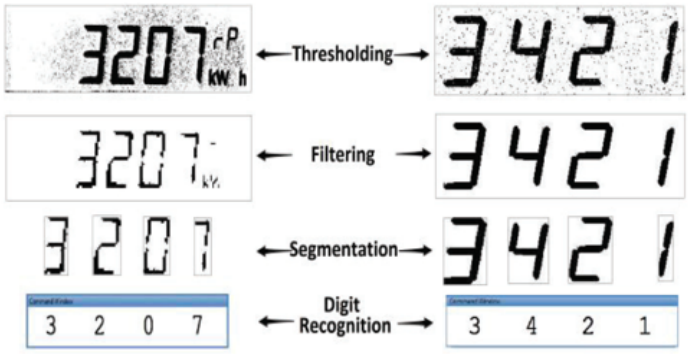


Figure 2.28: Stepwise Results of Optical Character Recognition for LCD and LED Meter [13].

Guidelines for developing and deploying CNN based applications with experimental results on CIFAR-10 dataset are given. It is the requirement to choose appropriate hyper-parameters i.e. structure, learning rate, training algorithm, regularization techniques etc. for optimized solutions. CNN structure has parallelism in its associated algorithms so for its parallel implementation, hardware choices are

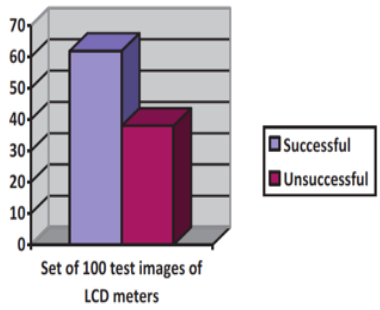


Figure 2.29: Success Rate of Set of 100 Test Samples [13].

Field Programmable Gate Arrays (FPGA) and Graphic Processing Units (GPU). FGPA is suitable for final stage of implementation in real-time and has lower power consumption on the other side GPU trains the model many times faster than CPU. Another cheaper solution is using Amazon Web Services. Software requirements are operating system and deep learning frameworks. DL frameworks are developed on 64-bit linux machines, so the latest 64-bit Ubuntu can be picked. cuDNN is a GPU-accelerated library especially for deep learning. Available DL frameworks are Torche, Caffe, Theano, TensorFlow. Optimization techniques can be selected as Stochastic Gradient Descent or Adam. Adam has given better results in terms of training accuracy and speed. The main aim should be to increase accuracy and network complexity should be increased gradually [19].

Gender recognition in computer vision due to its applications such as surveillance in public places, directed advertising, among others. The good result obtained using deep convolutional neural network in vision task make them an attractive tool for improving the capacity of gender recognition systems. A deep convolutional network architecture to classify as male or female person the candidate regions previously detected using Haar features embedded in an AdaBoost is proposed. The data set used for training and testing come from the Labeled Faces in the Wildand Gallagher's dataset. We have evaluated the classification results on the proposed architecture and have obtained an average of 95.42% and 91.48% accuracy for the training set and for the test set, respectively[25].

Chapter 3

Proposed Handwritten Devanagari Character Recognition Framework

After reviewing the different works on character recognition using deep learning approach, research gaps have been found. Based on these gaps, problem is formulated whose solution is given in next chapter.

3.1 Research Gaps

This section tells about the gaps encountered during literature review in the area of Devanagari Character Recognition using different approaches.

(i) Ideal combination of classifier for character recognition of Devanagari is need to be investigated [1].

(ii) Recognition system for complete handwritten document in Devanagari script is yet to be implemented [8].

(iii) Character recognition in scene images for other similar scripts like Bangla and Kannada needs to be investigated [2].

(iv) Optimization of hyperparameters of deep learning model for Devanagari character recognition needs to be done [16].

(v) Low-end computing device based automated offline recognition system for handwritten Devanagari character using Raspberry Pi is to be implemented [9,12,13].

3.2 Problem Formulation

Recognizing handwritten characters of Devanagari has been done using various traditional state-of-the-art machine learning methods. Devangari Handwritten Character Dataset(DHCD) poses different challenges in character classification because the images are taken from handwriting of different people and moreover, all characters have different fonts, angles and styles. Deep Convolutional Neural networks are effective in image classification because of their tolerance to transitional and rotational variance. It is of interest to find out the impact of width i.e. number of filters and optimization function of deep convolutional neural networks on test accuracy of DHCD. Also, the performance of deep CNN can be further improved using dropout regularization. The model so implemented is finally deployed on Raspberry Pi 3 for the practical use of deep CNN.

3.3 Objectives

Following are the proposed objectives:

(i) To study and analyze the work done in the area of Devanagari character recognition and learning on low-end devices.

(ii) To propose an optimized deep learning framework for handwritten Devanagari character recognition.

(iii) To implement a proposed handwritten Devanagari Character Recognition framework using Raspberry Pi 3 and its camera module.

(iv) To test and validate the proposed framework.

3.4 Proposed Handwritten Devanagari Character Recognition Framework

The architectural layout of the framework is shown in Fig.4.1. The implementation is done in mainly two steps:

(i) Training the DCNN model on DHCD (A1 & A2 shown in Fig.4.1).

(ii) Creating a Graphical User interface (GUI) application using Raspberry Pi III and its camera module (A3 shown in Fig.4.1).

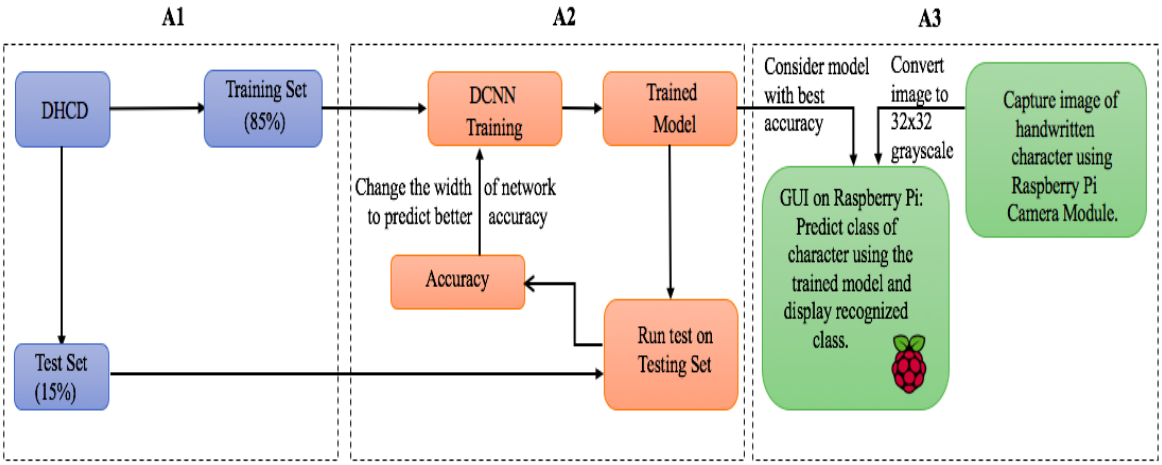


Figure 3.1: Architectural Layout of Character Recognition Framework.

Devanagari Handwritten Character Dataset is split into training and testing sets. DCNN is trained using training set for 12 epochs and accuracy is predicted on testing set. Number of filters of each layer are changed to predict better accuracy.

Following are the inputs of GUI application (A3 shown in Fig.4.1):

(i) Image of character captured using Raspberry Pi camera module.

(ii) Trained model with best predicted accuracy.

Image is processed and converted to 32×32 grayscale image. DCNN takes raw pixel values of image and predicts the class. Recognized class is finally displayed on screen.

Chapter 4

Design and Implementation of Proposed Framework

The sequence diagram showing the design of proposed handwritten Devanagari character recognition framework is given in this chapter.Its implementation are also given in this chapter.

4.1 Sequence Diagram of proposed handwritten Devanagari Character Recognition Framework

A sequence diagram shows object interaction in time sequence. It tells the objects and classes involved in the scenario and the sequence of messages exchanged between them.

The sequence diagram of proposed framework is shown in Fig.4.1.

As it can be seen in figure, objects are User, GUI and System. It also shows the direction of messages. GUI displays menu to user : Get Image and Recognize. User gets the image. GUI process image to system and generate output class by system.

