TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

THESIS NO.: 070/MSI/615

HANDWRITTEN DEVANAGARI CHARACTER RECOGNITION USING HYBRID CONVOLUTIONAL NEURAL NETWORK

By

Sudarshan Sharma

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING AS A PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE MASTER'S DEGREE IN INFORMATION AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

JANUARY, 2020

HANDWRITTEN DEVANAGARI CHARACTER RECOGNITION USING HYBRID CONVOLUTIONAL NEURAL NETWORK

By

Sudarshan Sharma

Thesis Supervisor

Dr. Basanta Joshi

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Information and Communication Engineering

Department of Electronics and Computer Engineering

Institute of Engineering, Pulchowk Campus

Tribhuvan University

Lalitpur, Nepal

January, 2020

COPYRIGHT ©

The author has agreed that the library, Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus, may make this thesis freely available for inspection. Moreover, the author has agreed that the permission for extensive copying of this thesis work for scholarly purpose may be granted by the professor(s), who supervised the thesis work recorded herein or, in their absence, by the Head of the Department, wherein this thesis was done. It is understood that the recognition will be given to the author of this thesis and to the Department of Electronics and Computer Engineering, Pulchowk Campus in any use of the material of this thesis. Copying of publication or other use of this thesis for financial gain without approval of the Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus and author's written permission is prohibited.

Request for permission to copy or to make any use of the material in this thesis in whole or part should be addressed to:

Head
Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus
Pulchowk, Lalitpur, Nepal

RECOMMENDATION

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a thesis report entitled "**Handwritten Devanagari Character Recognition using Hybrid Convolutional Neural Network**" submitted by **Sudarshan Sharma** in partial fulfillment of the requirements for the degree of "**Master of Science in Information and Communication Engineering**".

Dr. Basanta Joshi
Supervisor
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering
Tribhuvan University

Er. Manoj Ghimire
External Examiner

DEPARTMENTAL ACCEPTANCE

The thesis entitled "**HANDWRITTEN DEVANAGARI CHARACTER RECOGNITION USING HYBRID CONVOLUTIONAL NEURAL NETWORK**", submitted by **Sandeep Sigdel** in partial fulfillment of the requirement for the award of the degree of "**Master of Science in Information and Communication Engineering**" has been accepted as a bonafide record of work independently carried out by him in the department.

Dr. Surendra Shrestha
Head of the Department
Department of Electronics and Computer Engineering,
Pulchowk Campus,
Institute of Engineering,
Tribhuvan University,
Nepal.

ACKNOWLEDGEMENT

I am grateful to my Thesis supervisor **Dr. Basanta Joshi** for his incessant cooperative support, guidance and suggestion at various stages of the Thesis. I deeply value his work as my mentor by providing me with constant counseling and precious feedbacks on Thesis progress.

I would like to express my special thanks of gratitude to the Department of Electronics and Computer Engineering (DOECE) and to our Head of Department **Dr. Surendra Shrestha** for providing us with the golden opportunity to explore our interest and ideas in the field of engineering through this thesis. I would like to provide my sincere gratitude to our MSCSKE Coordinator **Dr. Aman Shakya** for providing us with necessary details and ideas for preparing the mid-term report of Thesis. I would also like to thank **Prof. Dr. Shashidhar Ram Joshi, Prof. Dr. Subarna Shakya, Dr. Diwakar Raj Pant, Dr. Sanjeep Prasad Pandey** and many other staffs from this DOECE for their constant support and encouragement on research activity during master's program.

I am really thankful to my friends **Er. Sabin Devkota, Er. Manoj Acharya, Er. Everest KC** and **Er. Dharma KC,** who helped and encouraged me to this level of accomplishment from abroad and with the presence.

Finally, I would like to thank all our teachers and friends who have helped me directly or indirectly for encouraging me with this thesis topic and research decision.

ABSTRACT

Handwritten character recognition is very popular field of research work in modern technology as information are stored and used as written script in different languages since long ago. This research work mainly focuses to recognize handwritten Devanagari character through preprocessing the image data and deep learning techniques.

The handwritten character is collected and processed to fit into popularly used neural networks for classification of those images into their related label of character. Collected image data are clipped, normalized and processed and then converted into numpy dataset to feed into the neural network that we adopted and developed.

The data is trained through the managed sequence of convolutional and fully connected layers of network and proper activation and pooling is done in between to optimize and speed up the training process. Here we have used ReLU as activation function and maxpooling as pooling function. We have used two convolution layer and each Convolution Layer are followed by activation and pooling functions. These layers are then followed by three fully connected layer to produce better neural network.

12051 images of handwritten Devanagari character are fed into this neural network and the trained to produce a neural network and the accuracy of this produced neural network is tested upon the different set of 290 validation images test set.

The validation of this model is observed through the confusion matrix and seems to work good.

TABLE OF CONTENT

COPYRIGHT© III

RECOMMENDATION.....IV

DEPARTMENTAL ACCEPTANCE.....V

ACKNOWLEDGEMENT.....VI

ABSTRACT.....VII

LIST OF FIGURES.....VIII

LIST OF ABBREVIATIONS.....IX

1. INTRODUCTION.....1

1.1 Background.....1
1.2 Problem Definition.....2
1.3 Objective.....2
1.4 Scope.....3

2. LITERATURE REVIEW.....4
2.1 Neural Networks.....4
2.2 Convolutional Neural Networks.....6
2.3 Deep Learning.....8
2.4 Deep Convolutional Neural Networks.....10
2.5 Related works.....11

3. METHODOLOGY.....12
3.1 Data preparation.....12
3.2 Data preprocessing.....15
3.3 Classification using CNN.....15
3.3.1 Structure of Network and forward pass.....17
3.3.2 Loss Calculation.....20
3.3.3 Optimization of parameters.....20
3.4 Validation of the output.....21
3.5 System Block Diagram.....22
3.6 Tools Used.....23

4. EXPERIMENTS AND OUTPUTS.....24
5. EPILOGUE.....38
6. REFERENCES.....39

LIST OF FIGURES

Figure 2.1: Sample artificial neural network.....5
Figure 2.2 Neural network training mechanism.....6
Figure 2.3: Difference of convolutional network with MLP.....8
Figure 2.4: Deep neural network vs rule based or intuition-based approach.....9
Figure 2.5: Deep learning compared with other techniques.....10
Figure 2.6: Example of CNN architecture. 11
Figure 3.1: Previously assumed probable Convolutional Neural Network.....16
Figure 3.2: Detailed structure of the customized network.....19
Figure 3.3 System block diagram.....22
Figure 4.1:Sample of collected image data 24
Figure 4.2 Screenshot of one batch of data fed into network.....24
Figure 4.3 Screenshot of display of details of network 24
Figure 4.4: Screenshot of progress on last 10 Training epochs.....25
Figure 4.5 : Top-1 Accuracy vs epochs of Experiment_1.....27
Figure 4.6: Loss vs Epochs of Experiment_1.....27
Figure 4.7: Top-1 Accuracy vs epochs of Experiment_2.....28
Figure 4.8: Loss vs Epochs of Experiment_2.....28
Figure 4.9: Top-1 Accuracy vs epochs of Experiment_3.....29
Figure 4.10: Loss vs Epochs of Experiment_3.....29
Figure 4.11: Top-1 Accuracy vs epochs of Experiment_4.....30
Figure 4.12: Loss vs Epochs of Experiment_4.....30
Figure 4.13: Top-1 Accuracy vs epochs of Experiment_5.....31
Figure 4.14: Loss vs Epochs of Experiment_5.....31
Figure 4.15: Top-1 Accuracy vs epochs of Experiment_6.....32
Figure 4.16: Loss vs Epochs of Experiment_6.....32
Figure 4.17: Top-1 Accuracy vs epochs of Experiment_7.....33
Figure 4.18: Loss vs Epochs of Experiment_7.....33
Figure 4.19: Top-1 Accuracy vs epochs of Experiment_8.....34
Figure 4.20: Loss vs Epochs of Experiment_8.....34

Figure 4.21: Top-1 Accuracy vs epochs of Experiment_9.....35
Figure 4.22: Loss vs Epochs of Experiment_9.....35
Figure 4.21: Losses of different folds.....36
Figure 4.22: Top-1 accuracy of different folds over train data.....36
Figure 4.23: Top-5 accuracy of different folds over train data.....35
Figure 4.24: Top-1 accuracy of different folds over validation data.....36
Figure 4.26: Confusion Matrix of the classifier without class name.....37

LIST OF ABBREVATIONS

CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
DHCR	Devanagari Handwritten Character Recognition
GPU	Graphical Processing Unit
MLP	Multilayer Perceptron
SVM	Support Vector Machine

1. INTRODUCTION

1.1 Background

So many generations of our ancestor contributed to the development of scripts and languages, we are now in right state of use of languages and scripts to transfer, translate, transmit and store knowledge.

Development of modern technologies forced the medium of knowledge transmission to adopt the computer related platforms and technologies. Hence its is now mandatory to translate the languages and knowledge we had already into digitally readable data format. For this purpose we need to translate the scripts and literature written in different language into digital data. And this process requires system to correctly recognize the written data by the computer and this process is known as character recognition process.

As a rich source of knowledge and history devanagari script is widely used by so many languages majorly in South Asian region. In oldest known civilization of Hindu Philosophy the knowledge is transferred through Sanskrit language and this Sanskrit language also scripted in Devanagari script. Many languages such as Hindi, Nepali, Maithili etc. have their rich literature and scriptures in Devanagari scripts and near about 15% of world still use Devanagari script in their daily life officially or unofficially[1]. Character can be written by hand in own style or can be printed with the machine with the standard format.

Image classification has been researched very intensely as so many field of its application is very useful in daily life. Since most of the information used in daily life are closely associated with the visual information, image processing and classification is a pronounced field of study in modern technological development.

As mentioned above this field of study is applied for the classification of character recognition widely[2]. And in the recent years a huge part of image classification is accomplished by the deep learning algorithms.

These days deep learning has been very pronounced field of scientific vision and image processing. Being a sub field of machine learning, deep learning uses application of artificial intelligence mimicking the process of learning by human brain. In the field of

image processing and image recognition deep learning has given very satisfying results with outstanding performance over previous approaches.

Optical character recognition is a process of recognizing the character from its image. Many approaches have been adopted to optically recognize the character by computer. But these days deep learning outperformed almost all other approaches. Written scripts were very tough to read by computer before the techniques developed to optically read by computer. But nowadays written scripts are very efficiently recognized by computer using different techniques. But reading Devanagari scripts by computer is still a bit more challenging job in this field. Despite being the scripts used by very dense population in south Asian region as well as being scripts used in very rich Sanskrit literature Devanagari character recognition is very important task to be done more efficiently.

1.2 Problem definition

With the increased usage of computer vision, efficiency and performance in wider range of scripts becomes more important. So developing better approaches of recognizing Devanagari characters is really important task. It will make computer vision more versatile as well as lead to a wide area of exploring many of the ancient literature written in Sanskrit literature. Previously many efforts have been adopted in this field as well. Furthermore it is more important to develop better algorithm to recognize handwritten characters to make it more versatile. Very huge mass of people of this century depend upon Devanagari scripts in their daily life practice and also in commercial field, because Nepali, Hindi, Sanskrit and many more languages use same Devanagari scripts in their written form. Therefore to address such important field of application using newer technology it is important to develop a better approach to recognize handwritten Devanagari character using recent artificial intelligence to support machine learning.

1.3 Objective

The objective of the thesis is to develop the customized Hybrid algorithm to recognize handwritten Devanagari characters using Convolutional Neural Network.

1.4 Scope

The scope of this research work is to develop a hybrid algorithm to recognize handwritten Devanagari character using CNN. In addition to this, the performance in terms of accuracy and computational complexity of different parameters will be compared.

2. LITERATURE REVIEW

2.1 Neural Networks

Being inspired from biological neurons a neural network is a network of artificial neuron that learn from input examples. Different layer of neural networks are interconnected to each other to learn in a coherent manner. A neural network may consist from one to several layer as per the complexity of the learning process and every layer can consist many neurons. Every link and neurons have their own weight as well as biases to calculate the weight of the effect from input to the output. The output from the neural network for the given set of input data is matched with the expected output and the difference between them gives the error of the network. This error value is used to update the weights and biases of the network. Hence this error value is used to optimize the network in repeated manner with this process. Neural networks optimizes errors and updates the weights from the set of input data and expected output one by one, unlike batch methods, such as SVMs.

Neural network have more significant progress in the field of computer vision, natural language processing and speech processing during late 2000s. But the history of neural network goes back to 1940's, when McCulloch and Pitts in 1943 proposed a computational model for biological neural networks called threshold logic based on mathematics and algorithms. Minski and Papert pointed out two major flaws of slow speed evolution of neural net at that time in 1969. Firstly the simple perceptron model was incapable of learning exclusive-OR function, secondly the lack of high processing power calculation to train neural networks at that time.

