OBJECT DETECTION IN VIDEO BASED ON TRANSFER LEARNING USING CONVOLUTION NEURAL NETWORK

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

Master of Engineering
in
Computer Science and Engineering

Submitted By
Pranav Kapur
(801632037)

Under the supervision of:
Dr. Parteek Bhatia
Associate Professor, CSED

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

June 2018

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Object Detection in Video Based on Transfer Learning using Convolution Neural Network*", 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. Parteek Bhatia* 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.

Signature:

(Pranav Kapur)

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

(Dr. Parteek Bhatia)
Associate Professor
CSED

Acknowledgement

First of all, I would like to express my gratitude towards **Thapar University**, for pro-viding me a platform to do my thesis work at such an esteemed institute.

I wish to express my respect, deep sense of gratitude and indebtedness to my guide **Dr. Parteek Bhatia**, Associate Professor, Computer Science And Engineering Department, Thapar University, Patiala for their invaluable and enthusiastic guidance, useful suggestions, unfailing patience and sustained encouragement throughout this work.

I would like to thank **Dr. Maninder Singh**, Head of Computer Science And Engineering Department, Thapar University, Patiala for kind help, guidance, encouragement and providing the necessary facilities to carry out my research. I am indebted to the faculty members of the department for valuable suggestions, friendly support and full cooperation rendered by all of them.

Any vote of thanks is not enough to thank the supreme power "**The GOD**" one who has always guided me to work on the right path of the life. Without his grace, this would never come to be today's reality. With special thanks, I dedicate this thesis to GOD.

Last but not the least I would like to thank my **family** and my friends specially **Sonia Mittal, Kanika Sharma, Sheena Nanda, Aditi Thakur** and **Sukriti Sharma** for their help and support throughout the thesis.

Pranav Kapur

Abstract

Object Detection has been an active area of research and development since the past few years and due to its diverse applications it continues to be a challenging research topic. It is indeed an evident fact that convolution neural network have shown a remarkable progress for various vision related tasks such as image classification object detection etc.

In this thesis, we introduce a complete framework of object detection in a real time video using the concept of transfer learning. In this thesis the model used to train the system is a Deep Neural Net. The whole concept of how deep neural net learns to recognize the pattern in the data is termed as Deep Learning. It originates from Machine Learning which is basically a sub-branch of Artificial Intelligence (AI). Deep Learning is mainly based on Artificial Neural Network with a concept to mimic the human brain i.e. Artificial Neural Networks are computational models which work similar to the functioning of human brain. Here, we use convolution neural networks to train our system on ImageNet CIFAR-10 dataset and use this trained system to detect objects in YTO dataset. We have applied various image processing techniques so as to achieve a better accuracy in comparison to the state-of-the-art methods for object detection.

Basically, in this thesis, the technique that we have incorporated is transfer learning. Transfer Learning, as the name indicates is a machine learning method where the system make use of the knowledge gained while solving one problem and applying it different but related problem. Here, we are training our system with the image dataset (CIFAR-10) and we are testing it on video dataset and videos are basically collection of related and continuous images.

Our framework has an accuracy of 85.99 on our own dataset and an accuracy of 61.96% on YTO dataset which is better than the state-of-art results.

Table of Contents

Title **Page No.**

Certificate……..... **i**

Acknowledgement…… **ii**

Abstract…… **iii**

Table of Contents…… **iv**

List of Figures……. **vi**

List of Tables…….. **vii**

Chapter 1 A Brief Introduction to Object Detection And Convolution Neural Networks…… **1**

1.1 Introduction…… 1

1.2 Why We Used Neural Networks…… 2

1.3 Types of Neural Networks Used in Machine Learning……. 3

1.3.1 Feed Forward Neural Network……... 3

1.3.2 Radial Basis Function Neural Network…….. 4

1.3.3 Kohonen Self-Organizing Neural Network…… 5

1.3.4 Recurrent Neural Network……. 5

1.3.5 Convolution Neural Network……. 6

1.3.6 Modular Neural Network…… 6

1.4 Why We Choose Convolution Neural Network Over Simple Neural Network… 6

1.5 Layers And Other Parameters of Convolution Neural Network……..... 7

1.5.1 Convolution Layer…….. 8

1.5.2 Pooling Layer……. 8

1.5.3 Fully Connected Layer……... 9

1.5.4 Padding……... 9

1.5.5 Stride……... 10

1.5.6 Rectified Linear Unit…….. 11

1.5.7 Back Propagation……..... 11

1.6 What is Transfer Learning …… 11

1.7 Applications Of Transfer Learning ……... 12

1.8 Steps Used in Object Detection……. 13

Chapter 2 Literature Review……. **15**

Chapter 3 Problem Statement……... **18**

3.1 Problem Formulation ……. 18

3.2 Solution …….. 19

3.3 Research Objectives ……... 19

Chapter 4 Data Collection, Pre-Processing And Segmentation……. **20**

4.1 Data Collection ……..... 20

4.1.1 CIFAR-10 Dataset ……. 20

4.1.2 Youtube Objects Dataset ……... 21

4.2 Data-Preprocessing ……..... 22

Chapter 5 Implementation And Experimental Results…….. **25**

5.1 How does Tensorflow works ……. 25

5.2 Hardware and Software Requirements …….. 26

5.3 Tensorflow Installation ……. 26

5.4 Implementation ……. 27

5.4.1 Implement Deep Learning Model …… 27

5.4.2 Training Phase …….. 28

5.5 Results & Discussions ……... 29

Chapter 6 Conclusion And Future Work……. **34**

6.1 Introduction …… 34

6.2 Summary of Contribution …….. 34

6.3 Future Work ……... 35

References…… **36**

List of Publications…….. **37**

List of Figures

Figure No.	Title	Page No.
1.1	Simple neural network…..……	1
1.2	MultiLayer perceptron neural network…….	2
1.3	Eucledian distance based Radian based function.....	4
1.4	Basic structure of RNN...……..	5
1.5	Visual Representation of Modular Recurrent Neural Network…….	6
1.6	A very general architecture of Convolution neural Network……	7
1.7	Simplified example of CNN.....	8
1.8	Example of Max Pooling layer having kernel window size of 2*2……..	9
1.9	Example of zero padding on 6 * 6 volume size image……..	10
1.10	Example of stride value of 2, applied on 7 * 7 that will output 3 * 3 image.	11
1.11	Methodology…….....	14
3.1	Various approaches in the field of Artificial intelligence to mimic human brain…….....	18
4.1	Ten classes with ten random images in CIFAR-10 dataset…….	21
5.1	How optimization flow occurs……	25
5.2	General architecture of CNN……..	28
5.3	Graph representing the nature of decrease of the training and validation loss as the number of epochs increase……	29
5.4	Methodology for our own video dataset…….	30
5.5	Accuracy Graph for our own video dataset……	31
5.6	Comparison of accuracy of our proposed system with Prest. Et al.[1] method……	32
5.7	Comparison of accuracy of our proposed system with Jouline. Et al.[2] method……	32
5.8	Comparison of accuracy of our proposed system with Kwak. Et al.[3] method……..	32

