TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

Thesis No.:

Image Classification based on Convolution Neural Network

By

Shailesh Singh

A THESIS REPORT

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SYSTEM AND KNOWLEDGE ENGINEERING (MSCSKE)

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

January, 2017

COPYRIGHT

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a thesis entitled **"Image Classification Based on Convolution Neural Network"** submitted by Shailesh Singh in partial fulfillment of the requirements for the degree of Master of Science in Computer System and Knowledge Engineering(MSCSKE)

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

External Supervisor,

Committee Chairperson, Dr. Diwakar Raj Pant
Head
Department of Electronics and Computer Engineering
Pulchowk Campus
Institute of Engineering, Tribhuvan University

Abstract

Apart from image processing techniques, the problem of object recognition can also be solved by using machine learning techniques. The main concern of this thesis is to classify images using machine learning techniques. To tackle with such problems, artificial neural network i.e. Convolutional Neural Network has been developed. In order to design the Convolutional Neural Network, different parameters like filter size, number of convolution layers, drop out layers etc were taken. Careful choosing and study of these parameters shows that efficient architecture can be designed. Different neural network architectures for CIFAR-10 and MNIST dataset have been developed. Being different in terms of number of hidden layers, filter size and other measures, they have been trained on Central Processing Unit. Drop out techniques has been used to reduce over-fitting issues. Accuracy of these architectures has been calculated by feeding the networks with the test data. Lastly, results are compared and analyzed to find out best architecture. Thus, this study gives a way to design efficient architecture for image classification.

Acknowledgements

It is great pleasure for me to complete a thesis. I am indebted to several personnel who have lent me hands in various ways during the thesis period.I would like to express my deep sense of gratitude to my supervisor, **Dr. Basanta Joshi**, for his immense guidance, constant inspiration and encouragement. It has been a privilege to work under his supervision.

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. Dibakar Raj Pant** 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 express my sincere appreciation towards my program co-coordinator, **Dr. Sanjeeb Prasad Panday**, for his constant guidance and motivations.I am also grateful to **Prof. Dr. Shashidhar Ram Joshi**, **Prof. Dr. Subarna Shakya** and **Dr. Aman Shakya** for their invaluable knowledge sharing, support and encouragement during my master degree.

Last but not the least my classmates (071 Batch, M.Sc. in Computer Systems and Knowledge Engineering) and all the people who are directly or indirectly involved in this thesis deserve special thanks for their kind cooperation, interest and support in my research work.

Table of Contents

Table of Contents

List of Figures

Abbreviations

AI	Artificial Intelligence
CNN	Convolution Neural Network
CONV	Convolution Neural Network
DNN	Deep Neural Network
MSGD	Minibatch Stochastic Gradient Descent
NLL	Negative Log Likelihood
POOL	Pooling Layer
RELU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent

Chapter 1: Introduction

1.1 Background

With the advent of modern technologies, computers are becoming powerful day by day. They have become perfect companion with high speed computing capabilities over the time. Few decades ago it was believed that machines are only for arithmetic operations but not for complex tasks like speech recognition, object detection, image classification, language modeling etc. But now a day, situation is inverted. Machines are capable of doing these things more easily with very much high accuracy.

Conventional algorithm consisting of finite arithmetic operations cannot provide capacity to do such complex tasks for machine. For this, Artificial Intelligence provides lots of techniques. Learning Algorithms are used for such purpose. Huge dataset is required for training the model with appropriate architecture. Testing is required to evaluate whether the model is working properly or not.

Image processing focuses on two major tasks. They are:

i. Processing of image data for storage, transmission and representation for autonomous machine perception.
ii. Improvement of pictorial information for human interpretation.

Following are the different uses of image processing:

i. Medical Applications
ii. Industrial Inspection
iii. Artistic effects
iv. Law enforcements
v. Human computer interfaces etc

Neural Network is one of AI techniques emerged long ago in 1940s but technology at that time was not so advanced. It was then wakening time to time but could not impress the Computer Science Community very much. It was up at time in 1980s with the development of back-propagation. Later it was again discarded due to slow learning and expensive computation. In 2000s, it picked up again in AI field with lot of researches. It was believed that only 2 to 3 hidden layers are sufficient for Neural Network to work properly but later on it is observed that even more layers can represent high dimensional features of the input signals. Such neural networks are referred to as Deep Neural Networks (DNN).

Connections between layers in DNNs, known as receptive fields (RF), are very important parameters. They have to learn good feature representations of the dataset and these representations involves learning linear filter weight values form input data. Results from (Eugenio, Dundar, Jin, & Bates, An Analysis of the Connections Between Layers of Deep Neural Networks, 2013) shows that even fully connected Convolutional Neural Network (CNN) Model performs poorly in Image Classification Tasks.

Defining appropriate architecture for object detection is the main task that this thesis is intended to do. Suitable feature representation methods are required to make the DNN work properly with high accuracy.

Supervised Learning and Multi-Class Classification

Supervised learning is one of the common machines learning which consists of model which is trained with training data. The training data consists of “labels” or “right answers”. Under this, the model is trained. After completion of training, the model is supposed to give us more right answers on new set of training examples.

There are two types of problems under supervised learning, Classification and Regression. In Classification problem, the model classifies data into one of multiple discrete classes, no in between the classes where as in Regression problem, the model classifies predicts some continuous real valued problem.

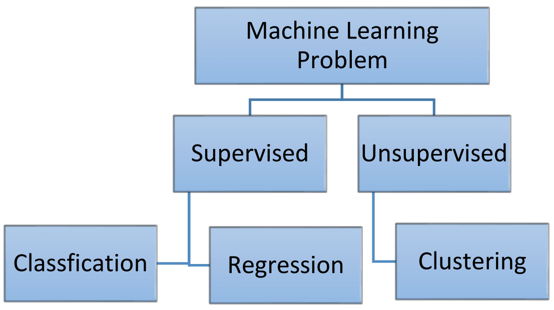


Figure 1:- Machine Learning Overview

Needs of Image Classification:

The intent of the classification process is to categorize all pixels in a digital image into one of several objects. The image classification is required in various field like medical sector, remote sensing, security, aviation etc. This is also used to classify the satellite images. The image classification is done mainly by three methods: Unsupervised image classification, Supervised image classification and Object-based image analysis

Classification of Image:

Image classification is a task of assigning an input image one label from set of categories. This is one of core problems in computer vision.In computer vision, image is represented as one large 3-dimensional array of numbers. Consider RGB image of size 32x32. What computer sees is $3x32x32$ numbers or a total of 3072 numbers. And the task is to obtain a single label (such as chair) from these numbers. Though recognizing an object is trivial task for human to perform, It is challenge for computer vision algorithm to work correctly.A single object can be oriented in many ways with respect to camera. So there may be viewpoint variation to consider. Other challenges for the algorithm may to taking into account of scale variation, deformation, Occlusion, Illumination Conditions, Brightness Clutter and Intra-class variation. A good image classification model should address all of these variations.

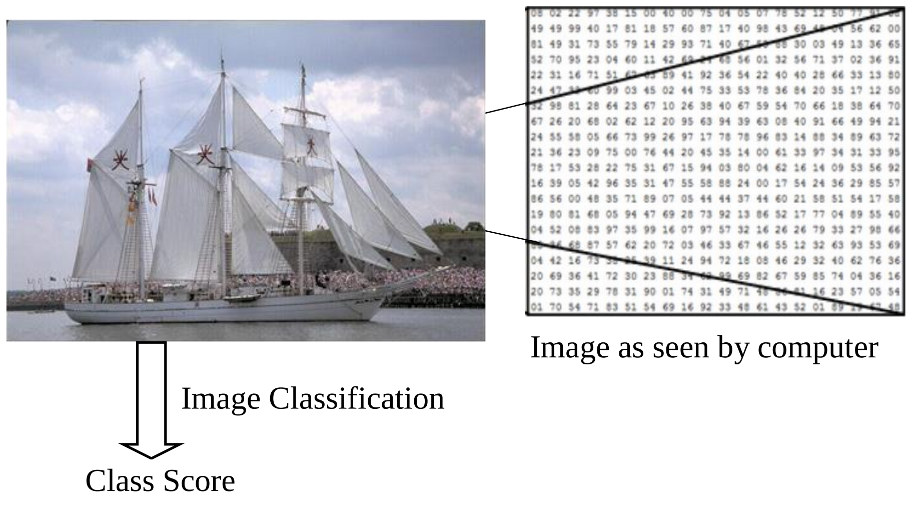


Figure 2:- Concept of Image Classification Problem

1.2 Problem Statement

Now a day the application of artificial intelligence is growing in various sector to make the decision automatic or to enhance the system so that they can cope with the changing environment. Neural Network is a special technique in artificial intelligence field. Deep Neural Networks are applicable to fields like image recognition (Eugenio, Dundar, Jin, & Bates, An Analysis of the Connections Between Layers of Deep Neural Networks, 2013), Speech Recognition (Sainath, Kingsbury, Mohamed, & Ramabhadran, 2013; Alex Graves), text prediction and handwriting generation [4], Language Generations [5]. DNN architecture for one task does not work well with another task. For task related to specific field, it is required to define which architecture to pick. For Image Classification or Object Detection task, and designing its structure is challenging job.

To develop the neural network architecture based image classification is a challenging job and also a booming topic in research community. This motivates me to take this thesis as an opportunity to work in this sector and give some contribution in research community in this fied.

1.3 Objectives

The main objectives of this thesis are:

- To design effective architecture of CNN for image classification task.
- To classify images using Convolutional Neural Network

Chapter 2: Literature Review

2.1 Neural Networks

Neural Networks are biologically inspired connectionist models that receive some input and transform it through series of hidden layers and finally calculates output. In regular neural network, neuron in each layer is connected to all neurons in previous layers.