Neural networks again gained popularity when the backpropagation algorithm solved the problem of exclusive-OR (Werbos, 1974). But although the problem of exclusive-or was solved, due to the continued lack of availability of machines with enough processing power, neural networks did not make much progress. Despite the rediscovery of backpropragation for multilayer perceptrons by McClelland, Rumelhart, and Hinton (1986), and a host of new applications, SVMs and simple linear classifiers seemed to rule the field by the mid to late 90s. Similarly, lack of a high amount of data needed to train neural network delayed progress.

With the evolution of powerful graphics processing units (GPUs) capable of general purpose computing, parallel processing in these GPUs has recently become the standard method for training neural networks. As GPUs have an an architecture built for parallel processing, training with GPUs leads to shorter training times for neural network algorithms. So advancement in powerful parallel processing GPU units along with availability of more training data for these neural networks to learn, has enabled advancements in the fields of machine vision, natural language processing, and speech recognition.

A simple artificial neural network is shown in Figure 2.1.

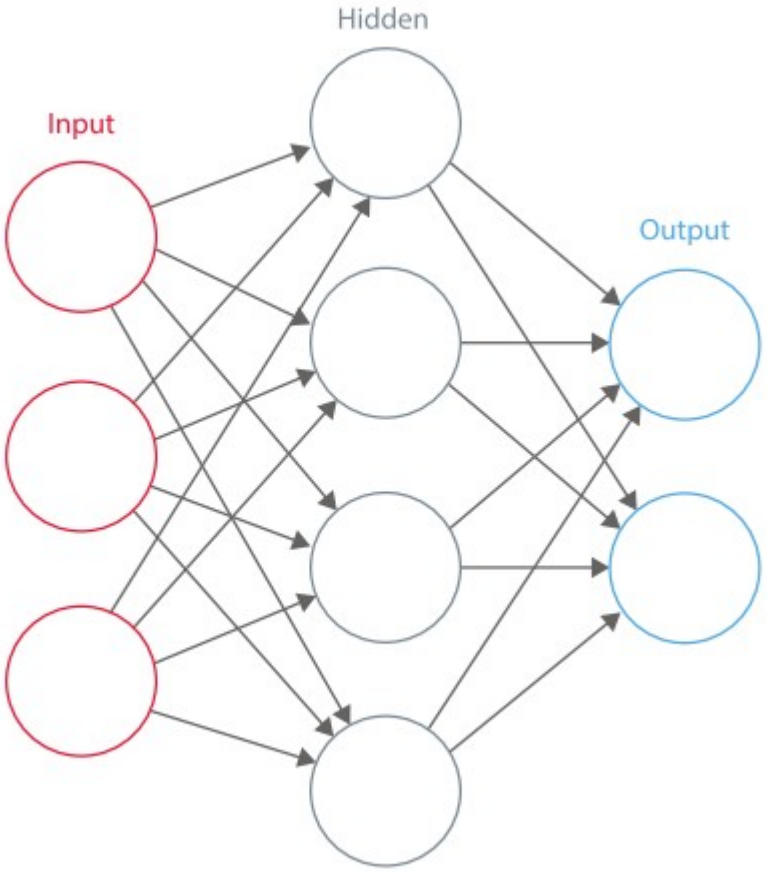


Figure 2.1: Sample artificial neural network. Reprinted from Samer, Rishi, and Rowen (2015)

Training a neural network requires a paired set of input data (X_i) and input labels (Y_i) as shown in figure 5.2. Weights (W) of neural networks are randomly initialized and trained with the set of input output pairs. This process repeatedly improves the weights and biases

of the neural networks by minimizing the error with the comparison of the predicted and expected output labels.

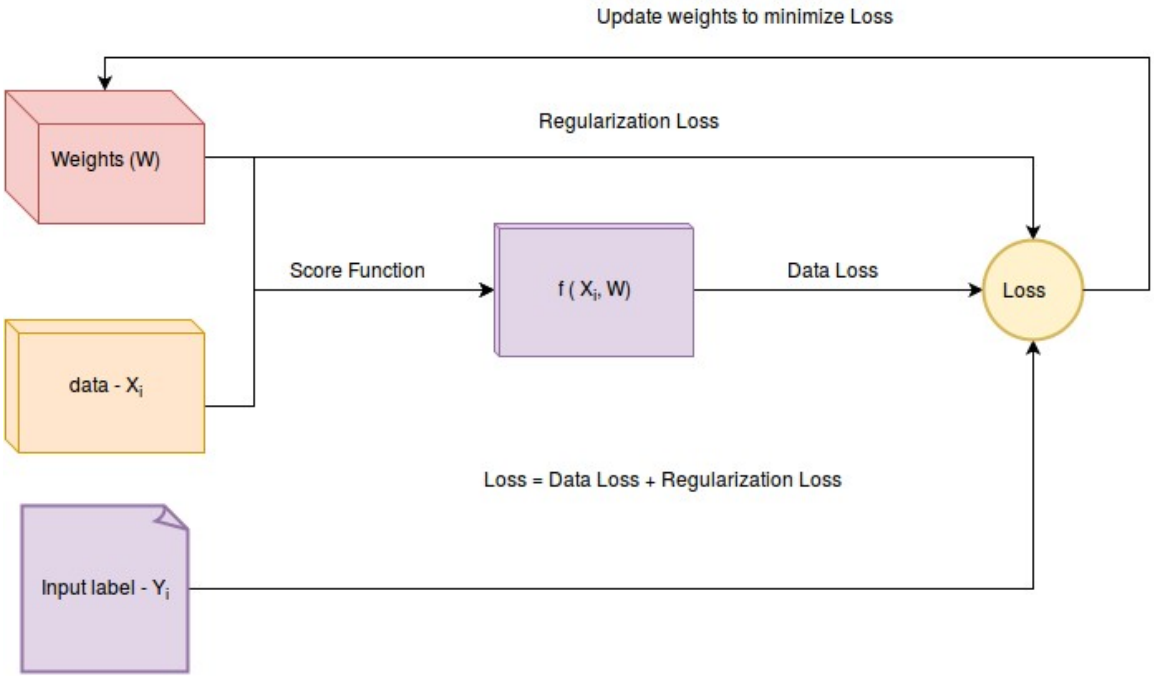


Figure 2.2 Neural network training mechanism.

2.2 Convolutional Neural Networks

The evolution of neural network research led to the development of special algorithms to implement perceptrons. The perceptron learning rule was a supervised learning algorithm for linear classification. As single-layer perceptrons did not show good results for multiclass nonlinear classification, a new method was developed called the multilayer perceptron (MLP). The evolution of the MLP and the backpropagation algorithm used to train it led to the first “convolutional” neural networks back in the 1990s. The structures and advances made in the 1990s can be summarized from the works of a pioneer in the field of machine learning, Yann LeCun.

Becker and Le Cun (1988) shows improvements in the slow convergence rule of backpropagation by simple approximation of second derivative terms with a slight increase in complexity. One of the hard parts of neural network is choosing efficient hyperparameters such as the step size (learning rate). LeCun, Simard, and Pearlmutter (1993) proposed a method for automatically calculating and adjusting learning rates of multilayer neural networks.

Although multilayer perceptrons were being used for image recognition, due to the full connectivity of neurons, i.e., all neurons of one layer being connected to all the neurons of

the previous layer, they suffer from the problem of increased dimensionality and correspondingly increases in computational requirements. For example, if we take an input image of 256 × 256 × 3 (256 pixels wide, 256 pixels height, 3 color channels), a fully connected neuron in the first hidden layer will have 256 × 256 × 3 = 196, 608 weights. The fully connected architecture doesn't take into account of the spatial structure of data; it treats input pixels that are far and close to each othre on the same basis. Taking spatial structure into account for image recognition, full connection seems to be wasteful at best, and at worst, the large number of parameters leads to a problem called overfitting.

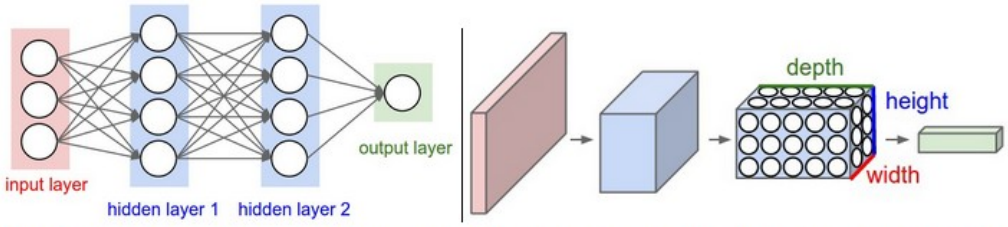
A new neural network architecture was designed for image recognition that takes the spatial structure of data into an account. It was called convolutional neural network. Convolutional neural networks, inspired by the structure of visual cortex in the brain, has following properties.

• 3D volume of neurons: A CNN contains neurons arranged in three dimensions: width, height, and depth. A neuron in one layer is connected to only some of the neurons in the previous layer, i.e., based on the fact that pixels in this layer are more affected by pixels from the previous layer. A CNN architecture consists of a stack of locally connected neurons and fully connected neurons.

• Local connectivity: CNN takes into account of the spatial structure of the data by local connection between neurons of adjacent layers. This will eventually decrease the number of parameters and the cost of computation and increases the accuracy. Stacking many such layers makes the filter global, i.e., is responsive to a larger region of pixels.

• Shared weights: CNN filters are replicated across the entire visual field, so that they consist of shared weights, which will lead to a lower total number of independent parameters. Repeating filters give the ability to detect certain features in an image irrespective of their location, giving the ability of translation invariance.

Figure 5.3 shows these representations schematically.



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Figure 2.3: Difference of convolutional network with MLP. Reprinted from Standford course on CNN.

LeCun et al. (1989) shows the improvement in neural networks for image recognition with local connection and shared weights. The authors compare their results with the performance of single-layer, fully-connected networks, two layer network (12 hidden units fully connected), locally connected networks (2 hidden layers locally connected), constrained network 1 (two hidden layers with constraints) and constrained network 2 (2 hidden layers with local connection and constraints) for small digit recognition problem.

2.3 Deep Learning

According to Deng and Yu [3], deep learning can be defined as:

A sub-field of machine learning that is based on learning several levels of representations, corresponding to a hierarchy of features or factors or concepts, where higher-level concepts are defined from lower-level ones, and the same lower- level concepts can help to define many higher-level concepts. Deep learning is part of a broader family of machine learning methods based on learning representations. An observation (e.g., an image) can be represented in many ways (e.g., a vector of pixels), but some representations make it easier to learn tasks of interest (e.g., is this the image of a human face?) from examples, and research in this area attempts to define what makes better representations and how to learn them.

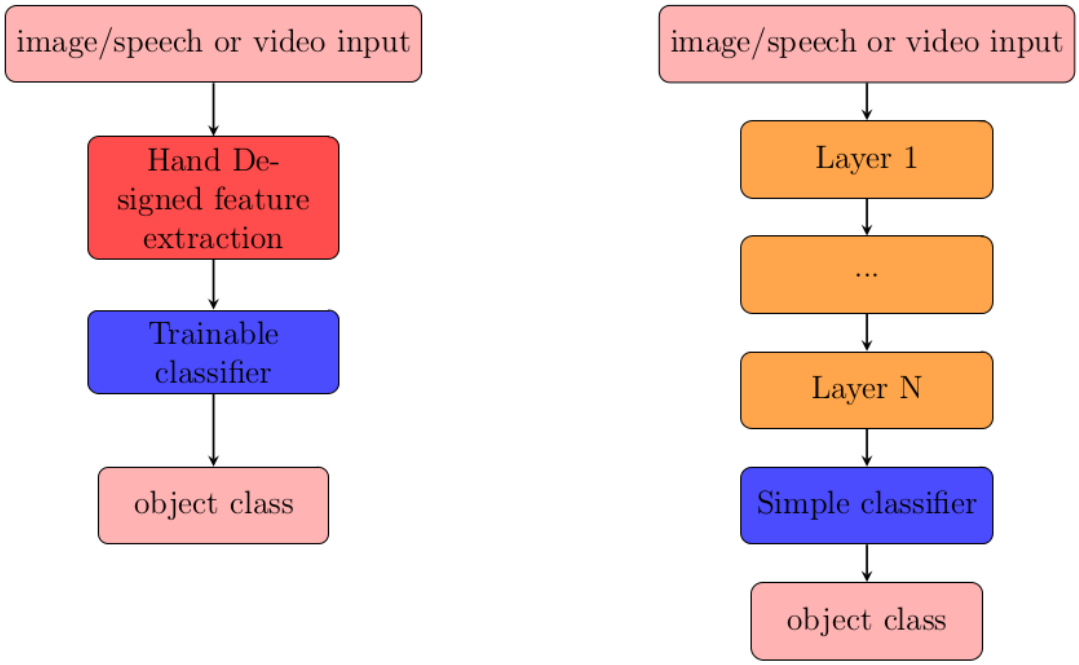


Figure 2.4: Deep neural network vs rule based or intuition-based approach.

