Convolutional Neural Network Models for Image Classification and Object Detection

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

Master of Engineering
in
Computer Science and Engineering

Submitted By
Shweta Upadhyay
(Roll No. 801632047)

Under the supervision of:
Dr. R. K. Sharma
Professor, CSED

Dr. P. S. Rana
Assistant Professor, CSED

COMPUTER SCIENCE AND ENGINEERING DEPERTMENT, THAPAR INSTITUTE OF ENGINEERING, PATIALA, PUNJAB

JULY 2018

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Convolutional Neural Network Models for Image Classification and Object Detection"**, in partial fulfillment of the requirements for the award of degree of **Master of Engineering** in **Computer Science and Engineering** submitted in Computer Science and Engineering Department of Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out under the supervision of **Dr. R. K. Sharma** and **Dr. P. S.Rana** and refers other researcher's work which are duly listed in the reference section.

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

(Shweta Upadhyay)
Roll No. 801632047

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

31.7.18
(Dr. R. K. Sharma)
Professor, CSED

31/7/2018
(Dr. P. S. Rana)
Assistant Professor, CSED

Acknowledgment

I would like to express my gratitude to my supervisors, Dr. R.K. Sharma, Professor, Computer Science and Engineering Department, TIET, Patiala and Dr. P.S. Rana, Assistant Professor, Computer Science and Engineering Department, TIET, Patiala, for the time, patience, thought-provoking discussions, criticism and suggestions given by them during the course of this work. Without their experience and encouragement, it would not have been possible to present this dissertation in its form.

I would also like to extend a sincere thanks to Mr. Malinder Singh, Lab Incharge, Computer Science and Engineering Department, TIET, Patiala, for helping me with setting up the right software environment.

I am thankful to all my friends and well-wishers for supporting me in every possible manner to complete this dissertation.

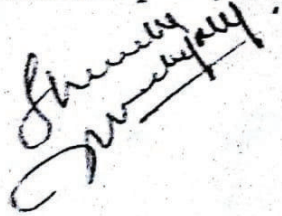
It is very hard to express my feeling in proper words for my parents who have always encouraged me in all endeavors.

Lastly, though it is not possible to mention every one, none can be forgotten for their direct or indirect help.

Date: July 2018

Place: TIET, Patiala

(Shweta Upadhyay)



Abstract

Computer vision is one of the most frontier and revolutionary fields of computer science. It aims at understanding and managing the enormous visual content available in the present day. Vision begins with eyes but really takes place in the brain; hence the concept and functioning of ANNs are studied, which enable the computers to understand images by examining its features.

The study focuses on a category of deep neural networks typically used to deal with visual data, known as convolutional neural networks (CNNs). CNNs eliminate the need to hand-engineer the features of the object class, thus making the process more like a child learning to see and interpret things.

The thesis presents the implementation of image classification and object detection which are two fundamentals to build any computer vision system. Certain well-known algorithms are applied, including Alexnet for classification and Faster R-CNNs and Mobilenet-SSD for detection. Their performances are analyzed and compared.

The outcome of the study shows that Mobilene-SSD takes less detection time as compared to Faster R-CNNs, on the other hand, Faster R-CNNs can detect more object instances in an image, with higher class scores.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Photos and videos are becoming an integral part of global life. These are being generated at a pace far beyond any organization reports. We emote, tell a story, explain a concept, and start a new idea through pictures. However, our most advanced machines in computers still struggle at this task. The task of understanding or making sense of what they see. Today, there are prototype cars that can drive by themselves but without stark vision, they cannot really tell the difference between a crumpled paper bag thrown on the road which can be driven over, and a rock of that size which should be avoided. Security cameras are everywhere but they do not alert when a child is drowning in a swimming pool.

The study and attempts made in this direction are all grouped in the field known as Computer Vision, and the ultimate goal is to see just like the humans do. The first step to complete this is to teach computer to see objects. An attempt has been made to cover this step in this thesis work.

This chapter briefly describes the concept of Artificial Neural Networks, associated terminologies and their mathematical representation. The need for introduction of Deep Neural Networks is highlighted; focusing on the fundamentals of Convolutional Neural networks, which are majorly used to solve computer vision problems and their brief working methodology. The last section of this chapter presents an organization of the thesis content.

1.1 Artificial Neural Networks

The term Neural Network derives its origin from the human brain or the human nervous system. The brain is a highly complex, non-linear and massively parallel information processing system. Hence, it can perform certain computations (Eg: perception, motor control and recognition) in an entirely different manner and in an amazingly small amount of time as compared to the conventional digital computer. For example: Human vision: which is an information processing task. Out of so many faces one sees every day, the brain possesses the ability to recognize a familiar face embedded in an unfamiliar scene, even if it was years ago since they saw that person. The brain does this in 100-200ms, whereas tasks of much less complexity may take days on a conventional computer. Another example: Eco-location by bats, using sonar. The bat sonar provides information about the distance, size, elevation and even size of various features of the target. Such computational abilities of the biological neural systems, is what motivated the researchers to mimic their highly complex structure, hence forming the basic idea behind the conception of ANNs.

According to Simon Haykin (1994): "A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two ways: (1) Knowledge is acquired by network from its environment through a learning process. (2) Interneuron connection strengths, known as synaptic weights, are used to store acquired knowledge."

1.1.1 Brief Working of the Brain

The function of the brain is to accept, process and reply to the electro-chemical signals which may be coming through sensors like the sense organs, to register things like pressure, temperature, light etc.

A neuron is the elementary processing unit of the brain, and like every processor they have an input and an output. Each neuron is connected to its neighboring neurons. A human brain consists of about 10^{10} neurons, and about 10^{14} interconnections. The input signals are electro-chemical stimulus, received by neurons through

special input lines, called dendrites (each neuron has approx. 10,000 to 100,000 dendrites) and produce output signals along its output line (single), called axon. Signals are transferred through synapses. A synapse serves as a unique inter-neuron interface to transfer the information. In the basic model, dendrites carry the signals (impulses) to the cell body, where all of them get summed up. If the final sum is above a certain threshold, the neuron become active, i.e., it 'fires', sending a spike (or output) along its axon, to another connected neuron.

1.1.2 Mathematical Representation of Neuron

Mathematical models are suitable for computer simulations and can be understood using mathematical formulas.

The Neuron

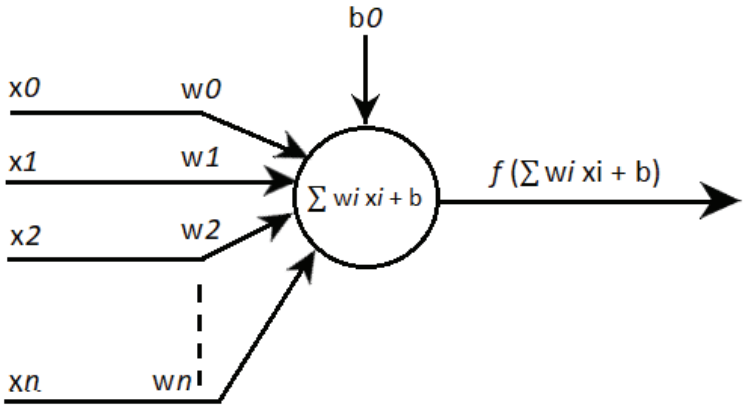


Figure 1.1: Mathematical model of a neuron

In Figure 1.1, $x1,x2,\ldots xn$, represent the input signals and $w1,w2,\ldots wn$ represent the weights of the input symbols respectively. $b0$ represents an additional class of weights known as bias. Each neuron performs a dot product with the input and its weights and adds them all (weighted sum is performed). Biases are added to the sums calculated at each node (except input nodes) (The use of biases in a neural network improves the properties of a neuron, by allowing in moving the threshold.)

$$S = \sum_{i=1}^{n} w_i x_i \tag{1.1}$$

The weighted sum of input values represents the excitation level of the neuron. When the excitation level exceeds a certain threshold, a neuron fires. But this weighted sum does not help to reach a conclusion or a decision. So, in order to achieve a decision, a nonlinear processing unit is applied after summation. This is called the activation function. Thus, the activation function of a node defines the output of that node given an input or set of inputs. Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it.

1.1.3 Commonly used activation functions

Some activation functions that are usually encountered in practice are:

1. **Sigmoid:**

   Sigmoid is a common choice of activation function (non-linearity). The main reason for this is that it takes a real-valued inputs (the values represent signal strength of neuron after the sum) and brings them down to a range between 0 and 1. The sigmoid curve resembles an 'S' shape. The function is mathematically defined as follows:

   $$f(x) = \frac{1}{1 + e^{-x}} \tag{1.2}$$

   The sigmoid function gives non zero-centered outputs. This is undesirable, since the neurons in latter layers of processing in a Neural Network would receive data that is not zero-centered. This impacts the gradient descent dynamics, because if the input data into a neuron is always positive, then the gradient on the weights during backpropagation either become all be positive, or all negative (depending on the gradient of the entire expression). This may lead to undesirable zig-zag dynamics in the gradient updates for the weights.