4.2 Dataset description and processing

Devanagari script is used to write Nepali, Hindi, Marathi and similar other languages. DHCD follows the Nepalese writing system. It contains base form of 36 consonants shown in Fig.4.2 and 10 numeral characters shown in Fig.4.2 . These images are taken from the dataset. The dataset is generated by cropping char-

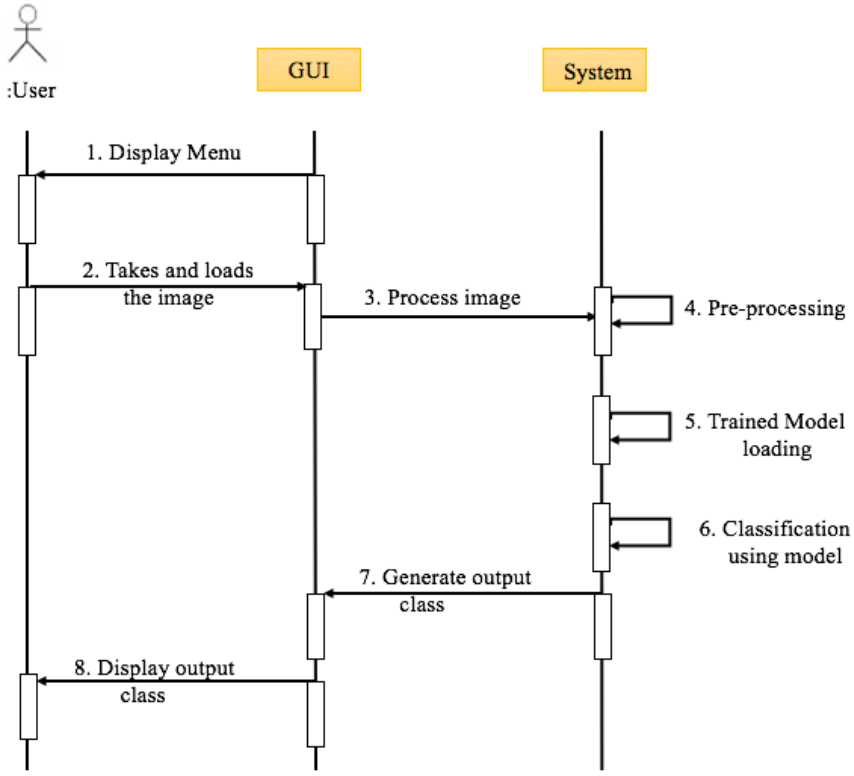


Figure 4.1: Sequence diagram of proposed framework.

acters from different individuals handwritten documents. Characters are then preprocessed and converted to grayscale 32×32 images. There are two thousand images of every character. Every image in dataset is unique. Total 92,000 images are split into training set (85%) of 78,000 images and testing set (15%) of 13,800 images. The dataset poses challenge as every character is written differently and also, some characters has the same structure in Devanagari script [8].

The dataset is given in Train and Test folders. Each folder contains 46 folders of 46 different classes. Each class folder contains images of that class as shown in Fig.4.3.

This data is need to be processed and converted into format that machine understands. The preprocessing is done with the help of pickle library and data is stored in pickle file. The pickling of data is shown in Fig.4.4. After processing, data is partitioned into three sets:training, validation and testing set. Each of

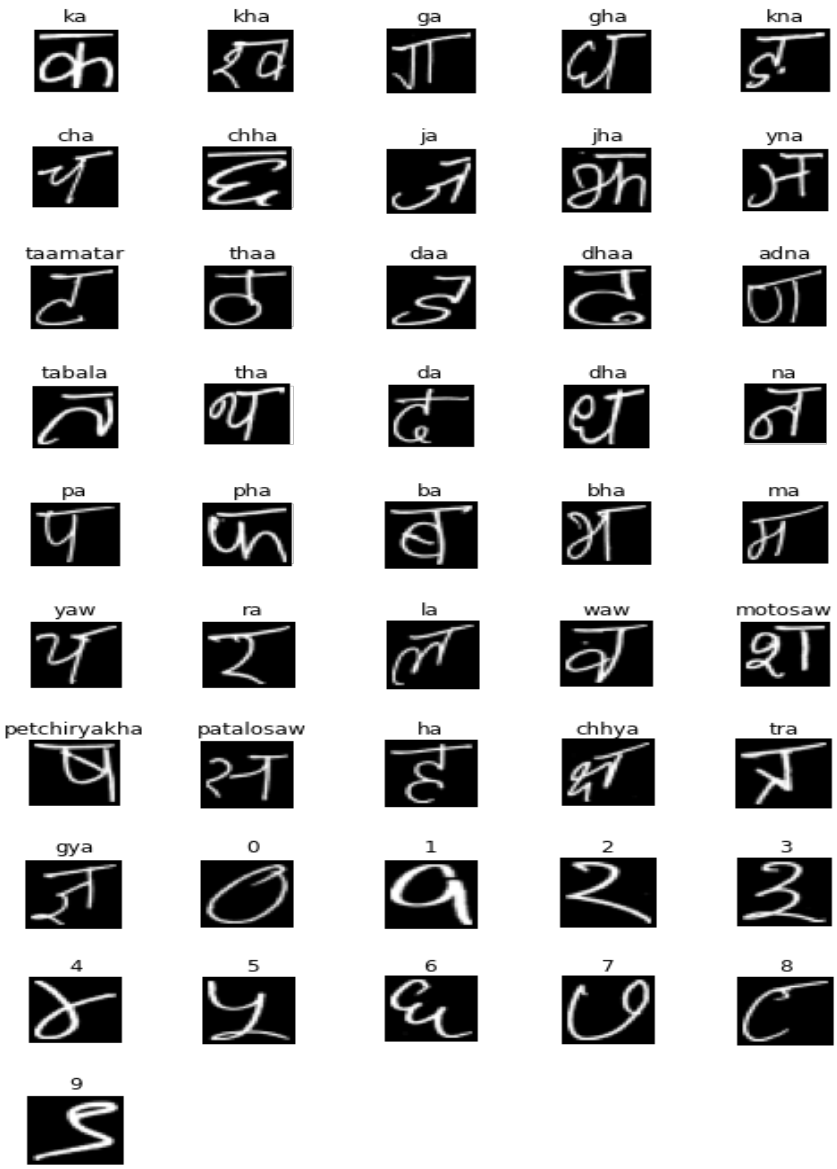


Figure 4.2: Characters from dataset with their class name.

these sets contain the input image in the form of a numpy array and its associated labels specifying the actual character in the image. Shapes of these sets are given in table 4.1.

Figure 4.3: 46 Folders containing 46 different classes in each train and test folder.

```
Pickling /Users/jasminejaidka/Desktop/Test_label/Train/digit_6.pickle.
/Users/jasminejaidka/Desktop/Test_label/Train/digit_6
('Full dataset tensor:', (1700, 32, 32))
('Mean:', -0.26933062)
('Standard deviation:', 0.38140231)
Pickling /Users/jasminejaidka/Desktop/Test_label/Train/digit_7.pickle.
/Users/jasminejaidka/Desktop/Test_label/Train/digit_7
('Full dataset tensor:', (1700, 32, 32))
('Mean:', -0.24291903)
('Standard deviation:', 0.40167698)
Pickling /Users/jasminejaidka/Desktop/Test_label/Train/digit_8.pickle.
/Users/jasminejaidka/Desktop/Test_label/Train/digit_8
('Full dataset tensor:', (1700, 32, 32))
('Mean:', -0.27318266)
('Standard deviation:', 0.38458139)
Pickling /Users/jasminejaidka/Desktop/Test_label/Train/digit_9.pickle.
/Users/jasminejaidka/Desktop/Test_label/Train/digit_9
('Full dataset tensor:', (1700, 32, 32))
('Mean:', -0.228149)
('Standard deviation:', 0.40514278)
Pickling /Users/jasminejaidka/Desktop/Test_label//Test/character_10_yna.pickle.
/Users/jasminejaidka/Desktop/Test_label//Test/character_10_yna
('Full dataset tensor:', (300, 32, 32))
('Mean:', -0.28681192)
('Standard deviation:', 0.37168479)
```

Figure 4.4: Pickling of data.

Table 4.1: Partitioned sets and their shapes

Set	Image data Shape	Labels Shape
Training	(69000,32,32)	(69000)
Validation	(9200,32,32)	(9200)
Testing	(13800,32,32)	(13800)

Table 4.2: Reshaping of partitioned sets for input layer

Set	Image data Shape	Labels Shape
Training	(69000,32,32,1)	(69000,1)
Validation	(9200,32,32,1)	(9200,1)
Testing	(13800,32,32,1)	(13800,1)

4.3 Deep Convolutional Neural Network

The baseline model used in our recognition system comprises of two convolutional layers, each followed by pooling layer and one fully connected layer as shown in Fig.4.3. Non - Linear function applied to all the layers is Rectified Linear Unit (ReLU). The input is a 32×32 grayscale image and output is a softmax layer which predicts the probability that an image belongs to each class. The model is trained for 12 epochs with no batch normalization. Dropout layer is added to prevent overfitting in network.