Figure 2.4 describes how deep learning differs from traditional recognition strategies. In the traditional recognition approach, we extract hand-designed features from the input image/speech or video pixels and we use trainable classifiers such as SVMs for the classification task. But in the deep learning approach, we pass the input image/speech or video to input layer, and this input goes through a series of hidden layers (so called deep) to give a certain label.

Then we calculate the derivative of difference between this output label and actual label with respect to input (known as gradient) and back propagate it to the previous layers from the output and use it to adjust the weights of the neural network. So our network learns the parameters and uses them for feature extraction, unlike traditional systems, in which we provide a set of parameters for feature extraction. This is how the deep learning approaches are able to learn more complex features than traditional rule-based systems. Figure 5.5. describes the difference of Deep learning methods with other methods.

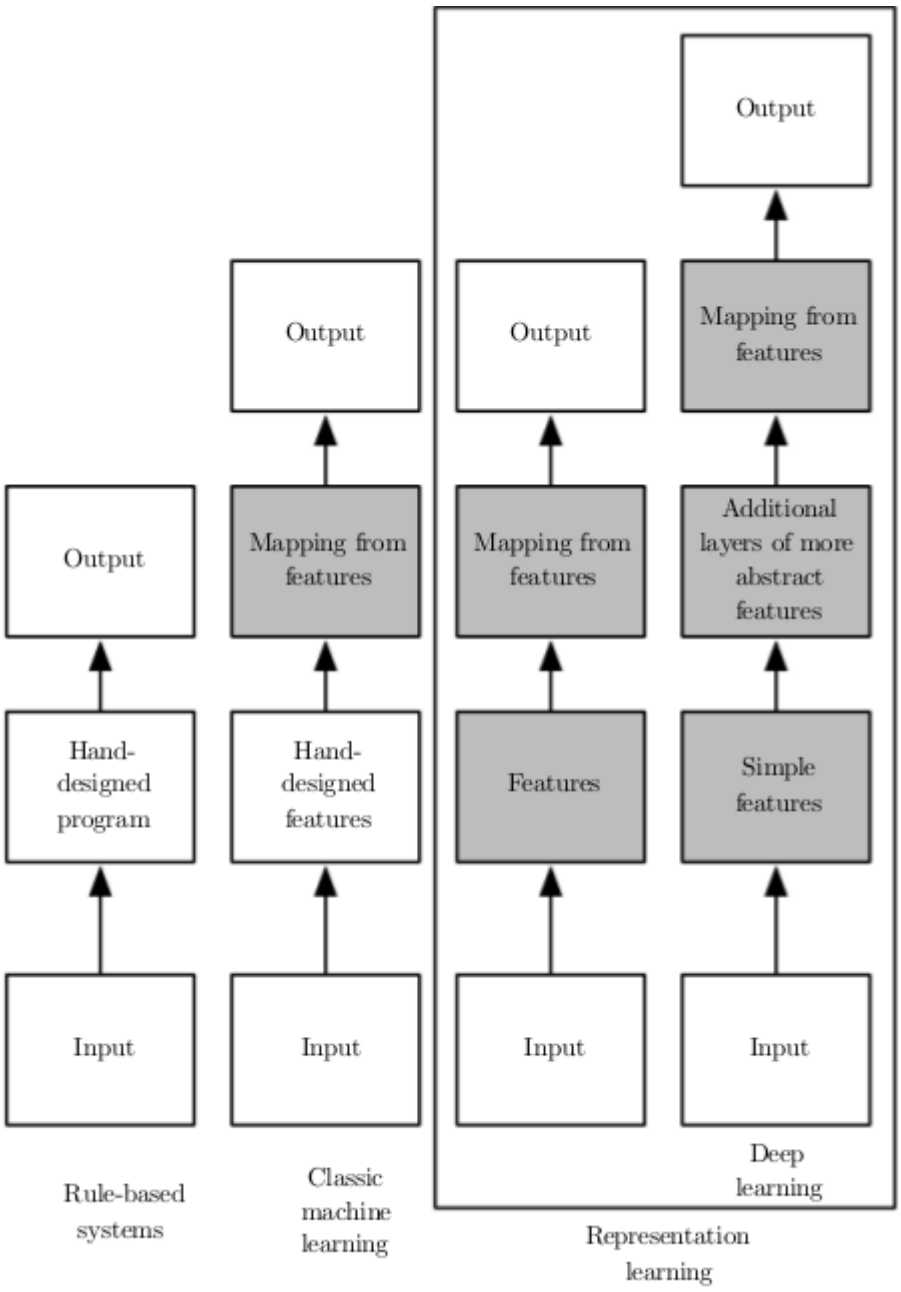


Figure 2.5: Deep learning compared with other techniques. Reprinted from Ian Goodfellow and Courville (2016).

2.4 Deep Convolutional Neural Networks

Deep convolutional neural networks (DCNNs) represent a deep learning mechanism inspired by the working of neurons in our visual cortex. A DCNN consists of multiple layers, with each layer consisting of many neurons. The lower layers, i.e., the layers near the input layer learn basic fundamental features such as edges, and the higher layers learn higher layer features such as structure and relative position of these structures as is

apparently done in our visual cortex, which is also a layered architecture in which the simple cells first learns basic features and complex cells learn object-specific features. Here the CNN learns the weights by itself, so we do not need to provide features for our network. This is the main advantage of DCNNs, which learn object-specific features during training. The drawback of DCNNs compared to existing methods is that they require powerful processing power, but this problem is has been solved to some extent using the parallel processing of powerful GPUs. A convolutional neural network consists of following layers.

• Convolutional layer
• Pooling Layer
• Non-linear Layer
• Fully Connected Layer
• Loss Layer

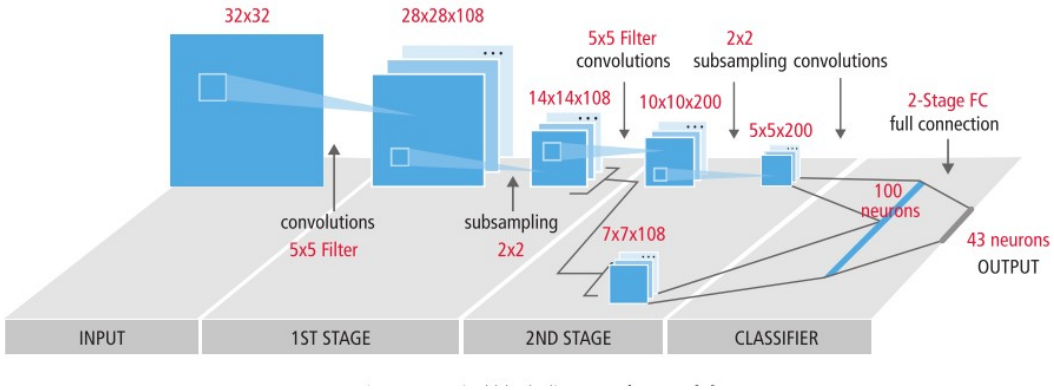


Figure 2.6: Example of CNN architecture. Reprinted from Samer et al. (2015).

2.5 Related Work

So many work have been done in the field of handwritten Nepali character recognition. Recently Acharya, Shailesh & Pant, Ashok & Gyawali, Prashnna proposed a model for recognizing handwritten character of Devanagari script [4][5]. They have purposed DCNN model for this purpose and limited to the character set of language only.

But previous approach of similar work on any other language or script still supports the way towards Devanagari scripts. Hence this field has been widely explored field but still lacks some better approach in Devanagari scripts.

3. METHODOLOGY

An approach to recognizing handwritten character using CNN will require huge set of database and higher computational power as well. This section explains applied methodology for Devanagari Handwritten Character recognition.

3.1 Data set preparation

Although Devanagari script is widely used by varieties of languages such as Nepali, Hindi etc., I will be focused in the data set used in Nepali language. We have prepared a moderately large handwritten data set for 58 Devanagari isolated characters, which contains 36 consonants, 12 vowels and 10 numbers. For handwritten character scripts, we will consider the handwriting of various individuals from different ages and education levels. These scanned images are then This data set contains wide variation of distinct characters because of different peoples' writing styles. Some of these character images may be very complex shaped and may also closely correlated with others. List of characters, numbers and their corresponding English phoneme and sample handwritten images are shown in table 6.1.

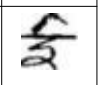
Preprocessing cleans the arbitrary images into common shape or form that makes appropriate to feed into classifiers. At first handwritten characters will be scanned and produces gray scale image files. In a grayscale image, each pixel value is a single integer number (from 0 to 255) that represents the brightness of the pixel. Typically white pixel has value 255 whereas black pixel has value 0. The image files even for a character are often found different sizes for different persons. The arbitrary images will be resized into dimension to maintain appropriate and equal inputs for all the characters. Since we consider black color for writing on white paper (background), the grayscale image files contain more white pixels than black for writing.

Scanned images are filtered and clipped to 28*28 pixel gray-scale .jpg images. Our prepared data set size is differentiated into two different set of training set and test set. Train set consists of array of 12051 images of 58 handwritten characters and test set contains array of 290 images of same 58 characters having nearly equal samples for each character. Each train, and test folder is made as a collection of folders with 58 classes of images and then are named with the integer from 1-58.

SN	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Devanagari character	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ
English	Ka	Kha	Ga	Gha	Nga	Cha	Chha	Ja	Jha	Nya
Handwritten Images										

SN	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Devanagari character	ट	ठ	ड	ढ	ण	त	थ	द	ध	न
English	TTa	TTha	Dda	Ddha	Nna	Ta	Tha	Da	Dha	na
Handwritten Images										

SN	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Devanagari character	प	फ	ब	भ	म	य	र	ल	व	श
English	Pa	Pha	Ba	Bha	Ma	Ya	Ra	La	Va/ Wa	Sha
Handwritten Images										

SN	31	32	33	34	35	36	37	38	39	40
Devanagari character	ष	स	ह	क्ष	त्र	ज्ञ	अ	आ	इ	ई
English	Sha	Sa	Ha	Ksha	Tra	Jnya	A	Aa	I	Ee/ii
Handwritten Images										

SN	41	42	43	44	45	46	47	48	49	50
Devanagari character	उ	ऊ	ए	ऐ	ओ	औ	अं	अः	०	१
English	U	Uu/ Oo	E	Ai	O	Au	Am	Aha	0	1
Handwritten Images										

SN	51	52	53	54	55	56	57	58
Devanagari character	२	३	४	५	६	७	८	९
English	2	3	4	5	6	7	8	9
Handwritten Images								

Table 3.1 List of Devanagari characters and numbers, corresponding English phoneme and sample handwritten images.

3.2 Data preprocessing

These set of images are not so easy to handle with the network. So we have to make them appropriate to fed into the network. Hence we prepared a numpy array of each train and test dataset each consisting the array of image data provided with the corresponding image label. Images are loaded using Image() function from PIL library and converted into numpy array and the stored as a higher dimensional array in the big data set. These images are provided with the corresponding labels sidewise. Furthermore the information content of the picture are more variant in the black pixel, which are of lower value of the pixel. Hence after realization of the situation as the net will do its better work when higher values carry more distinctive information among the images, we have inverted the black and white color. Now onward higher values or white pixels are presented as the handwriting in the black backgound.

This arrays are then zipped into npz file to feed into the net work. This npz file also provided with the array of labels of the image which consists of the corresponding labels of the image in train and test image data with matched indexes.This image data are loaded to the iterator using numpy array. So all this image data from different folders are loaded through PIL library and converted to numpy array to fit for feeding into neural net.

3.3 Clsasification using CNN

Handwritten character classification is a high- dimensional complex task and traditional MLP require much computation to work with grayscale image. Therefore, a number of traditional methods first extract features from the input image and then use MLP based methods for classification task. On the other hand, CNN itself extract features from the input image or speech signals through its convolution operation [6]. Moreover, CNN has ability to perform right operation on invariance to scaling, rotation and other distortions. Therefore, CNN is considered for classification for Devanagari handwritten character in this study as it is found to perform well for English. CNN automatically obtains the relevant features like invariance to translation, rotation by forcing the replication of weight configurations of one layer to a local receptive field in the previous layer. Thus, a feature map is obtained in the next layer. By reducing the spatial resolution of the feature map, a certain degree of shift and distortion invariance is also achieved. Also, the number of free parameters is significantly decreased by using the same set of weights for all features in the feature map. Figure 3.1 shows probable reference of CNN structure of this study for classification Devanagari handwritten characters that holds two

convolutional layers with 5×5 receptive fields (i.e., kernel) and two subsampling layers with 2×2 averaging area with input and output layers. Input layer contains 784 nodes for 28×28 pixels image. 1 st convolutional operation produces first level six feature maps. Distinct kernel having different weights and biases from other kernels are used to produce a 1 st level feature map so that it can extract different types of local features. Where the output nodes will be updated for 58 number of nodes.