2. **Tanh:**

   The tanh non-linearity, shown in Figure 1.2(b), compresses a real-value to the

range [-1, 1]. It gives zero-centered outputs, unlike sigmoid neuron. Therefore, in practice the tanh non-linearity is always preferred over the sigmoid nonlinearity. Also, the tanh neuron is simply a scaled version of sigmoid neuron. The following holds in particular:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.3}$$

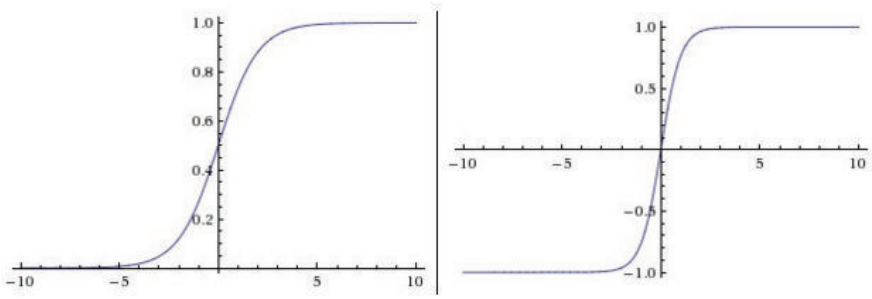


Figure 1.2: (a) Left: Sigmoid non-linearity (b) Right: Tanh non-linearity

Figure 1, shows a comparison between ranges of Sigmoid(a) and ReLU(b) non-linearities.

3. **ReLU:**

   The Rectified Linear Unit is the most widely used activation function in the last few years, since it is used in almost every CNN or Deep Learning based application. In a CNN architecture ReLU function is preformed after each convolutional layer. ReLU is defined as:

$$f(x) = max(0, x) \tag{1.4}$$

   where x represents the input to a neuron. The activation is thresholded at zero (Figure 2(a)). In other words, f(x)=0 when x0 and f(x)=f when x=0.$Figure 1.2(b) displays a plot fro$

1.1.4 Backpropogation

The goal of any ANN is to find a function that best maps a set of inputs to their correct output. Eg: Input is an image of an animal, and the correct output is the name of

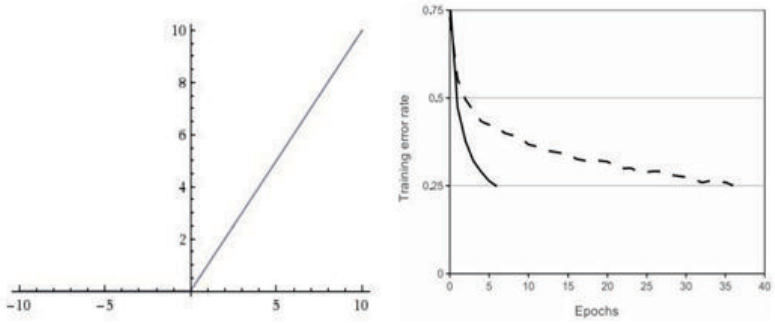


Figure 1.3: (a) Left: ReLU activation function (b) Right: Comparision plot

the animal. The motivation for backpropagation is to train a multi-layered neural network such that it can learn any arbitrary input-output mapping. Backpropagation is a method used in ANN for calculation of each neuron's error contribution, after a batch of data (in case of image recognition, multiple images) is processed. And adjust the weight of each neuron, to complete the learning process for that case, and get an optimized result.

A brief description of the algorithm is given below. Each iteration of the algorithm involves two major steps:

Phase 1: **Propagation**

• Propagating forward through the network to generate the output value.

• Calculating the cost (or error term).

• Calculating The difference between the targeted and actual output values, for all output and hidden neurons, in order to propagate backwards and update the weight.

Phase 2: **Weight Updation**

The weight updation rule is:

$$w_{ij} = w_{ij} + \delta w_{ij} \tag{1.5}$$

where,

$$\delta w_{ij} = -\eta E w_{ij} \tag{1.6}$$

1.2 From Conventional Neural Networks to Deep Neural Networks

Conventional ANNs were limited in their ability to process natural data in raw form. Therefore, constructing these ANNs (say, for the purpose of pattern-recognition or image-classification), required careful engineering and eminent domain expertise for designing feature extractors that could transform the raw data (eg: pixel values of an image) into suitable internal representations or feature vectors which could be used to detect and classify input patterns by the network.

Representation learning is a set of methods in which, a machine is fed with raw data and representations required for detection or classification, are automatically discovered. These learning methods form the basis of deep neural networks and obtain multiple representation levels, by composing simple, non-linear modules. Every module transforms the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. When enough of such transformations are composed, very complex functions can be learned.

For example, an input image is in the form of an array of pixels, and the features learned in the first layer of representation usually represent the presence or absence of edges at specific locations in the image. The second layer generally detects patterns or shapes or figures (collectively termed as 'motifs'), by spotting certain arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations, corresponding to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep neural networks is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

1.3 Convolutional Neural Networks

Convolutional neural networks were designed to mimic the visual cortex of humans/mammals and thus are majorly used to solve problems in the domain of Computer Vision. The concept was introduced by Yann LeChun of New York University, cur-

rently the Director of Facebook's AI unit.

The architecture of CNN is inspired by 3 features of the human/mammal visual cortex:

1. Local connections
2. Layering
3. Spatial Invariance. Eg: We are able to recognize a chair under a variety of conditions, or the position of a shoe.

There component layers of a CNN are described below:

1. **Input Layer:**
   The input is an image sequence or a single image. Each image is an array of pixel values. It is comprised of 3 spatial dimensions: width, height and depth. Say size 227*227*3, where 3 represents RGB channels. There can be other channels depending on the image properties.

2. **Convolutional Layer:**
   Convolutional Layer is the core building block of a CNN. It makes use of the mathematical operation 'convolution', which uses two functions to produce a new, modified function, and thus finds patterns. Unlike conventional neural networks, every neuron the convolutional layer is only connected to a small region of the input volume.
   For example, if the input to the network is an image of size 64 * 64 * 3 (a RGB-coloured image with a dimensionality of 64 * 64) and we set the receptive field size as 6 * 6, we would have a total of 108 weights on each neuron within the convolutional layer. (6 * 6 * 3 where 3 is the magnitude of connectivity across the depth of the volume) To put this into perspective, a standard neuron seen in other forms of ANN would contain 12, 288 weights each.

3. **Pooling Layer or (sub-sampling):**
   The pooling layer reduces the spatial size (height and width, but not depth) of

representation of the image. Thus, it reduces the number of parameters and computation in the network, which helps control overfitting. It is a common practice to insert a pooling layer at intervals, in-between successive convolutional layers in a CNN architecture. It operates autonomously on every input depth slice and re-sizes it spatially, using operations like average, MAX etc.

4. **Fully Connected Layer:**
   Fully connected layers generally form the latter layers of a CNN. Each neuron in a fully connected layer, is connected to all activations in the previous layer, but neurons within a single layer share no connections. Their activations can hence be computed with a matrix multiplication. Fully-connected layers are the most common layer type, of regular or conventional neural networks.

1.4 Image Classification

In machine learning, classification is the problem of determining the category to which a new observation (or instances) or set of observations belong, on the basis of already known data or the data which the system is trained to identify. Some examples include, determining the status of an email as spam or non-spam, assigning diagnosis to patients based on their observed characteristics.

It is important to understand the relationship between the existing classes and data in order to classify the data into different categories. To achieve this on a machine, the machine must be trained.

In this work, classification is being used to deal with image data. The goal of image classification process is to categorize all the pixels of an image into one of the various classes present in the training dataset. This categorized data may then be used to produce thematic maps of the land cover present in an image.

Image classification is the first and most important step in the field of Computer Vision. It was originally developed to work in the field of Pattern Recognition and is considered to be an instance of supervised learning.

1.5 Object Detection

Classification works well only if there is a single object in the image or multiple objects but belonging to the same category or class. When a variety of objects or classes are present in a single image, the classifier is 'confused' as to which object to focus on, and thus ends up delivering incorrect results. For instance, in the task of classifying pet animals, if in an image the true class is 'Dog', but the image also has a cat in it, i.e., both cat and dog in the image, then the classifier might end up classifying the picture as 'Cat'.Hence, a need arises for the machine to perform well even in case there are multiple object instances present in the image, which is a concept similar to human vision.

To achieve this, the process of 'Localization' comes into play. Localization bounds the object instance to a smaller region or defines the space taken by the object inside the image. It uses the method of bounding boxes to do so. The classification of all the localized objects in the image is called object detection. It is defined as the task of identifying or finding and classifying various objects instances (such as faces, animals, buildings etc.) present in an image or a stream of images.

In the present day, Object detection is one of the most prominent tasks in the field of computer vision. Some of it's applications are listed below:

1. **Self-driving cars -** Object Localization is used extensively to train autonomous driving perception models for objects like lane obstacles, traffic signs and pedestrians.

