TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

CENTRAL CAMPUS, PULCHOWK

THESIS NO.: 070-MSCS-670

EFFICIENT CONVOLUTIONAL NEURAL NETWORK FOR IMAGE CLASSIFICATION

BY

YOGENDRA TAMANG

A THESIS

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

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

NOVEMBER 2015

Efficient Convolutional Neural Network for Image Classification

By

Yogendra Tamang

(070/MSCS/670)

Thesis Supervisor:

Dr. Sashidhar Ram Joshi

Professor

A thesis submitted in partial fulfillment of the requirement for

the degree of Master of Science in Computer System and Knowledge Engineering.

Department of Electronics and Computer Engineering

Institute of Engineering, Central Campus, Pulchowk

Tribhuvan University

Lalitpur, Nepal

November, 2015

COPYRIGHT

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

Ananda Niketan, Pulchowk, Lalitpur, P.O. Box 1175, Kathmandu, Nepal.
Tel : 5534070, 5521260 extn. 315, Fax : 977-1-5553946, E-mail : doece@ioe.edu.np

Our Ref :

DEPARTMENTAL ACCEPTANCE

The thesis entitled "Efficient Convolutional Neural Network Architecture for Image Classification", submitted by Yogendra Tamang in partial fulfillment of the requirement for the award of the degree of "Master of Science in Computer System and Knowledge Engineering" has been accepted as a bona fide record of work independently carried out by him in the department.

Dr. Dibakar Raj Pant
Head of the Department

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

Ananda Niketan, Pulchowk, Lalitpur, P.O. Box 1175, Kathmandu, Nepal.
Tel : 5534070, 5521260 extn. 315, Fax : 977-1-5553946, E-mail : doece@ioe.edu.np

Our Ref :

CERTIFICATE OF APPROVAL

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a thesis report entitled "Efficient Convolutional Neural Network Architecture for Image Classification" submitted by Mr. Yogendra Tamang in partial fulfillment of the requirement for the degree of Master of Science in Computer System and Knowledge Engineering.

Supervisor, Dr. Sashidhar Ram Joshi
Professor
Department of Electronics and Computer Engineering

External Examiner, Mr. Om Bikram Thapa
Chief Technical Officer
Vianet Communications Pvt. Ltd.

Committee Chairperson, Dr. Sashidhar Ram Joshi
Professor
Department of Electronics and Computer Engineering

DATE OF APPROVAL: November 6, 2015

Acknowledgement

I would like to express my sincere thanks to Department of Electronics and Computer Engineering (DOECE) for providing me the opportunity to explore my interest and ideas in the field of engineering through this thesis.

I owe a debt of gratitude towards my Thesis supervisor, **Prof. Dr. Sashidhar Ram Joshi**, for his proper guidance, encouragement and support.

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

Yogendra Tamang (070/MSCS/670)

Abstract

Problem of object recognition and image classification cannot be only solved by image processing techniques but can also by machine learning technology. This thesis is about classifying images using machine learning techniques. Artificial Neural Networks are be deployed to work with this problem. One of deep learning architectures, Convolutional Neural Network (CNN) is specifically used to learn the features of images. To design such CNN, different parameters are taken into account. These parameters may be filter size, number of convolution layers, Dropout layers etc. By careful choosing and study of these parameters, somehow efficient architecture is designed. On this thesis, Different neural network architectures for CIFAR-10 and MNIST dataset are developed. These networks are different in terms of number of hidden layers, filter sizes and other measures. They are trained on Graphical Processing Unit (GPU). Dropout technique is used for reducing over-fitting issues. Networks are then fed with testing data and the accuracy of these architectures are calculated. The research is run by continuous evaluation of accuracy of different architectures. Finally results are compared and analysed to find out best architecture thus giving a way to design efficient architecture for image classification.

Table of Contents

Chapter

List of Figures

List of Tables

List of Abbreviations

AI	=	Artificial Intelligence
ANN	=	Artificial Neural Network
CNN	=	Convolutional Neural Network
CONV.	=	Convolutional Layer
DNN	=	Deep Neural Network
FC	=	Full Connection Layer
MSGD	=	Mini-batch Stochastic Gradient Descent
NLL	=	Negative Log Likelihood
POOL.	=	Pooling Layer
RELU	=	Rectified Linear Unit
RNN	=	Recurrent Neural Network
SGD	=	Stochastic Gradient Descent

Chapter 1

1. Introduction

1.1 Background

In this modern era, Computers are being 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 modelling etc. But now a days, situation is inverted. Machines are capable of doing these things more easily with very much high accuracy.

Usual 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.

Neural Network is one of AI techniques emerged long ago in 1940s but technology at that time was not so advanced. It was then waken 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 [1] 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 most common type of machine learning in which a model 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.

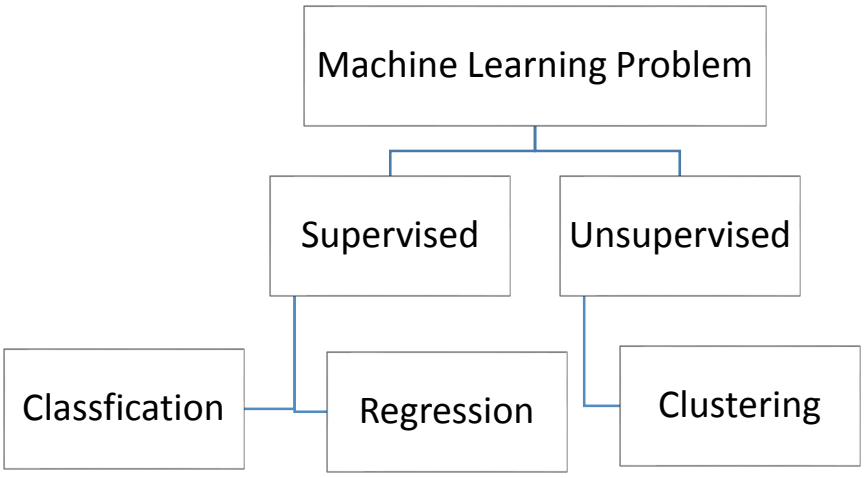


Figure 1:- Machine Learning Overview

Image Classification

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.

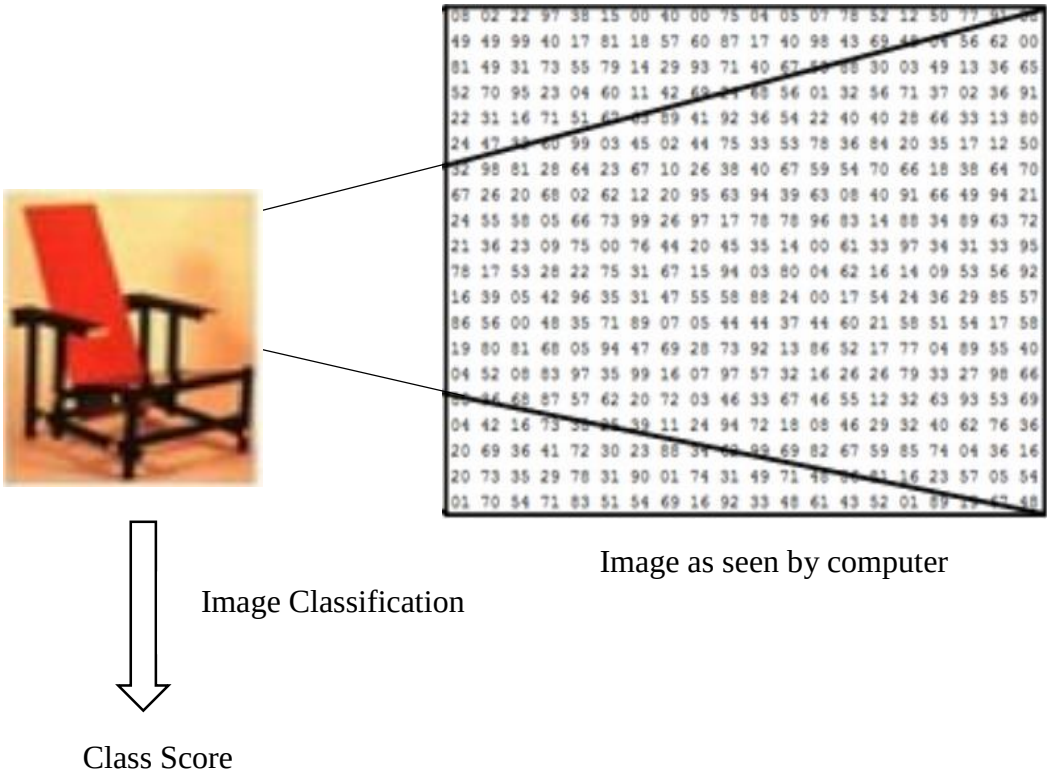


Figure 2:- Image Classification Problem

1.2 Problem Statement

DNNs are applicable to fields like image recognition [1], Speech Recognition [2, 3], text prediction and handwriting generation [4], Language Generations [5]. DNN architecture for one task does not work well with other 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.

1.3 Objectives

The Prime objectives of this thesis are

- To classify images using CNN
- To design effective architecture of CNN for image classification

Chapter 2

2. Literature Review

2.1 Neural Networks

Neural Networks (NNs) are biologically inspired connectionist models that receive some input, transform it through series of hidden layers and finally calculate the output. In regular NN, neuron in each layer is connected to all neurons in previous layers. Each neuron on a networks works just like biological neurons.