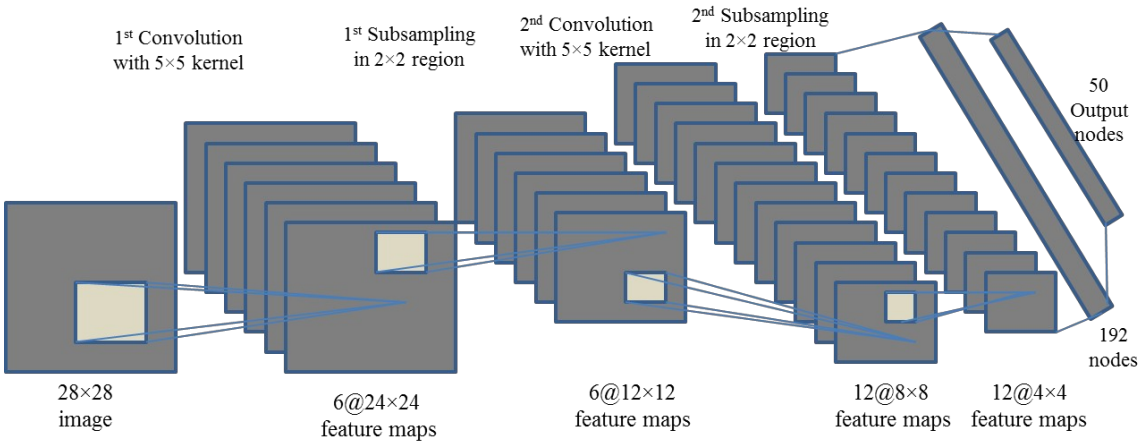


Figure 3.1: Previously assumed probable Convolutional Neural Network

Convolution operation with kernel spatial dimension 5 converts 28 spatial dimension to 24 (i.e., 28-5+1) spatial dimension [7]. Therefore, each 1 st level feature map size is 24×24. This local receptive field can extract the visual features such as oriented edges, end-points, corners of the images. Higher order features are obtained by combining those extracted features.

In first subsampling layer, the first level feature maps will be down-sampled from 24×24 into 12×12 feature maps by applying a local averaging with 2×2 area, multiplying by a coefficient, adding a bias and passing through an activation function.

Second convolution and second subsampling operations are similar to first convolution and first subsampling operations, respectively. second convolutional operation produces 12 distinct feature maps; a receptive field size of 5×5 produces a feature map size of 12×12 into 8×8. Then second subsampling operation resizes each feature map to size of 4×4. These 16 features map values are considered as 192 (=12*4* 4) distinct nodes those will be fully connected to 58 feature maps (the output nodes) for character set. For total 58 character set, each output node represents a particular character and the desired value of the node was defined as 1 (and other 57 desired output nodes value as 0) for the input set of the pattern. These image data is fed into the developed network and trained

continuously in loop to optimize a better values of weights and biases to produce a best weighted network.

3.3.1 Structure of network and forward pass

As we mentioned already we are using a CNN to classify these handwritten Devanagari character images, we have to make it with the composition of various components of CNN. Some building blocks of our network are briefly discussed below.

A) First convolutional layer (net.conv1)

First convolutional layer is provides with the input of one channel image array of 28*28= 784 pixels. This layer is constructed with the help of predefined module *torch.nn.Conv2d* defined by pytorch library which has following function prototype

`torch.nn.Conv2d`(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *padding_mode='zeros'*)

This layer is provided with the input of single channel image (in_channels=1) to produce 6 output feature maps (out_channels=6) which are obtained through 5*5 kernal map (kernel_size=5) and moving one stride in each move and without adding any pixels to its end (ie. setting remaining arguments default).

As per the document provided with the pytorch library this function in general applies a 2D convolution over an input signal composed of several input planes[8].

In the simplest case, the output value of the layer with input size (N,C_{in},H,W) and output ($N_i,C_{out},H_{out},W_{out}$) can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

B) Second convolution layer (net.conv2)

Second convolutional layer is also similar to the first convolutional layer but with different input and output attributes. The second convolutional layer is applied only after the data goes through first convolutional layer, activation function and pooling layer. Hence the this layer is provided to produce 12 output feature maps from those 6 channels obtained from first layer and with the help of same sized kernel as before.

C) Fully connected layer

After certain level of convolutional operation on those images some of fully connected linear layer are added to this network for better result. Here we have used three fully connected layers which are linear layer and applied after the application of two of above Convolutional layer. These three layer transform the input to out put 12*4*4 to 192, 192 to 96 and from 96 to our final required number of classes, ie. 58.

C) Activation functions

The activation function we used here is a very popular function known as Rectified Linear Unit function which nowadays outperforms in many of the neural networks and this is a functional application that holds the activation as high as the value is positively high and remains neutral as the value remains negative. This is also applied with the functional library of pytorch, *torch.nn.functional* .

D) Pooling Layer

Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Here in this research we have used max pooling techniques with the window of size 2*2 and stride movement of two pixels. This process of pooling reduces the spatial dimensions without decreasing the depth of the image. This is also defined through pytorch library as two dimensional 2*2 pooling window which samples the maximum pixel value included by the window. This action is carried out by *nn.MaxPool2d(2,2) function.*

E) Dropout layer

During training process of the network the waits and biases are adjusted based upon the error generated by the network. Hence they are adjusted in every run of the epoch. But while doing so there exits a probability of network overfitting due to similar pattern of adjustment in each run. So we need to make some of the attributes to become silent during the adjustment phase. This goal is achieved through the process of dropout layer. Here we have used 20% of the weights to dropout while optimizing those values. Which is applied through the *nn.Dropout(0.2) layer.*

After defining those layers mentioned above we have merged them to form a standard pattern of network. The sequence of network is mentioned below figure.

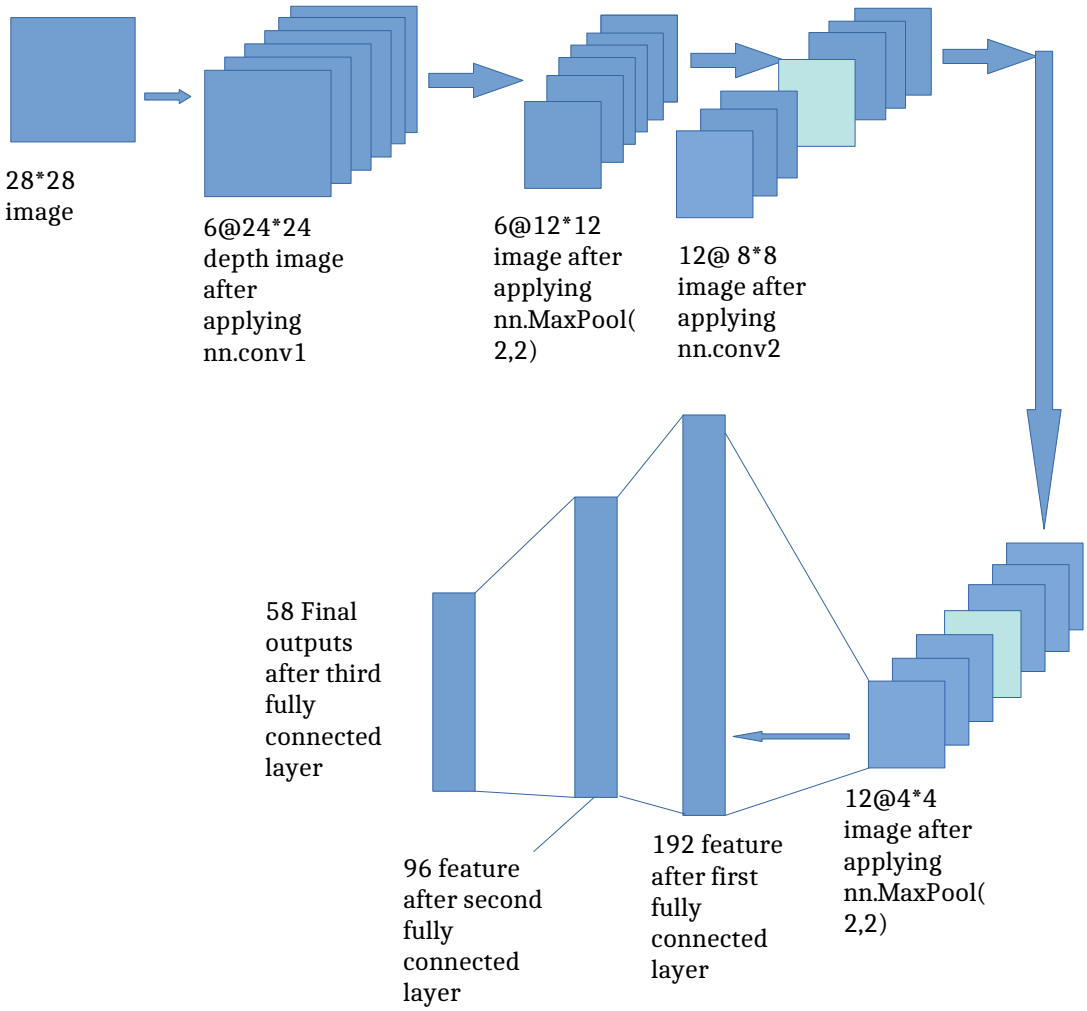


Figure 3.2: Detailed structure of the customized network

As shown in figure the 28*28 image is applied with the first convolutional layer to produce 6 feature maps each of containing 24*24 features. Then this features are applied with the ReLU functional operation and then they are applied with the pooling layer which we selected to be a max pooling with 2*2 windows and with 2 stride move. Hence this layer downsizes the spatial features to half of its value. So the output from this layer becomes 6 feature maps of 12*12 features. Then again second convolutional layer is applied and then similarly applied with the max pooling to produce 12, 4*4 feature maps. This output is then sent through three fully connected layers to produce 192, 96 and 58

output respectively, the final output number is matched to produce probability of the resemblance with the 58 character classes.

Hence in each forward pass, array of image produces different probability levels of the 58 output classes, indicating the probability of the matching of input image to the output classes with the value provided by the forward pass of network.