2. **E-commerce -** Object Detection is used to train visual search models for recognizing various products like gadgets, furniture and fashion accessories.

3. **Insurance -** Damage Detection used to Identify damage in structures, vehicles etc. in real world and detecting the degree of damage by training ML models for insurance claims.

4. **Drones and Robots -** Labeled images are trained for smart surveillance by drones and robots for identifying variety of objects.

1.5.1 Some Object Detection Techniques

R-CNN (Region-CNN)

R-CNN by Girshick (2014) is an architecture specifically designed to address object detection problems i.e., it aims to localize and correctly identify the objects of interest in the images. It creates region proposals or bounding boxes, using the process of Selective Search. Unlike HOG based methods, Selective Search reduces the number of bounding boxes being fed to the classifier, to almost 2000 region proposals. It uses local indications like color, texture, intensity etc., in order to generate all possible object locations.

After creation of the proposals, R-CNN encloses the regions to standard square sizes, which are passed through a revised version of Alexnet. SVM is used train to the final CNN layer, which classifies the object, if it belongs to one of the given classes. The final step of R-CNN is to tighten or optimize the bounding boxes, so that they fit the true dimensions of the object. This is achieved by running simple linear regression on the region proposal, to reduce the box coordinates.

Fast R-CNN

R-CNN works really well, but is quite slow. This is because:

1. A forward pass of the CNN (AlexNet), is required for each region proposal in each image which estimates to around 2000 forward passes for one image.

2. Three different models, need to train separately

    - CNN - for generating image features
    - SVM classifier - for predicting the classes
    - Regression model – for optimizing the bounding boxes

Therefore, to overcome this limitation, Fast R-CNNs were introduced by Girshick (2015). Improvement- It was observed that the proposed regions of the image overlapped causing the same CNN computations to run many times. Therefore, to avoid this, Fast R-CNN used a technique called ROI Pooling. It involves pooling (usually

Table 1.1: Inputs and outputs of Fast R-CNN

Input	1. Images 2. Region Proposals
Output	1. Class labels and scores for each region 2. Optimized bounding boxes

max-pooling) the features in each region. This shares the CNN computation across the sub-regions of the image. For classification in Fast R-CNN, instead of using SVM classifier, a softmax layer is added on the top of the CNN. A linear regression layer is added parallel to the softmax layer for bounding box optimization. Thus, Fast R-CNN has the ability to multitask, i.e., a single network is used to train the CNN and the classifier, and to optimize bounding boxes.Thus the overall training time is reduced and accuracy is improved. Table 1.1 shows the inputs and outputs of the overall model.

1.6 Transfer Learning

Transfer Learning is a technique in machine learning, to store the gained knowledge from one problem or a set of problems and then applying it to solve new but similar problems.

A pre-trained model is selected from the available models according to the application where they are needed. And then they are reused. This helps to initialize the training with better weight values. The model may also optionally, be tuned to adapt to the new or refined dataset being used.

Why is it needed?

1. Practically, very few people train CNN right from scratch because it is not easy to get a dataset which is large enough. Arranging or building datasets is a major issue. Hence, pre-trained network weights are used as either initializations or fixed-feature extractors. This solves a significant portion of the problems.

2. Very deep NNs are expensive to train. Even the most expensive models can take weeks to get trained using many machines equipped with very expensive GPUs.

3. Determining the flavour/ hyper parameters/ topology/ training method for deep learning is not an easy task.

1.7 Thesis Organization

The work presented in this thesis has been organized into five chapters.

Chapter 1 provides an overview of the importance of the problems of classification and detection and gives the theoretical and mathematical background of conventional and deep neural networks.

Chapter 2 presents a review of the published literature in the area of this study and the state-of-the-art for each of the applications studied.

Chapter 3 gives the gaps in the study mentioned in the literature review, and also states the objective and scope of the work presented in the thesis.

Chapter 4 demonstrates the methodology to administer and implement image classification and objection.

Chapter 5 displays and analyses the qualitative and quantitative results in details.

Finally, chapter 6, the last chapter, discusses the conclusions of the study and yields recommendations for the future work.

Chapter 2

Literature Survey

Success in the field of computer vision has been a long-standing demand for the modern society. In literature, many methods have been proposed in this area of research. Till the recent past, some conventional machine learning algorithms (esp. Neural Networks) were in processing images, video, speech and audio. But deep convolutional networks have brought about breakthroughs in this domain since the time they were introduced.

The development in this field can be identified in two major categories:

1. Development in conventional Neural Networks
2. Development in Convolutional Neural Networks

2.1 Development in conventional Neural Networks

McCulloch and Pitts (1943) tried to perceive how the brain produced extremely complex patterns by using neurons connected together. A highly simple model of a neuron, known as the MCP model, was presented, which served as a major contribution of development in the field of artificial neural networks.

Rosenblatt (1959) introduced the notion of a perceptron. Perceptrons are MCP neurons, where the inputs are initially passed through association units also known as

'preprocessors'. The preprocessors detect the presence of some specific features in the input. The perceptron, as suggested by the name, was meant to be a feature recognition system, where the preprocessors correspond to pattern or feature detectors.

Wolberg (1998) presented a study on morphphing a well-trained neural network to a new one so that its network function can be completely preserved. This was defined as network morphism in their research. After morphing a parent network, the child network is expected to inherit the knowledge from its parent network and also has the potential to continue growing into a more powerful one with much shortened training time.

LeCun (1998) described a gradient based learning technique which is used to generate complex decisions to classify high dimensional patterns with minimal preprocessing. A new paradigm of learning called Graph Transformer Networks (GTN) is introduced, which uses gradient based methods to train multi-model systems globally, in order to minimize overall performance. A GTN is presented and deployed commercially to read bank checks.

Zhang (2000) in his paper summarized and focused on some important issues and developments in the field of neural network classification. These include feature variable selection, generalization-learning tradeoff in classification, posterior probability estimation and effect of misclassification costs. Also, the connection between neural and conventional classifiers is examined and insights are gained into neural networks performing the task of classification.

This review analyzed the techniques and state-of-the-art of Neural Networks in the field of medical imaging and showed that, NNs are not restricted by the application. The paper focused on the detection and diagnosis of breast cancer. For this purpose, medical imaging applications are used, which include mammogram, ultrasound, thermal imaging and MRI imaging. Types of NNs used, along with various types of feeding data, have been reviewed. Adaptations of hybrid NNs in breast cancer detection

are also discussed.

2.2 Development in Convolutional Neural Networks

LeCun (1989) applied back-propagation networks to classify low-resolution images handwritten digits. Because of the exceedingly constrained architecture, the network required minimal data preprocessing, as compared to feature extraction methods. The network delivered better scaling properties and shorter learning time due to the constraints imposed on the network and redundant nature of data. Normalized images of isolated digits were provided as the input. The proposed method gave around 9% rejection rate, 1% error rate and great throughput rates, and could be directly applied to larger tasks.

Jin et al. (2014) presented the concept of flattened convolutional neural networks designed for fast feedforward execution. This technique reduces the number parameters in CNN for feedforward acceleration, by converting 3D filters of convolutional layers into a sequence of 1D flattened filters across channels, and horizontal and vertical directions. The flattened networks are trained and tested on different datasets and their performances are obtained and compared with convolutional CNNs. It is found that the flattened CNNs have similar or sometimes better accuracies on datasets and once the model is trained, efforts in post processing and tuning are not requires.

Srivastava (2014) proposed the idea of dropouts, in order to deal with overfitting in DNNS. The key idea is to omit feature detecting units (along with their connections) on each training case, from the neural network during training. Big improvements are achieved on benchmark problems By using random dropout, and new records are set for object and speech recognition.

LeCun et al. (2015) presented a review paper where they described the concept, significance and standards for Deep Learning. It summarized the major differences between the conventional machine learning techniques and deep learning. Deep learn-

ing allows multiple-layered computational models to learn data representations with multiple abstraction levels and discovers complicated structures in large data sets by making use of the learning backpropagation algorithm. Backpropogation indicates how the internal parameters of a machine should change in order to compute the representations in every layer from the representations in the previous layer.

Chen et al. (2015) introduced techniques for rapidly transferring the information stored in one neural net into another neural net. The main purpose is to accelerate the training of a significantly larger neural net.

In this paper, Wu (2016) proposed Quantized CNN, which is an efficient framework that simultaneously accelerates and compresses CNN models, in order to reduce their memory and storage overhead. Network parameters are quantized to allow efficient test-phase computations. Each layer's response of the estimation error is minimized by quantizing filters in convolutional layers and weight matrices in fully-connected layers. MNIST and ILSVRC-12 datasets are used to conduct experiments. This approach achieves exceptional compression and speed-up rates, with insignificant loss in the classification accuracy. With this quantized CNN model, even mobile devices can classify images, accurately, within a second.