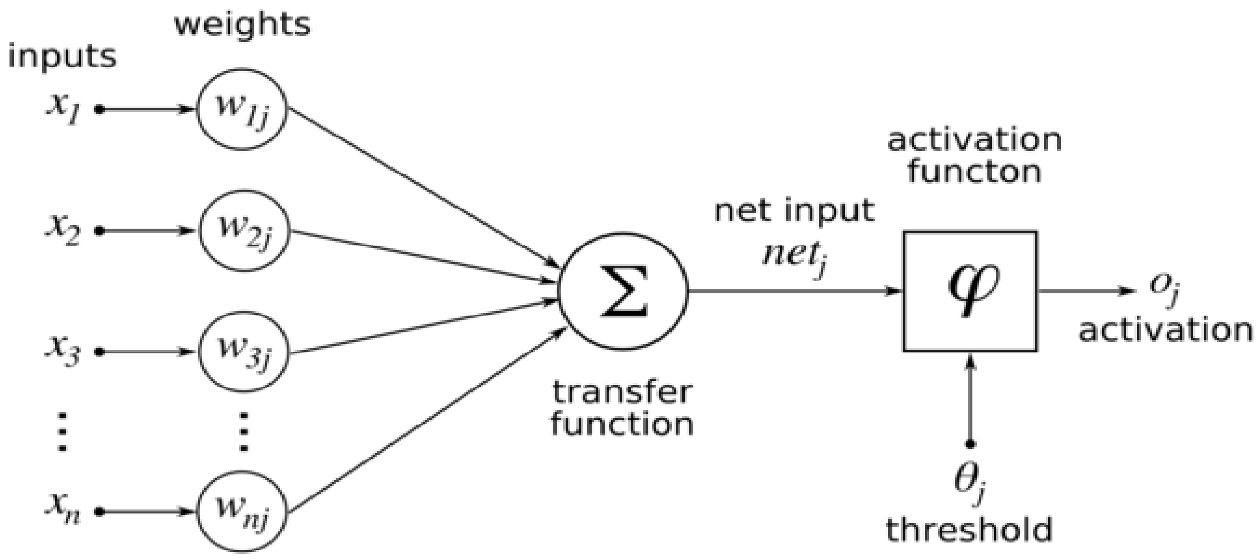


Figure 3:- An Artificial Neural Network

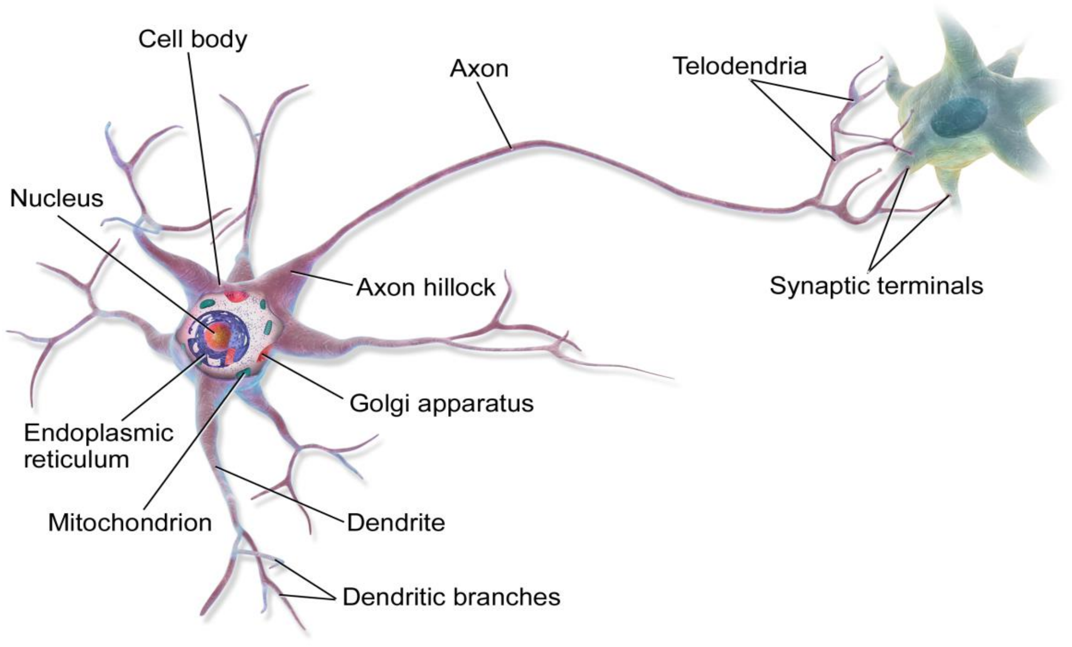


Figure 4 : - A Human Neuron Example

Multilayer Perceptron (MLPs)

Every neurons are independent and do not share connection among them. Problem with these types of NNs is they cannot scale well (Grishick, Donahue, Darrel, & MAlik, Rich Features Hierarchies for accurate object detection and semantic segmentation.). Consider a simple example where input is RGB image of 32 x 32, there will be 32x32x3=3072 weights. This may be somehow manageable but when the image is like 200x200 then there will be about 120000 weights, in this full connection model is wasteful and they possess over-fitting [7].

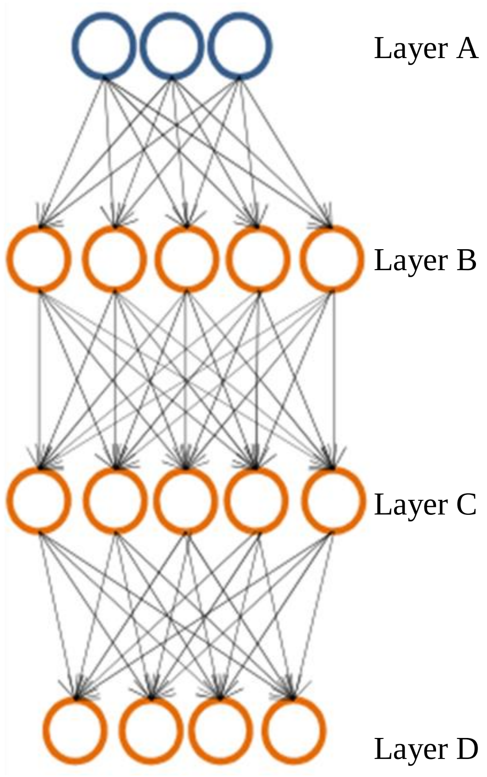


Figure 5 :- Multilayer Perceptrons with 4 layers.

2.2 Deep Neural Network (DNNs)

Deep Neural Networks are highly recognized Neural Network Architecture and are used extensively used in computer vision, Speech Recognition, Language modeling and Natural Language Processing. They have deeper architecture than conventional neural network. Convolutional Neural Network and Recurrent Neural Network (RNN) are examples of DNN. RNNs are used for sequence learning tasks like Speech Recognition, Handwriting Generation, language modelling(Alex Graves; Graves, Generating Sequences with Recurrent Neural Networks, 2014; Sutskever, 2013).

2.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Network is the powerful tool for image classification and object detection. Instead of full connection model they modify neurons to be connected to small region of the neurons in previous layer. It consists of many layers.

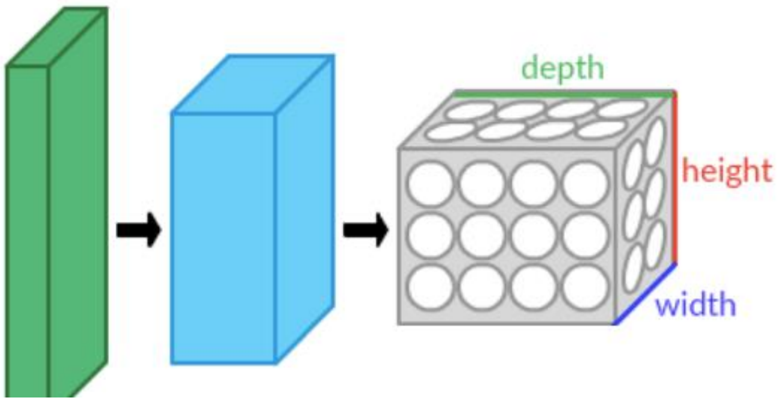


Figure 6:-CNN with its neurons arranged in three dimensions

i. Convolution Layer (CONV)
ii. Pooling Layer (POOL)
iii. Fully Connection Layer (FC)

Simple example of CNN structure is like [INPUT-CONV-RELU-POOL-FC]. CONV and FC apply transformations and the operations done are the function of parameters, while POOL/RELU Layers have fixed functions [7].

Convolutional Layers

Convolutional layers consist of set of learnable filters. Network will learn filters that activate when they see some specific type of feature at some spatial position of the image. Stacking number of these filters makes depth of Convolution Layer. Each neuron takes inputs from a rectangular section of the previous layer; the weights for this rectangular section are the same for each neuron in the convolutional layer. This rectangular section is receptive field that runs whole across the image. Thus, the convolutional layer is just an image convolution of the previous layer, where the weights specify the convolution filter. [9]

$$\mathrm{O}[\mathrm{m},\mathrm{n}] = \mathrm{f}[\mathrm{m},\mathrm{n}] * \mathrm{g}[\mathrm{m},\mathrm{n}] = \sum_{u=-\infty}^{\infty} \mathrm{f}[\mathrm{u},\mathrm{v}]\mathrm{g}[\mathrm{m}-\mathrm{u},\mathrm{n}-\mathrm{v}]$$

Depth, Stride and Zero-padding

Depth(D) is number of neurons in convolution layers that connect to same region of input volume.

Stride(S) is measure of shifting of filter. Stride(S) =1, means that filter is moved to 1 unit (or Pixel) at a time in x and y directions. More the value of S, there will be less alignment of filters.

Zero-padding (P) means adding zeros to the start and end of rows and columns. It is done in order to properly extract the features of boarder pixels.

If W is width of input image, F is filter size, P is zero-padding and S is stride then Convolving image would result output with size given by:

$O = \frac{W-F+2P}{S}+1.$

Also image can be convolved only when O is integer.

Pooling Layer

The pooling operation decreases dimensions of convolved image. It may be min-pooling, max-pooling or averaging. The pooling regions are very small. If pooling region is just 2×2 then this will have effect of subsampling the output maps by a factor of 2 in both dimensions. It progressively reduces spatial size and reduces over-fitting.

Fully Connection Layer

After reducing image to suitable feature-maps, fully connected MLP is used. This layer consists of neurons that are connected to every neurons of previous layer. [As in figure 1]

CNN Architecture

CNN in figure 6 consists of input layer, convolutional layers, and subsampling layers. Input layer is input image. After this layer there is hidden convolution layer. Input image is convolved with 4 different convolution filter and we get 4 feature maps of the same image. These 4 feature maps are then pooled and down-sampled at next layer. The down-sampled feature map is again convolved with convolution filter to get 6 other feature maps. These are again sub-sampled. This final feature map is then fully connected with output classes.