The overall definition of the network is as given below

```
Net(
  (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv2): Conv2d(6, 12, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=192, out_features=192, bias=True)
  (fc2): Linear(in_features=192, out_features=96, bias=True)
  (fc3): Linear(in_features=96, out_features=58, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

3.3.2 Loss calculation

The output of the network is not exactly resembles with the labels of the classes provided in the input. Hence this output from the network is compared with the input labels. Which produces certain error in every pass. There are so many functions to calculate these errors. Here we applied *nn.CrossEntropyLoss()* function to calculate this loss. This process uses negative log loss of the output compared with the input, after each forward pass.

3.3.3 Optimization of parameters

After calculating the loss we update the parameters of the network by calculating the contribution of each parameter to the error obtained in forward pass. This process involves very cleaver approach of calculating gradient of the parameters over the loss and optimizes the parameters as per the direction indicated by the gradient. So many optimizer are being researched to speed up the right convergence of the network since long ago. Here in our research we have used a very popular approach among them namely Adam Optimizer. This optimizer is loaded from *torch.optim.Adam()* and applied through all of the network parameters at the given learning rate initially as *lr=0.001* and updated in every 30 epochs.

In every run of the training batch the accuracy and the loss are calculated and recorded and displayed. Due to lack of the resource available to me this process is done on Google colab and the recorded outputs are stored to analyze the direction of progress. After completing the training and optimization up to 50 epochs this net is applied to the test dataset to validate the accuracy. The accuracy of the network are differently calculate as

top_k accuracy model to calculate the accuracy of network as top-1 and top-5 accuracy. Which defines the confidence of the network to classify the image with topmost probability and the probability of being the right class within the top 5 highest probabilities respectively.

Further more we have changed the network by changing different hyper parameters of the network in different Experiments. We particularly varied learning rate, drop out and Batch size.

Firstly we have done different experiments with variation of first two hyper parameter ie . Learning rate and drop out with the constant batch size of 64. The table containing different experiments by varying different hyper parameter is shown below.

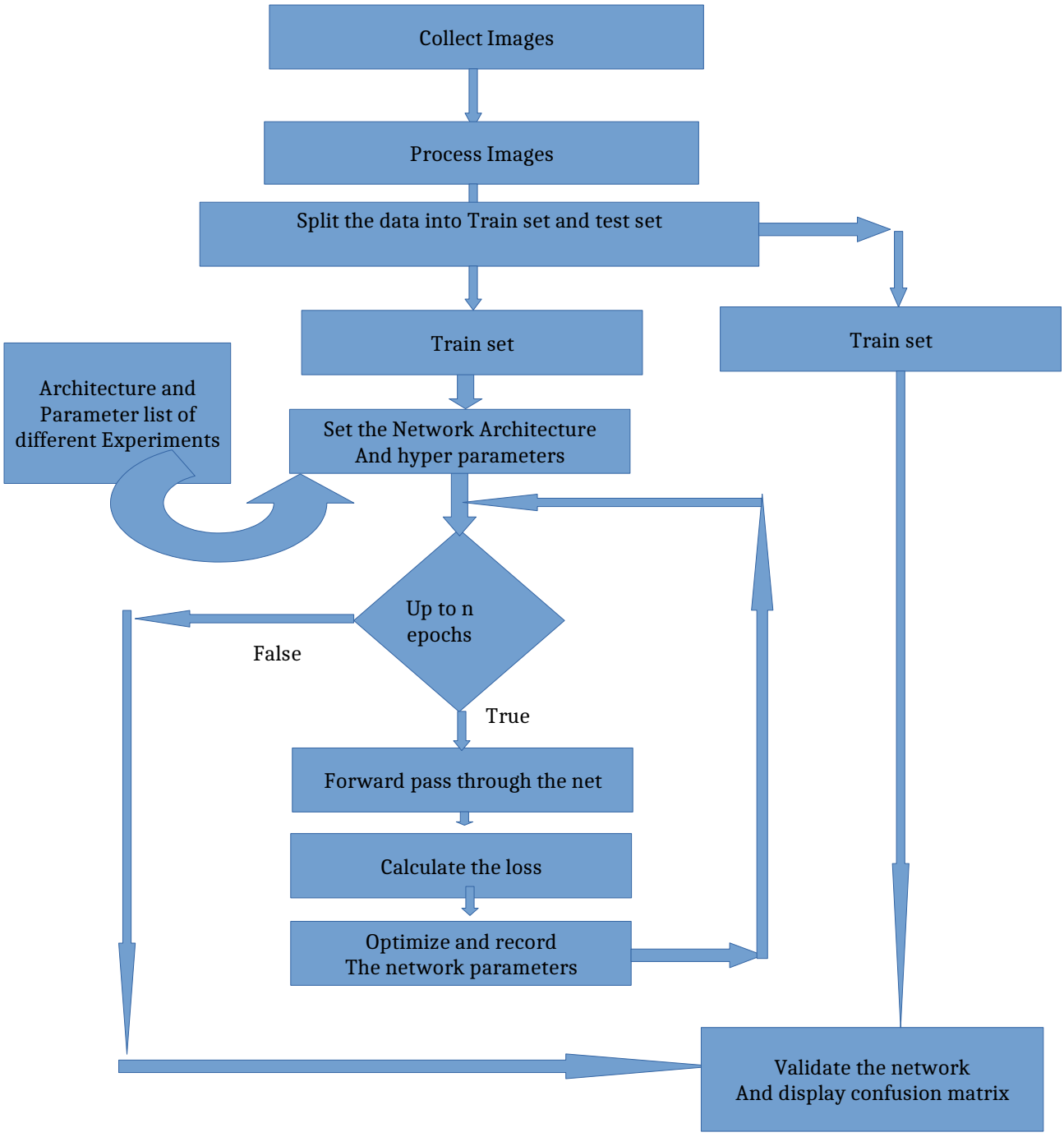
Experiment Name	**Learning Rate**	**Dropout**	**Number of epochs**
Experiment_1	0.01	20%	50
Experiment_2	0.001	20%	50
Experiment_3	0.0001	20%	50
Experiment_4	0.01	30%	50
Experiment_5	0.001	30%	50
Experiment-6	0.0001	30%	50
Experiment_7	0.01	50%	50
Experiment_8	0.001	50%	50
Experiment_9	0.0001	50%	50

After analyzing the output from these nine experiments, the model with the best accuracy is again tested over different batch size of 32, 64 and 128.

3.4 Cross Validation

Best performing network is trained for 40 epochs running through each of the image in every epoch and resulted an optimized neural network. But this network is again tested over different splits of data set to give validation to the obtained output. Train data is divided into 10 equal sets and 10 fold cross validation is applied. In every run one among 10 set is taken as validation data set and other 90% data is used to train the model. The average of those 10 data pair is taken as valid accuracy of network. Again this network is applied over the test data set to view its accuracy over external wold problems. Test of the network is also viewed with the help of confusion matrix showing miss interpretation of the classes of the inputs.

3.5 System Block Diagram



3.6 Tools used

1. Jupyter notebook
2. Google colab
3. Python
4. Shutter
5. Image viewer

4. EXPERIMENTS AND OUTPUTS

a) Collected image data

Raw image of a character obtained after cropping 28*28 image size:

Figure 4.1:Sample of collected image data

The process of collection is found to be so problematic and hectic during data collection.

b) Batch of input data

This set of data is input to the network as 32, 64 and 128 image per batch as per the defined architecture. A sample of representation of a batch of data as torch of size [32,28,28], images and true label of first 4 image is obtained as shown below for the case of Experiment_3.

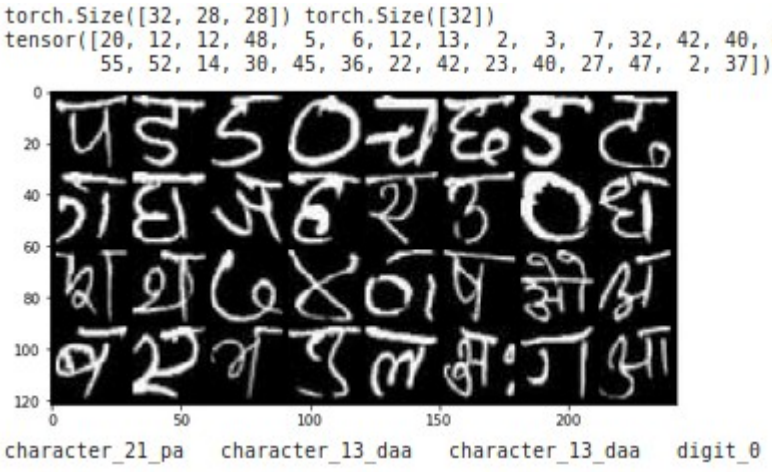


Figure 4.2 Screenshot of one batch of data fed into network

c) Network displayed after we define it through pytorch library

As we define the Convolutional neural network, this is displayed in detail as below for the case of Experiment_3.

```
Net(
  (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 12, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=192, out_features=192, bias=True)
  (fc2): Linear(in_features=192, out_features=96, bias=True)
  (fc3): Linear(in_features=96, out_features=58, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

Figure 4.3 Screenshot of display of details of network

d) Experiments over different values of learning rates and drop outs.

Learning rates are varied as 0.01, 0.001 and 0.0001 and the dropout values is varied as 0.2, 0.3 and 0.25 respectively. Results of these variation over those hyper parameters are tabulated as below.

Learning Rate> / **Dropout**	0.01	0.001	0.0001
20%	Experiment_1 Accuracy=64.92% Loss=1.11	Experiment_2 Accuracy=90.59% Loss=0.355	Experiment_3 Accuracy=67.35% Loss=1.14
30%	Experiment_4 Accuracy=56.03% Loss=1.52	Experiment_5 Accuracy=85.04% Loss=0.47	Experiment_6 Accuracy=61.33% Loss=1.37
25%	Experiment_7 Accuracy=55.27% Loss=1.52	Experiment_8 Accuracy=88.72% Loss=0.46	Experiment_9 Accuracy=65.145 Loss=1.22

Hence by analyzing these results we can conclude that the hyper parameters set of Experiment_2 are appropriate for the network.

e) Experiments over different Batch sizes

After finding appropriate learning rate and dropout this network is applied over different batch size of input data to find optimal batch size. Results over different batch size of input data in model as in Experiment_2 with learning rate of 0.001 and dropout of 0.2 is as shown below.

Batch Size		
32	64	128
Accuracy=90.12% Loss=0.36	Accuracy=90.59% Loss=0.355	Accuracy=87.56% Loss=0.374

f) 10 fold cross validation

After obtaining adjusted hyper parameters, the optimized network with the learning rate=0.001, dropout = 0.2 and batch size=64 is subjected to K-fold cross validation method. Here we applied 10 fold of split data to validate the output. Results of the validation over 10 different sets of data are as shown below.

Folds	**Training loss**	**Training Accuracy % (Top-1)**	**Training Accuracy % (Top-5)**	**Validation Accuracy % (Top-1)**
0	0.30	91.07	99.59	80.95
1	0.31	89.84	99.64	80.70
2	0.31	90.57	99.54	82
3	0.32	89.44	99.52	81.7
4	0.35	88.80	99.47	80.62
5	0.34	87.91	99.58	79.93
6	0.28	91.14	99.69	82.41
7	0.32	90.23	99.67	82.41
8	0.33	89.56	99.55	80.99
9	0.28	91.01	99.56	81.66
Average	0.314	89.96	99.581	81.34

g) Results and plots of different Experiments

1. Experiment_1

Training loss = 1.11
Training accuracy (Top1) =64.92%
Training accuracy(Top5)= 93.44%
Validation accuracy(Top1)= 59.87%

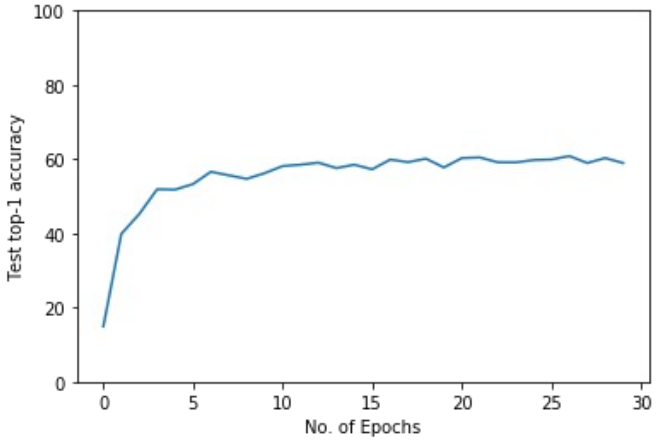


Figure 4.5 Top-1 Accuracy vs epochs of Experiment_1

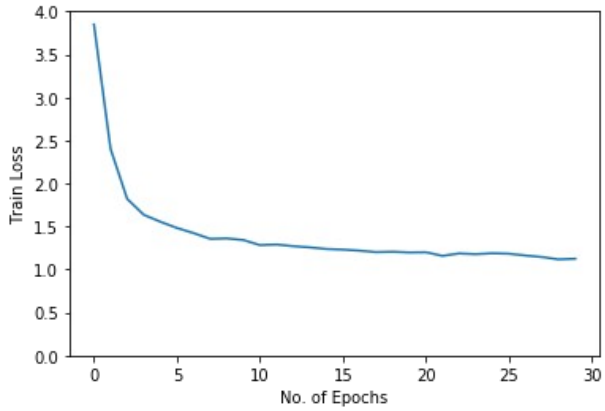


Figure 4.6: Loss vs Epochs of Experiment_1

2. Experiment_2

Training loss = 0.355
Training accuracy (Top1) =90.59%
Training accuracy (Top5)= 99.77%
Validation accuracy(Top1)= 81.59%