Iandola (2016) presented a CNN architecture called SqueezeNet, which was discovered while exploring design space of CNN architectures. It attains accuracy comparable to that of Alexnet, on ImageNet dataset with 50 times fewer parameters. They were able to further compress SqueezeNet to less than 0.5MB (510 times smaller than AlexNet), using model compression techniques. SqueezeNet is a suitable CNN architecture especially for applications that have small model sizes.

Chapter 3

Problem Description

3.1 Gaps in the Study

On the basis of the reviewed lierature, some gaps have been identified which need to be further adderessed.

1. The detection methods described above are expensive computationally, take a lot of training time, and yield not very high accuracies.
2. All CNN approaches suggested in the above section improve either the training time, or the computational expense or the accuracy. This accuracy-computation tradeoff has to be reduced.
3. The classification and detection time taken for real-world applications, is far more than the ideal time.

3.2 Objective

The work presented in this thesis, aims at the following:

1. Understanding the key concepts of Artificial Neural Networks and mathematical preliminaries associated.
2. Studying and understanding Deep-Convolutional Neural Networks, their need and functioning.

3. Progressively implementing well-known CNN architectures, for the tasks of image classification and object detection in real-time, and applying the concept of transfer learning for faster and more accurate outcomes.

Chapter 4

Methodology and Implementation

This chapter describes the implementation details of the algorithms and technology and platforms used during the course of this study.

4.1 Implementation Platforms

During the entire course of this thesis work, Python and MATLAB are used for learning and implementation purposes.

Python

Python is a high- level, general purpose programming language created by Guido van Rossom. It was conceived in 1980s and first released in 1991. A major part of implementation presented in this report, has been carried out using Python (version 3.6). This is because it has a comprehensive and large library of standard functions.

MATLAB

MATLAB is a powerful numerical computing tool, developed by Mathworks. During the course of this work, MATLAB is used to implement CNN models for image classification.

LabelImg

LabelImg is a graphical image annotation tool, written in Python. It uses Qt (cross-platform application for creating GUIs) for its graphical interface. It is used for creating bounded boxes in the images, so as to create dataset for the purpose of object detection. The annotations are saved to user-specified folders as XML files in PASCAL VOC format, which is the format used by ImageNet.

Libraries:

The following libraries are installed using Python 3.6 and pip:

1. **TensorFlow** - Tensorflow is an open-source deep-learning library created by Google Brain team. It performs numerical computations using computational graphs, where the nodes represent operations and edges represent flow of data in the form of tensors. Tensors are multidimensional matrices. After the generation of computational graphs, a session is created, that is executed by one or multiple CPUs and GPUs, in a distributed or non- distributed environment.

2. **OpenCV** – OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. It provides a common infrastructure for computer vision applications and accelerates the use of machine perception in in applications and commercial products. For this work, it is imported to allow web-camera access if tensorflow.

4.2 Algorithms and Procedural Details

The procedural details are presented out in two major steps:

1. Image Classification

2. Object Detection

4.2.1 Image Classification

As described in Section 1.4, classification is the first step towards Machine Vision. Hence, it is a basic necessity, to implement any form of Computer Vision technique. The approach followed during this study to implement classification is represented using the flowchart given in Figure 4.1. These steps are elaborated in the latter sections.

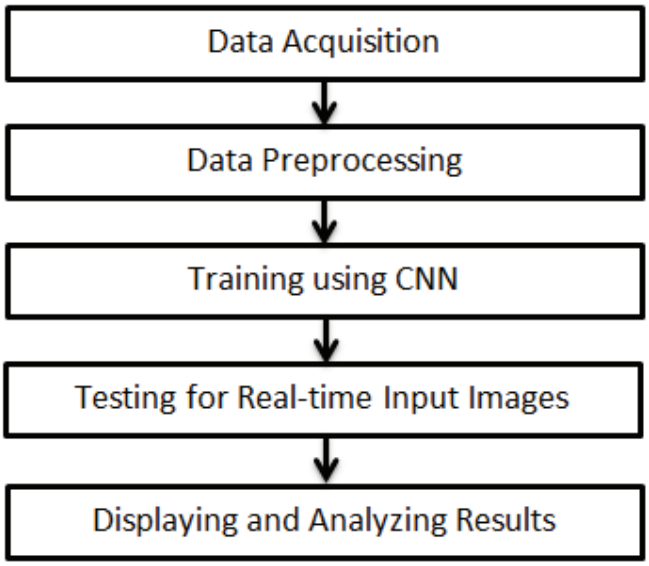


Figure 4.1: Classification Workflow

Dataset Description

The ASL (American Sign Language) Alphabet dataset is used for demonstrating classification task, in this work. Sign languages are built on the notion that vision is the most significant/vital tool possessed by a deaf person in order to communicate. The (ASL) is a complete, complex language which practices signs made by hand movements, facial expressions and body postures. It serves as the primary sign language of the deaf communities in the North American region. ASL includes a set of 26 signs known as the American manual alphabet, which are used to signify words from the English language. No form of sign language is universal.

The ASL Alphabet dataset, consists of collection of images for each alphabet in the

American Sign Language. There are 26 separate folders for alphabets A-Z, each folder containing about 3,000 images. The dataset also includes similar folders of images containing 3 special characters of ASL, namely 'SPACE', 'DELETE' and 'NOTHING', which are extremely useful in the real-time applications. This makes a total of 87,000 images and 29 classes in the dataset. The test data-set merely contains 29 images, one for each of the classes, to encourage the using real-time test images.

CNN Architecture Description

Alexnet

Alexnet is a Deep Convolutional Neural Network architecture developed by Krizhevsky et al. (2012). It was the first time that a CNN was used to solve the ILSVR Challenge and made a breakthrough by achieving the top-5 error rate of 15.3%, followed by 26.2% error rate at the second position in the 2012. For this work, Alexnet is implemented for the classification of ASL Alphabet image dataset.

Alexnet architecture takes in an input image of size 227*227*3. It consists of 5 convolutional layers with 96 filters (each of size 11*11*3), each followed by a ReLU operation for non-linearity and 3 fully-connected layers. Dropout layers are included in order to overcome the problem of overfitting. Max- pooling is used. The detailed layer description is given in Figure 4.2.

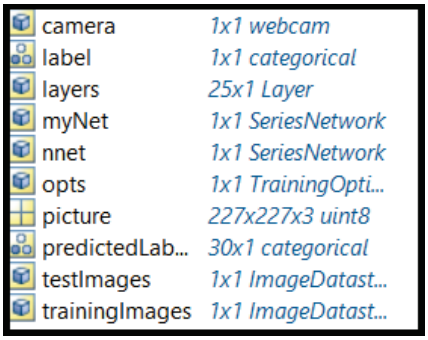


Figure 4.2: Network Details

1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	''	Fully Connected	2 fully connected layer
24	'prob'	Softmax	softmax
25	''	Classification Output	crossentropyex

Figure 4.3: AlexNet Layer Description

Training

As mentioned in the previous section, the AlexNet architecture is pre-trained on 2 GTX-580 GPUs for 5-6 days, for classifying 1000 possible categories and over a million images from ImageNet dataset Deng et al. (2009).

For this thesis, the ASL Alphabet dataset is trained using transfer learning with Alexnet, in order to achieve higher accuracy. This is executed on the MATLAB tool. Training settings are described below :

Table 4.1: Training Cycle Details

Training Time	71hrs 21min
No. of epochs	6
Iterations per epoch	489
Maximum Iterations	2934
Initial Learning Rate	0.0001
Learning Rate Schedule	Constant

4.2.2 Object Detection

In addition to classifying the network, object detection also involves training the regions of interest i.e., the localized regions where objects from the existing classes are present. The steps followed to implement object detection are shown with the help of the flowchart given in Figure 4.5.

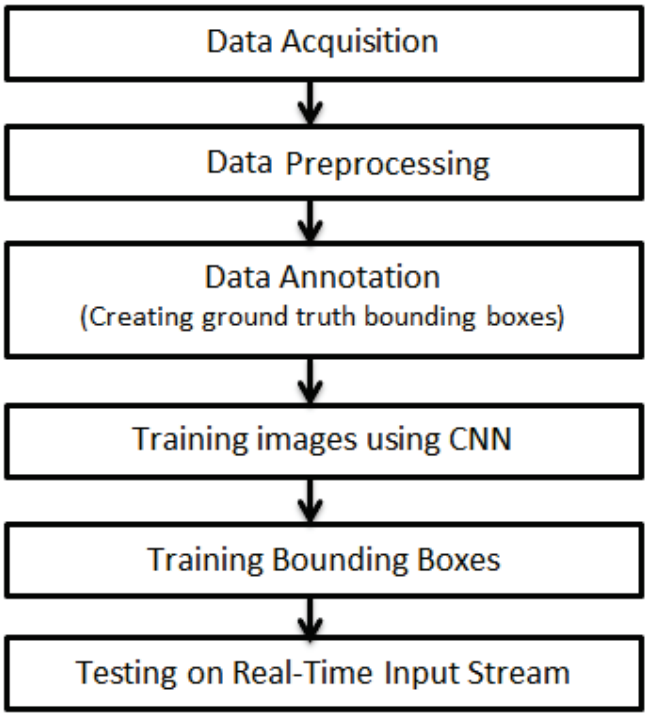


Figure 4.4: Object Detection Workflow

Dataset Description

The sparse availability of labeled data, was one of the obstacles faced in the early era of Machine Learning. However, in the present day, many image datasets are available on open platforms, which meet the rapidly increasing demands for larger datasets and are convenient to be used by students and researchers for learning purposes. Following are the datasets, used for studying object detection in this work:

1. MS COCO - MS COCO is elaborated as MicroSoft Common Objects in COntext dataset. COCO is a large-scale object detection, segmentation, and captioning

dataset. It contains over 320,000 images of the objects that can be commonly seen in our everyday lives. The dataset has 91 classes and contains about 2,500,000 objects of interest or instances. Precise pixel-level segmentation gives the spatial location of each object. Additionally, this dataset has a good no. of labeled instances per image, which is a critical distinction as compared to the other object-detection datasets available. This may aid to learn contextual information. This dataset is downloaded from the Common Visual Data Foundation (2018).