The input layer comprises of the dataset. The partitioned sets are reshaped as given in Table 4.2. because CNN take color channel of image as input layer also. Since the image is grayscale, number of color channels is 1.

In convolution, box of weights also called as filter is multiplied by every part of the image. The size of filter is small relative to the whole image. This small box of weights is a feature finder which outputs a very positive number when it detects a feature like corners, end-points, edges and very negative number when it detects something else. So, filters can find the character in image no matter where it is because the convolution uses the same weight everywhere. This is the main motivation behind convolution in the neural networks. Along with weight values, bias values are also associated with each layer which are added to the value obtained after multiplying with weights. We have kept the filter size same throughout the experiment which is 5x5 but the number of filters are changed to

train the network better. Stride is the number of pixels, filters move at a time. It is also fixed in our experiments i.e. stride of 1.

Next step is the pooling or sub-sampling of the output of previous convolution layer. It helps to know if the feature was present in the certain area of image or not. It is done by down sampling or changing the resolution of image. In our experiment, pooling is done by the factor of 2 which means 32x32 image is converted to 16x16 in both horizontal and vertical directions.

Last step is the fully connected layer which is a basic feed forward fully connected neural network. It takes the output from the last pooling layer and flatten it into a vector which will be the input of this layer. Finally, the softmax function predicts the output that is the class to which the input belongs.

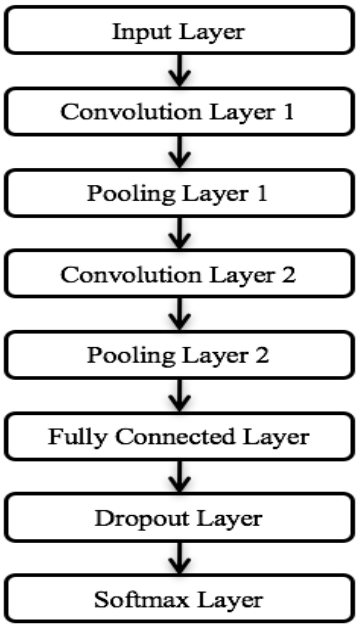


Figure 4.5: Baseline model of DCNN.

4.3.1 Training algorithm of model

The models are trained for 12 epochs on training set. After each epoch, validation cost and accuracy are calculated. At the end of training, testing accuracy is calculated. The training algorithm followed by deep convolutional neural network consists of the following steps:

(i) Load dataset into X and Y variables where X is the pixel values and Y tells its class.

(ii) Initialize weight and bias values associated with each layer randomly.

(iii) Train model on training dataset and predict accuracy. Non-linearity used after each layer is Rectified Linear Unit (ReLU) because training with gradient is faster for ReLU [8].

$$y = relu(x) = max(0, x) \tag{4.1}$$

$$\frac{\partial y}{\partial x} = \begin{cases} 0, x < 0 \\ 1, x > 0 \end{cases} \tag{4.2}$$

(iv) Find cost function. Cost function is a negative log likelihood function. The main idea behind the training is to minimize the cost function which will maximize the likelihood. Cost Function, J (for multiple classification):

$$J = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{n,k} log_{n,k} \tag{4.3}$$

where K is the number of classes, N is the total number of inputs, tn,k is the indicator matrix or one-hot encoded matrix of 1s and 0s.

(v) Update weight and bias values to minimize the cost function by gradient descent and repeat steps 3 to 5 till last epoch. Gradient descent is done as:

$$W \leftarrow W - \eta \frac{\partial J}{\partial W} \tag{4.4}$$

where W is the weight value, η is a small number used to take small steps in the direction of grading.

4.3.2 Training using Amazon Web Services

All the models were trained using Amazon Web Services (AWS) [Fig.4.4]. It offers a broad set of global compute, storage, database, analytics, application, and deployment services that help the computations to be done fast enough. The instance used is t2.2 $\times$ *large* of Deep Learning AMI (Ubuntu). It is pre-installed with deep learning frameworks like TensorFlow, Theano and Keras optimized for high performance on Amazon EC2 instances. After an instance is launched, public DNS is given for connection via Secure Shell (SSH) and jupyter notebook is accessed from workstation. The dataset and python code are loaded and run on AMazon Web Services using jupyter notebook as shown in Fig.4.7.

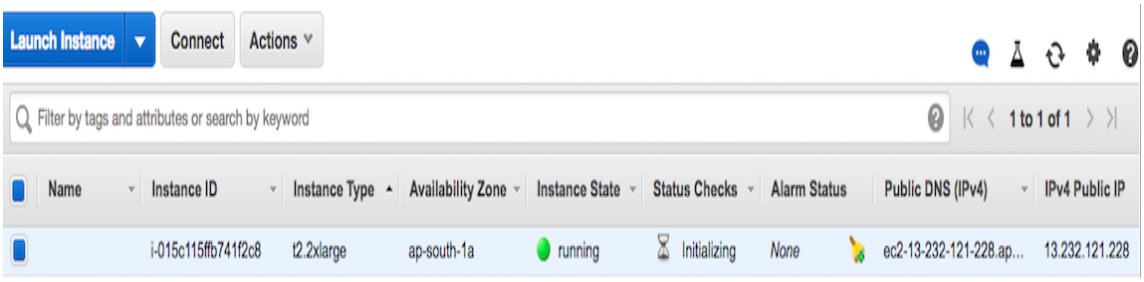


Figure 4.6: Running instance of Amazon Web Services.

4.3.3 Saving and Loading the model

Once the model with best accuracy is found, that entire model is saved in the h5 format. This single HDF5 file contains:

(i) The architecture of the model.

(ii) The weights of the model.

(iii) The training configuration.

(iv) The state of the optimizer.

This file is loaded in GUI using

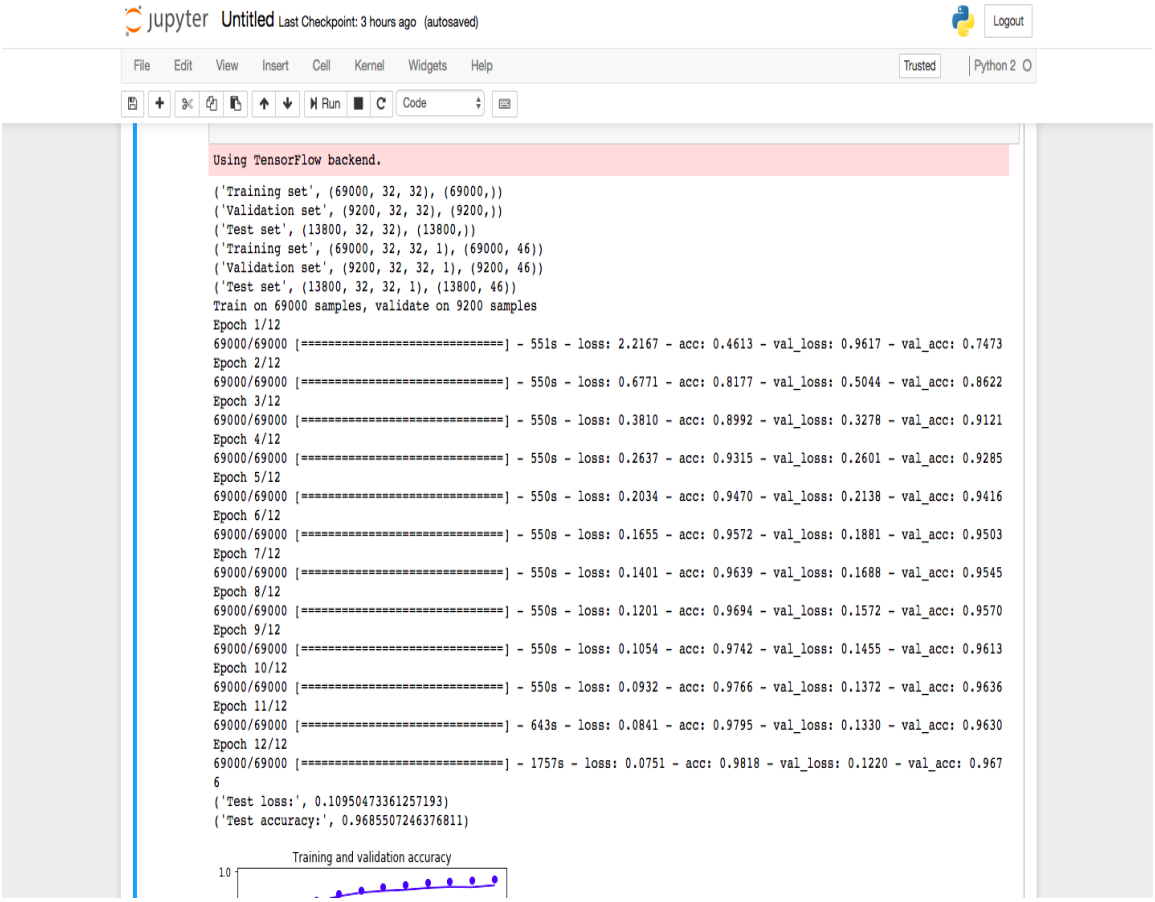


Figure 4.7: Training of model using Amazon Web Services.