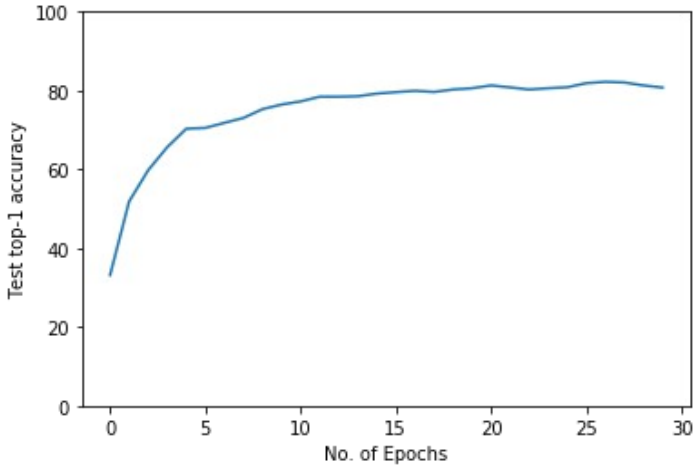


Figure 4.7 Top-1 Accuracy vs epochs of Experiment_2

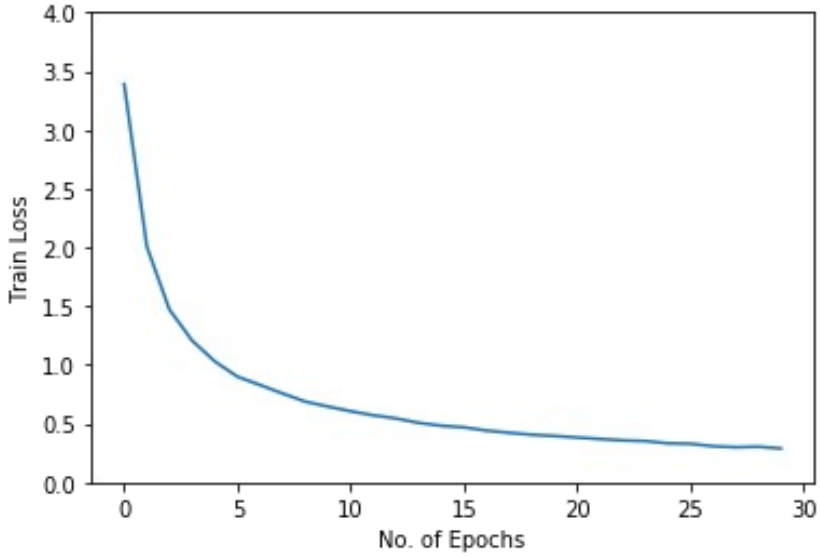


Figure 4.8: Loss vs epochs of Experiment_2

3. Experiment_3
Training loss = 1.14
Training Accuracy (Top-1) =67.35%
Training Accuracy (Top-5)= 92.87%
Validation accuracy(top1)= 65.34%

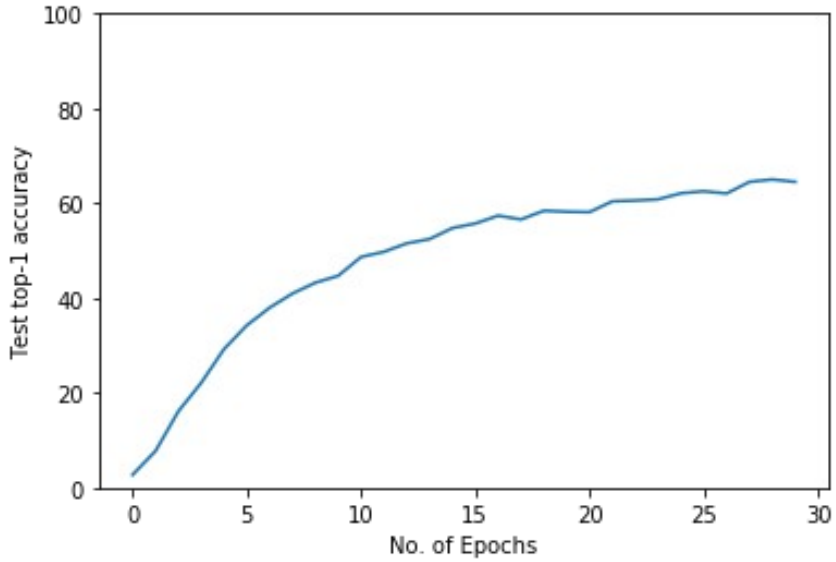


Figure 4.9: Top-1 Accuracy vs epochs of Experiment_3

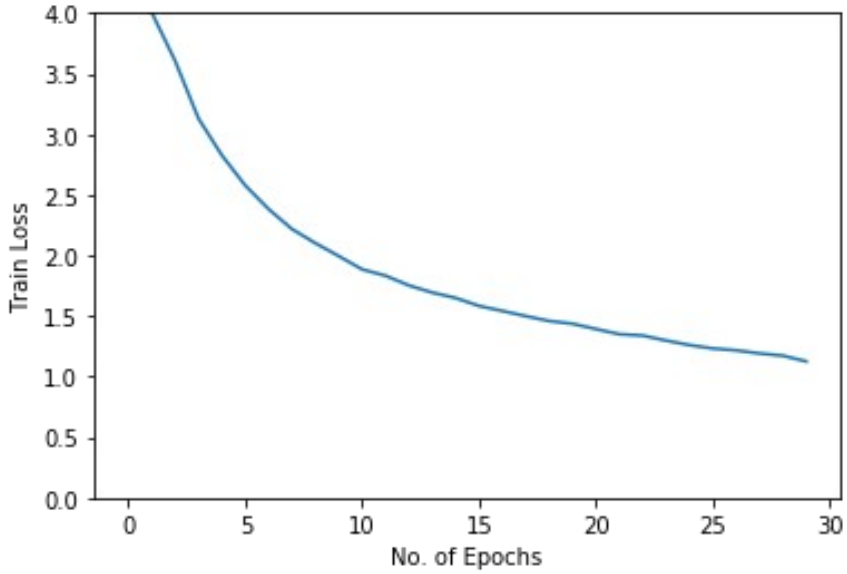


Figure 4.10: Loss vs epochs of Experiment_3

4. Experiment_4

Training loss = 1.52
Training Accuracy (Top-1) =56.03%
Training Accuracy (Top-5)= 87.86%
Validation accuracy(top1)= 53.07%

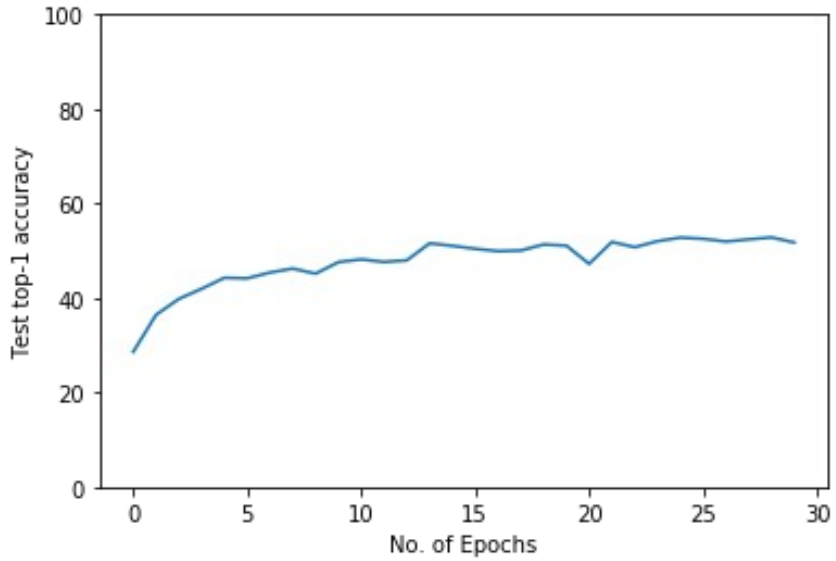


Figure 4.11 Top-1 Accuracy vs epochs of Experiment_4

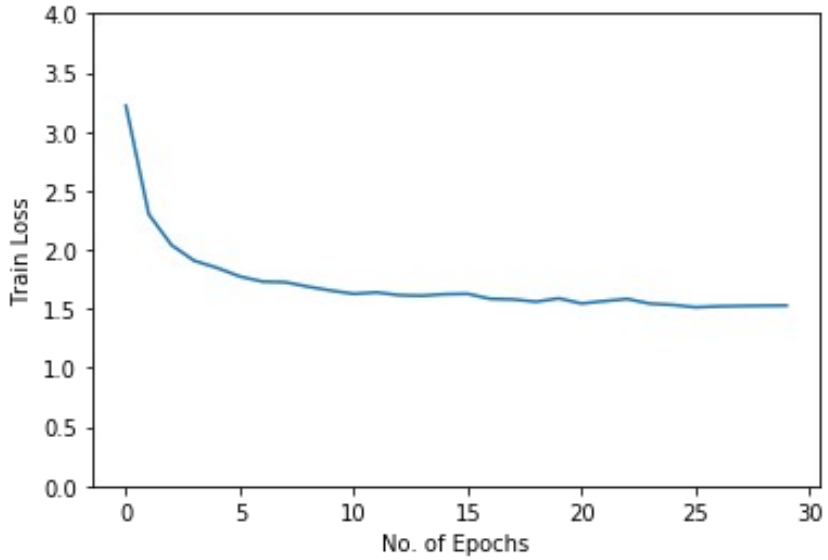


Figure 4.12: Loss vs epochs of Experiment_4

5. Experiment_5
Training loss = 0.47
Training Accuracy (Top-1) =85.04%
Training Accuracy (Top-5)= 98.87%
Validation accuracy(top1)= 77.99%

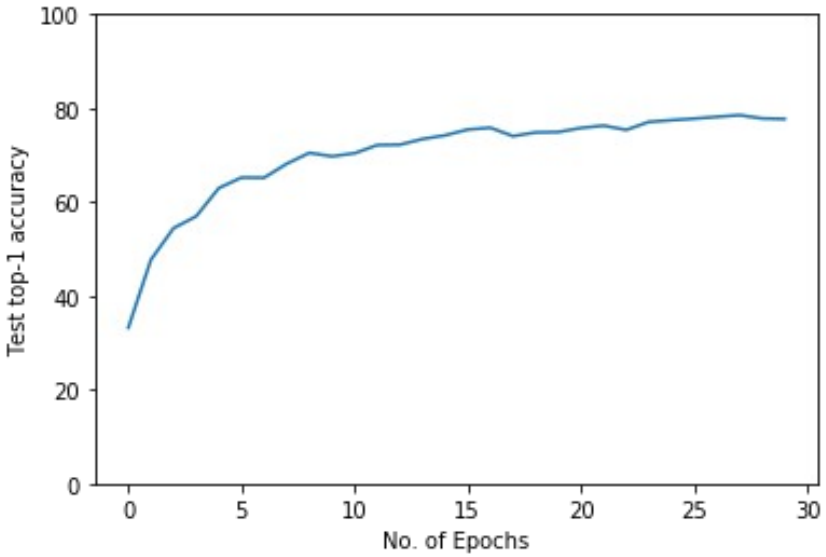


Figure 4.13: Top-1 accuracy vs epochs of Experiment_5

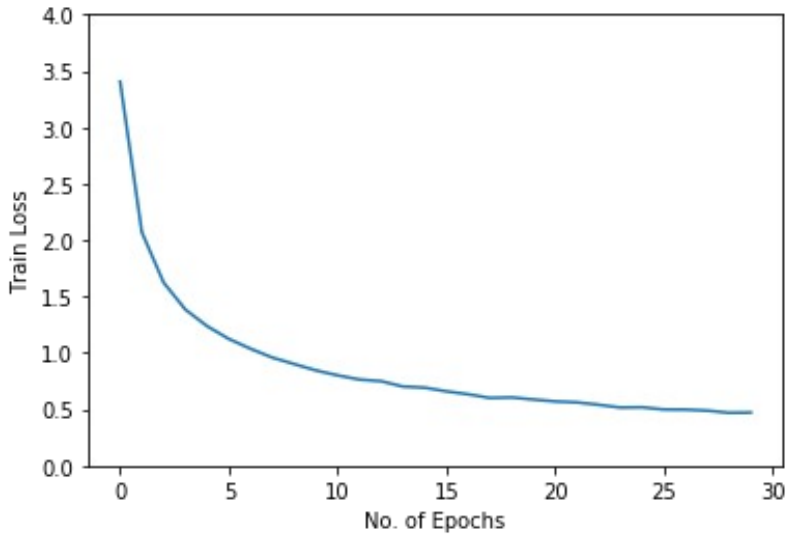


Figure 4.14: Loss vs epochs of Experiment_5

6. Experiment_6

Training loss = 1.37
Training Accuracy (Top-1) =61.33%
Training Accuracy (Top-5)= 89.44%
Validation accuracy(top1)= 58.04%