List of Tables

Table No.	Title	Page No.
4.1	Videos corresponding to each class of objects……	22
5.1	Values of training loss and validation loss as the number of epoch increase…….	28
5.2	Accuracy Graph for our own video dataset…….	30
5.3	Accuracy Analysis of each class of objects…….	31
5.4	Comparison of results of our proposed system with State-of-the-Art Results……	32

Chapter 1

A Brief Introduction to Object Detection & Convolution Neural Network

1.1 Introduction

Question Detection has been a functioning zone of innovative work since the past few years and due to its diverse applications, it continues to be a challenging research topic. Object Detection is essentially a computer innovation that is firmly identified with computer vision and image processing, that handles the acknowledgment of semantic objects of a specific class, (example humans, animals, cars etc) in computerized images and videos.

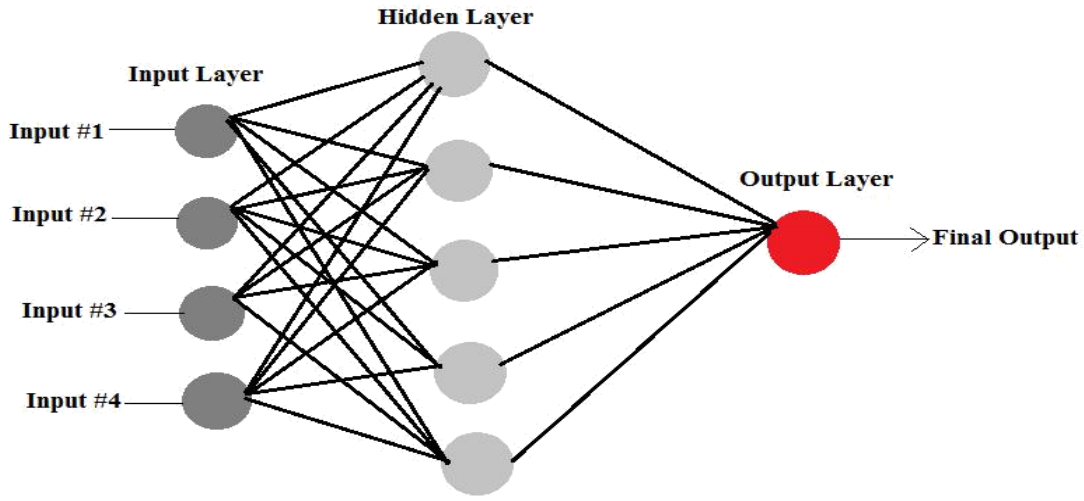

Figure 1.1 Simplified Neural Network

The prime aim of the thesis is to train the system with CIFAR-10 image dataset and to detect objects in a real time video. In this thesis the model used to train the system is a Deep Neural Net. The whole concept of how deep neural net learns to recognize the pattern in the data is termed as Deep Learning. It originates from Machine Learning which is fundamentally a sub-branch of Artificial Intelligence (AI). Deep Learning is mainly based on Artificial Neural Network with a concept to mimic the human brain i.e. Artificial Neural Networks (ANN) are mathematical and computational models which work like the working of human cerebrum. There are indeed 6 types

of ANNs that are currently being used in machine learning. These kinds of systems are executed in light of the mathematical expressions and parameters that decide the yield.

1.2 Why We Choose Neural Networks?

Artificial Neural Network (ANN) is proven to be a paradigm of information processing that follows the same paradigm as followed by the biological nervous system such as the human brain processes information. Artificial Neural Network fundamentally comprises of countless processing elements alluded to as neurons that work as one to solve complex problems. Many specific applications have been configured by the Artificial Intelligence like pattern recognition, following a learning process.

Neural networks takes different approaches to tackle a problem than used by traditional computers. In order to tackle a problem, traditional computers use algorithmic approach in which computer has to perform a specific set of instructions. In the event that the particular advances that computer needs to take after are not known to the computer, it can't have the capacity to determine the critical thinking ability of conventional computers.

On the contrary, neural networks processes information in the same way as the human brain does. Neural network learn by example. Number of layers constitutes to form a neural network. Number of interconnected nodes combined to form a neural network layers which includes an activation function. Different examples are given as a contribution to the information layer that is additionally associated with at least one hidden layers which goes about as the brain of neural system where the real processing is handled. The hidden layers at that point connect to a yield layer which gives us the required outcome.

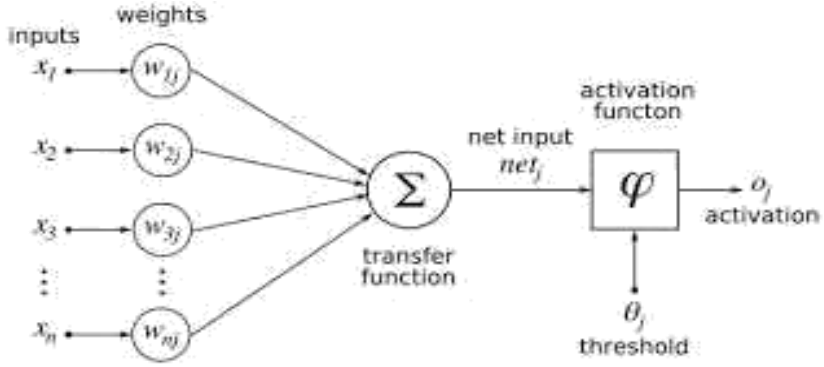

Figure 1.2 Multilayer Perceptron neural network

A more sophisticated example of neuron is shown in Figure 1.1. Here the weighted inputs are used, decision of the neural network depends on the weight of corresponding input. The inputs provided of the input layer are multiplied with their respective weight values are performed and then if computed sum is more than the pre-set threshold value, the neuron fires. If it is less than pre-set threshold value, the neuron does not fire. A simple example of how the multiplayer perceptron neural network works is shown in the above Figure where

$$x_1, x_2, x_3, \ldots, x_n \tag{1.1}$$

are the inputs;

$$w_{1j}, w_{2j}, w_{3j}, \ldots, w_{nj} \tag{1.2}$$

are the respective weights.