4.4 Graphical User Interface

GUI application is built on Raspberry Pi 3 which takes image of character and trained model as input and displays the class of character. The image is preprocessed to remove noise and is reshaped into the format (-1,32,32,1) before given as input to model. The model is loaded in the application and then, prediction is made.

4.4.1 Image processing

The image captured by Raspberry Pi camera module is processed using the OpenCV library of Python. The camera captures the image in RGB color. First step was to convert it into grayscale image so as to simplify the following steps. The

grayscale image of character pa is shown in Fig.4.8.

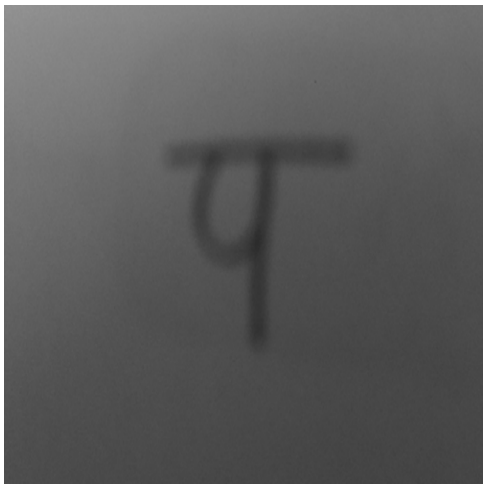


Figure 4.8: Grayscale Image.

From grayscale image, noise was removed from image using blur function. The blurred image is shown in Fig.4.9.

Figure 4.9: Blurred Image.

After eliminating noise, an adaptive threshold was applied to the image using the mean method to convert it into binary. It ensures that the threshold is applied properly regardless of non-uniform lighting or other such conditions. It uses the average pixel value of a cluster of pixels in the image to set a threshold value for that cluster. All the pixel values below that threshold become black and the rest

become white. This results in a white letter with black background as shown in Fig.4.10 .

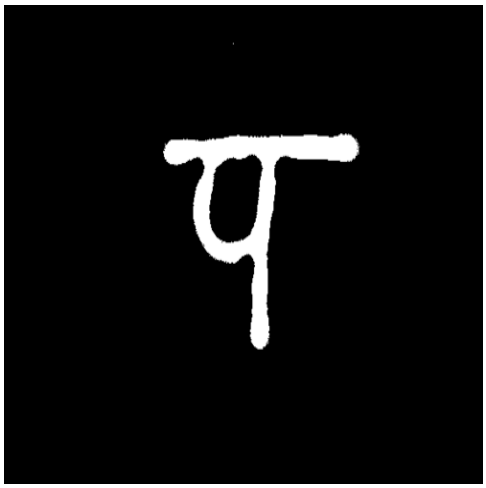


Figure 4.10: Adaptive Threshold Image.

The image was then dilated to reinforce the letter and ensure there are no small gaps that may have been removed by the adaptive threshold. The dilated image is shown in Fig.4.11.

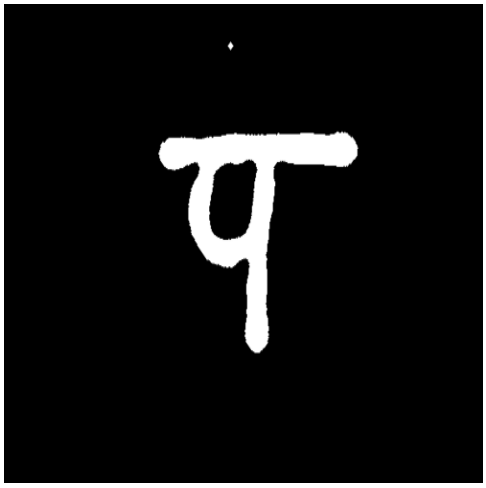


Figure 4.11: Dilated Image.

The contours of the image were then identified and the largest contour area is assumed to be the character. This area becomes the region of interest. The height and width of this region are returned for calibration reference. It helps to determine more appropriate values to input. It is marked on the original camera

capture by a red box as shown in Fig.4.12. This updated original image can be used in GUI to display the character.

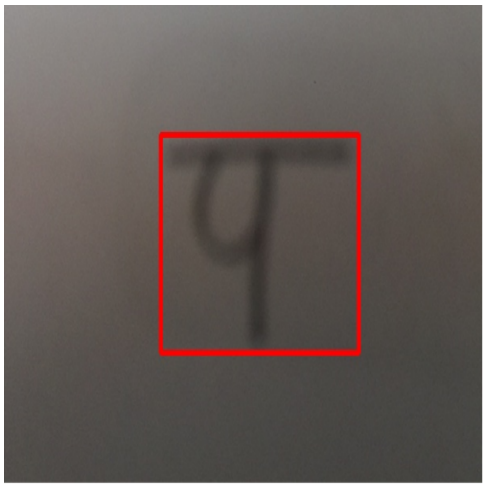


Figure 4.12: Original Image with Caliberation.

Finally, the image data in the region of interest is resized to a 32×32 size and converted to 32×32 binary matrix which is comprised of entirely ones and zeros. The matrix for processed "Pa" is shown in Fig.4.14 . This 32×32 matrix is reshaped to (-1,32,32,1) so as to be used as input to our trained model.

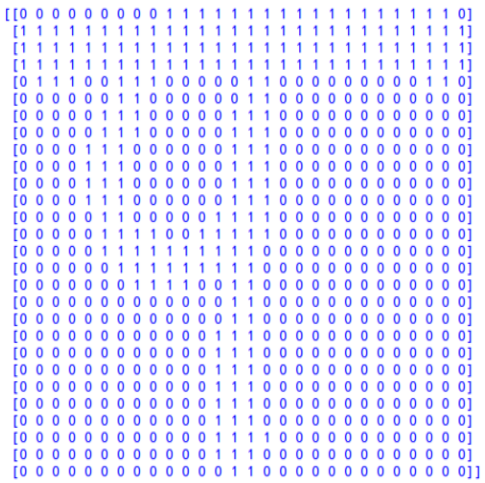


Figure 4.13: Binary Matrix with 0s and 1s.

4.5 Experimental Setup

The software and hardware used for training of models and building the GUI application are:

(a) Software:The programming language used is Python with below given additional libraries:

- Pickle:It is used for serializing and de-serializing a Python object structure. It converts image data into machine understandable format.
- Numpy:It is the fundamental package for scientific computing. It contains N-dimensional array objects in which dataset variables can be stored and is useful for linear algebra operations.
- Matplotlib:It is a Python 2D plotting lobrary which produces publication quality figures. It is used to plot the graph between Training and Validation Accuracy and cost function of all the models.
- Keras [with Tensorflow Backend]:Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. Tensorflow backend is used in our experiments. It is very simple to add layers in Keras.
- Open-CV:It is designed for computational efficiency with a strong focus on real world applications. CV2 library used in GUI application helps to process the image by removing noise from image and converting it into white character with black background image.
- Tkinter:It is a standard Python interface to the Tk Graphical User Interface(GUI) toolkit.It provides fast and easy way to create a GUI application.Widgets like various controls, such as buttons, labels and text boxes can be easily added to GUI using this

(b) Hardware:Below given are the details of the hardware used:

- Raspberry Pi 3 B: It is a series of small single-board computers with its own Raspbian operating system. It is released by Raspberry Pi Foun-

dation. Its specifications are Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1GB RAM, BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board, 100 Base Ethernet, 40-pin extended GPIO, 4 USB 2 ports, 4 Pole stereo output and composite video port, Full size HDMI, CSI camera port for connecting a Raspberry Pi camera, DSI display port for connecting a Raspberry Pi touchscreen display, Micro SD port for loading your operating system and storing data and Upgraded switched Micro USB power source up to 2.5A [Fig.4.14].