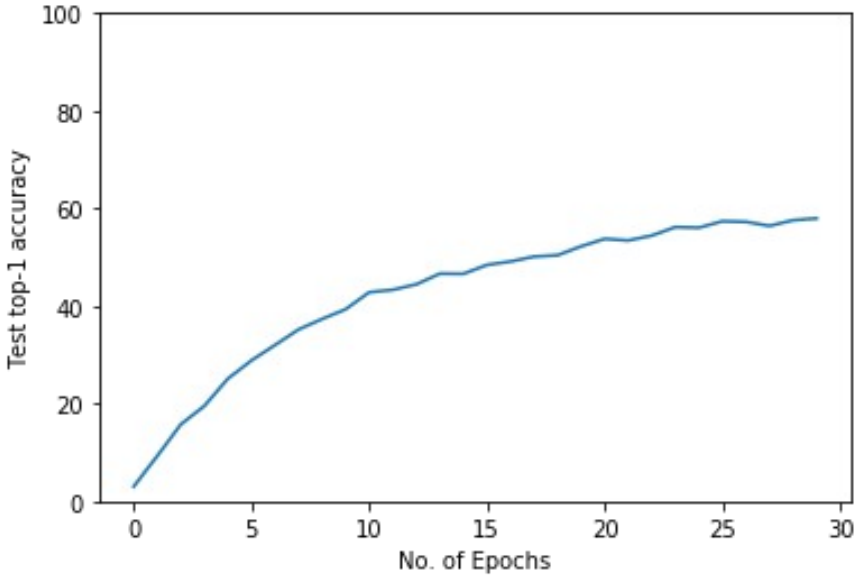


Figure 4.15: Top-1 Accuracy vs epochs of Experiment_6

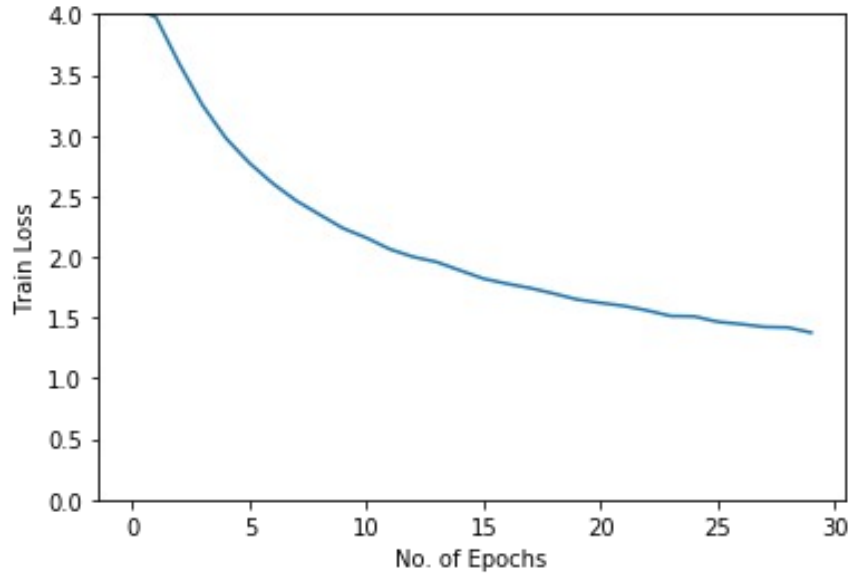


Figure 4.16: Loss vs epochs of Experiment_6

7. Experiment_7

Training loss = 1.52
Training Accuracy(Top-1) =55.27%
Training Accuracy(Top-5)= 87.54%
Validation accuracy(top1)= 51.87%

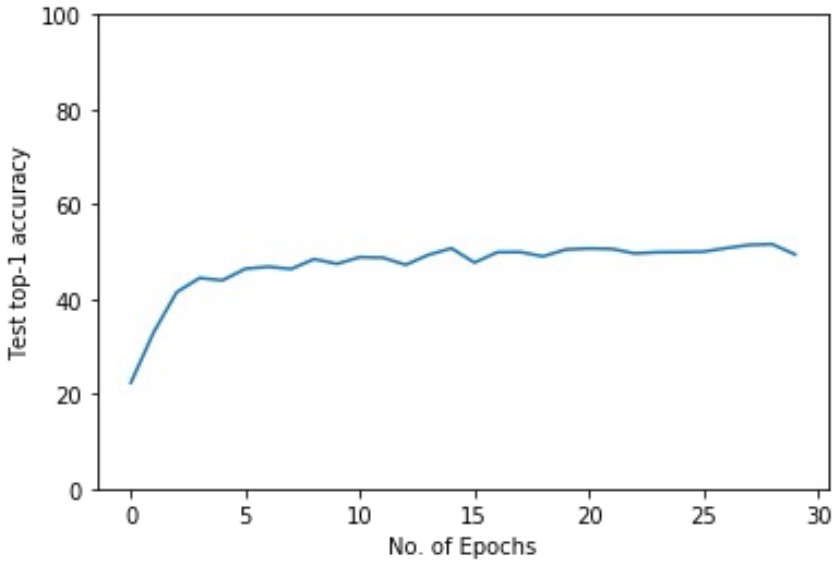


Figure 4.17: Top-1 Accuracy vs epochs of Experiment_7

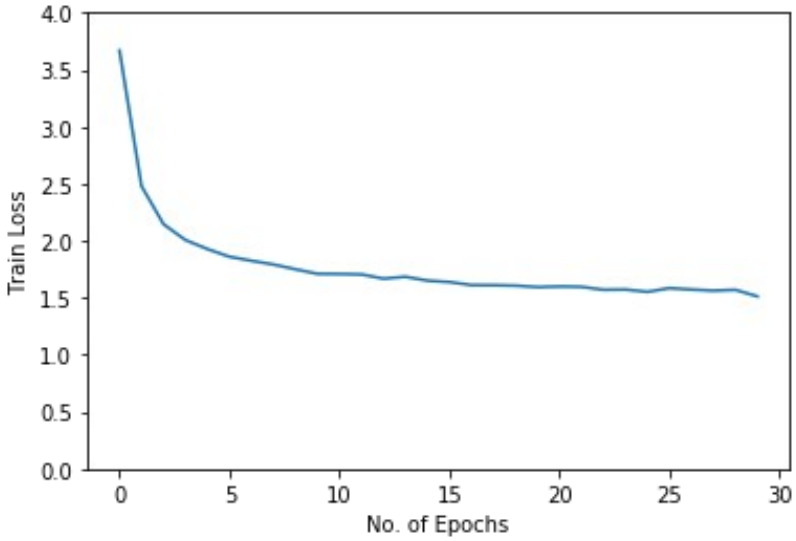


Figure 4.18: Loss vs epochs of Experiment_7

8. Experiment_8

Training loss = 0.4624
Training accuracy(Top-1) =88.72%
Training accuracy(Top-5)= 99.24%
Validation accuracy(top1)= 80.31%

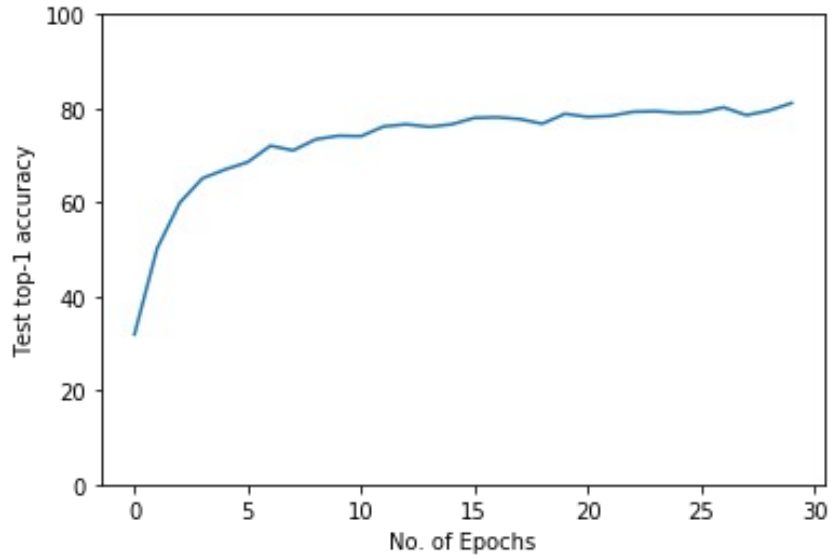


Figure 4.19: Top-1 Accuracy vs epochs of Experiment_8

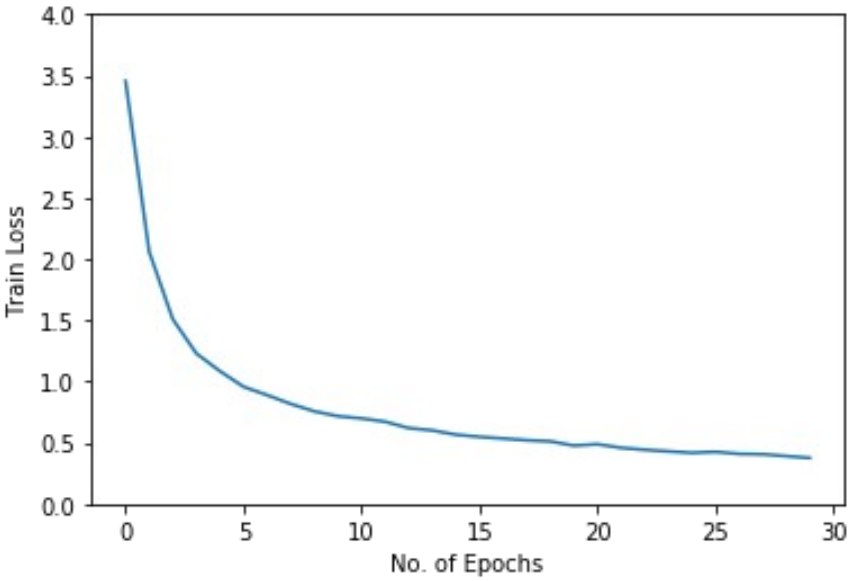


Figure 4.20: Loss vs epochs of Experiment_8

9. Experiment_9

Training loss=1.22
Training accuracy(Top-1) =65.14%
Training accuracy(Top-5)= 91.24%
Validation accuracy(top1)= 62.35%