2. ASL Alphabet - The ASL Alphabet dataset (described in Section 4.1.1) is used. The dataset is annotated using the LabelImg tool. Since the dataset is being applied for detection purposes, ground truth regions or bounding boxes are created for all the objects of interest in the image and the objects in each box are labelled. The dataset was downloaded from Kaggle (2018).

CNN Architecture Description

This study involves progressively implementing object detection using the paramount CNN architectures available. A brief description of the architectures is given in the following sections.

Faster R-CNN

As discussed in section 1.5.1, the region proposal step of Fast R-CNNs was found to be a bottleneck, since it was still fairly slow. To overcome this, an architecture by the name Faster R-CNN was introduced in 2015, by Ross Girshick's team at Microsoft Research, which made the region proposal process almost cost free. The basic idea behind Faster R-CNNs is to reuse the CNN results, which are used to calculate the image features, for generating region proposals i.e., a single CNN is used for generating region proposals as well as classification. Therefore, the entire process needs only one CNN to be trained. The inputs and outputs of this model are shown in Table 4.4.

In Faster R-CNN, Selective search is replaced with Region Proposal Network, for ROI generation. The RPN passes a sliding window over the CNN feature map and at each window, and outputs the potential bounding boxes and their scores. Maintaining

Table 4.2: Inputs and outputs of regression model in R-CNN

Input	Images (Region proposals not needed)
Output	1. Class labels and scores 2. Bounding boxes

Table 4.3: Inputs and outputs of regression model in R-CNN

Input	CNN Feature Map.
Output	1. Bounding box per anchor 2. Class Score

aspect ratios is significant for object detection because bounding boxes may be used to localize a variety of objects, for example, some boxes are required to resemble human shapes, some to resemble very small object, some to represent thin objects etc. The concept of anchor boxes (set of common aspect ratios) is introduced to handle the scale and aspect ratio variations of the objects.

In such a way, we create k such common aspect ratios we call anchor boxes. For each such anchor box, we output one bounding box and score per position in the image. The inputs and outputs of Region Proposal Network (wrt anchor boxes) are shown in Table 4.5.

Bounding boxes that have high scores or that are likely to contain the object of interest, are fed into Fast R-CNN for classification and optimization.

1	''	Image Input	32x32x3 images with 'zerocenter' normalization
2	''	Convolution	32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
3	''	ReLU	ReLU
4	''	Convolution	32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
5	''	ReLU	ReLU
6	''	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
7	''	Fully Connected	64 fully connected layer
8	''	ReLU	ReLU
9	''	Fully Connected	2 fully connected layer
10	''	Softmax	softmax
11	''	Classification Output	crossentropyex

Figure 4.5: Faster R-CNN Layer Description

MobileNet-SSD

In the previously discussed methods a part of the network was dedicated to provide vision proposals, and then a classifier of higher quality was used to classify the proposals. Thus come at great computational cost so both the stars are combined into one network. This can be achieved by having a set of predefined bounded boxes to look for object instances, rather than having a network to produce proposals.

All running CNN, spatial size of the input image in is compressed at each convolutional layer. Therefore, in the final layers, each pixel represents a larger area of image. Hence, pixels can be used to determine the object position sense is pixel represents large area of the input image.At this point a 1*1 convolutional layer can classify each cell as a class. Figure 4.7 shows the Mobilenet Layer Architecture.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 4.6: MobileNet Architecture Description

In this implementation MobileNet architecture is used for CNN in SSD. MobileNets are efficient models proposed mainly for mobile and embedded vision applications. These allow developers to build light weight deep neural networks using depthwise separable convolutions for their applications. MobileNets are based on a streamlined architecture. The standard convolutional is slow to perform. The core layer for these, is known as Depthwise Separable Convolutional layer.

4.2.3 Training

Both the datasets are trained using Google's Tensorflow Object Detection API. It is an open source framework built on TensorFlow and makes it easier to construct, train and deploy object detection models. Following are the steps to do so:

- **Installing the Object Detection API and its dependencies**

  The list of dependencies required for this API include tensorflow, Cython, protobuf-

compiler, pillow, lxml and conrtextlib2.

- **Annotating images using LableImg**

  All the images are fed into the LableImg tool. Rectangular boxes are drawn over the objects of interest present in them and all the boxes are hand-labeled. These act as the ground truth bounding boxes. Figures 4.8, 4.9 and 4.10 show the process of creating and labeling bounding boxes using LabelImg software.

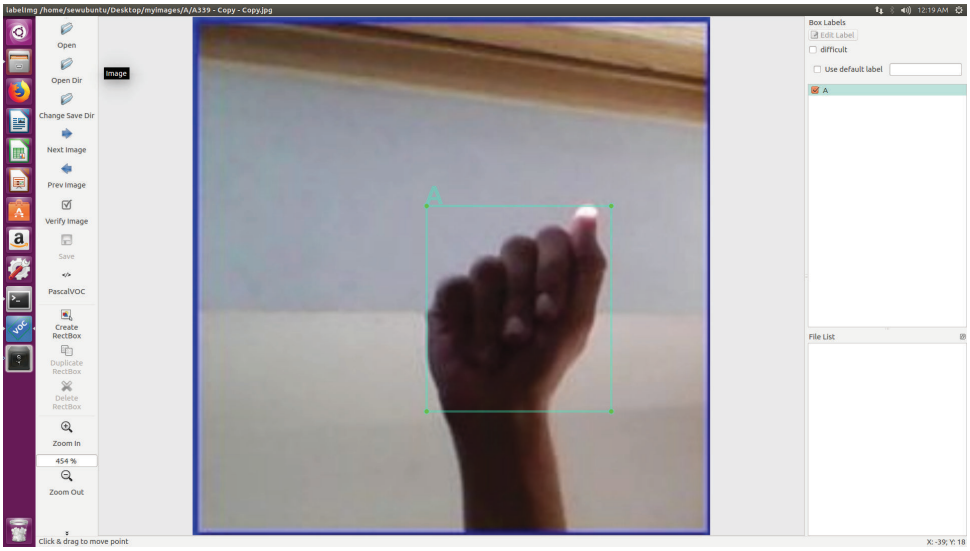


Figure 4.7: LabelImg: Creating Bounding Boxes

This annotated image information is stored in the form of .xml files.

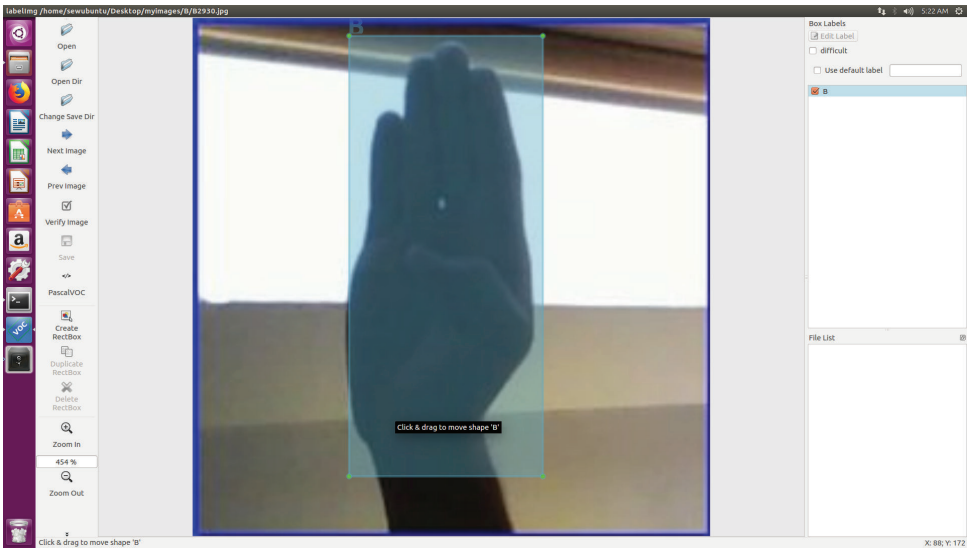
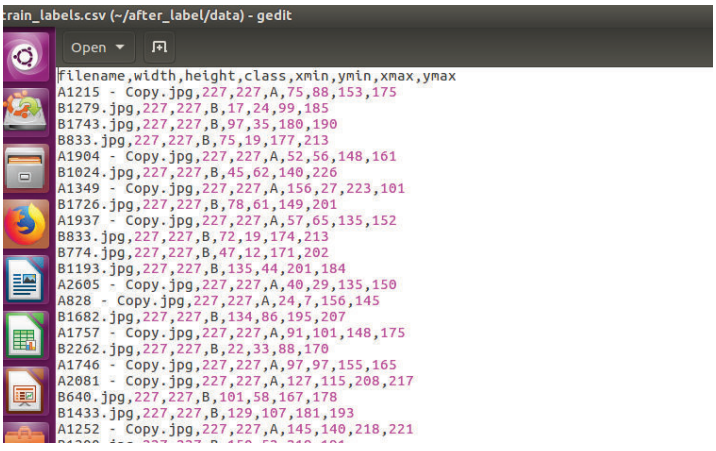