- Raspberry Pi Camera Module V2:The Raspberry Pi Camera Module v2 is a high quality 8 megapixel Sony IMX219 imagesensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It'scapable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and640x480p90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses thededicated CSi interface, designed especially for interfacing to cameras [Fig.4.14].

Figure 4.14: Raspberry Pi attached with its camera module

Chapter 5

Test and Analysis

The experiments were conducted by changing the width and optimization function of baseline model shown in Chapter 4. The details of the configurations of 10 models and accuracy achieved by each model is shown in Table 5.2. Architecture of model is written in the form:

$$(m, f1)_{/1} + (m, f2)_{/1} + n \tag{5.1}$$

where m is the size of filter which is fixed in all the observations, f1 and f2 are the number of filters used in first and second convolution layer respectively and n is the number of units in the last fully connected layer.

The mini-batch size is 200. Batch-normalization is not applied. Number of classes and epochs are 46 and 12 respectively. Learning rate of SGD and Adam is 0.001. The size of filters is constant i.e. (5,5). After training, accuracy and loss function of training and validation are plotted for each epoch.

5.1 Impact of width

It can be observed from Table 5.2 that with the increase in number of filters, accuracy increases but after Model 8, it starts decreasing.The highest accuracy achieved was of Model 8 i.e. 98.75% which is more than 98.26% achieved by the model in [8]. Also, the training time increased with the increase in width. It results in overtraining of network as well. It is evident from the increase in difference between validation and training accuracy shown in Fig.5.2. So, more width is not always better.

Table 5.1: Accuracy of models with respect to change in width of architecture and optimization function

Model	**Architecture of Model**	**Optimization Function**	**Test Accuracy**
1	$(5,16)_{/1}$+$(5,32)_{/1}$+64	SGD	95.37%
2	$(5,20)_{/1}$+$(5,50)_{/1}$+100	SGD	96.14%
3	$(5,30)_{/1}$+$(5,70)_{/1}$+150	SGD	96.72%
4	$(5,50)_{/1}$+$(5,90)_{/1}$+180	SGD	96.86%
5	$(5,64)_{/1}$+$(5,128)_{/1}$+250	SGD	96.94%
6	$(5,100)_{/1}$+$(5,200)_{/1}$+500	SGD	97.47%
7	$(5,100)_{/1}$+$(5,200)_{/1}$+500	Adam	97.55%
8	**$(5,100)_{/1}$+$(5,200)_{/1}$+500+Dropout(0.5)**	**Adam**	**98.75%**
9	$(5,100)_{/1}$+$(5,250)_{/1}$+1000+Dropout(0.5)	Adam	98.42%
10	$(5,200)_{/1}$+$(5,400)_{/1}$+1000+Dropout(0.5)	Adam	98.24%

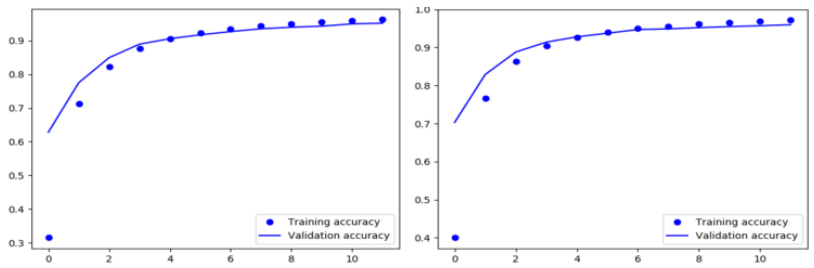
5.2 Impact of optimization function

Optimization functions used are SGD and Adam with learning rate of 0.001. It can be seen in Table 5.2 that Adam performed better as it converges very fast and the learning speed of the model is quite fast and efficient. It also rectifies problems like vanishing learning rate, slow convergence or high variance in the parameter updates which leads to fluctuating loss function.

5.3 Overfitting of Network

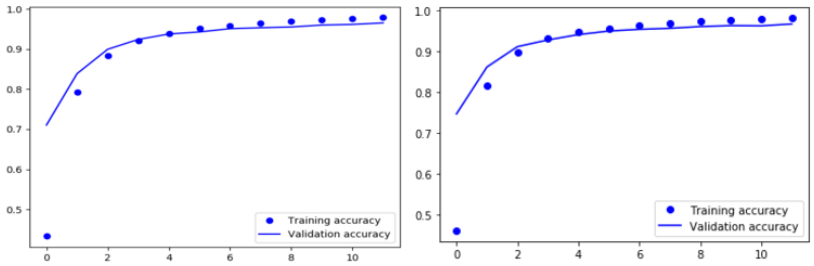
Overfitting of network was found by dividing the training dataset into validation and training sets. After each epoch, training and validation accuracy were calculated and plotted for each model as shown in Fig.5.1. Model is over trained if validation accuracy was more than training accuracy. Till Model 6, network was not overtrained but Model 7 was overtrained. To solve this problem, dropout layer was added to following models. Dropout layer drops units randomly in each iteration which results in different model every time thereby preventing overtraining of the network. Overtraining in Fig.5.1(g) is prevented by dropout in Fig.5.1(h).