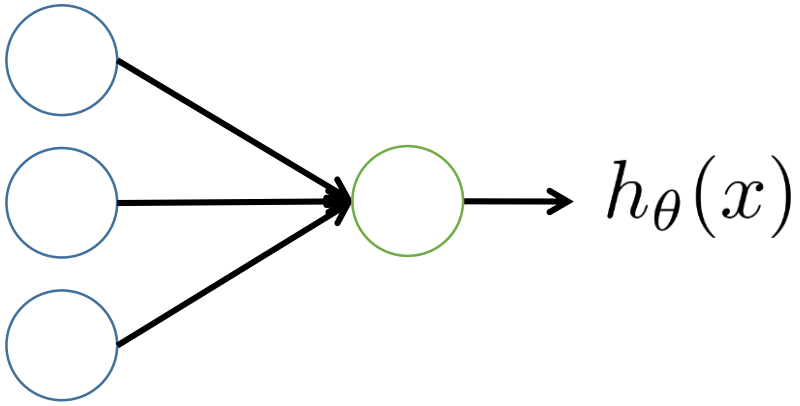


Figure 3:- An Artificial Neural Network

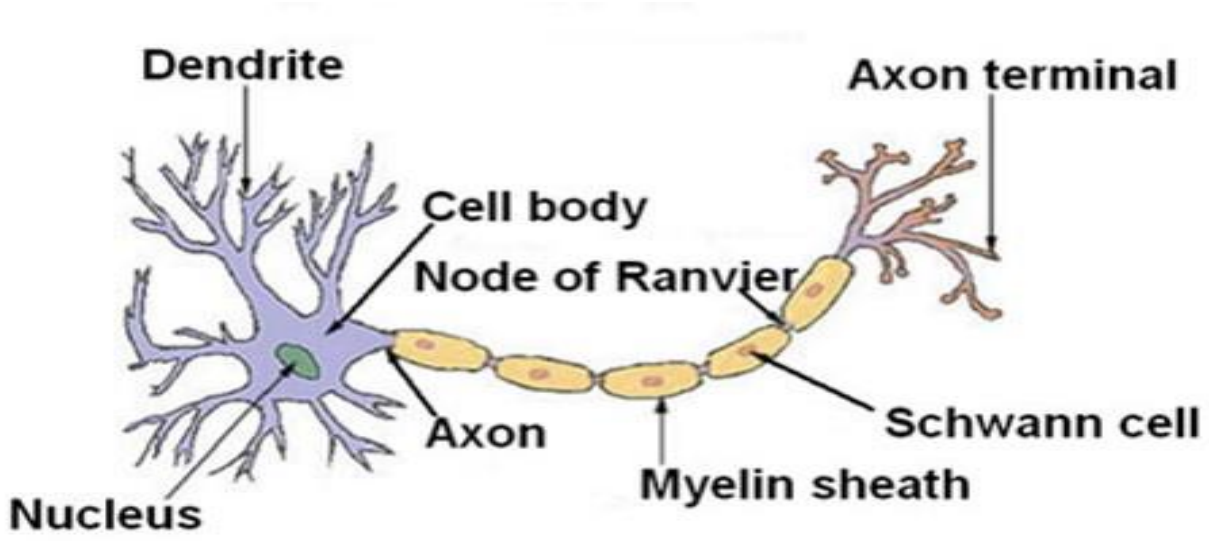


Figure 4:- A biological neuron example

A single neuron connects to other 10^4 to 10^6 neurons thus forming massive parallel structure. Two neurons are connected by Axon. The connection between two neurons strengthens or weakens depending upon how they work together. They simply follow the principle "The neurons that fire together, wire together".

Multilayer Perceptron (MLPs)

Each neurons are independent and do not share connection among them. Problem with these types of NNs is they cannot scale well [6]. 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].

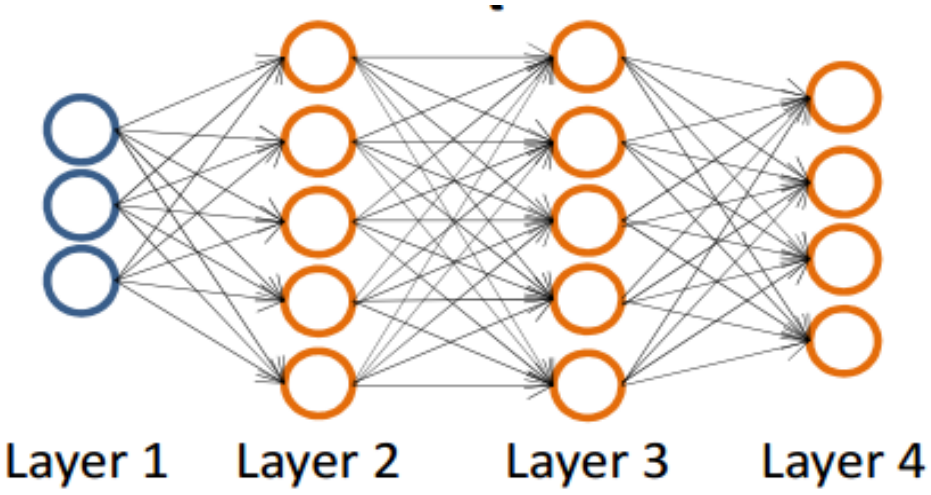


Figure 5:- Multilayer Perceptrons with 4 layers.

2.2 Deep Neural Network (DNNs)

DNNs are very popular Neural Network Architecture and are used extensively used in computer vision, Speech Recognition, Language modelling 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 [3, 4, 8].

2.3 Convolutional Neural Networks (CNNs)

Layers of CNN are arranged in 3 dimensions: width, height and depth. A single image from CIFAR-10 is a volume of dimensions 32x32x3. Convolutional Neural Networks are biologically inspired variants of MLPs. Visual Cortex contains small arrangement of cells. These cells are sensitive to small sub-regions of visual field and are known as Receptive Fields. The Sub-Fields are stacked and tiled together thus forming the entire visual field. These cells most closely behave as filters over the input space and are well suited to exploit the strong spatially local correlation present in natural images.

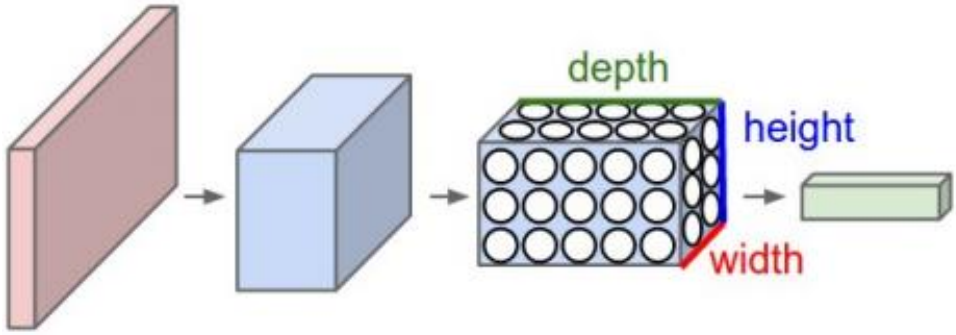


Figure 6:- CNN with its neurons arranged in three dimensions

CNN has been proved to be 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.

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].

Convolution 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]

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

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. It is nonlinear down-sampling of image.

Most common type of pooling layer is Max Pooling. Max-pooling partitions the input image into subset of non-overlapping rectangles and for each sub-region, outputs the maximum value. [9]

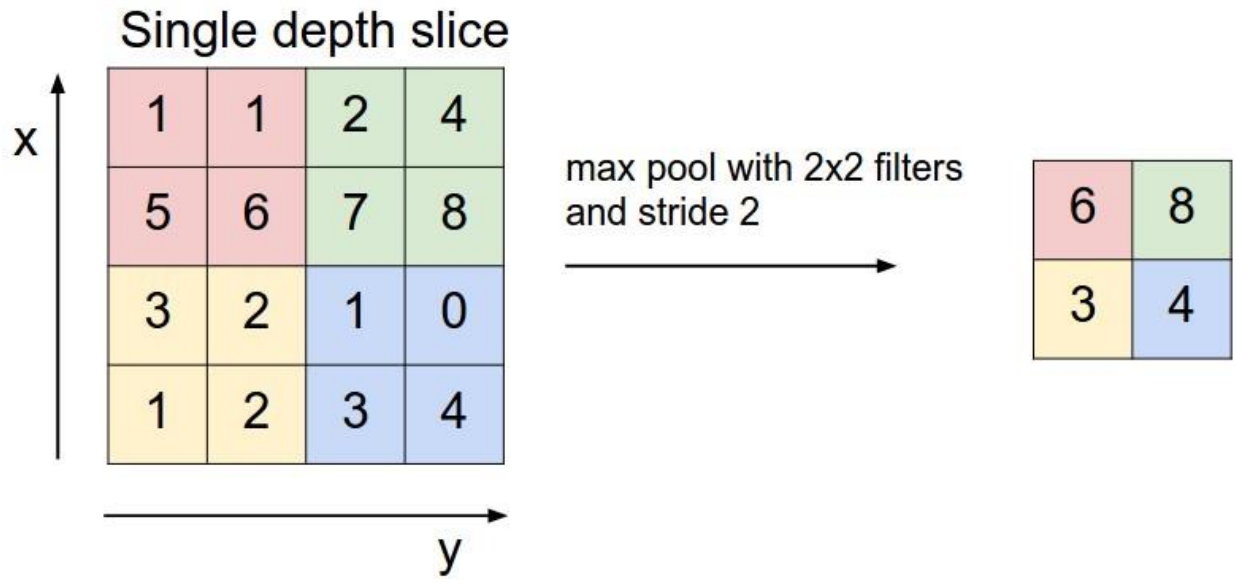


Figure 7:- Example of Max-pooling

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 below 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.