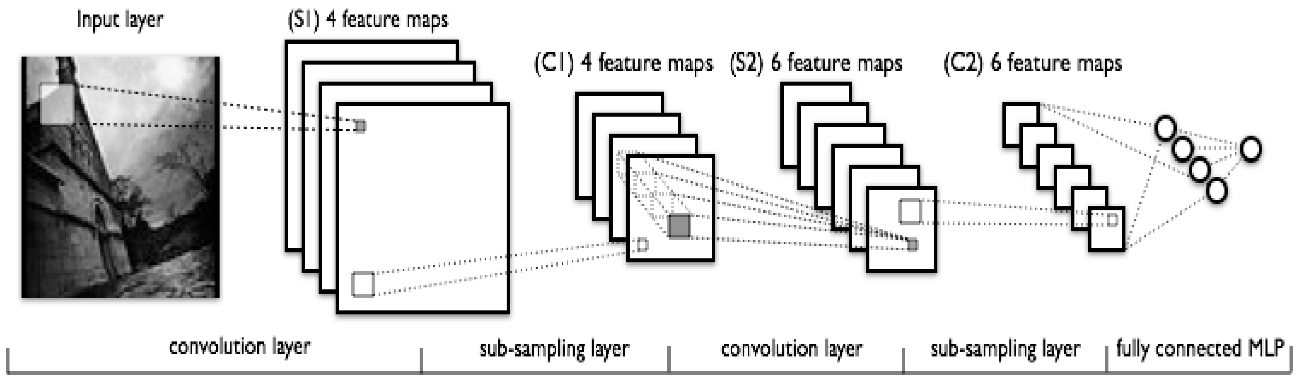


Figure 7 : - Different Layers within Convolutional Neural Network Architecture used in [9]

2.4 Related Works

First successful implementation of CNN was done by Yan LeCunn in 1990's which is used to read zip codes and digits [9]. CNN has been proved to be powerful tool for image classification and object detection. It became more powerful when AlexNet(Alex, Ilya, & Hinton, 2012) won ImageNet ILSVRC Challenge 2012 with significant performance from 2nd runner-up. AlexNet was CNN with some modification to LeNet. It was deeper, bigger and Conv Layers were stacked instead of immediate pooling layer. ZFNet(Zeiler & Fergus, 2013) won ImageNet ILSVRC Challenge 2013. It modified AlexNet by varrying hyper parameters and using larger filters on convolution layers.

Both CNN, along with Recurrent Neural Networks (RNN) are used to solve the problem like image captioning(Fie-Fie & L., 2014; Vinyals, Toshev, Bengio, & Erthan, 2014; IIya Sutskever, 2011).

Chapter 3: Methodology

Data Driven Approach

It’s quite difficult task to generate algorithm within numbers of the image to classify it. Instead this problem is solved as supervised learning. There are lots of examples of images for each individual class. The models are exampled and develop learning algorithms that look at images and learn about visual appearance of each class. This is data driven approach. [7]

Image Classification Pipeline

Image Classification is just reading array of pixels from image and assigning it a label. The complete pipeline can be viewed as below:

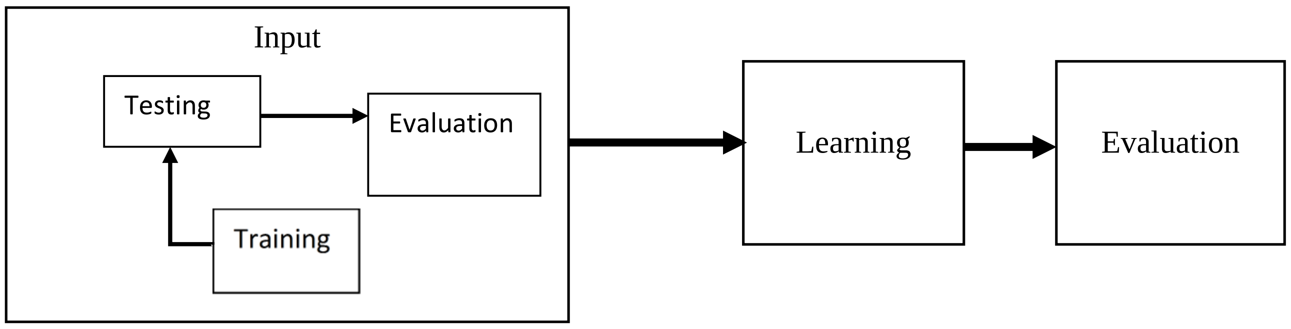


Figure 8 : - Image Classification Pipeline

Input

The first step in the pipeline is Input and it consists of set of images each labeled with one of the categories. It is known as training set.

Learning

The training set data is used by model to learn how each of class looks like. This step learns classifier.

Evaluation

The final step is Evaluation in which model is given new set of images to classify. These images are not seen by the model before and it calculates labels for these images. Then the predicted labels are compared with true label and expect most of predicted ones to be true. It is expected to have lot of predictions to be correct.

3.1 Model Development

For classification of data, there are various steps involved. Steps can be viewed as figure below:

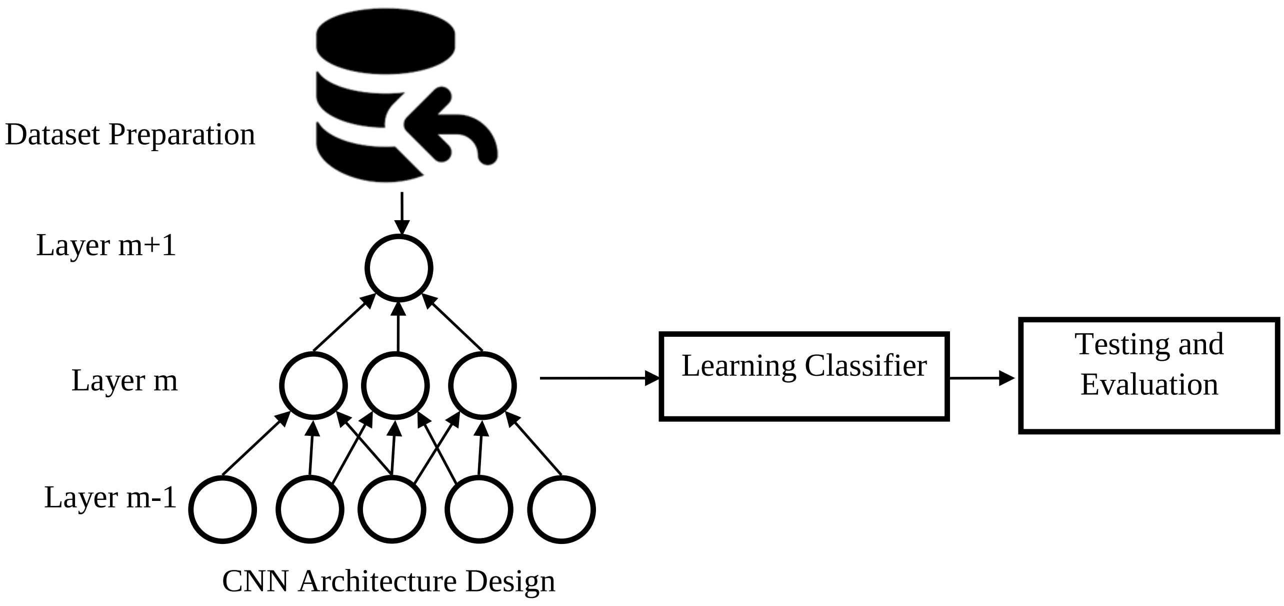


Figure 99: - Model Development Steps

3.1.2 Dataset Preparation

Data provided is first must be divided into training set, validation set and testing sets.

Training Set	Validation Set	Testing Set

Figure 10:- Dataset Division

Training Setare used to adjust various weights and parameters of the model.

Validation Setare not used to adjust the parameters of the model, instead they are used to reduce overfitting problem

Testing Setare used for evaluating the predictive power of the model.

3.1.2 CNN Architecture Design

CNN in Image Classification

CNN is to be designed for sole purpose of Image Classification. The architecture of network will be modified; its filters are changed to see which will give best output.

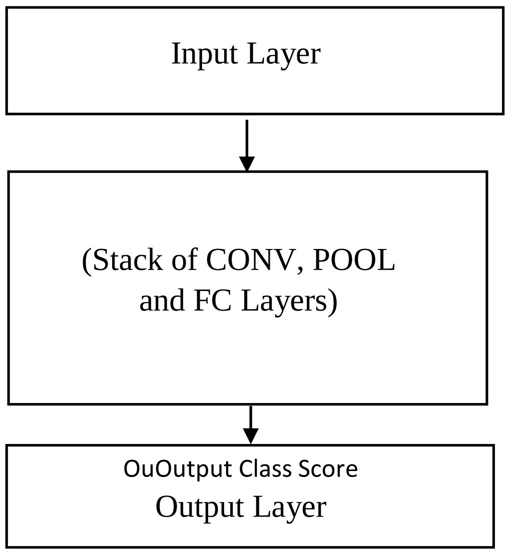


Figure 11:- CNN Design

Convolution Layer Design

Convolution Layer takes input as volume [$W_1xH_1xD_1$], and outputs another volume as [$W_2xH_2xD_2$]. It requires four hyper-parameters: Number of filters (K), Filter's spatial extent (F), amount of Stride (S), and Amount of Zero-padding (P).

Output volume is determined as follows:

$$\mathbf{W2} = \frac{W1 - F + 2P}{S} + 1$$

$$\mathbf{H2} = \frac{H1 - F + 2P}{S} + 1$$

$$\mathbf{D2} = K$$

Pooling Layer Design

Pooling Layer takes input as volume [W_1xH_1xD_1], and outputs another volume as [W_2xH_2xD_2]. It requires two hyper-parameters: Spatial extent (F) and amount of Stride (S)

Output volume is determined as follows:

$$\mathbf{W2} = \frac{W1 - F}{S} + 1$$

$$\mathbf{H2} = \frac{H1 - F}{S} + 1$$

$$\mathbf{D2} = D1$$

Example of Simple CNN Architecture

[INPUT—CONV—RELU—POOL—FC]

As already mentioned, RELU and POOL layers are fixed function but CONV and FC have transformation function whose value depends upon set of parameters (weights and biases). These values are learned through gradient descent.

3.1.3 Learning a Classifier

Log likelihood of a classifier can be defined as:

$$\mathcal{L}(\theta, \mathcal{D}) = \sum_{i=0}^{|\mathcal{D}|} \log P(Y = y^{(i)} | x^{(i)}, \theta)$$

Log likelihood of a classifier gives measure of correct classification. Increasing value of log-likelihood is better while training. Since it is more generalize to have minimization of objective function, we define Negative Log-Likelihood (NLL) as:

$$NLL(\theta, \mathcal{D}) = -\sum_{i=0}^{|\mathcal{D}|} \log P(Y = y^{(i)} | x^{(i)}, \theta)$$

Where $\mathcal{D}$ is Dataset

θ is weight parameter

$(x^{(i)}, y^{(i)})$ is ith training data. Y is target data.

NLL of a classifier is differential and it can be used by our model as cost function. Our model is trained with algorithm that minimizes the cost function over the training data. The algorithm we are going to use is Stochastic Gradient Descent with Mini-batches

Gradient Descent

A cost function is defined first. For system or hypothesis to work on the basis of examples, its cost function should be minimum so that it gives least error. Weights are initialized to some random values and iterating; we obtain value of weights that minimizes cost function. It is done by gradient calculation. Back propagation is applied to calculate gradient descent.

Stochastic Gradient Descent (SGD)

A loss function is defined by its parameters. In ordinary gradient descent algorithm, at each iteration, we move downward to error surface defined by the loss function. SGD works on same principle as gradient descent but it moves more quickly downward. Unlike gradient descent where gradients are calculated after running on entire dataset, SGD calculates over few examples at a time [Single data at purest form].

There is variant of SGD called Minibatch SGD (MSGD) in which training examples are divided into multiple batches and gradients are calculated after each batch.