Overtraining of network can also be seen in Fig.5.2(g) where validation loss is



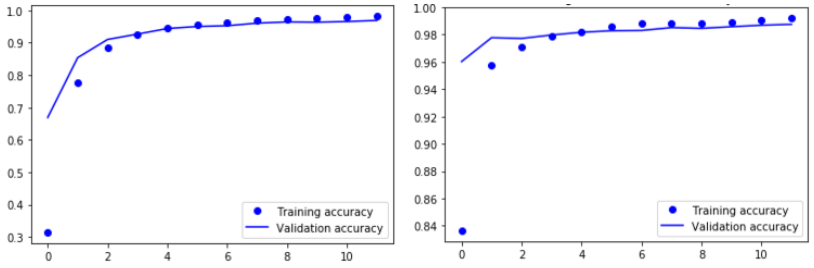
(a) Model 1

(b) Model 2



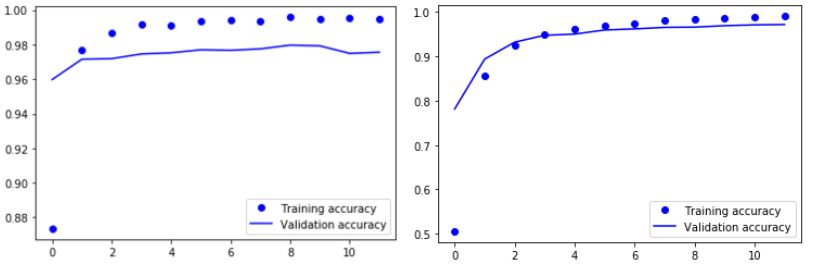
(c) Model 3

(d) Model 4



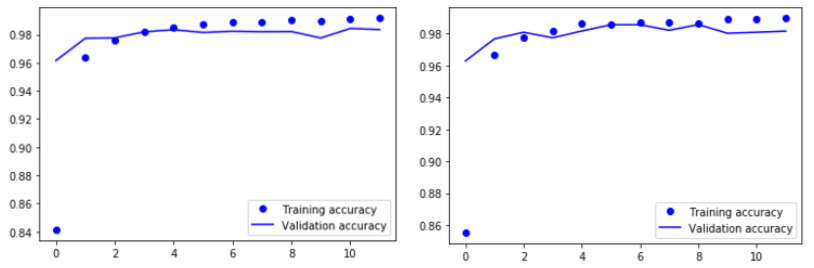
(e) Model 5

(f) Model 6



(g) Model 7

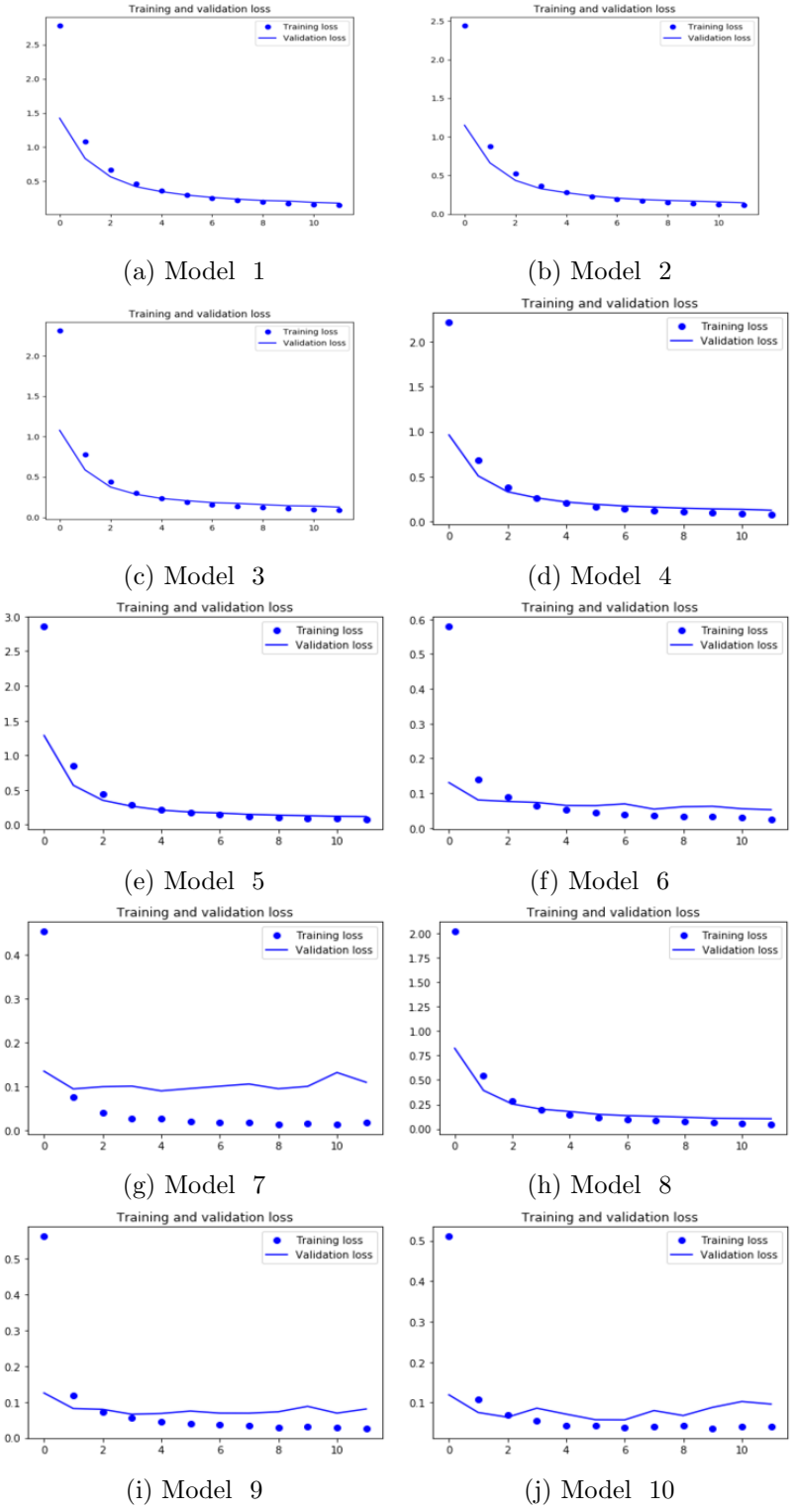
(h) Model 8



(i) Model 9

(j) Model 10

Figure 5.1: Training and validation accuracy of models.



(a) Model 1 (b) Model 2

(c) Model 3 (d) Model 4

(e) Model 5 (f) Model 6

(g) Model 7 (h) Model 8

(i) Model 9 (j) Model 10

Figure 5.2: Training and validation loss of models.

more than training loss. It is prevented by dropout in Fig.5.2(h).

5.4 Classification Report of Model 8

Classification report helps in identifying the misclassified classes. Table 5.2 shows the class for which the model underperformed of 46 classes.

Precision : It tells proportion of positive identifications was actually correct. Precision is defined as follows:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{5.2}$$

Recall: It tells what proportion of actual positives was identified correctly. A model that produces no false negatives has a recall of 1.0. Recall is defines as follows:

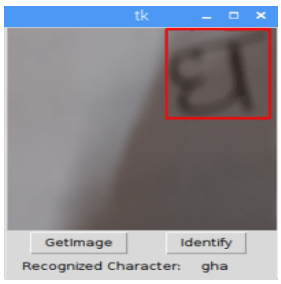
$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{5.3}$$

5.5 Classification of different characters

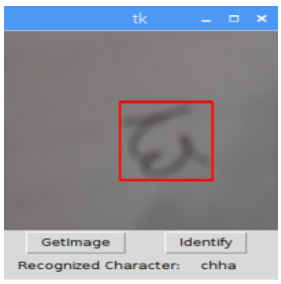
Model 8 was loaded in GUI and class of different characters was predicted as shown below Fig.5.5. Some of the characters were not correctly classified as shown in Fig.5.5

Table 5.2: Classification Report of Model 8

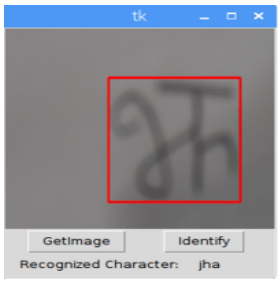
Class	**Precision**	**Recall**	**f1-score**
0	0.95	0.98	0.97
1	0.98	0.98	0.98
2	1.00	0.99	0.99
3	0.96	1.00	0.98
4	0.99	0.97	0.98
5	1.00	0.99	0.99
6	0.98	0.99	0.99
7	0.97	0.99	0.98
8	0.98	0.96	0.97
9	0.99	0.95	0.97
10	1.00	1.00	1.00
11	0.96	0.98	0.97
12	0.97	0.99	0.98
13	0.99	1.00	1.00
14	0.96	0.97	0.97
15	0.98	0.99	0.99
16	1.00	1.00	1.00
17	0.98	0.98	0.98
18	0.97	1.00	0.98
19	1.00	1.00	1.00
20	0.96	0.96	0.96
21	1.00	0.99	0.97
22	0.99	0.99	0.99
23	0.99	0.98	0.98
24	0.99	0.99	0.99
25	0.99	0.99	0.99
26	0.99	0.98	0.99
27	0.99	0.97	0.98
28	1.00	0.98	0.99
29	0.98	0.99	0.99
30	0.96	0.99	0.97
31	1.00	0.96	0.98
32	1.00	0.99	0.99
33	0.99	0.97	0.98
34	0.99	0.98	0.98
35	1.00	1.00	1.00
36	1.00	1.00	1.00
37	0.99	1.00	0.99
38	0.97	0.99	0.98
39	1.00	0.98	0.99
40	0.99	1.00	1.00
41	1.00	0.99	0.99
42	1.00	0.99	0.99
43	0.99	1.00	1.00
44	0.99	1.00	0.99
45	1.00	0.99	1.00



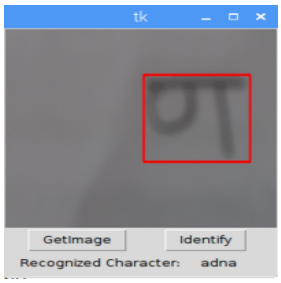
(a) *Character gha*



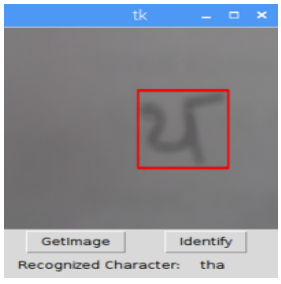
(b) *Character chha*



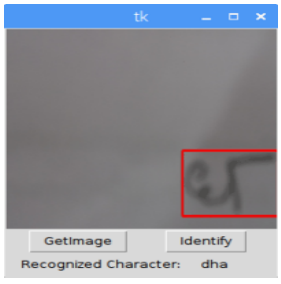
(c) *Character jha*



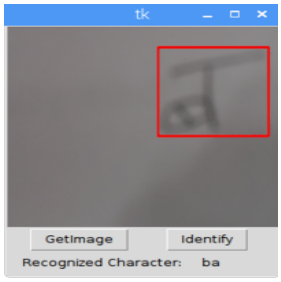
(d) *Character adna*



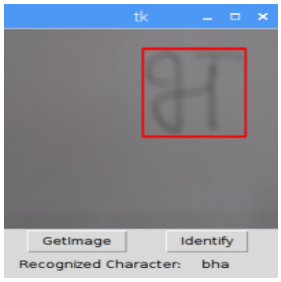
(e) *Character tha*



(f) *Character dha*

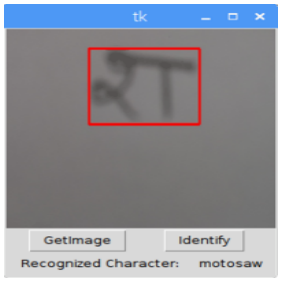


(g) *Character ba*



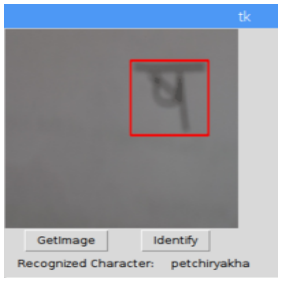
(h) *Character bha*

(i) *Character la*

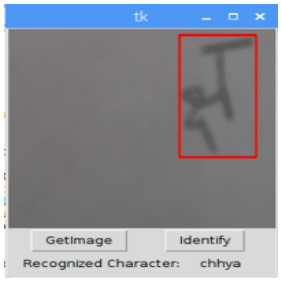


(j) *Character motosaw*

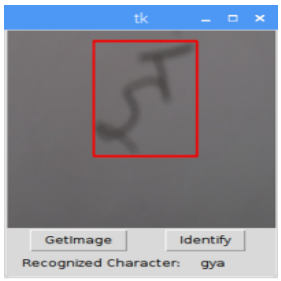
Figure 5.3: Correctly classified characters.