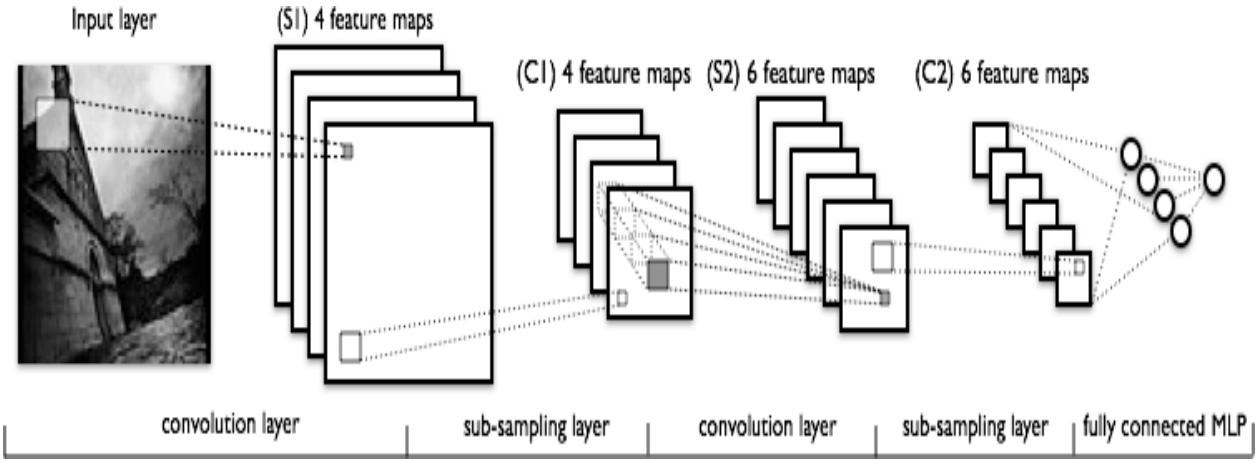


Figure 8:- Convolutional Neural Network Architecture used in [9]

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 [10] 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 [11] won ImageNet ILSVRC Challenge 2013. It modified AlexNet by varying hyper parameters and using larger filters on convolution layers.

Problem of over-fitting was one of disappointments for visual learning community as it degraded the performance of various networks. The network was under complex co-adaptation on training data. Dropout technique was used by [12] so as to reduce such problem and to better train the networks.

Both CNN, along with Recurrent Neural Networks (RNN) are used to solve the problem like image captioning [13, 14, 15].

Chapter 3

3. Methodology

3.1 Model Development

Data Driven Approach

It's quite difficult task to generate algorithm within numbers of the image to classify it. Instead we solve this problem as supervised learning. There are lots of examples of images for each individual classes. We examples to the model 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:

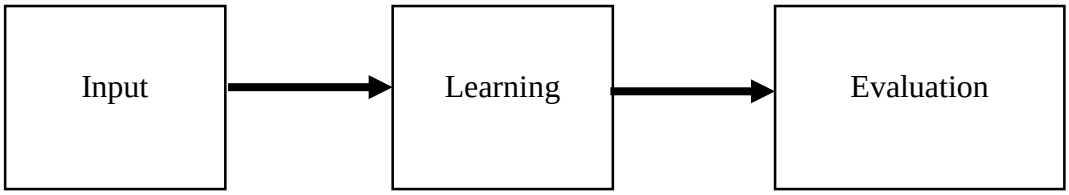


Figure 9:- Image Classification Pipeline

Input

The first step in the pipeline is Input and it consists of set of images each labelled 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. We then compare predicted labels with true label and expect most of predicted ones to be true. It is expected to have lot of predictions to be correct.

There are various steps involved for model development. Steps can be viewed as figure below:

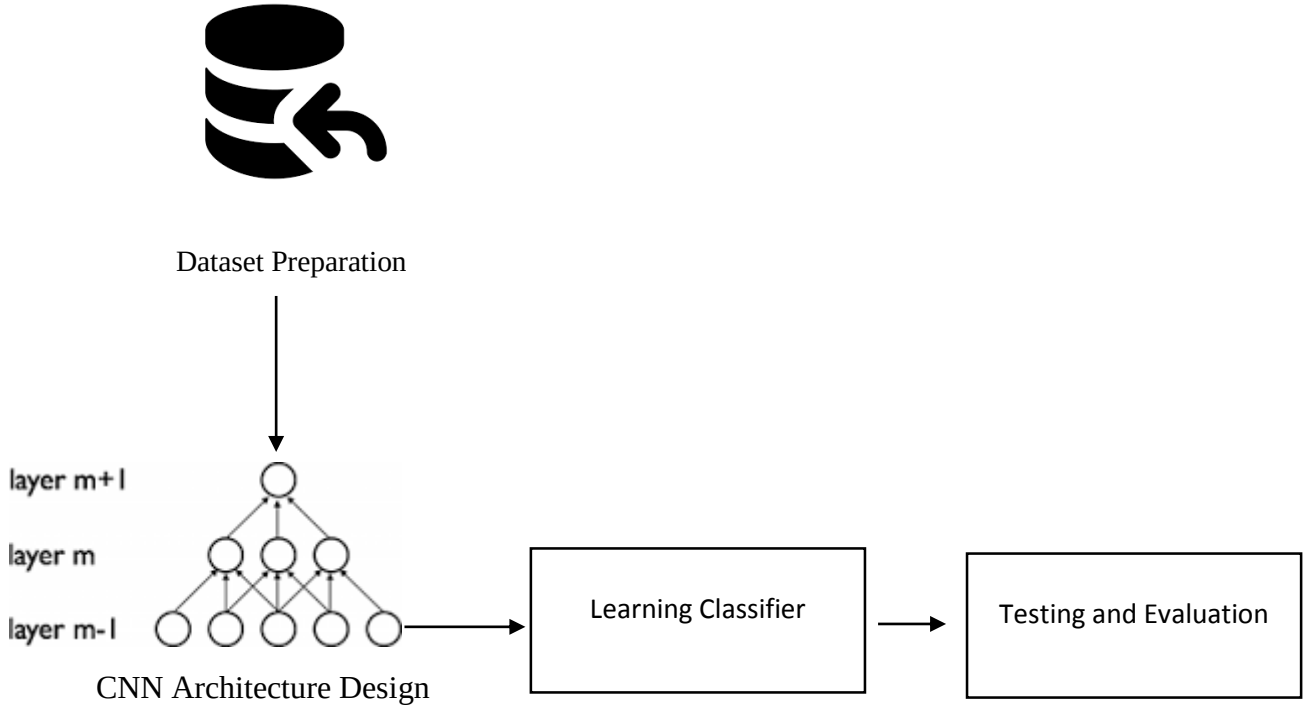


Figure 10:- Model Development Steps

3.1.2 Dataset Preparation

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

Figure 11:- Dataset Division

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

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

Testing Set are 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.