Mathematically, the neuron if fired if and only if;

$$x_1w_1+x_2w_2+x_3w_3+\ldots\ldots\ldots\ldots+x_nw_n > T$$

where, T denotes the specified pre-set threshold value.

The summation of input weights and of the threshold makes this neuron an uncommonly versatile and intense one. The multilayer perceptron can conform to a particular circumstance by changing its weight and threshold value. Back propagation can be used to update the values of weights and biases at the end of each epoch.

1.3 Types of Neural Networks Used in Machine Learning

There are varies sorts of Neural Networks. These kind of systems are executed in view of the scientific activities that includes mathematical expressions and the parameters that actually decide the implementation.

1.3.1 Feed-Forward Network – Artificial Neuron:-

This neural framework is one of the least cumbersome sort of ANN, where the data goes one way. The information experiences the info hubs and exit on the yield hubs. This neural framework may comprises of hidden layers. In straight forward words, it has a front propagated wave and no back multiplication by using an activation work.

Figure 1.3 is an example of a Single layer feed forward network. In this figure, the aggregation of the input is evaluated and then fed to the output. The yield is considered on the off chance that it is over a specific esteem i.e. edge (generally 0) and the neuron fires with an enacted yield (typically 1) and on the off chance that it doesn't fire, the deactivated esteem is radiated (normally 1). An example of this type is X-beam image fusion.

1.3.2 Radial basis function Neural Network:-

Radial basis function basically takes into consideration the distance between the center and the point on the periphery. Thus, RBF consists of 2 layers, first one being the layer where the features get associated with the Radial Basis. Inner layer functions and the outcome corresponding to these functions forms the basis for evaluating the same outcome in the succeeding step, which is the memory.

Figure 1.3 represents the distance between the center and the point in the periphery. Here, the distance measure used is Euclidean distance.

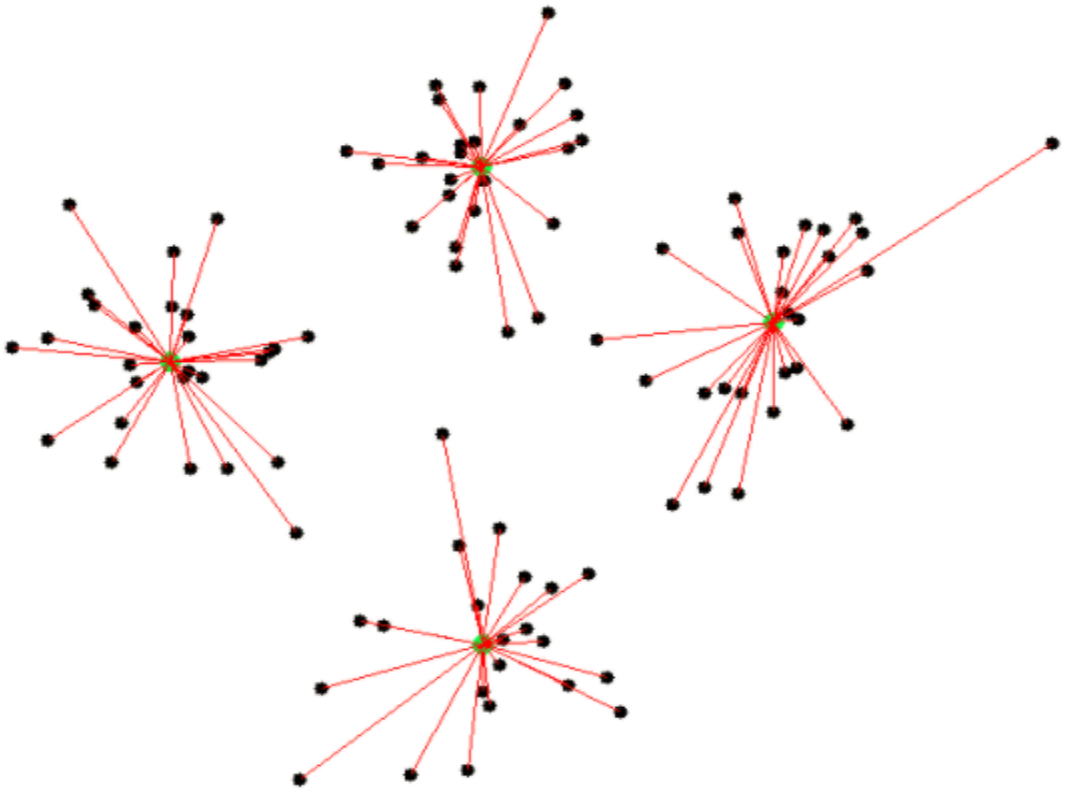

Figure 1.3 Euclidean Distance based Radian Based Function

1.3.3 Kohonen Neural Network:-

The prime goal of kohonen map is to enter vectors to a discrete guide included neurons. The map should be prepared to make its own association of preparing information. Amid training the map, the area of the neurons needs to stay steady, yet the weights contrast contingent upon the esteem. The self-association process has different stages. In the principal stage, every neuron stage is instated with a little weight and info vector. Amid the second stage, the neuron nearest closest to the point is the triumphant neuron and the neuron associated with the triumphant neuron will likewise move towards the point.