Figure 4.8: LabelImg: Labelling Objects

- **Converting the .xml files to .csv file**

  To obtain this annotated data in a single, readable file, .xml files are converted to .csv files. A screenshot is attached (Figure 4.11) to show the format of the .csv file obtained.

Figure 4.9: CSV file format



- **Generating tf-records using .csv files**

  Tf-record is tensorflow's native format that is used to shuffle, batch and split datasets with its own functions. These contain features as a field.

- **Training tf-records file using pipelining**

  Initializing the process of training is displayed in Figure 4.12 and Figure 4.13 is a screenshot of training executing. After the completion of training, inference graphs are obtained, which are then used for detecting objects and generating qualitative results.

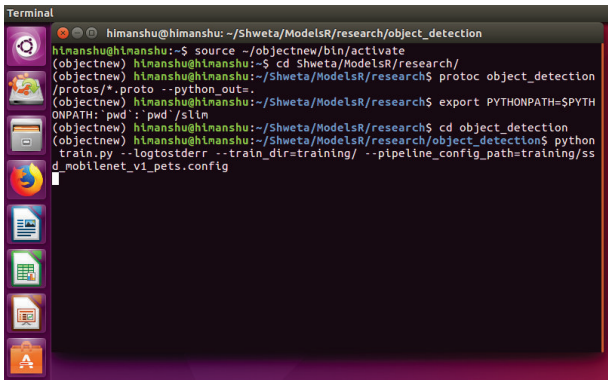


Figure 4.10: Training command

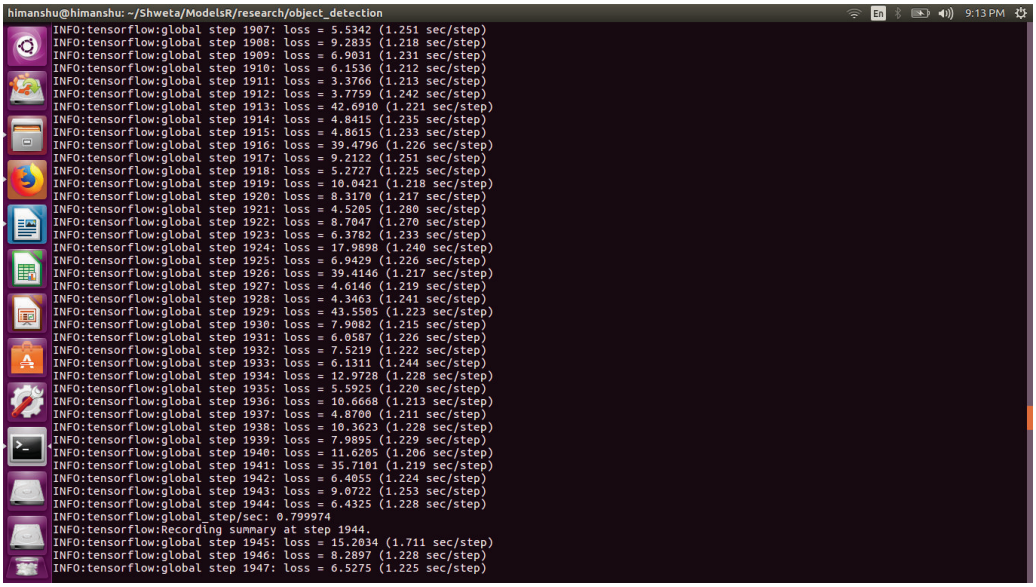


Figure 4.11: Training

Chapter 5

Results

This chapter discusses the results obtained after the training step. The results are arranged in 2 sections for each problem namely; Image classification and Object detection.

5.1 Image Classification

This section presents the outcomes of the implemented classification model, as discussed in Section 4.2.1. The outcomes are arranged in 2 sections:

- **Qualitative Results** - Outcomes of the actual classification process.
- **Quantitative Results** - Outcomes obtained during training the CNN model.

5.1.1 Qualitative Results

The output images and labels obtained after classification are displayed in this section.

Grids of qualitative results for classes A to Z and NOTHING are shown are displayed. Figures 5.1 shows signs A to I, Figure 5.2 shows signs J to R and Figure 5.3 shows signs S to Z, and 'NOTHING'. Figure 5.4 and Figure 5.5 show the classes 'DELETE' and 'SPACE' respectively.

This is real-time classification, since the labels are generated for a continuous input stream of images with the help of a web camera.

Figure 5.1: A to H Classification

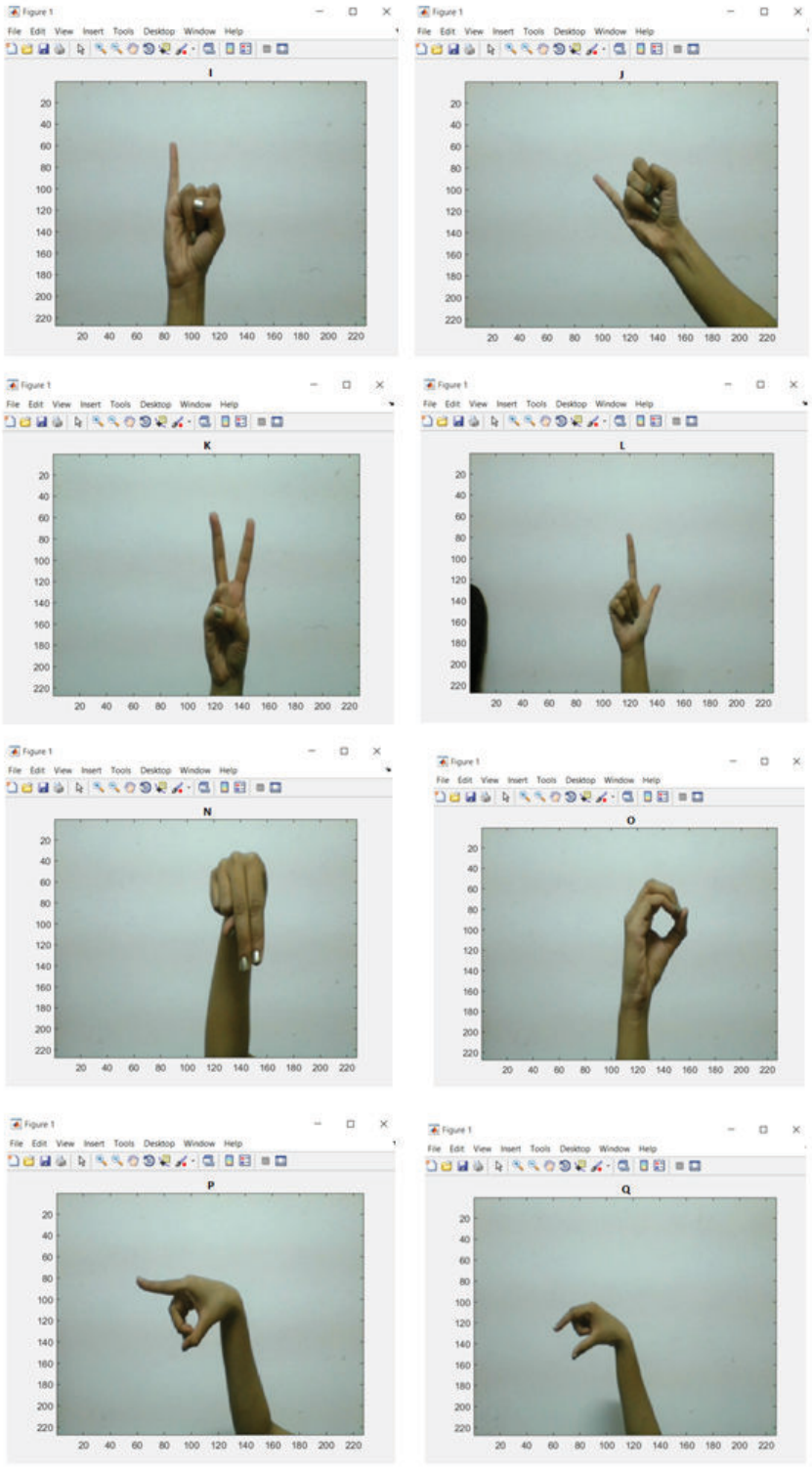


Figure 5.2: I to P Classification

Figure 5.3: Q to X Classification

Figure 5.4: Y, Z, 'DELETE', 'SPACE', 'NOTHING' sign Classification

5.1.2 Quantitative Results

The graphs in Figure 5.5 depicts the accuracy and Figure 5.6 depicts loss values, during the training process for AlexxNet, for image classification. The accuracy curve rapidly increases from under 10% to over 85% in epoch 1. Overall training accuracy for AlexNet came out to be 91.70%. Table 5.1 presents the training results. Tables 5.2 displays the test accuracy of all the component signs of the ASL Alphabet dataset.