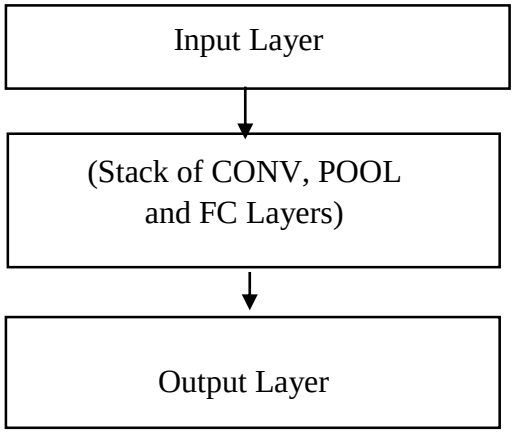


Figure 12:- CNN Design Flow

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
```

Rectified Linear Unit (ReLU) Activation

Standard Model used for activation functions are $f(x) = \tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$. In terms of training time with gradient descent these nonlinearities are much slower than the rectifier function defined by: $f(x) = max(0, x)$. The node that applies this type of function is known as ReLu node. Deep convolutional neural networks with ReLU train several times faster than their equivalents with hyperbolic tangent or sigmoid functions [10].

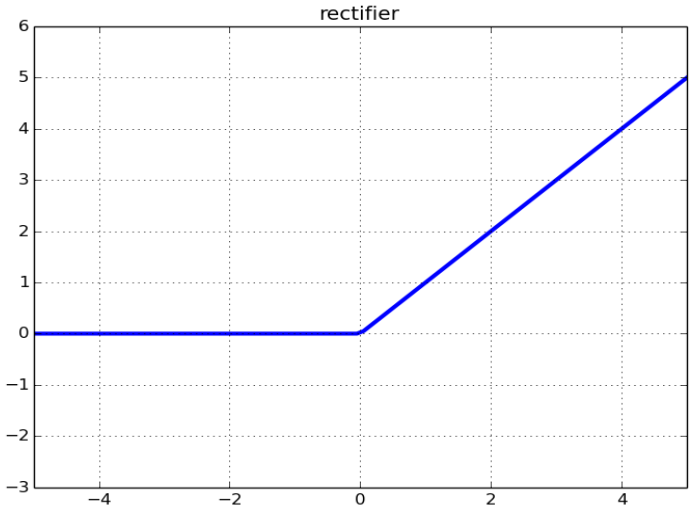


Figure 13:- Rectified Linear Unit

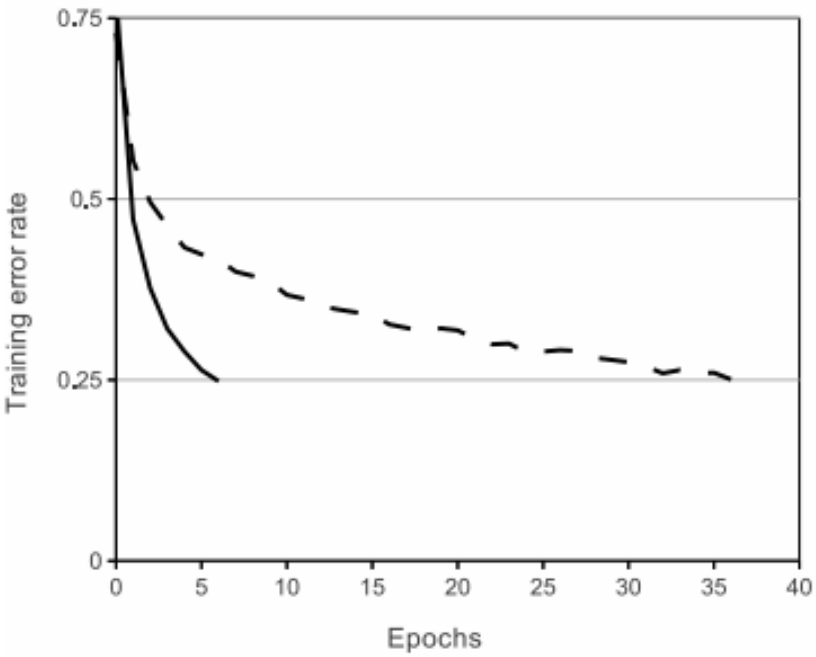
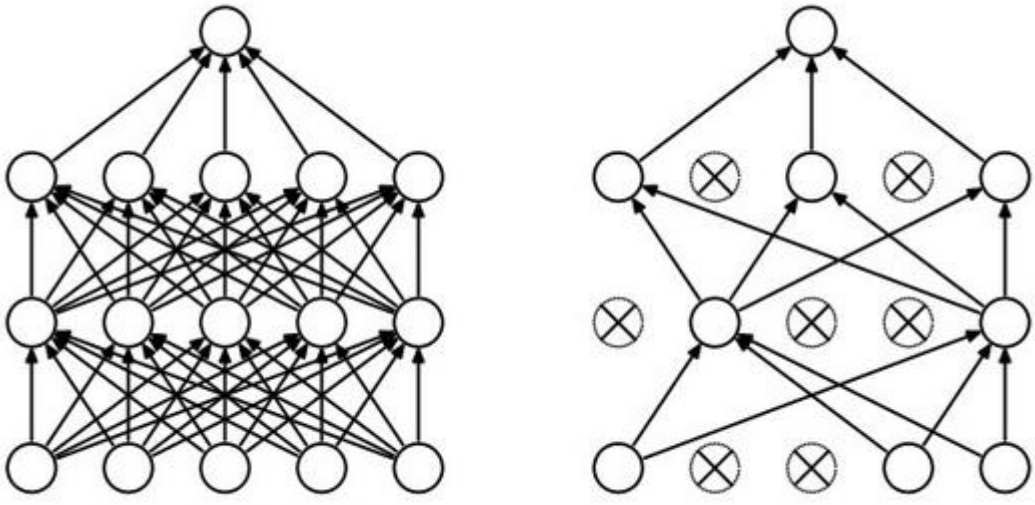


Figure 14:- Convolutional neural network with ReLU (solid line) reaches 25% training error rate six times faster than equivalent network with thanh neurons (dashed lines).

Overfitting and Dropout

There are non-linear hidden units between input and output on artificial neural networks. Weights are adapted continuously on these hidden units so that they learn feature detectors for predicting correct output for input vector. Due to less labelled training data the network will make different predictions on held-out test data. They perform worse on test data than on training data. Its because feature detectors are turned to work well together on training data but not on the test data. The figure below shows typical case of over-fitting in which after epoch 16, the train loss decreases smoothly but the validation loss does not.

Dropout is one of latest technology to combat with over-fitting. It is a tool for regularization. It prevents complex co-adaptation on the training data. It works by dropping out some activation units that sets given data of specific class to zero, so that a hidden unit cannot rely on other units being present [12]. It can also be viewed as model averaging neural networks. Test set error can be minimized by averaging predictions produced by very large number of different networks. One can do this by training many separate networks and then apply each of these networks to test data. This will be computationally expensive in both training and testing. Random dropout allows us to build such equivalent separate networks within single network at reasonable time.