MSGD Algorithm:

For every batch

i. Calculate cost function
ii. Calculate gradient
iii. Update parameters
iv. Until stopping condition

A pseudo-code of MSGD more clarifies the algorithm:

```
for (x_batch, y_batch) in mini_batches:
    loss = f(params, x_batch, y_batch)
    gradient_wrt_params=  #Calculate Gradiet
    params -=learning_rate*gradient_wrt_params
    if <stopping condition is met>:
        return params
```

3.1.4 Testing and Evaluation

A typical train and test scheme is: learn parameters from training set, minimize cost function, and compute test error.

Loss function is calculated for each batch. So, it is first quantity to track the training.

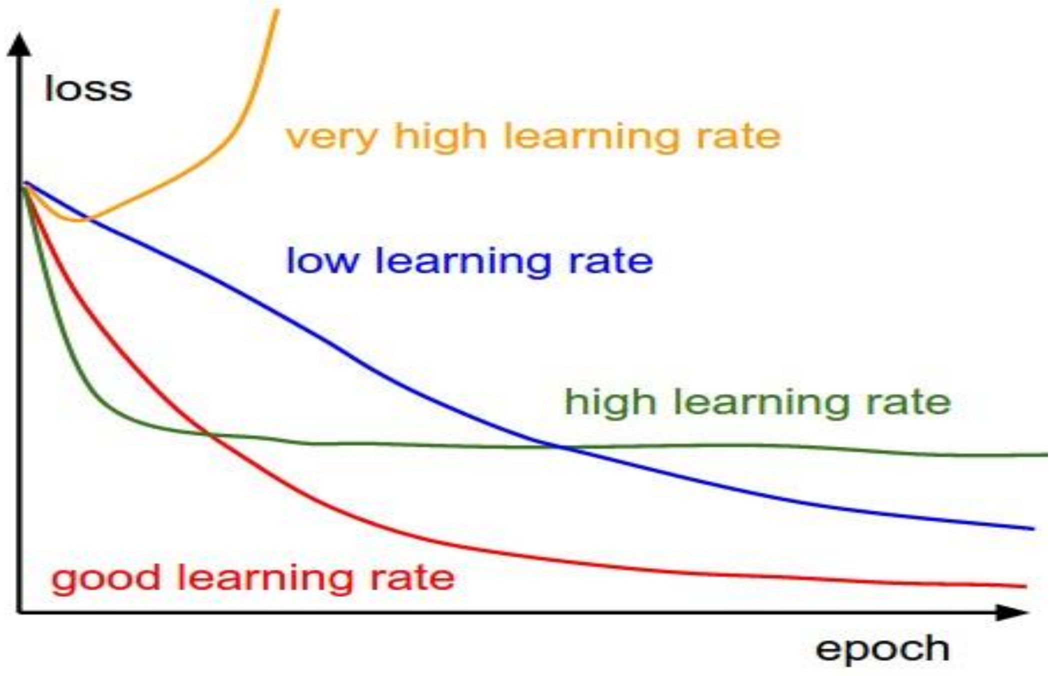


Figure 12:-Loss function with various learning rates

Figure above shows loss function over time with different learning rates. When learning, rate is very high it cannot converge, it overshoots the minima and hence the training loss increases with time.

If there is low learning rate, then the model learns at very slow rate. Thus, it takes much longer time to converge. Choice of learning rate is a tricky and it has to be in a way such that it should not be large enough to overshoot the minima and also should not be very small to get longer time to converge.

A good choice of learning rate give smooth decrease in loss over time as shown in figure above.

Another quantity to track training is Validation/Training Accuracy.

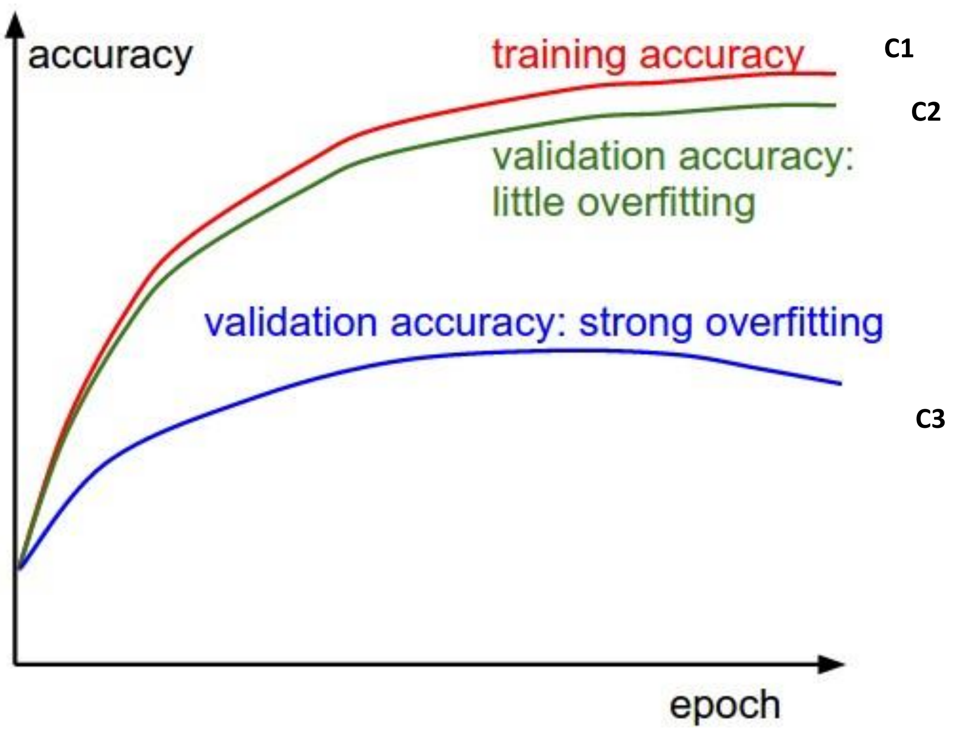


Figure 13:- Validation/Training Accuracy and over-fitting

The gap between training accuracy and validation accuracy clearly suggests that there is over-fitting. Curve C1 represents training accuracy. If the model gives validation accuracy represented by Curve C2 then there is little over-fitting and model works good for not just training examples but to new examples as well. But, in case, the model gives validation accuracy as depicted in Curve C3, then it has strong over-fitting, thus giving worse result for new unseen examples.

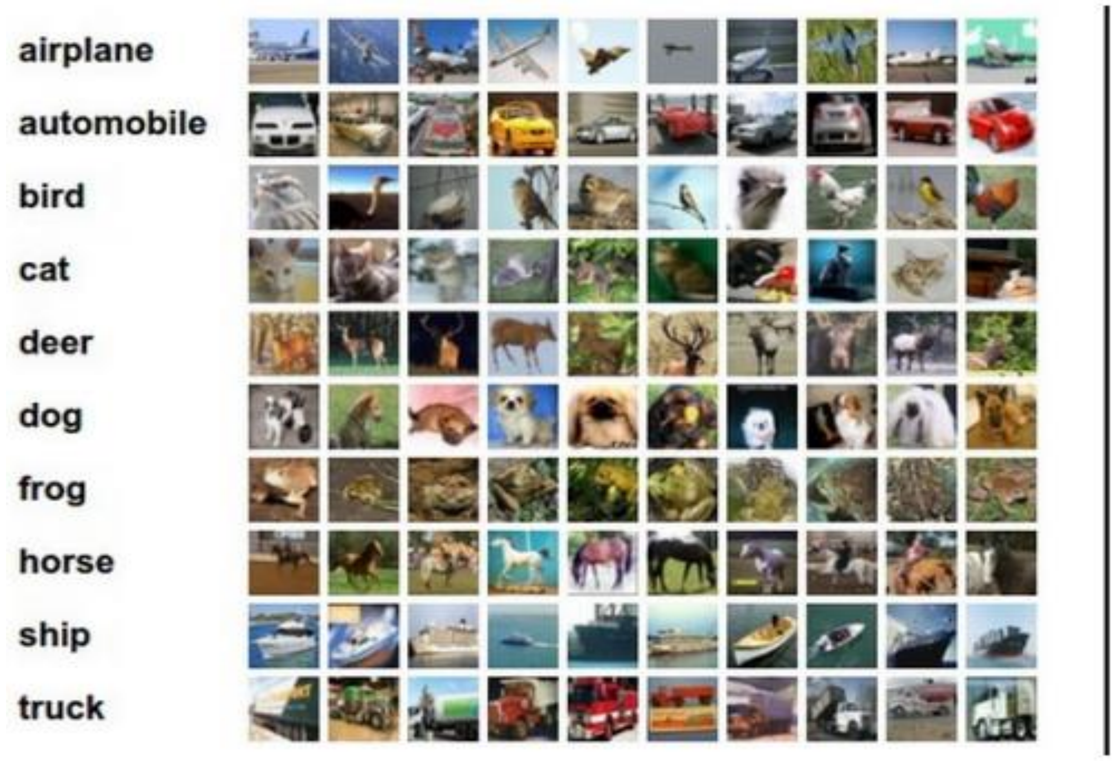


Figure 14: - Images in CIFAR-10 Dataset

Evaluation

The model can be evaluated with multiple measures:

i. Test loss/Test Accuracy
ii. Validation loss/Validation Accuracy
iii. Precision, Recall

A typical train and test scheme is: learn parameters from training set, minimize cost function, and compute test error.

3.2 Tools and IDE

- Python with Sci-kit, numpy, scipy, Tensor flow
- Core i5

3.3 Data Collection and Evaluation

For object detection CIFAR-10 and MNIST Dataset are be used. CIFAR-10 consists of 60K images of size 32x32. 50K images are used for Training while 10K are Test image.There are 6000 images per class. Different classes of CIFAR-10 Dataset are:

Label	class
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

Table 3.3.1: - CIFAR-10 Dataset Classes

MNIST Dataset also contains 60K grayscale images of hand-written digits, each of size 28x28. 50K images are used for Training and 10K are used for Test images.