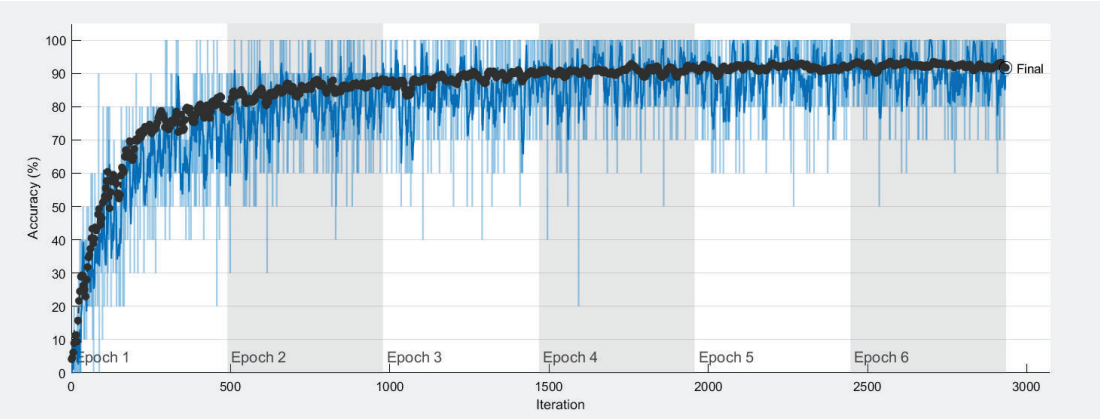


Figure 5.5: Training Accuracy for AlexNet

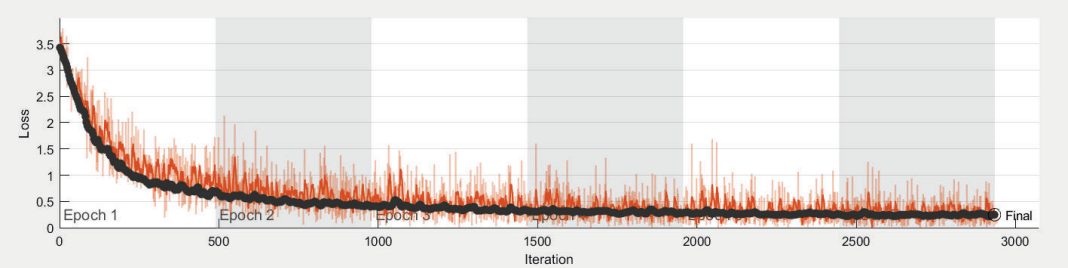


Figure 5.6: Training Loss for AlexNet

Table 5.1: Training Results

Training Time	71hrs 21min
Training Accuracy	91.70%

Table 5.2: AlexNet: Test Accuracy for signs A to Z

Symbol	Total no. of Test Images/Signs	Correctly Classified Signs	Accuracy
A	60	54	88.70%
B	60	58	96.33%
C	60	57	90.73%
D	60	59	89.01%
E	60	50	78.42%
F	60	60	90.21%
G	60	56	94.00%
H	60	54	92.28%
I	60	58	95.84%
J	60	56	92.14%
K	60	54	86.90%
L	60	58	93.78%
M	60	57	91.34%
N	60	59	92.99%
O	60	59	96.31%
P	60	52	86.22%
Q	60	55	87.90%
S	60	49	78.30%
T	60	50	82.05%
U	60	55	91.00%
V	60	53	83.77%
W	60	58	92.21%
X	60	51	86.25%
Y	60	57	94.46%
Z	60	53	92.55%
SPACE	60	60	98.02%
DELETE	60	60	96.91%
NOTHING	60	55	90.99%

5.2 Object Detection

The outcomes of the implemented Object Detection models are discussed in this section. These outcomes are presented in 2 sections:

- Qualitative Results - Displays the outcomes of the process of detection.
- Quantitative Results - Plots formed during the training of the CNN models.

5.2.1 Qualitative Results

This section displays the results of the actual detection process, i.e., that takes place after the training of the CNN model, using a web-cam. These are discussed with respect to two datasets; COCO and ASL Alphabet.

Results for COCO data detection:

The section displays screenshots of Object Detection process for COCO dataset, trained using Mobilenet-SSD and Faster R-CNN. Figures 5.7, 5.8, 5.9, 5.10 are screenshots for real-time object detection systems.

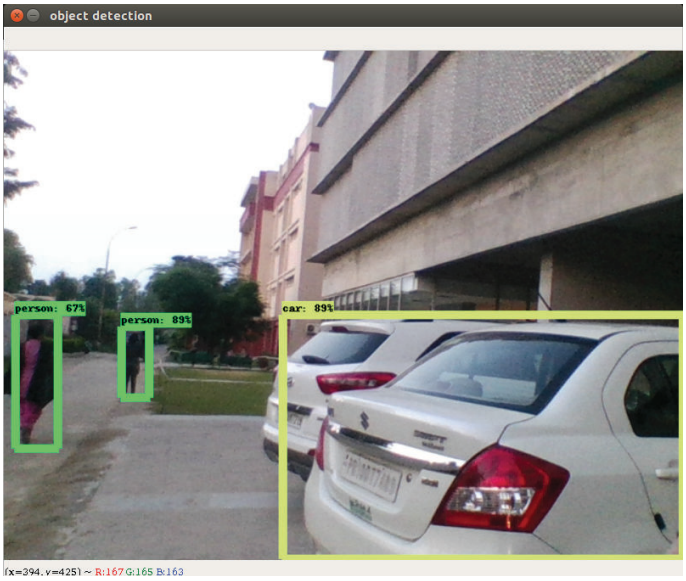


Figure 5.7: Detection: Mobilenet-SSD

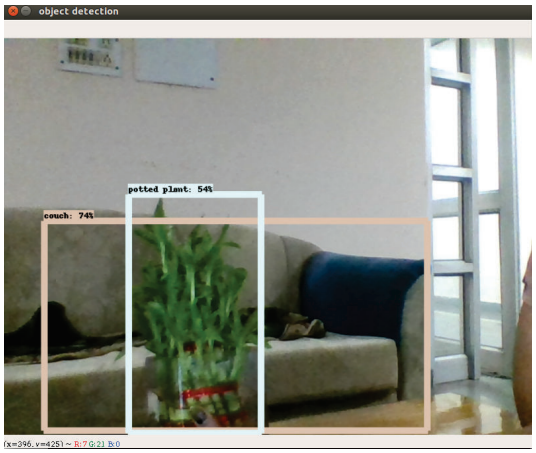


Figure 5.8: Detection: COCO Mobilenet- SSD

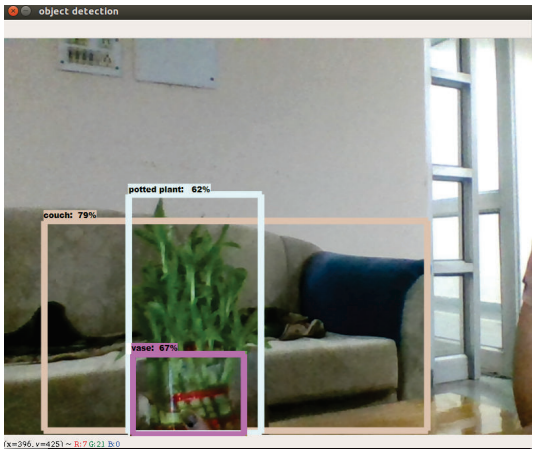


Figure 5.9: Detection: COCO Faster R-CNN

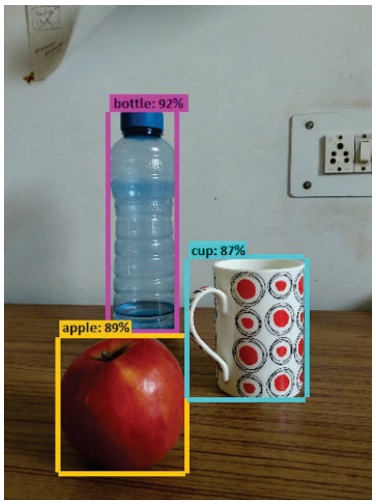


Figure 5.10: Detection: Faster R-CNN

The following observations during the detection process:

- Detection time taken by Mobilenet-SSD is quite less than by Faster-RCNN.
- Faster R-CNN is able to detect more object instances than Mobilenet-SSD.
- Faster R-CNN gives higher class scores for classes than Mobilenet-SSD.