a) Standard Neural b) After Applying Dropout

Figure 15:- Dropout Implementation to combat over-fitting

Figure above shows how dropout can disintegrate a complex neural structure to smaller networks and less connections thus achieving model averaging principle.

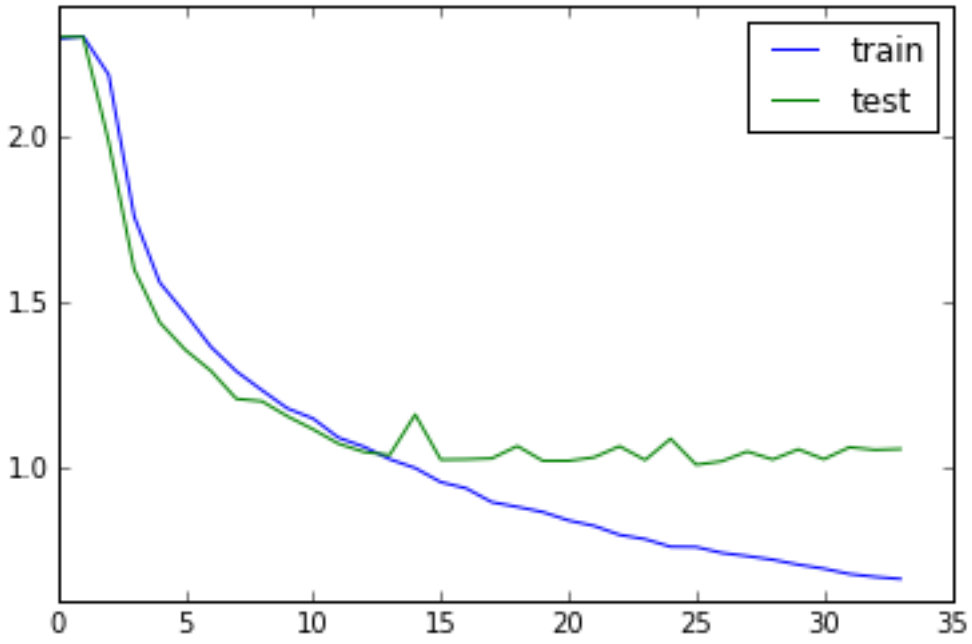


Figure 16:- Overfitting Issue

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 batches. So it is first quantity to track the training.

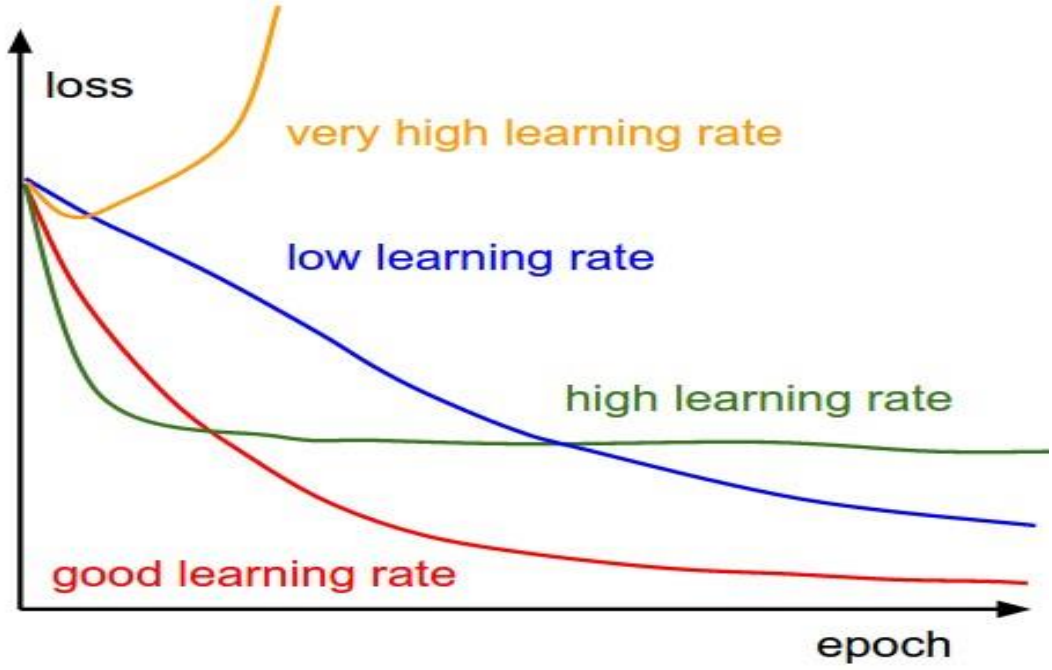


Figure 17:- Loss function with various learning rates

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.

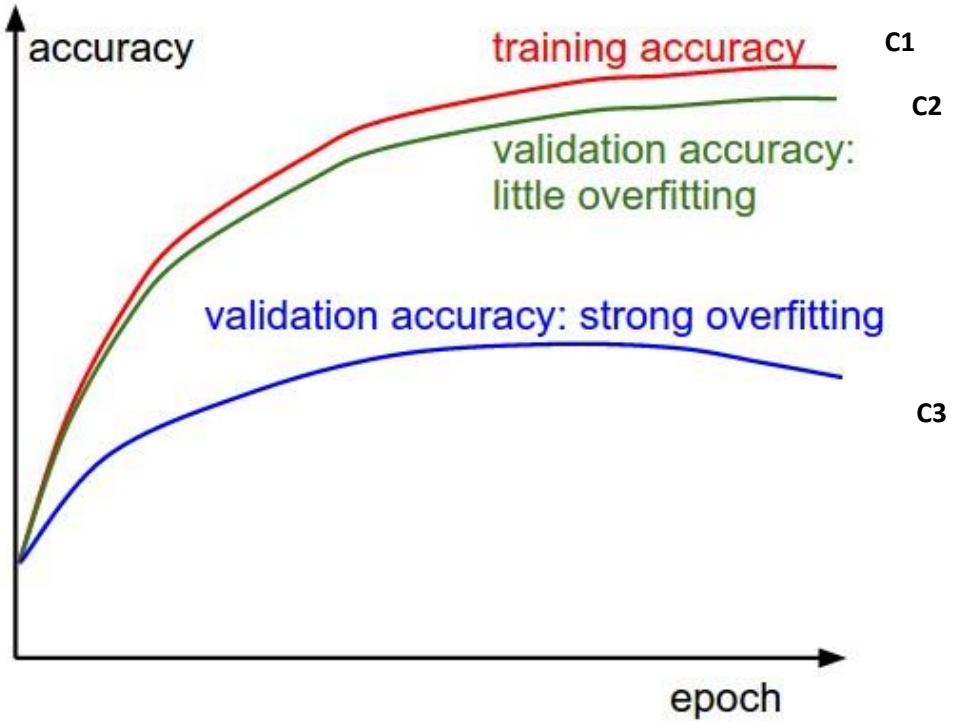


Figure 18:- Validation/Training Accuracy and over-fitting

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.

Evaluation

The model can be evaluated with multiple measures:

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

3.2 Tools and IDE

3.2.1 Python with Sci-kit, NumPyy, SciPy, Theano

Theano is a python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional are efficiently. It has been powering large-scale computationally intensive scientific investigations since 2007 [16]. Theano possesses tight integration with NumPy, which is fundamental package in python for scientific computing. It transparently runs over GPU achieving much faster than CPU.

3.2.2 GPU GeForce 820 M

Graphical Processing Unit (GPU) leverages Graphical Processors together with CPU to accelerate scientific, analytics, engineering, consumer, and enterprise applications [17]. Use of such graphics processor parallelizes the problem and gets solution at much faster time than central processing units (CPUs).

CUDA-Backend is used It is NVIDIA's GPU-Parallel programming tool-chain. CUDA is a parallel computing platform and programming model invented by NVIDIA that dramatically increases computing performance by harnessing power of GPUs.