1.3.4 Recurrent Neural Network (RNN):-

The Recurrent Neural Network takes a shot at the prime rule of sparing the yield of the layer and thus nourishing them back to the contribution to anticipating the result of the next layer. Here, the principal layer is framed like the feed forward neural system with the result of the total of the weights and the bias. The neural system process begins once this is figured, this implies starting with one time step then onto the next every neuron will recollect some data it had in the past time-step. This makes every neuron representation like a memory cell in performing calculations. In this procedure, we have to let the neural system to take a shot at the front proliferation and recall what data it requirements for later utilize. Here, if the forecast isn't right we utilize the learning rate or blunder revision to roll out little improvements with the goal that it will continuously work towards making the correct expectation of the back proliferation. RNN is shown in the figure below:

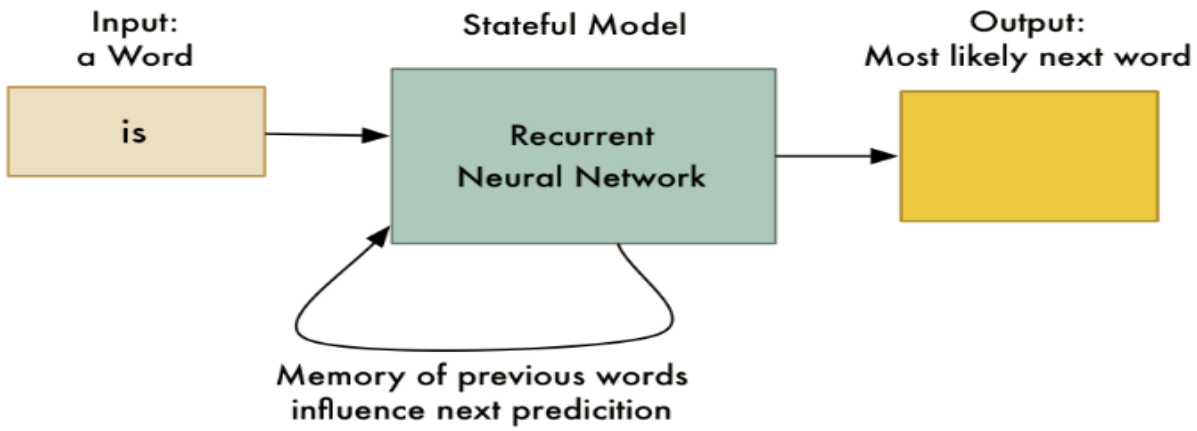

Figure 1.4: Basic Structure of RNN

1.3.5 Convolution Neural Network (CNN):-

Convolution Neural Network are indeed very similar in approach to the first type- feed forward network, where the neurons have parameters as weights and bias. It has various layers: Convolution layer, Pooling, reLu, fully connected layer. We will discuss about each of these layers along with padding, stride etc in detail later.

1.3.6 Modular Neural Network:-

Modular Neural Network can be considered as a whole bunch of different neural networks that perform the task in segregation to each other and hence generate the required output. Every network of this modular neural network has inputs and does separate sub-task as shown in fig 1.5 below:

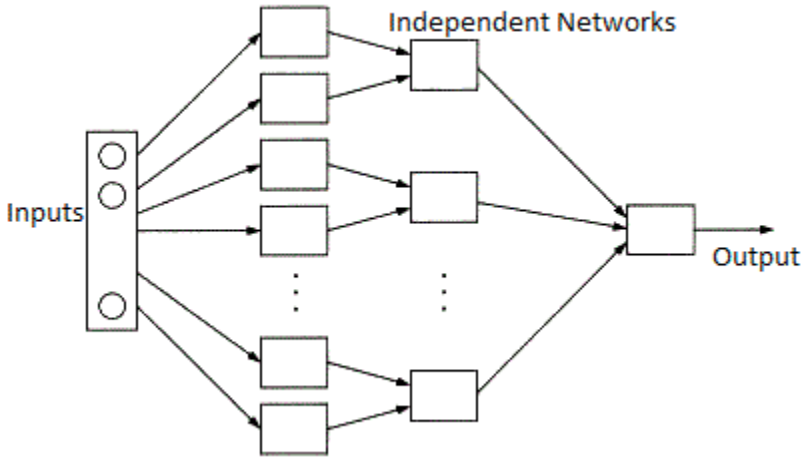

Figure 1.5: Visual Representation of Modular Neural Network

1.4 Why We Choose Convolution Neural Network Over Simple Neural Network?

Recurrent Neural Networks receives an input and convert it using a cascading hidden layers to respective output. Each hidden layer consists of set of neurons where each neuron is attached to all neurons in the previous layer in fully connected manner and neurons present in a single layer function are independent completely and have not any connections in shared manner. The last fully connected layer used in the network named as output layer and in classification it represents the

class scores. The problem occurs say if we have images of size 200*200*3 that would lead to 1,20,000 weights which is wasteful and huge number of parameters often results in over fitting.

On the other hand, the main advantage of using Convolution Neural Networks (CNNs) is that the images are provided as input that constrain the architecture in a more sensible way. The layers of a Convolution network have neurons arranged in three dimensions: width, height and depth unlike from regular neural networks. Depth belongs to number of channels used. For grey scale images depth is equal to one and for colored images depth is equal to three: for red, blue and green colors. Here, the neurons in a layer will be connected not to all neurons in the previous layer in a fully connected manner, rather have connections with a small regions of the layer proceeding it. The final output of the convolution layer network will be a vector array of size equal to the total number of available classes, because by the end of convolution network architecture, input full size image is converted into a single vector of class probabilities that are arranged along the depth dimension.

1.5 Layers And Other Parameters of Convolution Neural Network:-

As shown below, Convolution Neural Network consists of a number of layers. It basically uses all these layers to compute the calculations via the small region of the layer referred as activation map window.