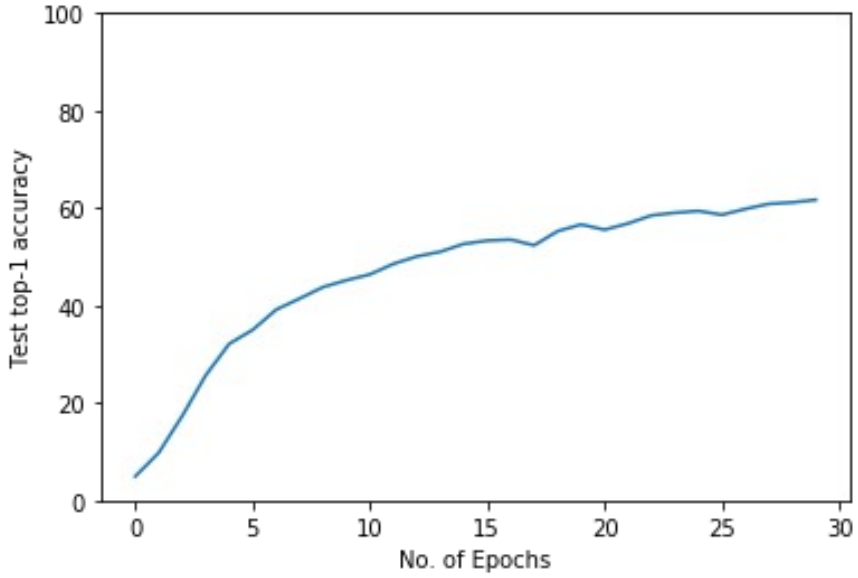


Figure 4.21: Top-1 Accuracy vs epochs of Experiment_9

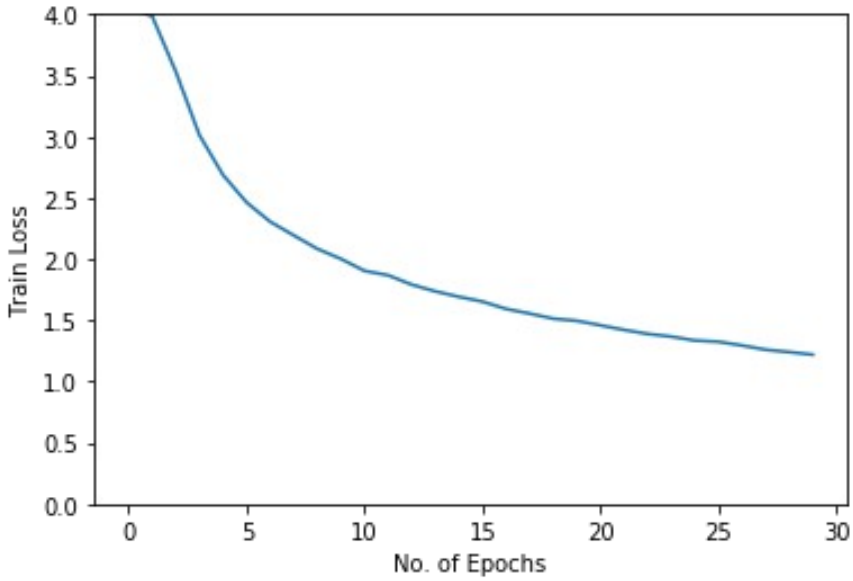


Figure 4.22: Loss vs epochs of Experiment_9

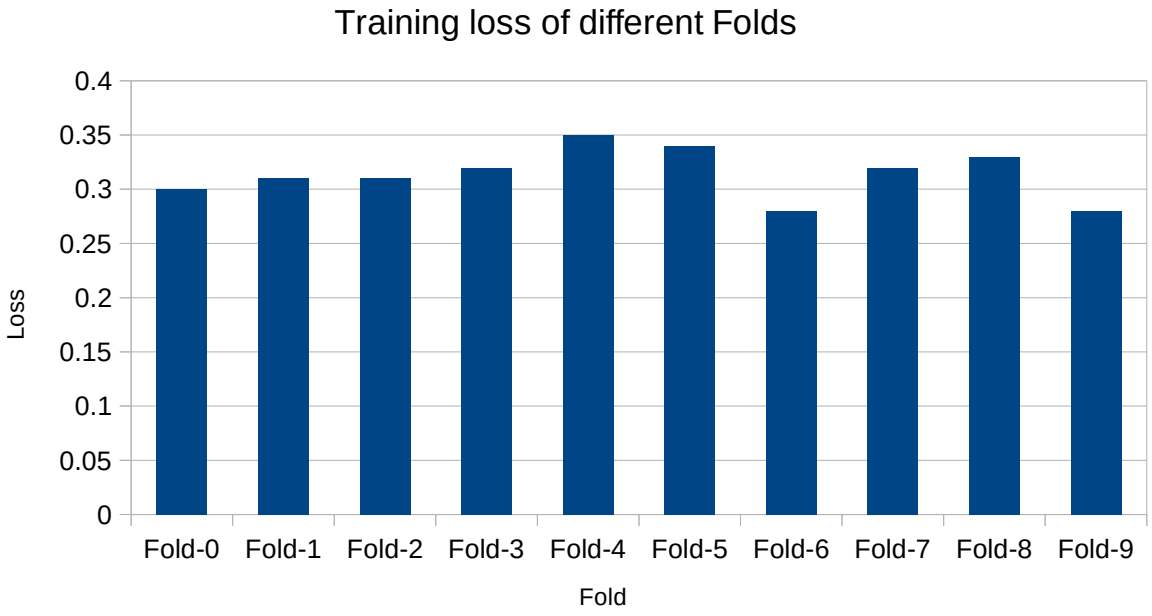


Figure 4.23: Losses of different folds of data

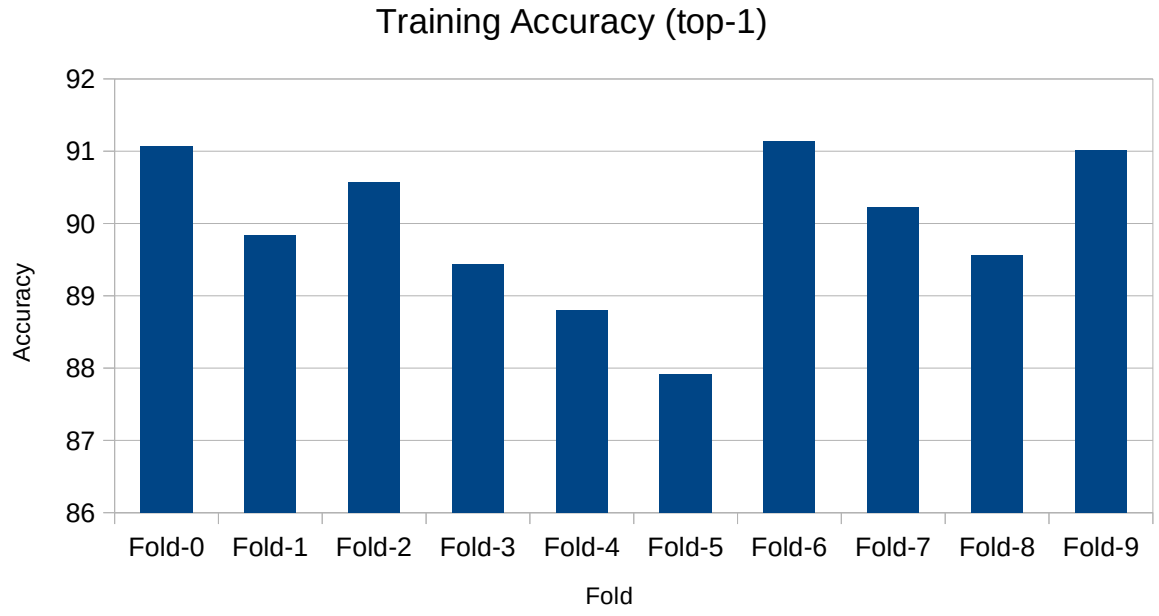


Figure 4.24: Top-1 accuracy of different folds over train data

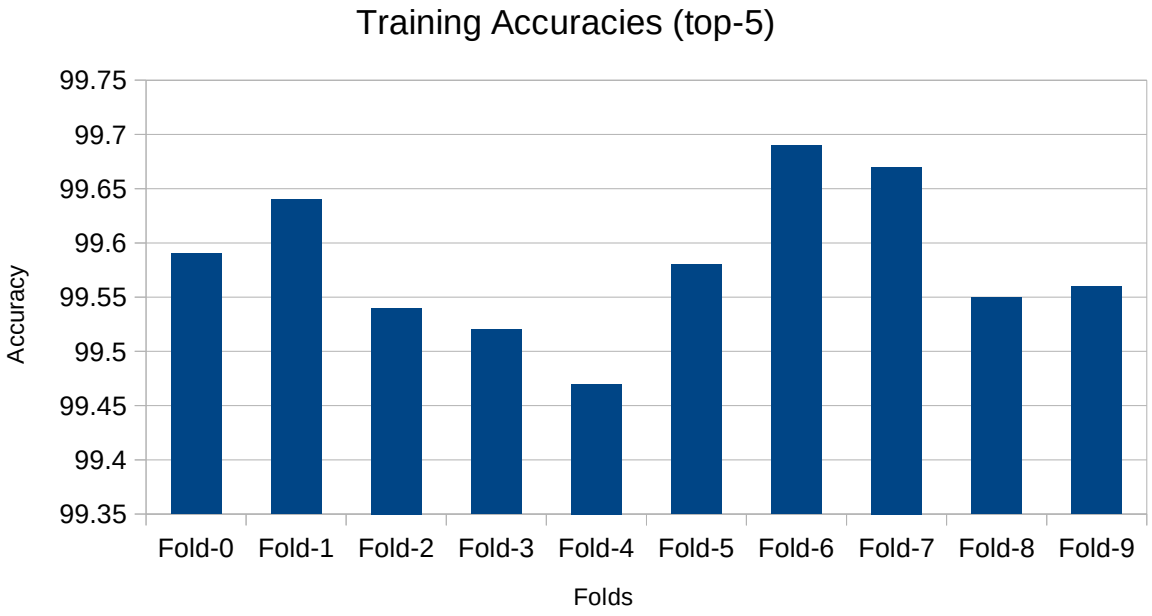


Figure 4.25: Top-5 accuracy of different folds over train data

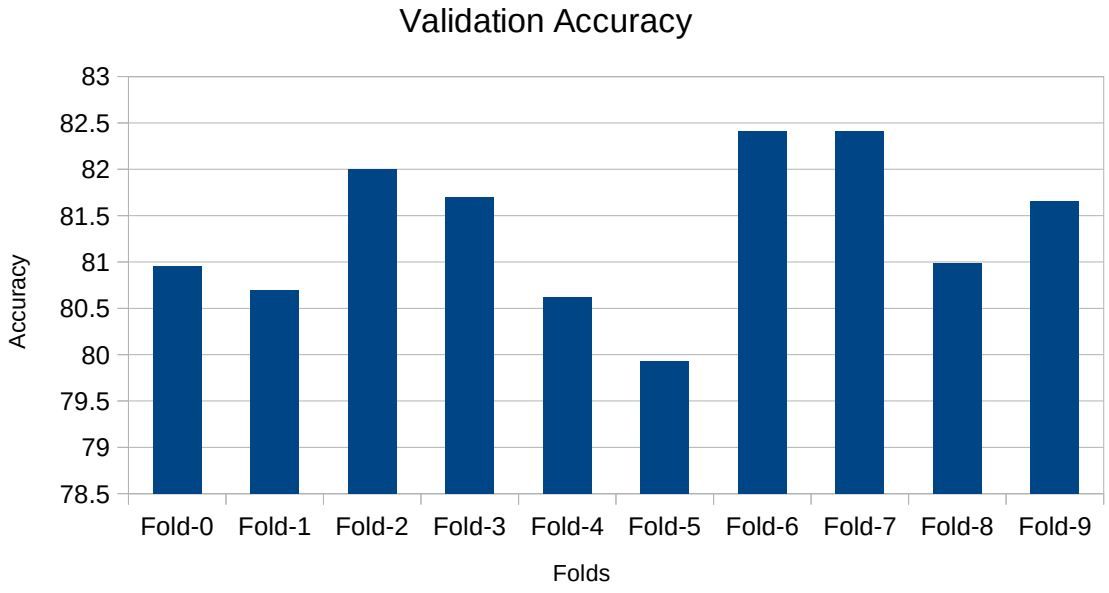
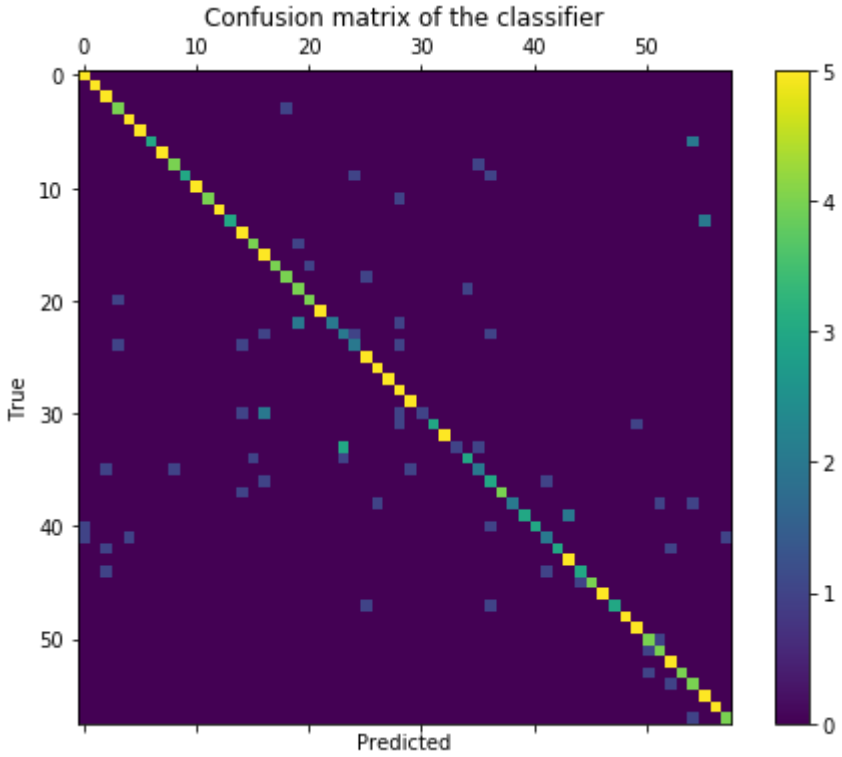


Figure 4.26: Top-1 accuracy of different folds over validation data

h) Confusion matrix

Predicted and actual image classes are plotted to visualize the accuracy through confusion matrix, and seen as below.



5. EPILOGUE

5.1 CONCLUSION

Hence from this research work we have developed a hybrid Convolutional neural network to classify Devanagari Handwritten characters. This can be used to recognize handwritten scripts and documents to digitize various works. This network performs over handwritten data with valid accuracy of 81.34% and this can be implemented with any hand written characters in Devanagari.

5.2 LIMITATIONS

In regard of recent developments on deep learning this percentage of accuracy is not still good for implementation. Hence there exist so many limitation of this work. This model can not be implemented with good level of confidence. These data are from only few personals. Hence the accuracy can be increased with the help of more data sets.

Further more this model is developed for image set of individual characters, but in Devenagari writing they are not alone and they are in connected way and this work can be further developed to work on it.

6. REFERENCES

1. https://www.worldstandards.eu/alphabets/*
2. http://neuralnetworksanddeeplearning.com/*
3. Deng, L., & Yu, D. (2013). Deep learning: Methods and applications. Foundations and Trends in Signal Processing, 7(3–4), 197–387.
4. Acharya, Shailesh & Pant, Ashok & Gyawali, Prashnna. (2015). Deep learning based large scale handwritten Devanagari character recognition. 1-6. 10.1109/SKIMA.2015.7400041.
5. Rahman, M. M., Akhand, M. A. H., Islam, S., Chandra Shill, P., & Hafizur Rahman, M. M. (2015). Bangla Handwritten Character Recognition using Convolutional Neural Network. International Journal of Image, Graphics and Signal Processing, 7(8), 42–49. doi:10.5815/ijigsp.2015.08.05
6. Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document Recognition, in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, November 1998.
7. Feature extraction using convolution. Available:http://deeplearning.stanford.edu/wiki/index.php/
8. https://pytorch.org/docs/stable/_modules/torch/nn/modules/conv.html#Conv2d