Label	Digit
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Table 3.3.2: -MNIST Dataset Classes

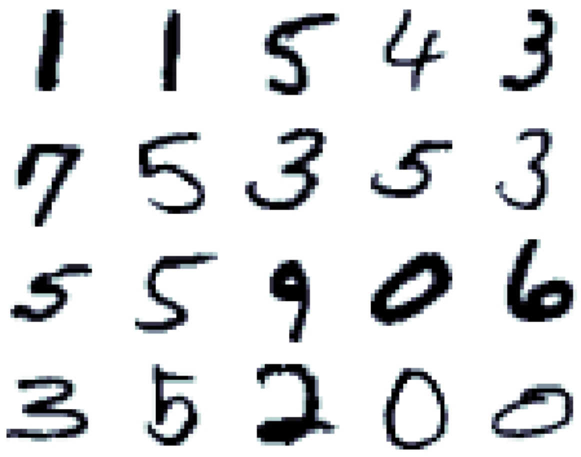


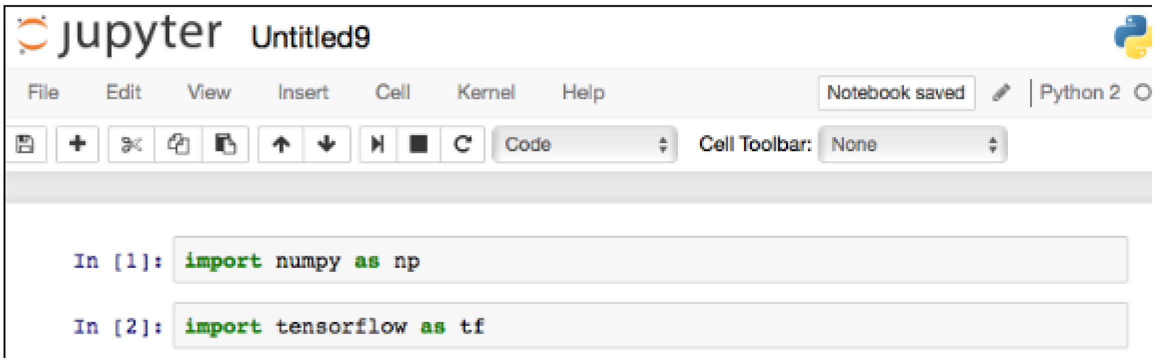
Figure 15: -Handwritten digits in MNIST Dataset

Chapter 4: Results and Discussion

4.1 Overview of Tasks

4.1.1 Deep Learning Framework Installation and CPU Configuration

A very effective deep learning framework ‘Tensor Flow’ is installed and is configured to run on Core i5.



```
In [1]: import numpy as np

In [2]: import tensorflow as tf
```

Figure 16 : - Importing tensorflow Library on IPython Notebook showing CPU enabled sign.

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture lets us deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. A deep learning framework ‘Lasagne’ along with ‘Nolearn’ are used for efficient implementation.

4.1.2 CNN Architectures

Different CNN architectures are designed. As image size of both MNIST and CIFAR-10 are 28x28 and 32x32 sizes respectively, more than more than three layers of convolution are not possible without sufficient pooling. Following each convolution with pooling layer, three architectures are proposed.

CNN for MNIST Dataset

A Convolutional Neural Network is designed for both MNIST and CIFAR-10 Datasets. An architecture is designed for MNIST Dataset with following layers’ configuration:

i. INPUT Layer accepting input of 28x28 image with single channel color

ii. First CONV Layer with 32 5x5 filters. With RELU activation function.

iii. MAXPOOL Layer with size 2x2.

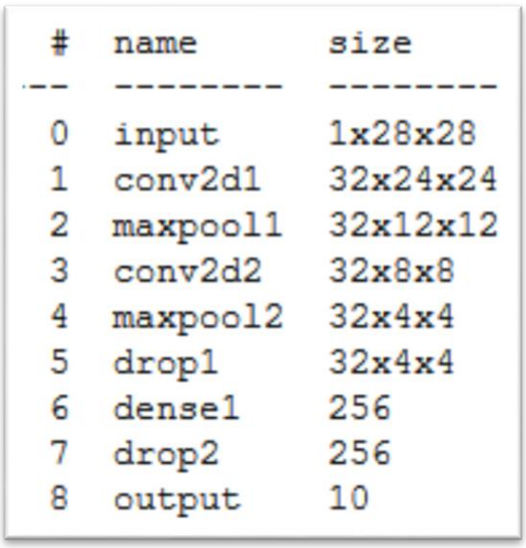
iv. Second CONV Layer with 32 5x5 filters. With RELU activation function.

v. Second MAXPOOL Layer with size 2x2

vi. Dropout Layer with 50% dropout.

vii. FULL Connection Layer with 256 units Followed by another 50% dropout layer.

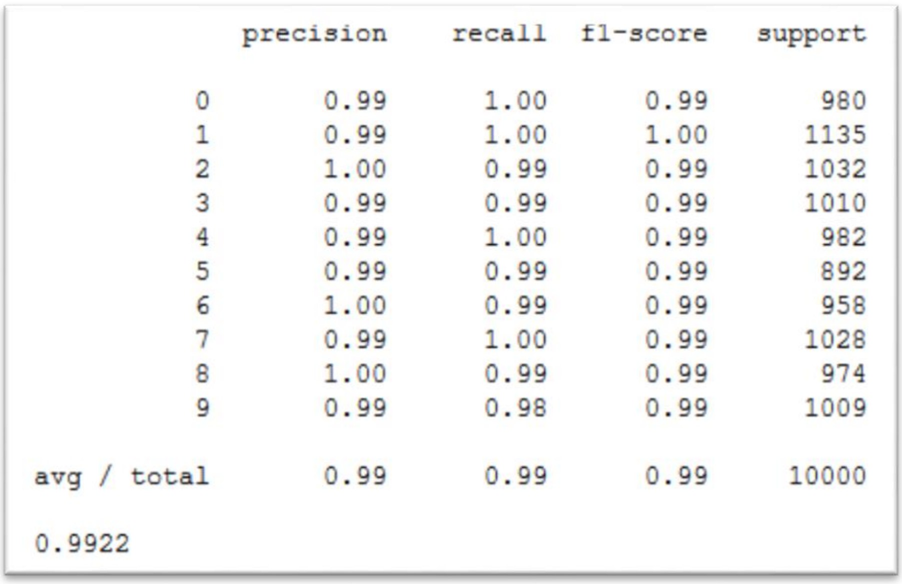
viii. OUTPUT Layer with 10 units.



```
 #  name      size
--  --------  --------
 0  input     1x28x28
 1  conv2d1   32x24x24
 2  maxpool1  32x12x12
 3  conv2d2   32x8x8
 4  maxpool2  32x4x4
 5  drop1     32x4x4
 6  dense1    256
 7  drop2     256
 8  output    10
```

Figure 17: - CNN Architecture for MNISTDataset

CNN after training over 10 epochs gives 98.70% accuracy.



```
             precision    recall  f1-score   support

          0       0.99      1.00      0.99       980
          1       0.99      1.00      1.00      1135
          2       1.00      0.99      0.99      1032
          3       0.99      0.99      0.99      1010
          4       0.99      1.00      0.99       982
          5       0.99      0.99      0.99       892
          6       1.00      0.99      0.99       958
          7       0.99      1.00      0.99      1028
          8       1.00      0.99      0.99       974
          9       0.99      0.98      0.99      1009

avg / total       0.99      0.99      0.99     10000

0.9922
```

Figure 18: - Precision, Recall, f1-score and Support for CNN after 10 epochs

This information can be viewed on Confusion Matrix can be obtained with above results. Obviously results are fascinating with just 10 epochs. If we train model for more epochs, then the output may be around 99% accuracy. This means this architecture is working perfectly for MNIST Dataset.

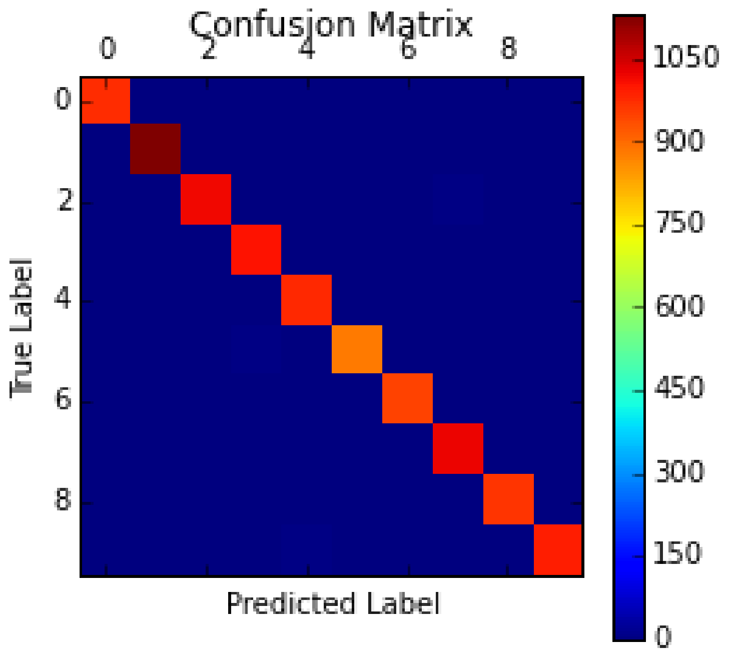
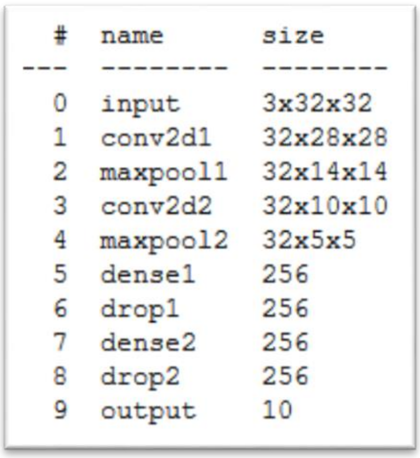


Figure 19: - Confusion Matrix for CNN designed for MNIST Dataset

CNN Architecture-1 for CIFAR-10 Dataset



```
  #  name       size
---  --------   --------
  0  input      3x32x32
  1  conv2d1    32x28x28
  2  maxpool1   32x14x14
  3  conv2d2    32x10x10
  4  maxpool2   32x5x5
  5  dense1     256
  6  drop1      256
  7  dense2     256
  8  drop2      256
  9  output     10
```

Figure 20:- CNN Architecture-1 for CIFAR-10 Dataset

Different Layers in CNN are:

i. Input layer is 32x32 images of 3 color channel i.e 3x32x32. Convolution of layer consists of 20 5x5 patches with padding size 2 and stride equal to 1.
ii. Pooling layer after first convolution consists of max function with 2x2 patch size. Hence it will reduce the image to 32x14x14.

iii. Second convolution layer consists of same properties as that of first convolution layer and the image it produces will be of size 32x10x10 followed by pooling layer that reduces image to 32x5x5
iv. Two full connection layer each of 256 units and with 50 % dropout is implemented.
v. Final Output layer with 10 outputs for different classes.

CNN Architecture-1 is trained for 34 Iterations.