Results for ASL Alphabet detection

Since it is observed that Faster R-CNN gives results with higher accuracy, it is used for the real-time detection of ASL signs. Screenshots are displayed in Figure 5.11 and 5.12. Image regions containing the sign alphabets are localized using bounding boxes, and the validation accuracy of each bounding box i.e., each sign, is displayed. From top-left to bottom right sequence, W, A, C, K, L, Y are the displayed alphabets made using hand gestures.

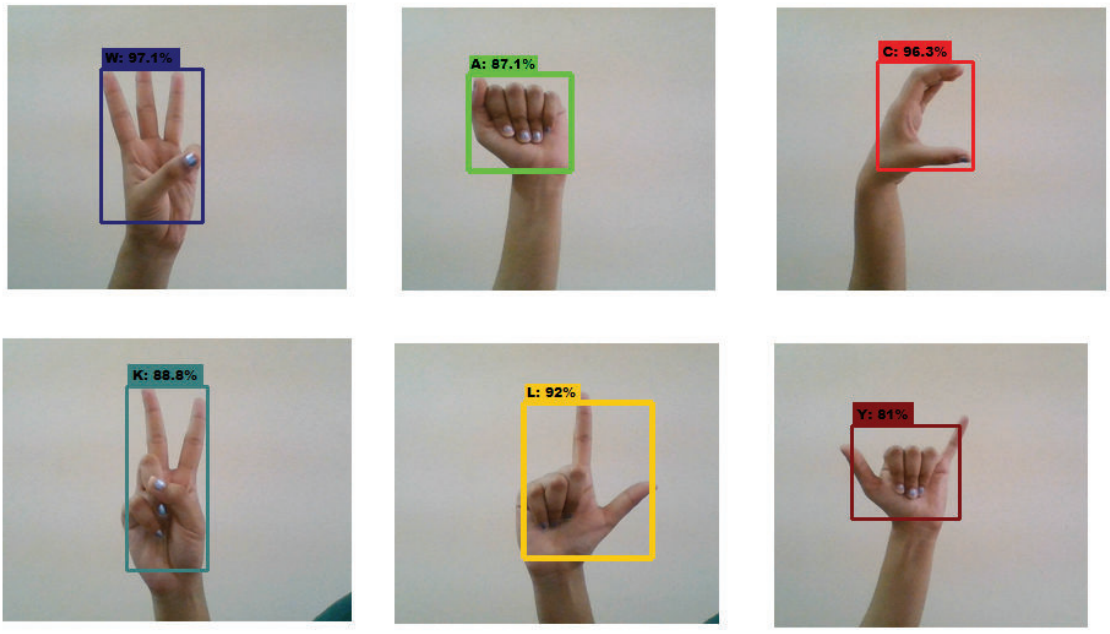


Figure 5.11: Object Detection: ASL dataset

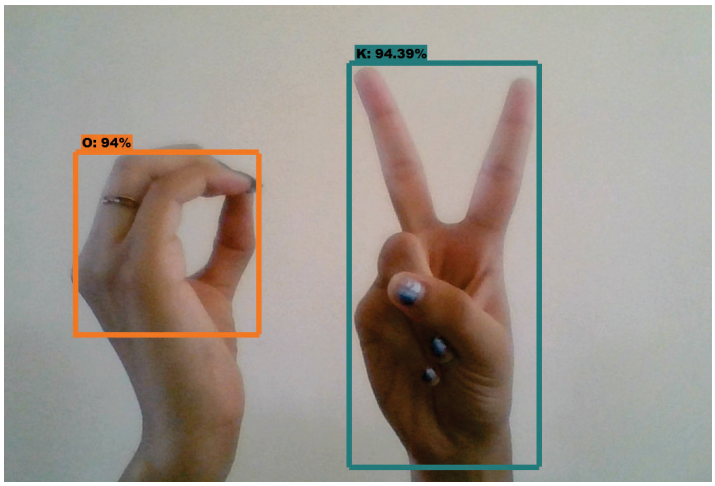


Figure 5.12: Object Detection for ASL

5.2.2 Quantitative Results

The loss-curves obtained during the training process are displayed in this section. Figure 5.14 shows loss in Faster R-CNN for 10,000 steps and Figure 5.15 shows loss in MobileNet-SSD for 10,000 training steps.

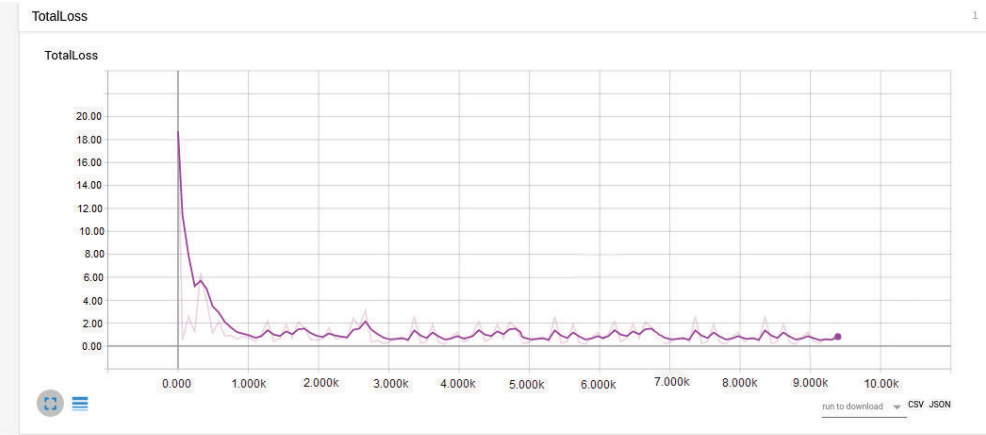


Figure 5.13: Faster R-CNN

Table 5.3: Faster R-CNN Loss

Step No.	Loss	Step	Loss
1	41.886	6000	1.823
1000	2.333	7000	1.076
2000	1.568	8000	0.997
3000	1.667	9000	1.506
4000	1.800	10,000	1.480
5000	1.006		

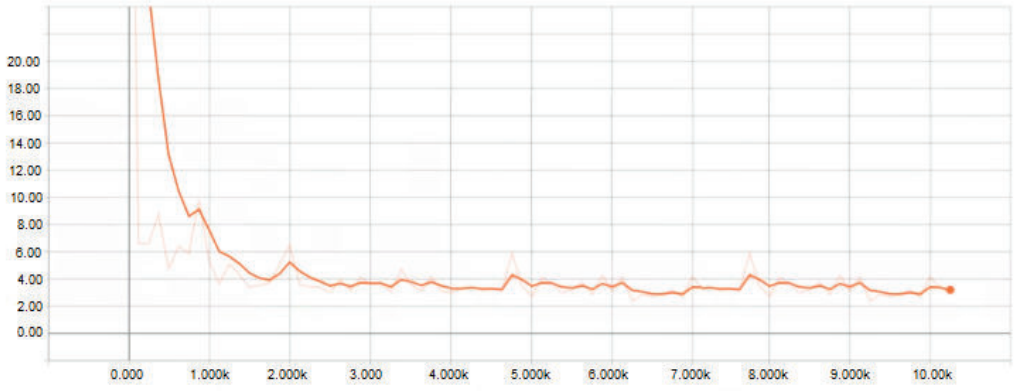


Figure 5.14: MobileNet-SSD

Table 5.4: MobileNet-SSD Loss

Step	**Loss**	**Step**	**Loss**
1	43.9227	6000	4.5522
1000	9.500	7000	6.8377
2000	9.0722	8000	3.9224
3000	7.5986	9000	4.8938
4000	8.3489	10,000	3.3276
5000	7.4106		

From the above plots, it is evident that Faster R-CNN converges much faster than Mobilenet-SSD. Also final loss value for Mobilenet-SSD is 3.326, which is more than 1.480, the loss value of Faster R-CNN .

Application of CNN for image classification and object detection

ORIGINALITY REPORT

11%	9%	5%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	2%
2	en.wikipedia.org Internet Source	2%
3	www.nature.com Internet Source	1%
4	Submitted to International American University Student Paper	1%
5	cs231n.github.io Internet Source	1%
6	proceedings.mlr.press Internet Source	1%
7	Submitted to King's College Student Paper	<1%
8	Submitted to University of Florida Student Paper	<1%
9	brage.bibsys.no	