GPU Engine Specs:	
GeForce Performance Score[1]	2.5x
Memory Specs:	
Memory Interface	DDR3
Technology Support:	
NVIDIA Optimus Support ™[2]	Yes
NVIDIA GPU Boost™	2.0
NVIDIA GameWorks™ Support	Yes
Microsoft DirectX	12 API
CUDA	Yes
OpenGL	4.5
Bus Support	PCI Express 2.0
OS Certification	Windows 8 and Windows 7

Figure 19:- NVIDIA GPU GeForce 820M Specifications [18]

3.3 Data Collection and Evaluation

For object detection CIFAR-10 and MNIST Dataset are be used.

CIFAR-10 Dataset

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 tabulated below with assigned class:

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

Table 1:- CIFAR-10 Dataset Classes

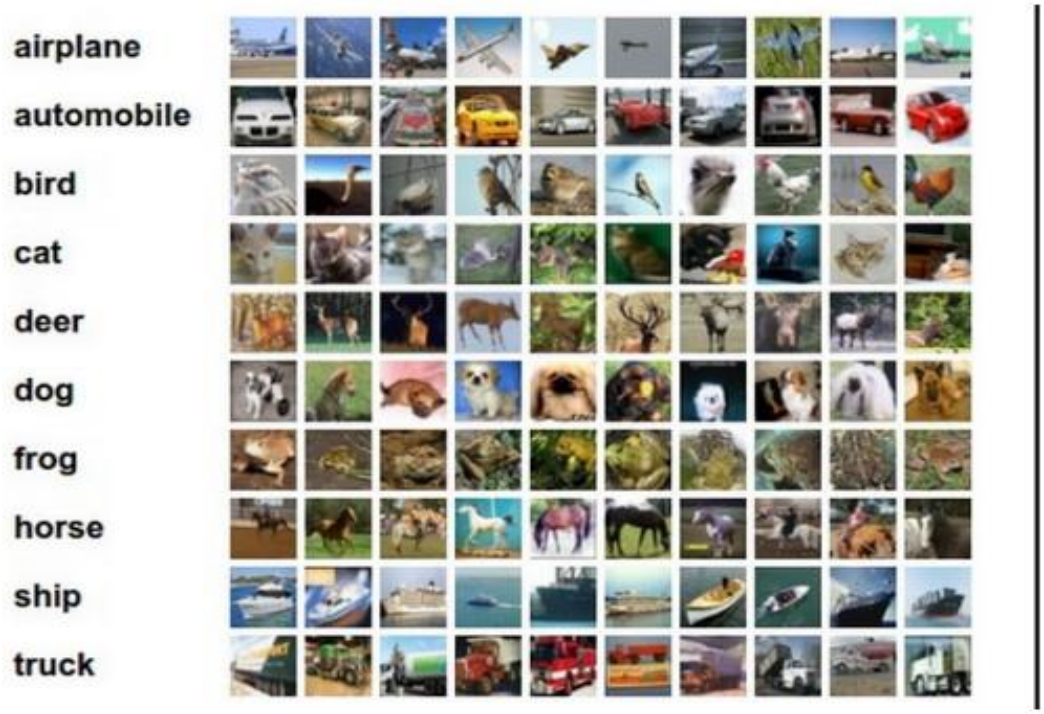


Figure 20:- Images in CIFAR-10 Dataset

MNIST Data

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 2:- MNIST Dataset Classes

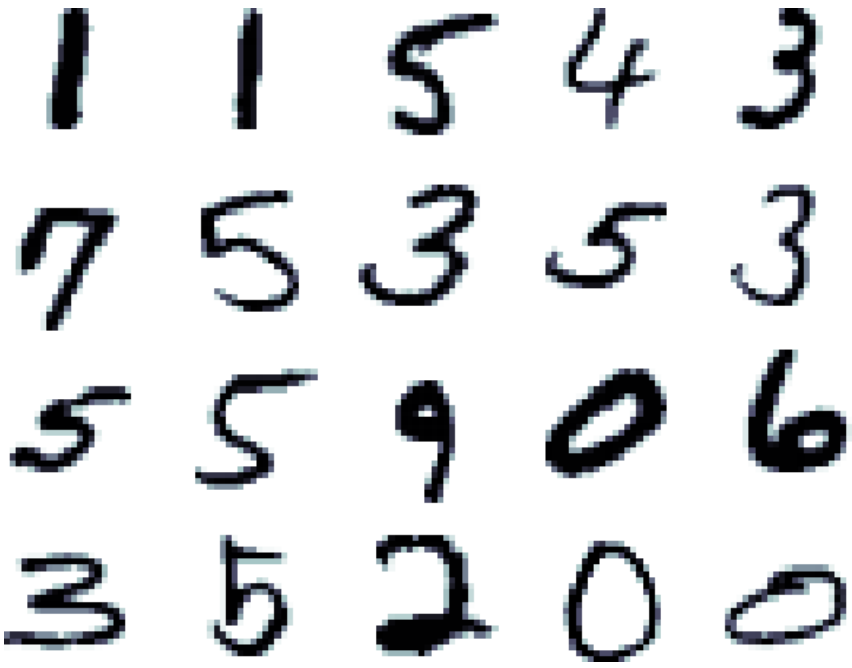


Figure 21:- Handwritten digits in MNIST Dataset

Chapter 4

4. Results, Analysis and Comparison

4.1 Overview of Tasks

4.1.1 Deep Learning Framework Installation and GPU Configuration

A very effective deep learning framework 'Theano' is installed and is configured to run on GPU GeForce 820M.

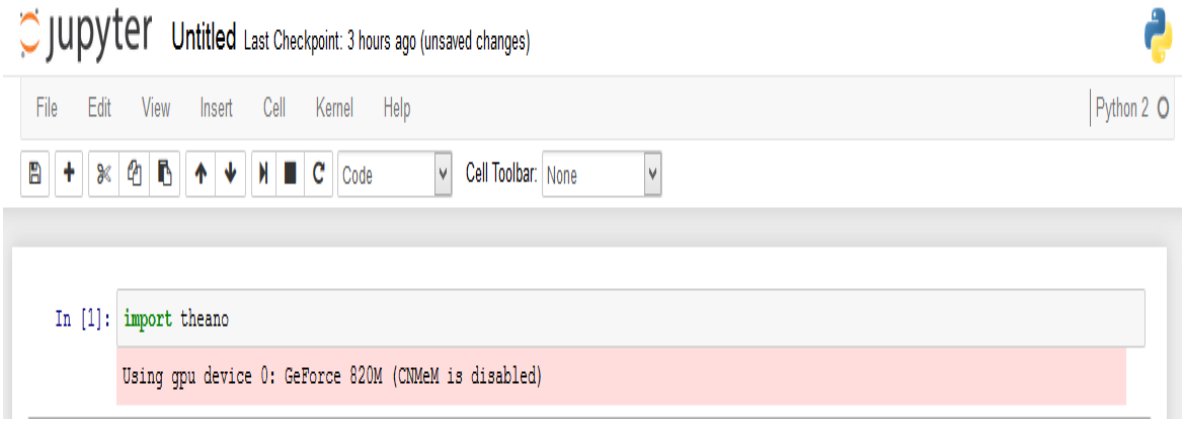


Figure 22:- Importing Theano Library on IPython Notebook showing GPU enabled sign.

Theano is python library allowing you to compute mathematical expression with multi-dimensional arrays efficiently. Main benefits of Theano is it is transparent to GPU usage, thus providing efficient and high performance training of model.

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 23:- CNN Architecture for MNIST Dataset

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 24:- 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.

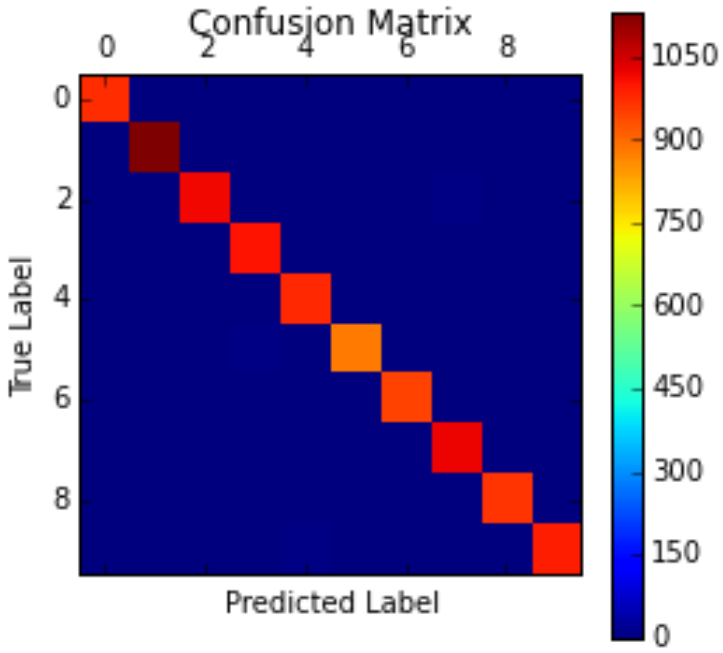


Figure 25:- Confusion Matrix for CNN designed for MNIST Dataset

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 26:- 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.

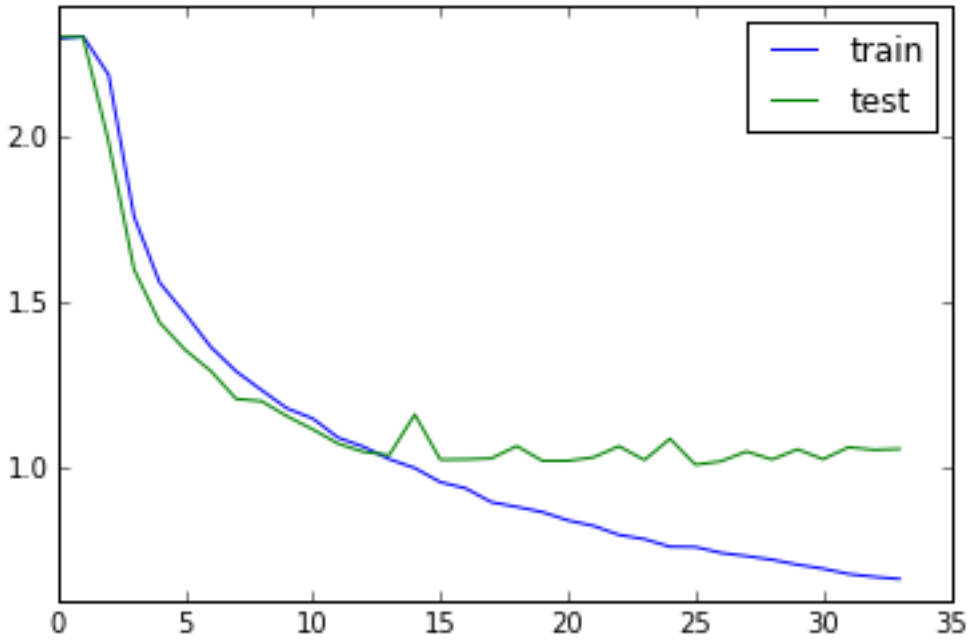