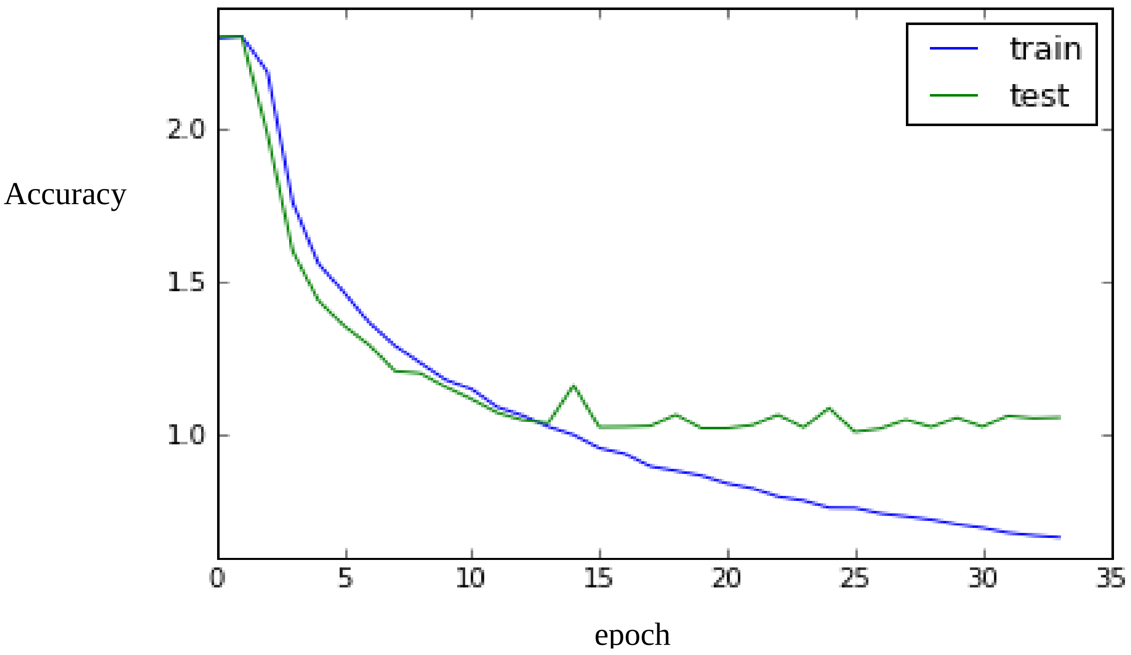
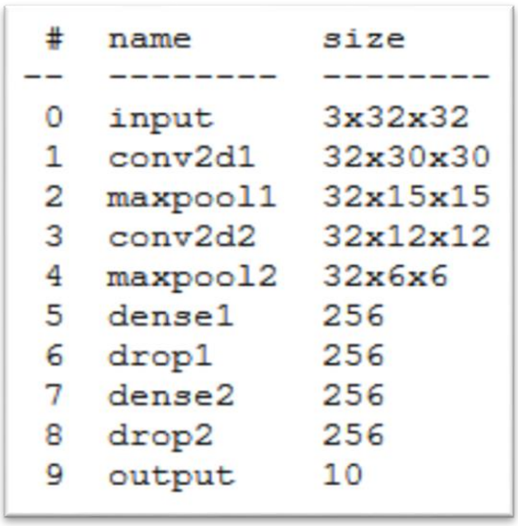


Figure 21: - CNN Architecture Train and Test Loss

CNN Architecture-2 for CIFAR-10 Dataset



#	name	size
0	input	3x32x32
1	conv2d1	32x30x30
2	maxpool1	32x15x15
3	conv2d2	32x12x12
4	maxpool2	32x6x6
5	dense1	256
6	drop1	256
7	dense2	256
8	drop2	256
9	output	10

Figure 22: - Layer Information for CNN Architecture-2

The figure above shows architecture-2. This consists of small two 3x3 convolutions layers, two full connection layers with dropout 50%. In architecture-1, we have used (5x5) convolution filters.

The CNN is trained for 40 epochs.

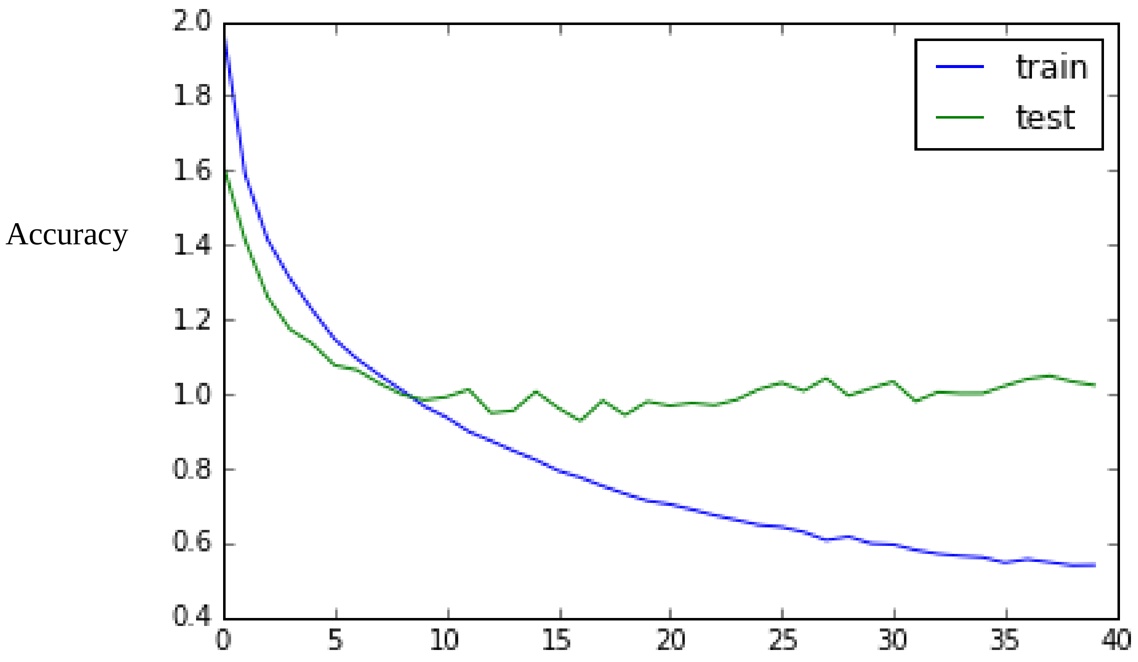
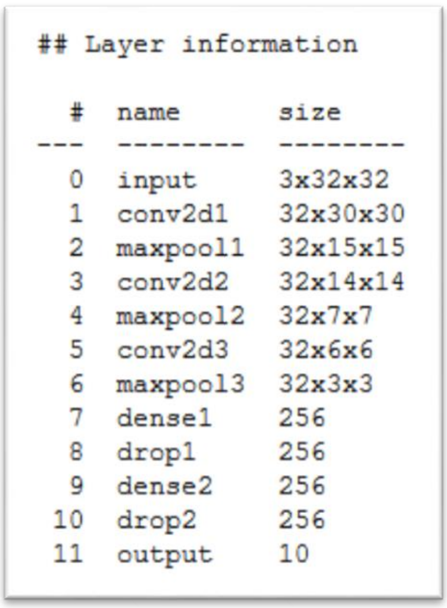


Figure 23: - Architecture trained for 40 epochs

CNN Architecture-3 for CIFAR Dataset



```
## Layer information

  #  name       size
---  ---------  ---------
  0  input      3x32x32
  1  conv2d1    32x30x30
  2  maxpool1   32x15x15
  3  conv2d2    32x14x14
  4  maxpool2   32x7x7
  5  conv2d3    32x6x6
  6  maxpool3   32x3x3
  7  dense1     256
  8  drop1      256
  9  dense2     256
 10  drop2      256
 11  output     10
```

Figure 24:- Layer Information for CNN Architcture-3 for CIFAR-10 Dataset

This Arhitecture-3 is different from Architecture 1 and 2 above. It consists of 3 convolutions layers of different sizes. First convolution layer is of size (3, 3), Second and Third Convolution Layers are of size (2, 2). Using such small patches and deeper network allows to better model the problem.

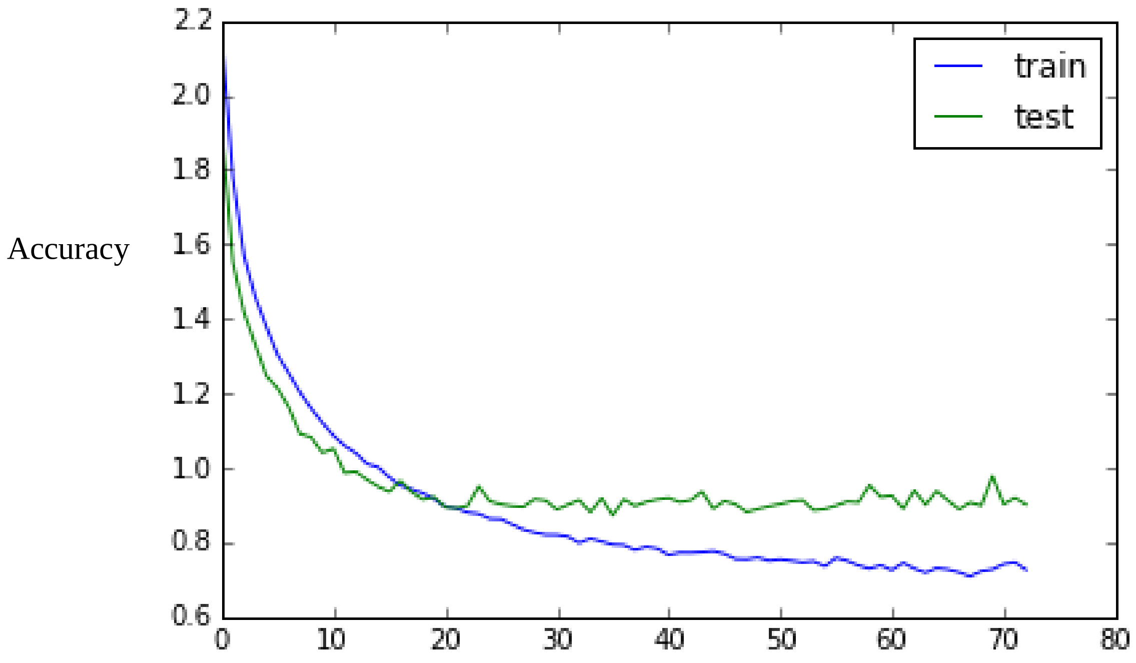


Figure 25: - CNN Architecture-3 Trained for 73 epochs

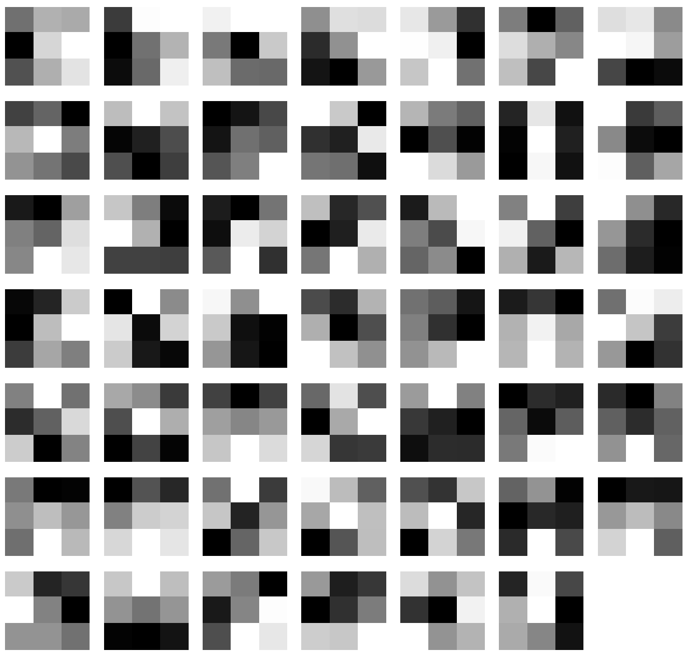


Figure 26:- Convolution Filter of first convolution layer of Architecture 3

4.2 Final Results

	Architecture 1	**Architecture 2**	**Architecture 3**
Maximum Accuracy (%)	65.526	66.564	71.025
Epochs to obtain Max. Accuracy	27	31	48
Average Training time(seconds)	72.04	70.13	49.84

Table 1:- Comparison of Three Architectures

Results obtained from all three architectures are listed above. The first architecture, having two convolution layers each with filter size 5x5 and two dropout layers with 50% dropoutgave 65.526% accuracy on 27th epoch. After that, it began to over-fit the model.