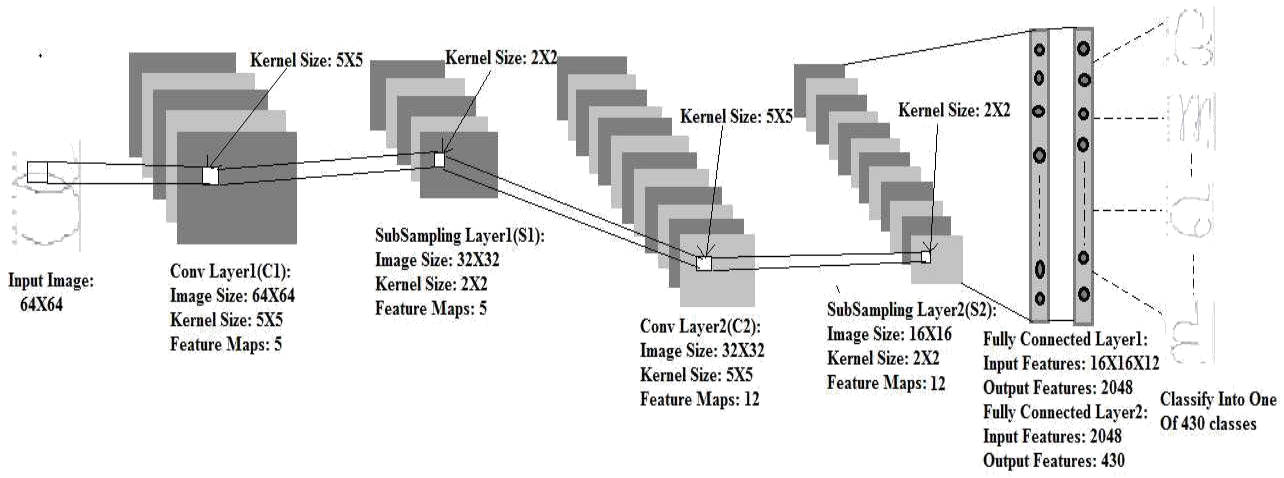

Figure 1.6: A very general architecture of Convolution Neural Network

1.5.1 Convolution Layer

The first layer used in Convolution Neural Network is the Convolution layer. As shown in the above figure, a small window called as the kernel window or filter window slide across all the regions of the input image. The regions covered by it is called the receptive field. This field basically consists of an array of numbers which are weight values or parameters. The depth of this filter will be same as that of the depth of input image. The initial position of the filter window will be at the top left corner of the input image. Now as the filter is convolving around the input image, it multiplies the values in the filter with original pixel values of the image. Then these multiplications are summed up to get a single number. In this way, we repeat this process for every position on input volume. Every unique location on the input volume produces a number. After sliding the filter all the possible locations of the input image, you get activation map or feature map. The second convolution layer then works on this layer.

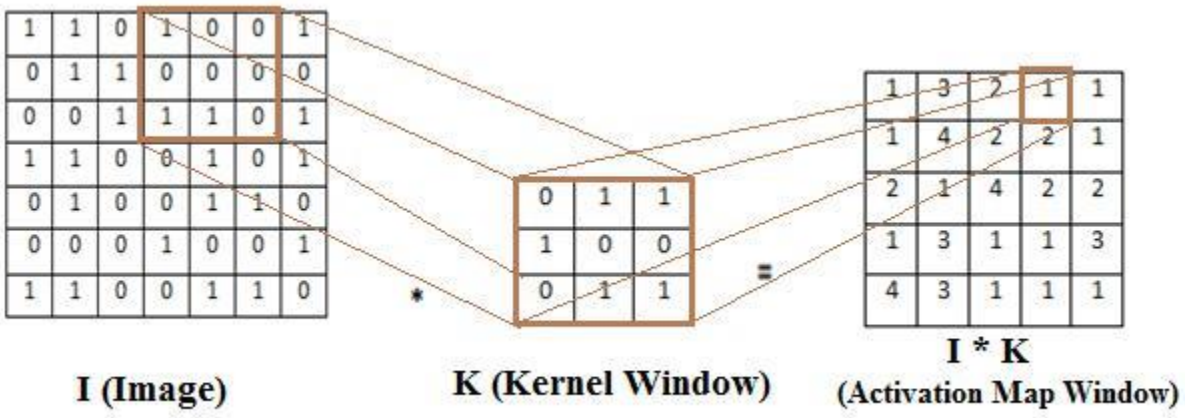

Figure 1.7: Simplified example of Convolution Layer

1.5.2 Pooing Layer

Pooling layer is generally in between the two convolution layers. The function of pooling layer is to constituently reduce the spatial dimensions of the input image, thus reducing the dimensionality of input image and computations required in the network, thereby controlling the rate of overfitting. Every depth slice of input image is taken into consideration by this pooling layer that resizes it spatially using AVERAGE or MAX operation. In Max operation, it picks the highest value amongst the kernel size window i.e. 2*2 window. The figure below shows the max pooling operation on kernel window of size 2*2.

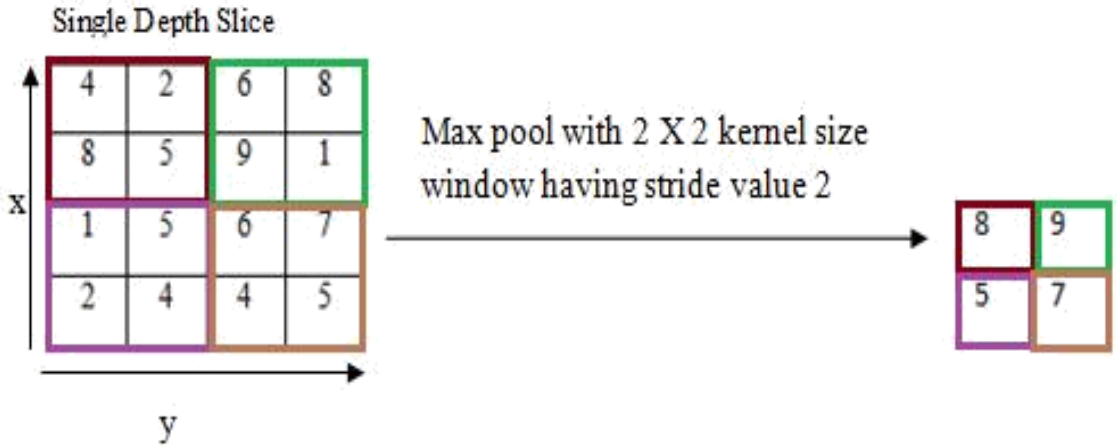

Figure 1.8: Example of max pooling layer having kernel window size of 2*2

1.5.3 Fully Connected Layer

Fully Connected Layer basically consists of neurons that are fully connected to all the neurons that are present in its previous layer, in the same way as in regular neural networks. Then we perform the matrix multiplication followed by bias offset, which is also referred to activation. The main difference that lies the fully connected layer and the convolution layer is that in the convolution layer, neurons are connected only to a small local region in the input and also they share parameters. However, the neurons in both the layers compute the dot product, thus their functional form is identical. Fully connected layer takes an input volume which will be the output of its previous layer and it outputs an N dimensional vector where N is the number of classes from which the program has to select from. Each number in the N dimensional vector represents the probability of a certain class. The way in which fully connected layer functions is that it looks at output of the previous layer and determines which feature must correlate to a particular class. Class containing the highest probability will be considered as the winning class in which we finally classify the input dataset.