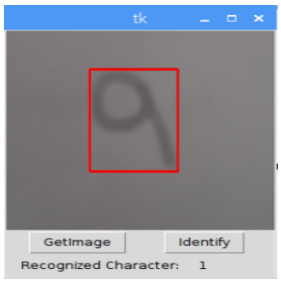
(a) *Character petchiryakha*



(b) *Character chhya*

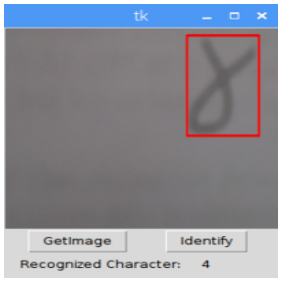


(c) *Character gya*

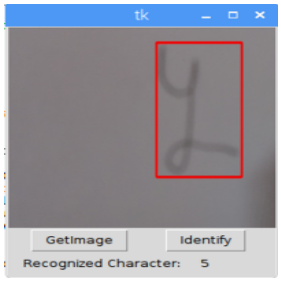


(d) *Numeral* 1

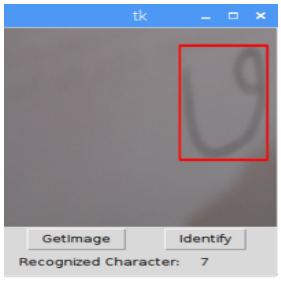
(e) *Numeral* 2



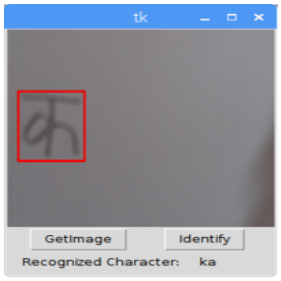
(f) *Numeral* 4



(g) *Numeral* 5



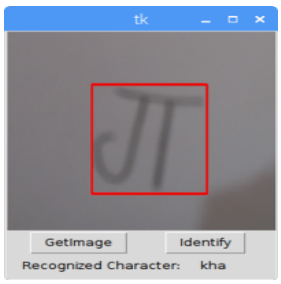
(h) *Numeral* 7



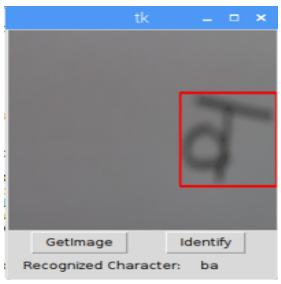
(i) *Character ka*

(j) *Character kha*

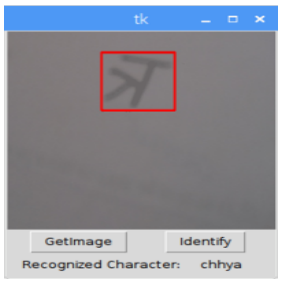
Figure 5.4: Correctly classified characters(cont.).



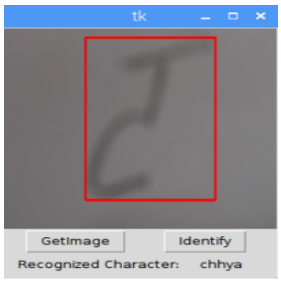
(a) *Character ga*



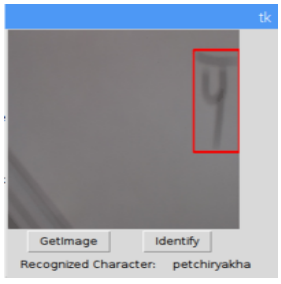
(b) *Character waw*



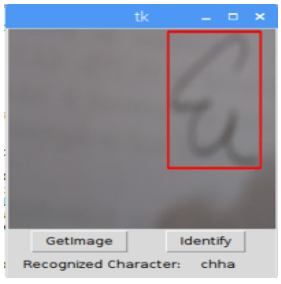
(c) *Character tra*



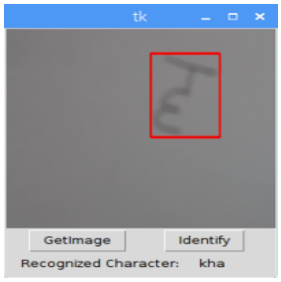
(d) *Character tamatar*



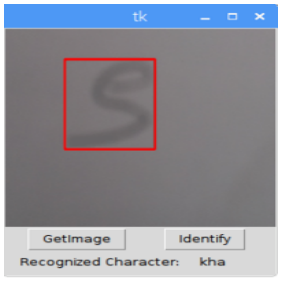
(e) *Character pa*



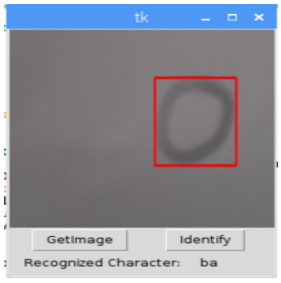
(f) *Numeral* 6



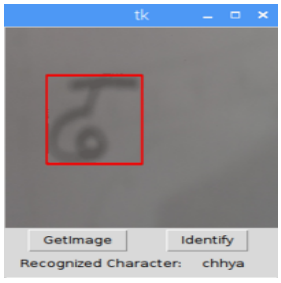
(g) *Character ha*



(h) *Numeral* 9



(i) *Numeral* 0



(j) *Character dhaa*

Figure 5.5: Incorrectly classified characters.

Chapter 6

Conclusion and Future Scope

After proposed framework done and result analaysed, conclusion and future scope are given.

6.1 Conclusion

The observations show that more width is not always good for accuracy as with the increase in width, accuracy increases but starts decreasing eventually. Also, with the number of units increasing, training time increases. Dropout has prevented overtraining of network as a result in increase in width of network. The highest accuracy achieved is 98.75%. GUI application processes the image captured and convert it into format required for the input to trained model and recognizes the character.

6.2 Future Scope

The proposed framework recognizes only base consonants and numeral characters. Hence, it can be extended to recog- nize character combined with vowels, words, sentences and then, complete documents. The network can be trained with comparatively large dataset so as to recognize character in real world application more effectively. Also, the framework can be implemented as a part of robotics for large Optical Character Recognition applications.

References

[1] R. Jayadevan, S. R. Kolhe, and P. M. Patil, U. Pal, *"Offline Recognition of Devanagari Script:A Survey."*, IEEE Transactions on Systems, Man and Cybernetics,vol.41, no.6, pp. 529-551, November 2011.

[2] V. Narang, S. Roy, O.V.R. Murthy, M. Hanmandlu, *"Devanagari Character Recognition in Scene Images."*, IEEE 12th International Conference on Document Analysis and Recognition, 2013.

[3] S. Shelke, S. Apte, *"Performance optimization and comparative analysis of neural networks for handwritten Devanagari character recognition."*, IEEE International Conference on Signal and Information Processing, 2016.

[4] V. J. Dongre, V. H. Mankar, *"Devanagari offline handwritten numeral and character recognition using multiple features and neural network classifier."*, 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015.

[5] K.V. Kale, S.V. Chavan, M.M. Kazi, Y.S. Rode, *"Handwritten Devanagari Compound Character Recognition using Legendre Moment an Artificial Neural Network Approach."*, IEEE International Symposium on Computational and Business Intelligence, 2013.

[6] K. Peymani, M. Soryani, *"From Machine Generated to Handwritten Character Recognition; A Deep Learning Approach."*, IEEE 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA), 2017.

[7] J. B.Z. Chen, B. Feng,B. Xu, *"Image character recognition using deep convolutional neural network learned from different languages."*, IEEE International Conference on Image Processing (ICIP), 2014.

[8] S. Acharya, A. K. Pant, P. K. Gyawali, *"Deep Learning Based Large Scale Handwritten Devanagari Character Recognition."*, 9thInternational Conference on Software, Knowledge, Information Management and Applications

(SKIMA), 2015.