In Second Architecture, We have used two convolution layers but with small filter size than that of Architecture 1. Each of them are of 3x3 size. With just this arrangement, the accuracy improved to 66.564%. Again after 31st epochs the model begins to over-fit.

Third Architecture exploits 3 convolution layers, the first convolution layer is of size 3x3, second and third convolution are of sizes 2x2 each. After three sequence of convolution and pooling, just like other architectures, two dropout layers along with two fully connected layer are designed before the final layer. On doing so the model gave the accuracy of 71.025 % which is quite good than other two architectures.

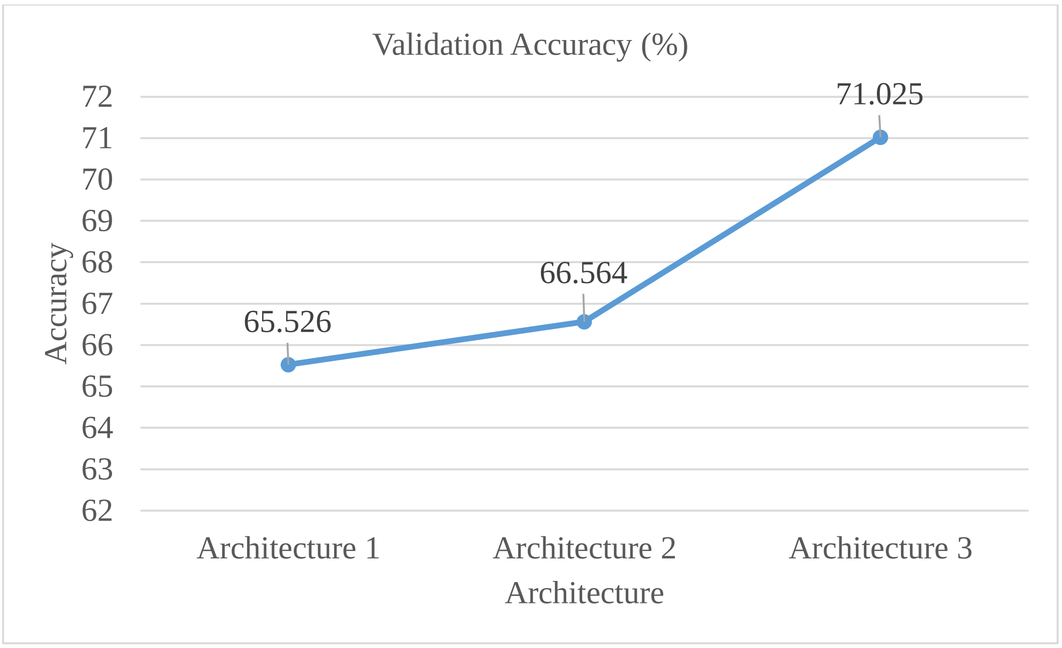
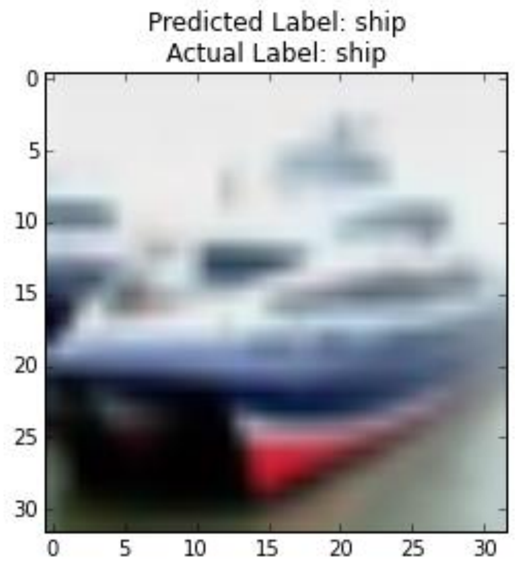
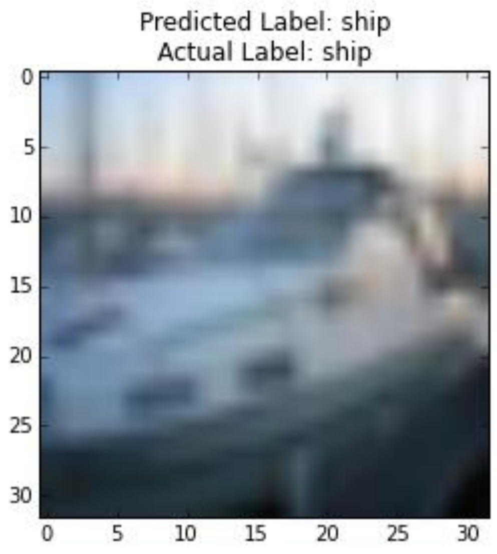


Figure 27:- Accuracy of different architectures

The figure 28 shows the classification results obtained from Architecture-3.

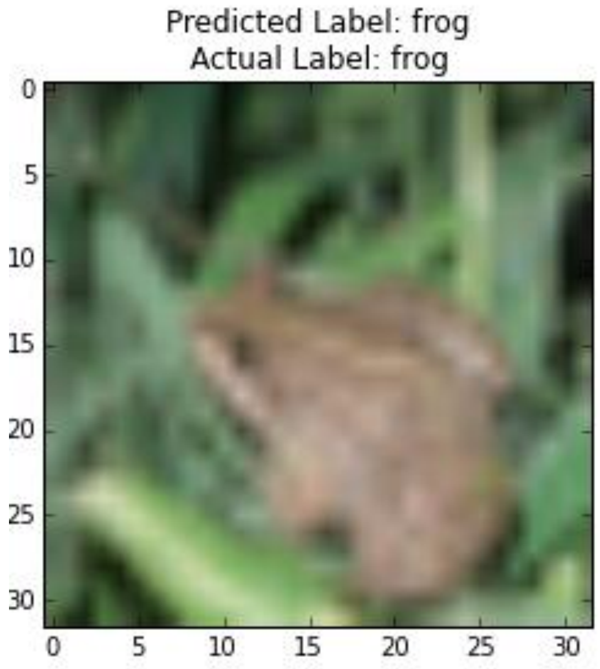


(a)

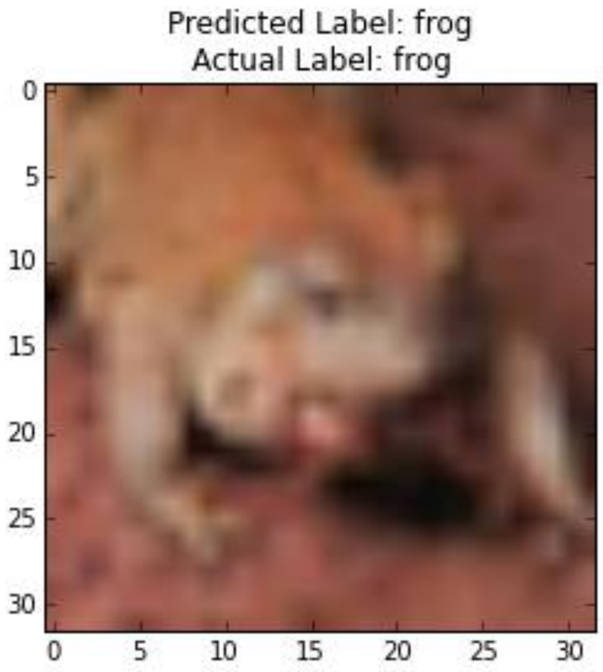


(b)

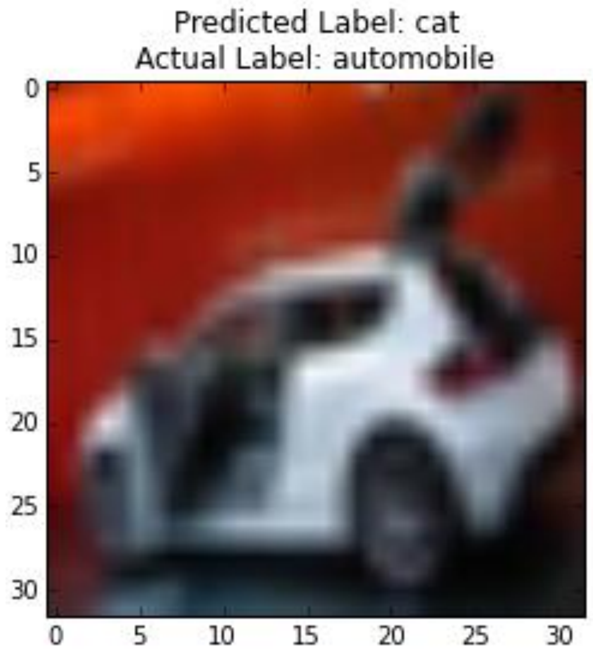
(c)



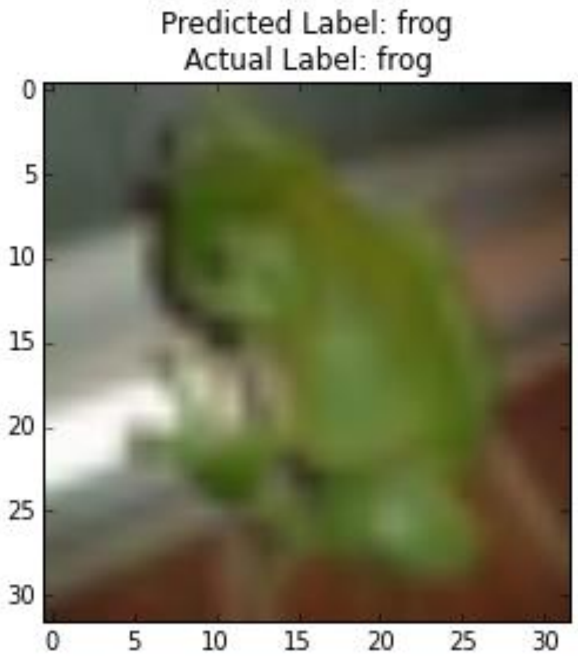
(d)