Figure 27:- CNN Architecture Train and Test Loss

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 28:- Layer Information for CNN Architecture-2

The figure above show 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.

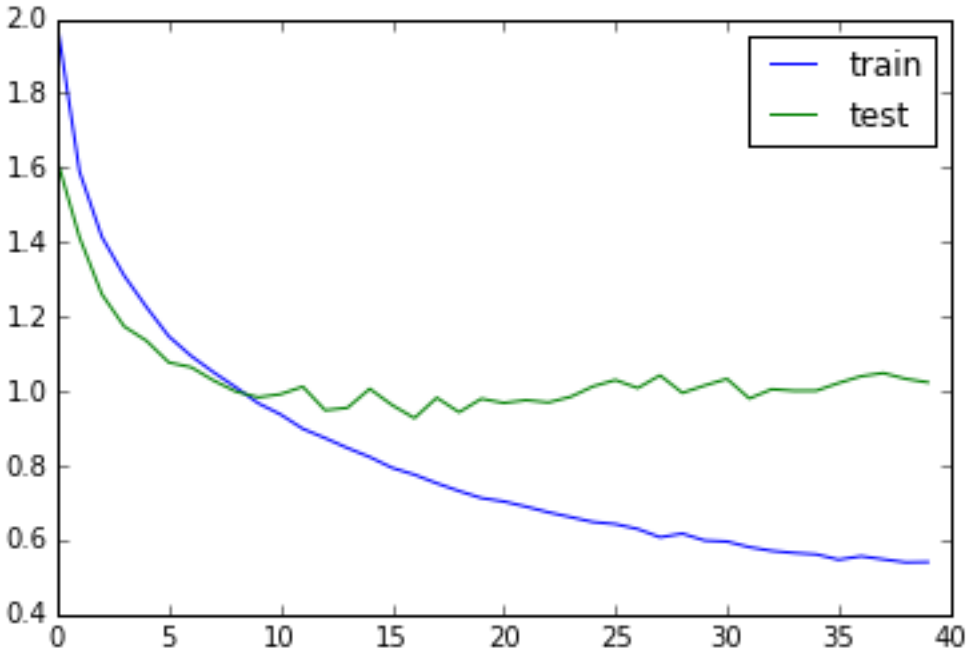


Figure 29:- Architecture trained for 40 epochs

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 30:- 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.

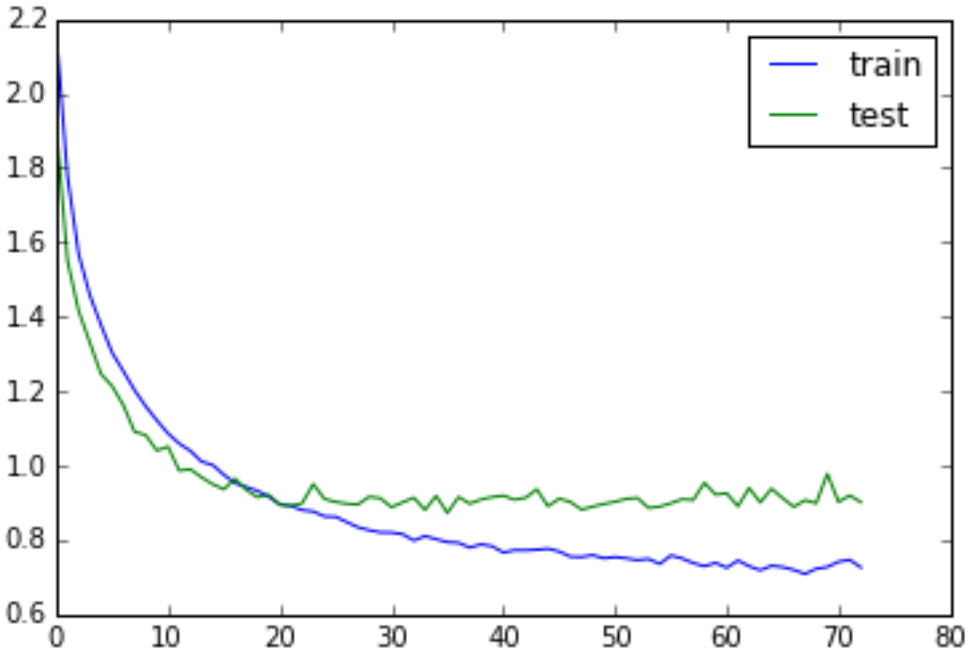


Figure 31:- CNN Architecture-3 Trained for 73 epochs

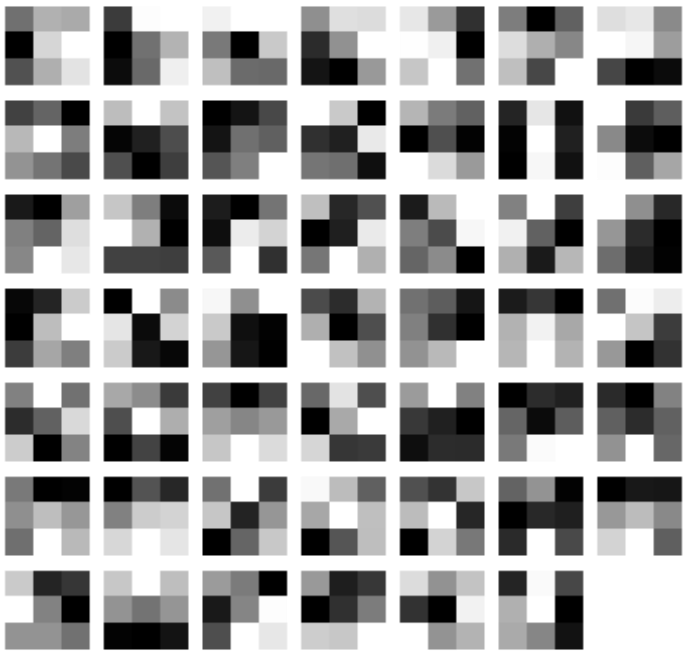


Figure 32:- Convolution Filter of first convolution layer of Architecture 3

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 3:- 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% dropout gave 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.

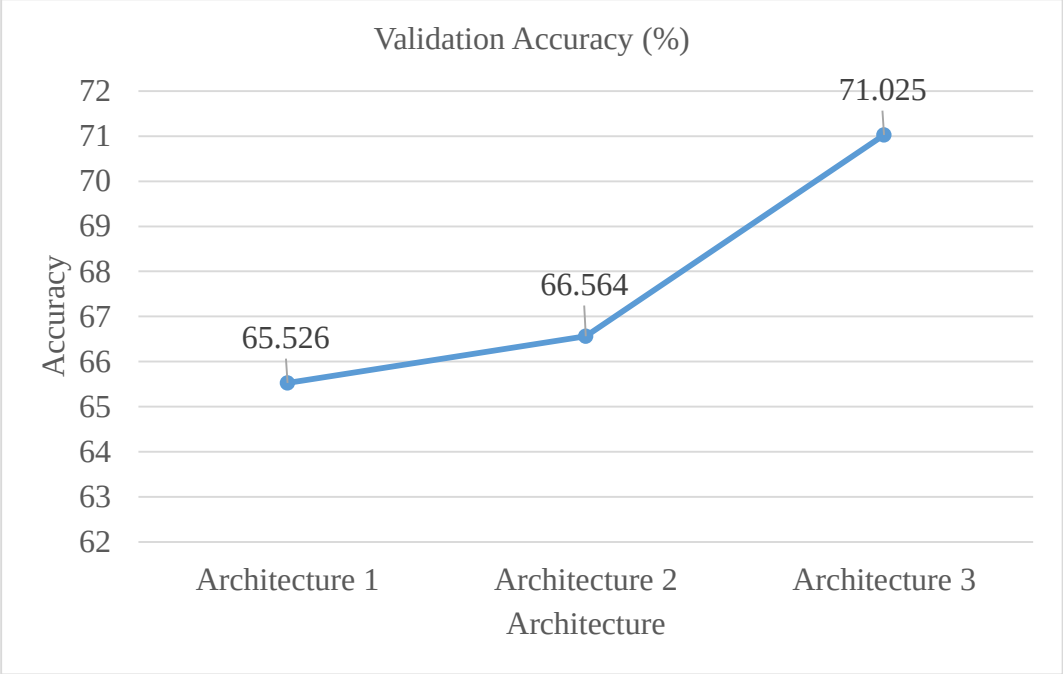
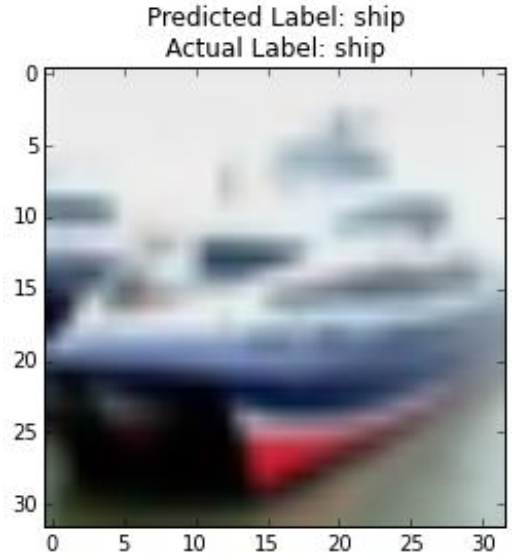
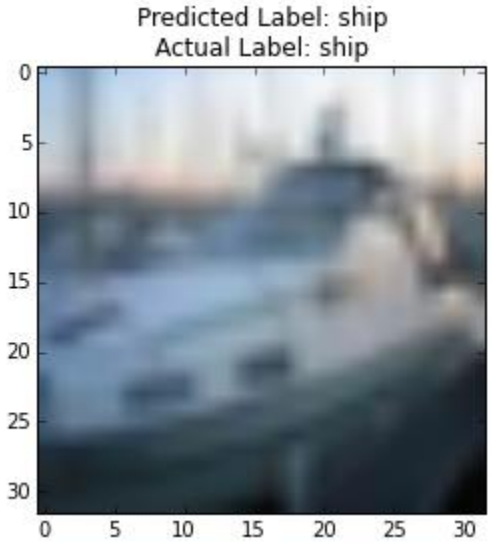
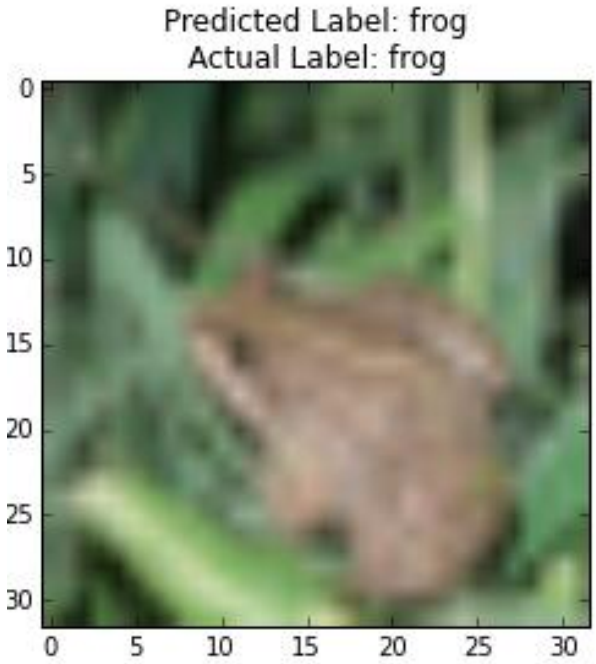


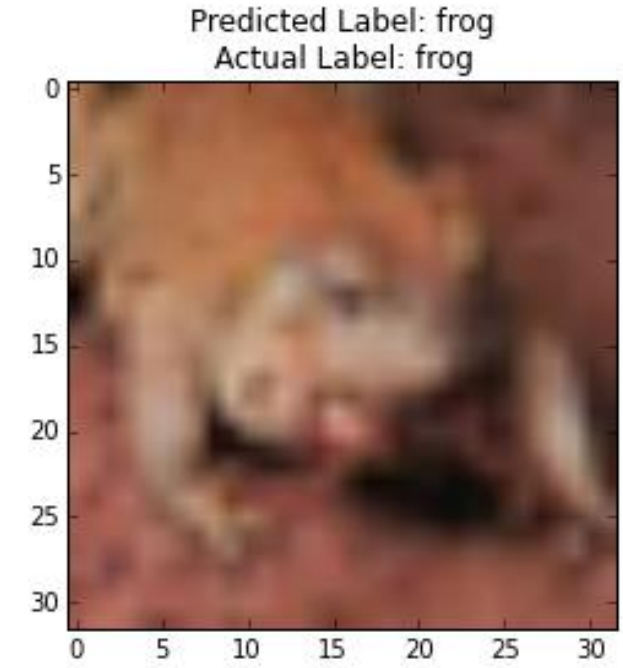
Figure 33:- Accuracy of different architectures

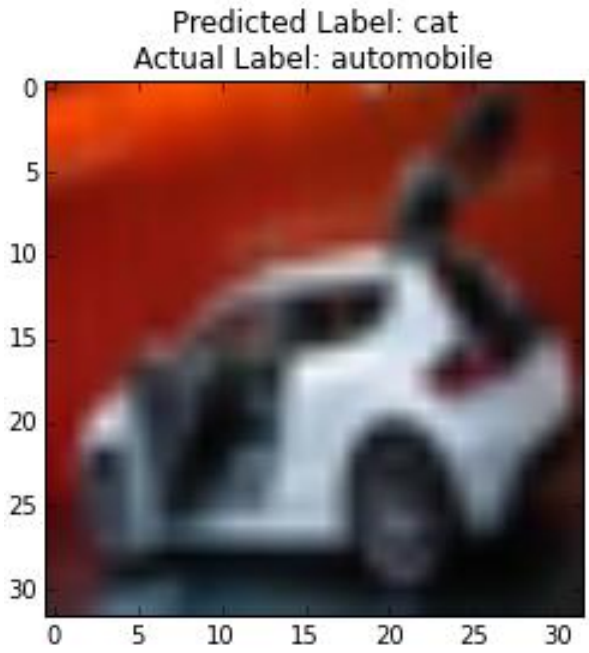
Figures below show classification results obtained from Architecture-3.

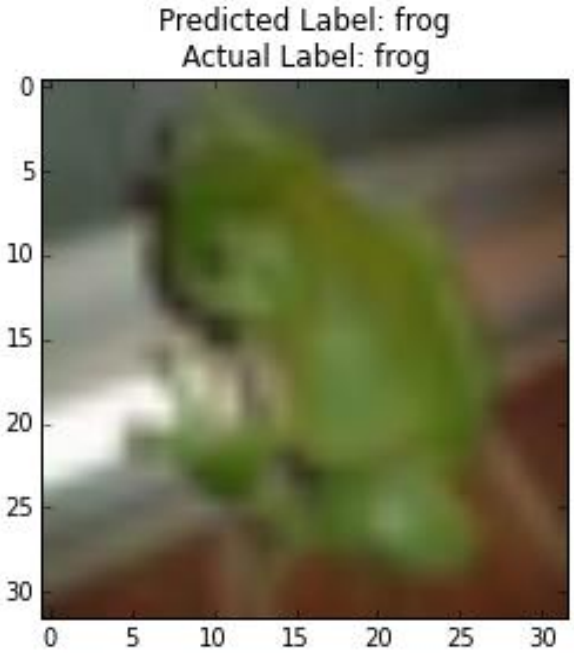


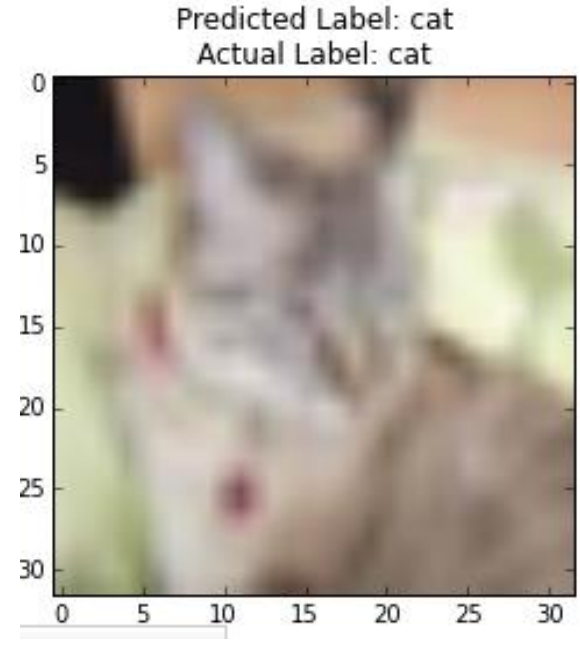





Predicted Label: frog
Actual Label: frog


Predicted Label: cat
Actual Label: automobile


Predicted Label: frog
Actual Label: frog


Predicted Label: cat
Actual Label: cat

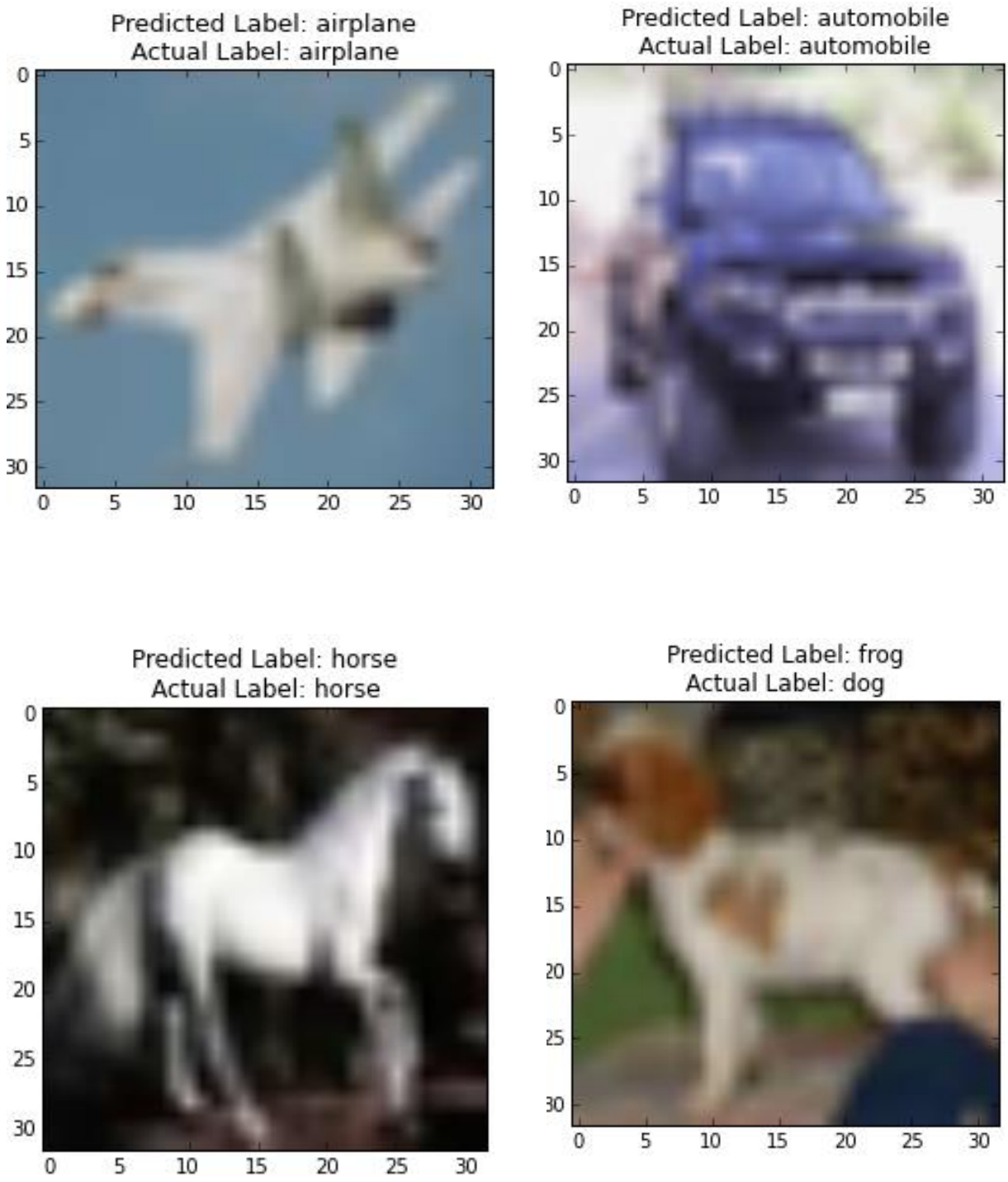


Figure 34:- Results obtained from architecture-3 over test dataset

Chapter 5

5. Conclusion

This thesis focuses on finding suitable architecture of convolutional neural network for solving image classification problem. Instead of using and applying advanced image processing techniques, machine learning technology is used to solve the problem. The model developed here implements deep neural network architecture. Deep layers extracts information from data at different level of abstractions. CNN is trained and image is classfied.

It was general idea 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.