[9] J. Tang, D. Sun, S. Liu, J. L. Gaudiot, *"Enabling Deep Learning on IoT Devices."*, IEEE Journals and Magazines, vol.50, no.10, pp. 92 96, October 2017.

[10] S. Park, M. Sohn, H. Jin, H. Lee, *"Situation Reasoning Framework for the Internet of Things Environment using Deep Learning Results."*, IEEE International Conference on Knowledge Engineering and Applications, 2016.

[11] S. Valipour, M. Siam, E. Stroulia, M. Jagersand, *"Parking-stall vacancy indicator system, based on deep convolutional neural networks."*, IEEE 3rd World Forum on Internet of Things (WF-IoT), 2016.

[12] Suchitra, P. Suja, S. Tripathi, *"Real-Time Emotion Recognition from Facial Images using Raspberry Pi II."*, 3rd International Conference on Signal Processing and Integrated Networks (SPIN), 2016.

[13] P. H. Kulkarni, P. D. Kute, V. N. More, *"IoT Based Data Processing for Automated Industrial Meter Reader using Raspberry Pi."*, International Conference on Internet of Things and Applications (IOTA), 2016.

[14] W. Rawat, Z. Wang, *"Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review."*, MIT Press Journal,s vol.29, no.9,pp. 2352-2449, June 2017.

[15] H. Wang, P. Chen, S. Kwong, *"Building Correlations Between Filters in Convolutional Neural Networks."*, IEEE Transactions on Cybernetics vol.47,no.10, pp. 3218 3229.

[16] K. He, J. Sun, *"Convolutional Neural Networks at Constrained Time Cost."*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[17] B. Ko, H. G. Kim, K. J. Oh, H. J. Choi, *"Controlled Dropout: a Different Dropout for Improving Training Speed on Deep Neural Network."*, IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017.

[18] P. Morerio, J. Cavazza, R. Volpi, R. Vidal and V. Murino, *"Curriculum*

Dropout.", IEEE Conference on Computer Vision, 2017.

[19] S. M. Maduta, C. D. Caleanu, *"Steps in Deployment and Development of Convolutional Neural Network based Applications."*, 12th IEEE International Symposium in Electronics and Telecommunications (ISETC) 2016.

[20] Q. Guo, J. Lei, D. Tu, G. Li, *"Reading Numbers in Natural Scene Images with Convolutional Neural Networks."*, IEEE International Conference on Security, Pattern Analysis, and Cybernetics (SPAC) (2014).

[21] M. D. McDonell, T. Vladusich, *"Enhanced image classification with a fast-lealning shallow convolutional neural network."*, IEEE International Conference on Neural Networks (2015).

[22] M. Y. Azar, M. D. McDonell, *"Semi-supervised Convolutional Extreme Learning Machine."*, International Joint Conference on Neural Networks (2017).

[23] T. S. Chang, *"End-to-end hardware accelerator for deep convolutional neural network."*, International Symposium on VLSI Design, Automation and Test (VLSI-DAT) (2018).

[24] C, Xin, Q. Chen, M. Tian, M. Ji, C. Zou, X. Wang, B. Wang, *"COSY: An Energy-Efficient Hardware Architecture for Deep Convolutional Neural Networks Based on Systolic Array."*, IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS) (2017).

[25] C. I. Orozovo, F. Iglesias, M. E. Beumi, J. J. Berlies, *"Real-time gender recognition from face images using deep convolutional neural network."*, 7th Latin American Conference on Networked and Electronic Media (LACNEM 2017).

Dataset Reference

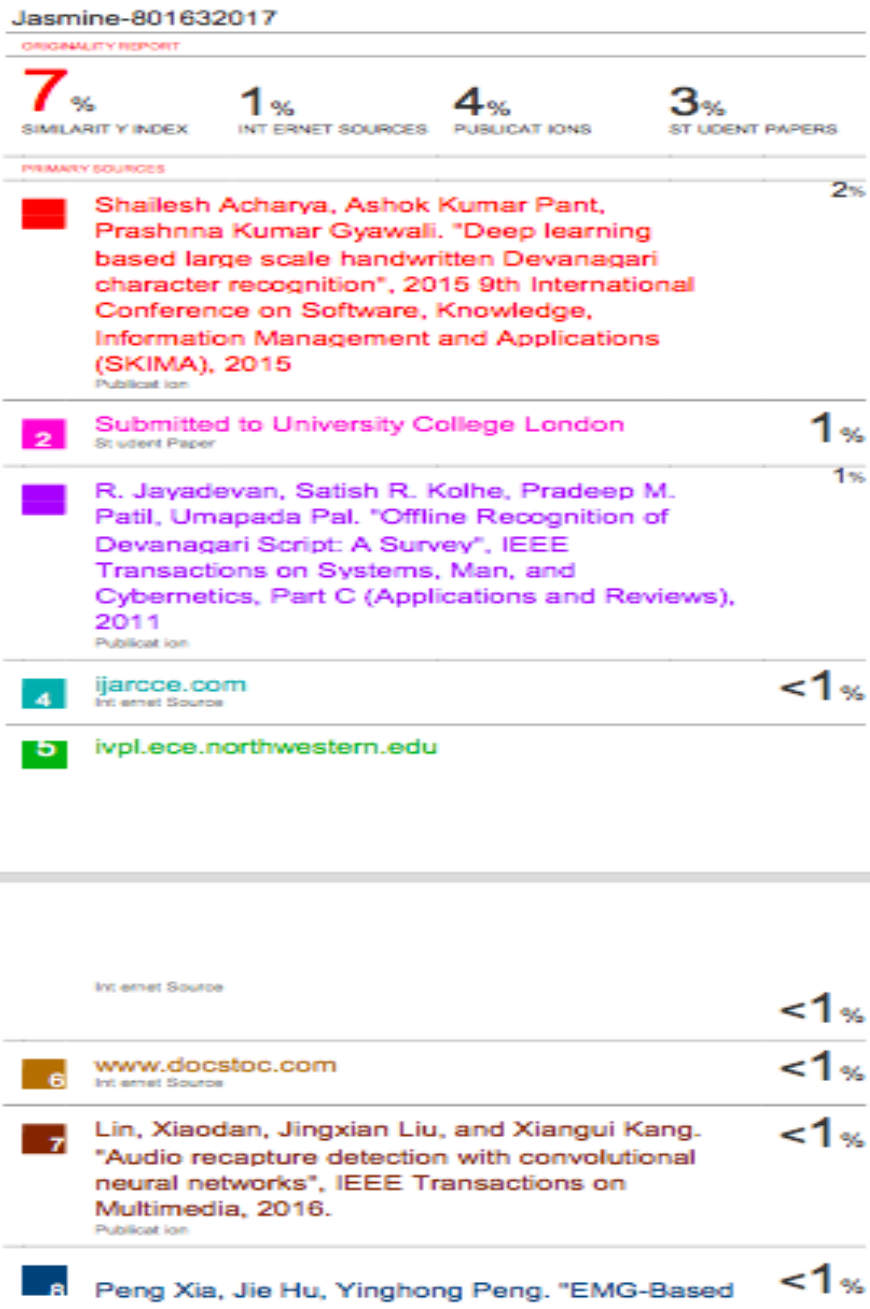
[26] https://archive.ics.uci.edu/ml/datasets/Devanagari+Handwritten+Character+Dataset

List of Publications

International Conference

[1] Jasmine and Seema Bawa, *"Deep Learning based Handwritten Devanagari Character Recognition using Raspiberry Pi 3"*, IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2018). [Communicated]

Plagiarism Report



Jasmine-801632017

ORIGINALITY REPORT

7%	1%	4%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Shailesh Acharya, Ashok Kumar Pant, Prashnna Kumar Gyawali. "Deep learning based large scale handwritten Devanagari character recognition", 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), 2015 Publication	2%
2	Submitted to University College London Student Paper	1%
3	R. Jayadevan, Satish R. Kolhe, Pradeep M. Patil, Umapada Pal. "Offline Recognition of Devanagari Script: A Survey", IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2011 Publication	1%
4	ijarcce.com Internet Source	<1%
5	ivpl.ece.northwestern.edu Internet Source	<1%
6	www.docstoc.com Internet Source	<1%
7	Lin, Xiaodan, Jingxian Liu, and Xiangui Kang. "Audio recapture detection with convolutional neural networks", IEEE Transactions on Multimedia, 2016. Publication	<1%
8	Peng Xia, Jie Hu, Yinghong Peng. "EMG-Based	<1%