Predicted Label: frog
Actual Label: frog

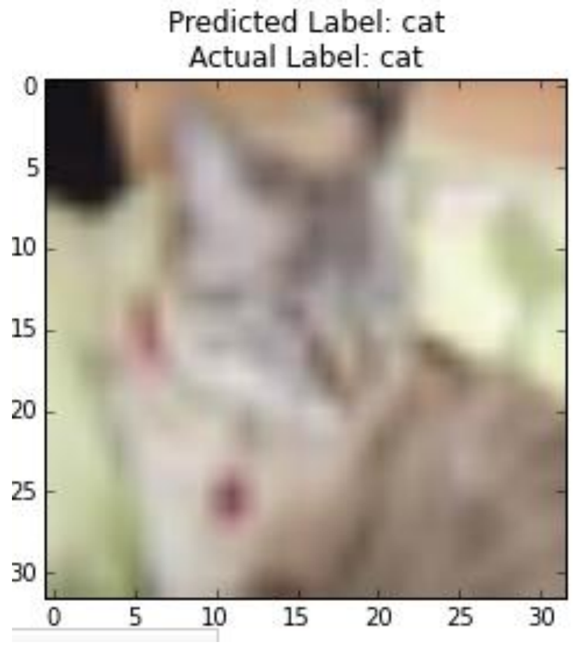
(e)


Predicted Label: cat
Actual Label: automobile

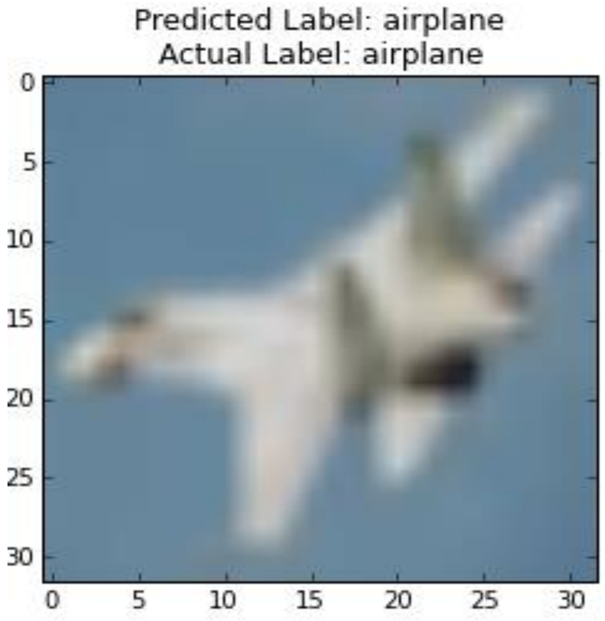
(f)


Predicted Label: frog
Actual Label: frog

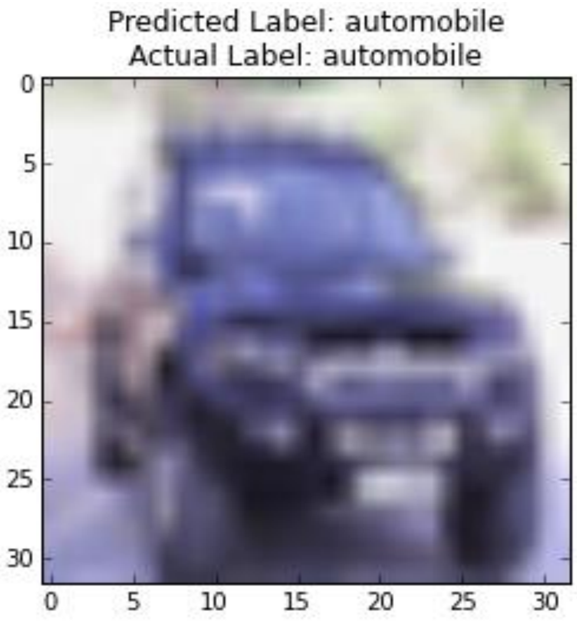
(g)


Predicted Label: cat
Actual Label: cat

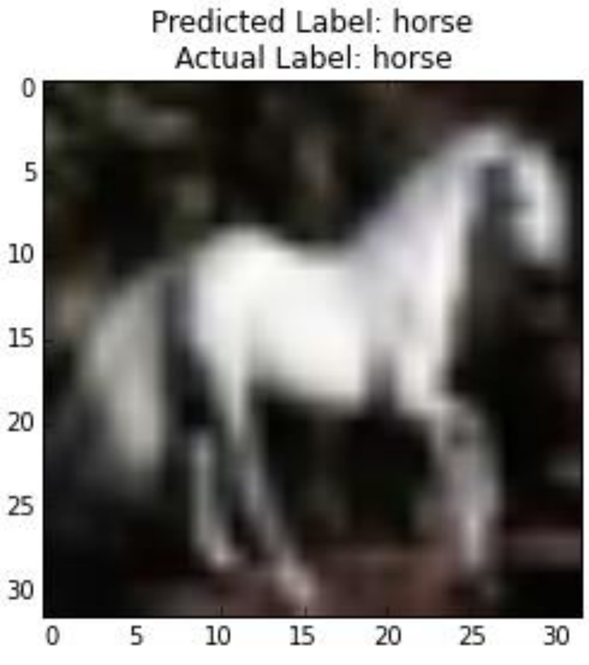
(h)



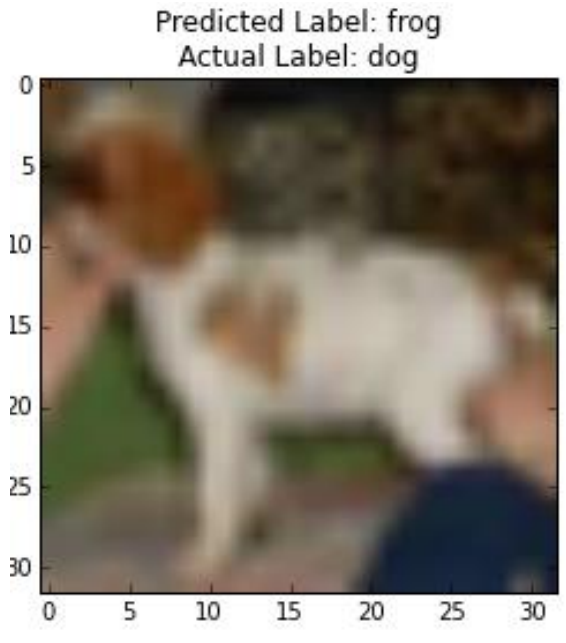
(i)



(j)



(k)



(l)

Figure 28:- Results obtained from architecture-3 over test dataset

Chapter 5: Conclusion

This thesis report presented suitable architecture of convolutional neural network for solving image classification problem. Machine learning technology has been used to solve the problem instead of using advance image processing techniques. The model of Deep Neural Network architecture is developed in which deep layer extracts information from data at different level of abstractions. Convolutional Neural Network is trained and the images are classified. General idea is to use constant filter sizes for convolution layers. New approach discussed on this research is of using variable filter sizes for extracting convolved features. The filters must be of small size of about 3x3 or 2x2 for obtaining smooth train and valid loss.

For both MNIST and CIFAR-10 datasets, the three architectures are developed. The architecture designed for one dataset did not work well with other. It depends upon image properties like size, channels etc. To better train CNN, overall procedure is to use small patches of convolution filters instead of bigger one. It will be better of using variable sized filters instead of constant filters among all convolution layers. This thesis showed that using some slightly bigger size filter at initial convolution layers and then using smaller filters long the depth gives good accuracy and hence the efficient architecture.

In order for reducing over-fitting, dropout must be used. Dropout significantly improves performance of the networks, as it performs model averaging of different networks. It must be used in full connection layers. One or two layers of dropout must be implemented.

Chapter 6: Limitations and Future Enhancement

As with every research, the Architectures developed in this thesis have some limitations. Since it is very expensive to calculate convolution operations CPUs needs to train the architectures. CPUs can exploit their parallelism. High configuration CPUs are required to perform these calculations so that training can be done in small time.

Neural Networks with 2 or 3 layers of convolution performs well for CIFAR-10 and MNIST Dataset but when dealing with larger datasets containing larger image sizes, it won't be enough with just 2 or 3 layers. Networks should contain deeper architectures.

The architecture developed for MNIST Dataset gave 98 % of accuracy over test dataset with only 10 epochs. For CIFAR-10 dataset, accuracy did not do so well. Architectures depend upon size of images which is a quite disappointment.

The accuracy obtained was around 68%. Further processes can be done to improve accuracy of the Architectures. One of them is Data Augmentation. It can be used further to minimize over fitting. Basically what it does is by performing transformations over images, it produces much larger dataset to train the network. Deeper Architectures with more convolution layers can be used to better model the problems.

References

[1] C. Eugenio, A. Dundar, J. Jin and J. Bates, “An Analysis of the Connections Between Layers of Deep Neural Networks,” arXiv, 2013.

[2] T. N. Sainath, B. Kingsbury, A.-r. Mohamed and B. Ramabhadran, “Learning Filter Banks within a Deep Neural Network Framework,” in *IEEE*, 2013.

[3] A. Graves, A.-r. Mohamed and G. Hinton, “Speech Recognition with Deep Recurrent Neural Networks,” University of Toronto.

[4] A. Graves, “Generating Sequences with Recurrent Neural Networks,” arXiv, 2014.

[5] O. Vinyals and Q. V.Le, “A Neural Conversational Model,” arXiv, 2015.

[6] R. Grishick, J. Donahue, T. Darrel and J. MAlik, “Rich Features Hierarchies for accurate object detection and semantic segmentation.,” UC Berkeley.

[7] A. Karpathy, “CS231n Convolutional Neural Networks for Visual Recognition,” Stanford University, [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[8] I. Sutskever, “Training Recurrent Neural Networks,” University of Toronto, 2013.

[9] “Convolutional Neural Networks (LeNet),” [Online]. Available: http://deeplearning.net/tutorial/lenet.html.

[10] K. Alex, S. Ilya and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” 2012.

[11] Zeiler, M. D and F. Rob, “Visualizing and Understanding Convolutional Networks,” arXiv, 2013.

[12] L. Fie-Fie and A. Karpathy, “Deep Visual Alignment for Generating Image Descriptions,” Standford University, 2014.

[13] O. Vinyals, A. Toshev, S. Bengio and D. Erthan, “Show and Tell: A Neural Image Caption Generator.,” Google Inc., 2014.

[14] I. Sutskever, J. Martens and G. Hinton, “Generating Text with Recurrent Neural Networks,” in *28th International Conference on Machine Learning*, Bellevue, 2011.

[15] J. Martens, “Deep Learning via Hessian-Free Optimization,” in *Procedings of 27th International Conference on Machine Learning*, 2010.

[16] Nielsen and A. Michael, “Neural Networks and Deep Learning,” Determination Press, 2014.