In this thesis the three architectures are developed for both MNIST and CIFAR-10 datasets. 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.

Dropout must be used for reducing over-fitting. 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

6. Limitations and Future Enhancement

The Architectures developed in this thesis have many limitations. Since it is very expensive to calculate convolution operations, we must need GPUs to train architectures. GPUs can exploit their parallelism. High configuration GPUs 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 98.70 % 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 71%. Further processes can be done to improve accuracy of the Architectures. One of them is Data Augmentation. It can be used further to minimize overfitting. 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] S. Tara, Brian 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] Q. V. Oriol Vinyals, "A Neural Conversational Model," arXiv, 2015.

[6] R. Grishick, J. Donahue, T. Darrel and J. Mallik, "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] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," 2012.

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

[12] G. Hinton, N. Srivastava, A. Karpathy, I. Sutskever and R. Salakhutdinov, *Improving Neural Networks by preventing co-adaptation of feature detectors*, Totonto: arXiv, 2012.

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

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

[15] J. M. G. H. IIya Sutskever, "Generating Text with Recurrent Neural Networks," in *28th International Conference on Machine Learning*, Bellevue, 2011.

[16] "Theano," [Online]. Available: http://deeplearning.net/software/theano/index.html. [Accessed 27 10 2015].

[17] "What is GPU Computing ?," NVIDIA, [Online]. Available: http://www.nvidia.com/object/what-is-gpu-computing.html. [Accessed 27 12 2015].

[18] "GeForce 820M|Specifications," NVIDIA, [Online]. Available: http://www.geforce.com/hardware/notebook-gpus/geforce-820m/specifications. [Accessed 28 10 2015].