1.5.4 Padding

The spatial dimensions of the input image goes on decreasing as we keep applying convolution layers. We want to preserve as much information about the input image, so that we can extract the low level features. Say we want to apply the same convolution layer but we want the output volume to remain 32*32*3. For this to happen, we can apply a zero padding to that layer. For this to

happen, we can apply a zero padding to that layer. The zero padding pads the input with zero around its border. The value of zero padding to be used can be calculated as follows:

$$\textbf{Zero Padding} = (K-1)/2 \qquad (1.3)$$

where, K is the filter window size. The formula used for calculating the output size for any given convolution layer is given below:

$$O = ((W - K + 2P) / S) + 1 \qquad (1.4)$$

where, O is the output length/height, W is the input height/length, K is the filter size, P is the padding and S is the stride.

0	0	0	0	0	0
0	2	5	1	8	0
0	9	1	8	5	0
0	1	6	7	7	0
0	4	2	1	9	0
0	0	0	0	0	0

Figure 1.9: Example of zero padding on 6*6 volume size image

1.5.5 Stride

Stride determines how the filter convolves around the input volume. If stride value is one, the filter convolves around the input volume by shifting one unit at a time. The amount by which the filter shifts is referred as stride.

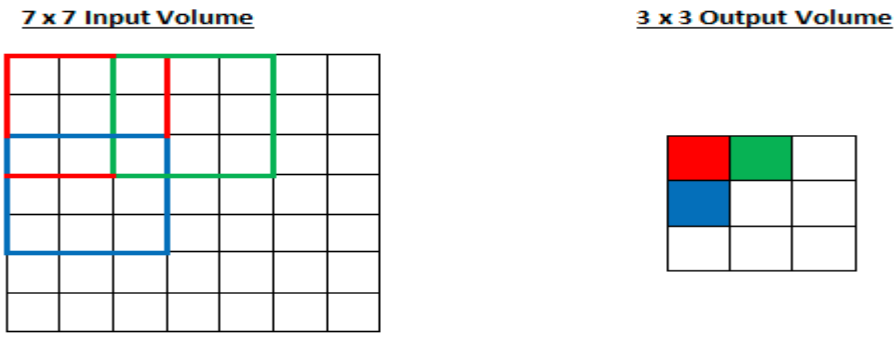

Figure 1.10: Example of stride value of 2 applied on 7*7 that will output 3*3 volume image

1.5.6 Rectified Linear Units (reLu):

After each convolution layer, it is necessary to apply a non-linear layer or activation layer immediately afterwards. The prime objective of this layer is to introduce non-linearity to the system that basically has been computing operations during convolution layers (element-wise multiplications and summations). reLu activation functions are better than tanh and sigmoidal functions as reLu layer works far better because the network is able to train faster without making a significant difference to the accuracy. The reLu layer applies the following functions:

$$\mathbf{F(x) = max(0,x)} \qquad \mathbf{(1.5)}$$

1.5.7 BackPropagation:

Backpropagation can be segregated into four segments: the loss function, forward pass, backward pass and weight update. In the training pass, the input is the training image and it is then passed through the whole network. For the first training example, since all of the weights or filter values were initialized randomly, the output will be something like [.1 .1 .1 .1 .1 .1 .1 .1 .1 .1], ie the output does not have any preference to any class in particular. Thus the network, with its current weights, is not able to look for those low level features or hence it does not make any reasonable conclusion about the classification. This goes to the loss function part of backpropagation. The loss function can be defined in terms of Mean Squared Error (MSE) which can be computed as ½ $(\text{actual-predicted})^2$. The Loss will be very high for the first few training images. Our main goal is to reach a point where the predicted label which is the output of the convolution network is same as training label. In order to achieve this target, we want to minimize this loss value. Thus, we perform backward pass through the network which determines the appropriate weights that we must set so as to have minimum loss value.

1.6 What is Transfer Learning

Transfer learning is fundamentally a machine learning strategy where a model outlined and produced for an undertaking is utilized as a beginning stage for a model on a second model. Consequently, Transfer learning is a streamlining that permits quick advance and enhanced execution when displaying the second model.

There are basically two approaches for transfer learning:

- Develop the Model Approach:
    - Selection of Source Task: We have to choose a related predictive modelling problem that has bounteous of information and there is a predictive relationship in the input information and yield information; additionally between ideas took in the mapping from input information to output information.
    - Development of Source Model: We have to build up an skillful model. This model must be superior to the naive model with the end goal that some feature learning has been performed.
    - Reusing the Model: The model that is fit on the source task can be utilized as beginning stage for the model on second undertaking.
    - Tuning Model: This is an optional step. The model may have to be tuned corresponding to the input-output pair.
- Pre-Trained Model Approach:
    - Selection of Source Task: A pre-trained research model is selected from the list of models.
    - Reuse Model: The model that is fit on the source task can be utilized as beginning stage for the model on second undertaking.
    - Tune Model: This is an optional step. The model may have to be tuned corresponding to the input-output pair.

1.7 Applications Of Transfer Learning

The 2 basic applications of transfer learning are as follows:

- Transfer Learning on Image Data: It is indeed common to use transfer learning technique for image data. For such tasks, the system is trained with ImageNet 1000-class photograph and then test it on a different related dataset. We have used this approach in our thesis.
- Transfer Learning on Language Data: Transfer Learning can likewise be utilized with Natural Language Processing issues that utilizes text as input or output. For these sorts, a word embedding is utilized that is a mapping of words to a high-dimensional ceaseless vector space where distinctive words with a comparative importance have a comparative vector representation.

1.8 Steps Used in Object Detection

On a broader aspect, our model consists of three phases: model building, Training and testing respectively. Firstly, our CNN model is build consisting of input layer of size 1*28*28; two convolution layers of size 32*24*24 and 32*8*8 respectively. Also, max pool operation is performed corresponding to the two convolution layers having filter size of 32*12*12 and 32*8*8 respectively. The model consists of two dropout layers of size 32*4*4 and 256 respectively; one fully connected layer and one output layer consisting of 10 neurons corresponding to the ten classes.

In the training phase, a CNN model is trained using 60k images, of 32*32*3 dimension corresponding to 10 object classes with 6000 images per class.
Now this trained model is tested on YouTube-Objects Dataset. The YouTube-objects dataset is fundamentally made out of videos extracted from YouTube of 10 object classes. It consists between 9 to 24 videos for each class. The length of each video may vary from 30 sec to 3 minutes.

Figure 1.11 shows the methodology of our project. Our project basically consists of five phases:

- Implementation of deep Learning Model
- Training Phase
- Preprocessing the testing dataset
- Testing Phase
- Output Evaluation

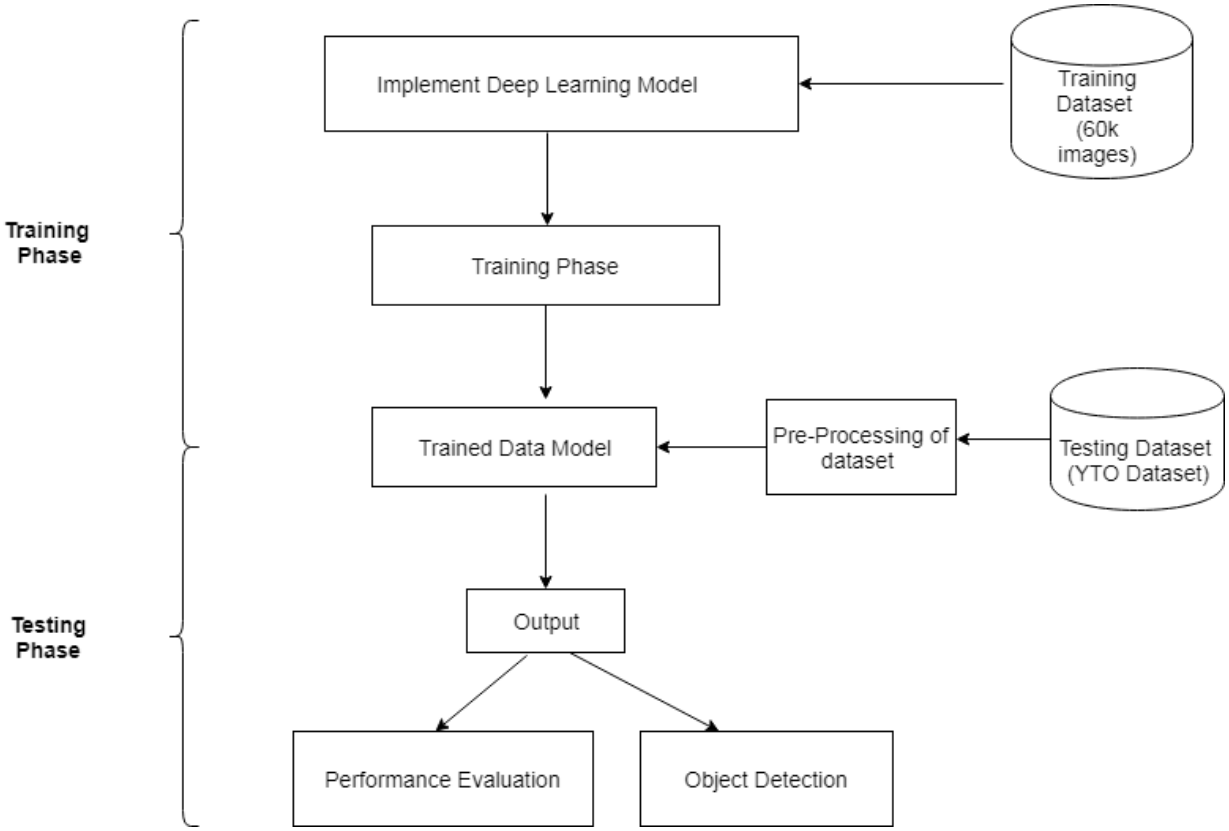

Figure 1.11: Methodology

Chapter 2

Literature Survey

In this chapter, the previous research carried by various researchers in the field of deep learning and transfer learning has been discussed.

Geoff Hinton in University of Toronto was one of the main analyst to device a leap forward thought for preparing deep net. His approach prompt the making of the RBM and DBN. As a result of his spearheading work, he is alluded to as father of machine learning.

Prest.et.al [1] introduced the concept of object detection by training the system with weakly annotated videos. The approach extracts candidate's spatio-temporal tubes in view of movement division and after that chooses one tube for every video together finished all videos. We will compare our results with its results.

Jouline et. al [2] tackles the problem of co-localization in images and videos. Furthermore, they also implement Frank-Wolfe algorithm to demonstrate how the enhancement plan for the two images and videos can be decreased to unraveling a progression of basic whole integer programs, prompting expanded proficiency in memory and speed. The accuracy of the framework presented by the authors is 53.9%.

Kwak et. al [3] tends to the issue of consequently localizing dominant objects as spatio-transient tubes in an uproarious collection of videos with minimal supervision. The task of detecting objects is essentially a mix of two assignments: discovery and tracking.

Kang[14] used Deep Convolution Neural Network for classifying images, detection of objects and semantic segmentation. He utilized the ImageNet object detection from video dataset to assess the general pipeline and individual segment of the proposed system. He even assessed the execution on the YTO dataset for the object localization task. Testing accuracy of his model was 47.5%.

Oquab, Bottou, Laptev, Sivic [2] used Convolution Neural Network for detecting images for VOC dataset. They also used the concept of transfer learning as they trained their model using ImageNet dataset and tested it on VOC dataset that is similar to the ImageNet dataset.

Shao, Loy, kang and Wang[38] utilized the Slicing Convolution Neural Network (S-CNN) to show its viability on the www crowd video dataset for attribute recognition and to watch critical execution upgrades to the state-of-art methods. Their model yielded an accuracy of 60.55%.

LeCun et al.[16] used the technique of backpropagation network for the recognition of handwritten digits. The error rate of 1% was achieved when he applied this technique on zipcode digits.

Lawrence et al.[5] designed a system for face recognition using convolution neural networks. Here, the database consisted of 10 images of each of 40 individuals thereby a total of 400 images. The system extract the deeper features using cascading set of layers. This system had an error rate of 3.8 %.

Girshick et al. [24] presented a multi-stage pipeline called as Regions with Convolutional Neural Networks (R-CNN) to prepare deep CNN so as to characterize regions proposals with the end goal of object detection. Essentially it arranges the object detection issue into different stages that incorporates the CNN pre-training, CNN fine-tuning, SVM training, and bouncing box regression. This sort of structure has demonstrated great execution and thus it was adopted by different techniques.

Szegedy et al. [25] presents the GoogLeNet that had a 22-layer structure and "inception" modules with a specific end goal to supplant the CNN in the R-CNN, which won the ILSVRC 2014 object detection task.

Ouyang et al. [26] presented a deformation obliged pooling layer and a box pre-training methodology, which accomplishes an accuracy of 50.3% on the ILSVRC 2014 test set.

To accelerate the preparation of the R-CNN pipeline, Fast R-CNN [27] is presented, where every image patch was never again wrapped to a steady size before being being fed to the Convolution Neural Network. In any case, rather, the relating highlights were cropped from the output feature map of last convolutional layer.

In the arrangement of Faster R-CNN pipeline [28], the bouncing box proposition were made by the Region Proposal Network (RPN), and thusly the general structure would thus be able to be set up in a conclusion to-end way. Be that as it may, fundamentally these pipelines are to be sure for detecting objects in still images. However, when these strategies are connected to movement

pictures or recordings, they may miss some positive examples. The reason may be that the objects may not be of their best poses in every frame of the videos.

Object tracking has been studied for decades [32,33,34]. CNN has been utilized for detection of objects and has accomplished a noteworthy following precision [35,36]. Wang et al. [36] planned a system to make a protest specific tracker by web based picking the most remarkable highlights from an ImageNet pre-prepared CNN, which has great outcomes.

Nam et al. [37] orchestrated a multi-region CNN for learning non representations for tracking objects. When following another objective, another system is made by consolidating the common layers in the pre-prepared CNN with another parallel classification layer, which is online refreshed. Be that as it may, be it CNN-based trackers, they may even now float in whole deal following since they for the most part utilize the protest appearance inside the video without semantic appreciation on its class.

Chapter 3

Problem Statement

3.1 Problem Formulation & Research Gaps

Image classification essentially refers to the assignment of separating data classes from a multi band raster picture. The subsequent raster from picture order can be utilized to make topical maps. Thus image classification is a huge area of research. Deep Learning has almost dominated computer vision over the last few years, achieving top scores on many tasks and their related competitions. The most popular and well known of these is the ImageNet.

Coming to Neural network, Deep Convolution Neural Networks (CNNs) have demonstrated a noteworthy and amazing execution in different vision assignments, for example, image classification and semantic division. For object Detection, particularly in still pictures the execution has been fundamentally expanded in a year ago because of powerful deep networks (GoogleNet).

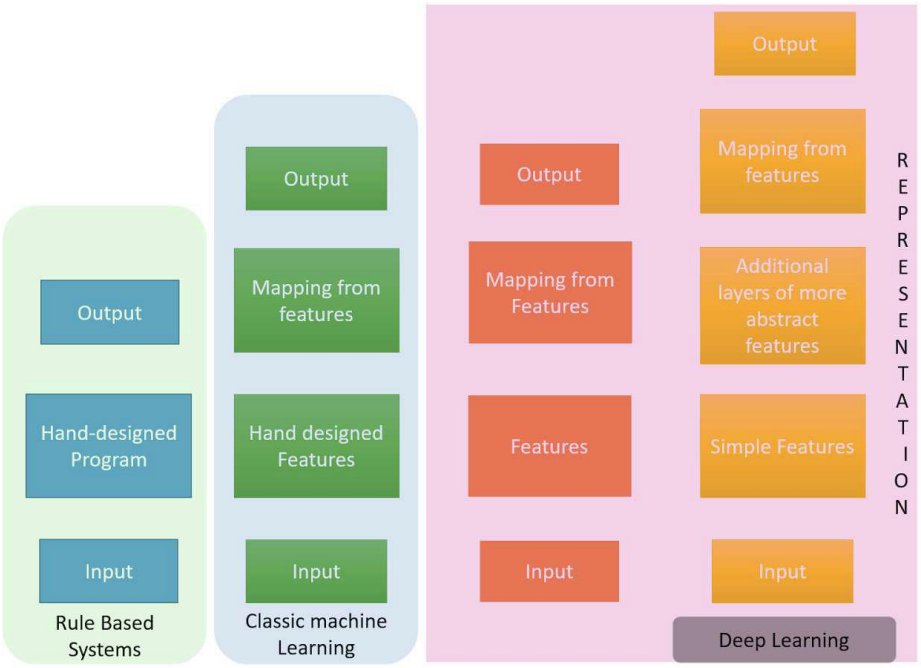

Figure 3.1: Various approaches in field of AI applied to mimic human brain

Object detection has been one of the most challenging research areas in recent times. The success of Convolution Neural Network for detecting objects in still images is evident. But, to detect objects

in case of real-time videos and to classify them into known set of classes efficiently is indeed a pretty challenging task.

3.2 Solution

In this thesis, we introduce a framework based on convolution neural network (CNN) for object detection in videos. The framework basically consists of three phases – a) Model building using CNN. b) Model Training on ImageNet CIFAR-10 dataset. c) Model Testing on YTO dataset. Here, we integrate the concept of Transfer Learning, an optimization technique that provides improved performance while modelling the second task. We have also incorporated various image processing algorithms on the testing video frames, to increase the accuracy of the model.

3.3 Research Objectives

Based on the literature survey, the objectives of the current research work can be listed as:

1. To propose a framework for object detection in video based on Transfer Learning using the concept of Convolution Neural Network.
2. To train our proposed model with the ImageNet dataset.
3. To implement Image processing technique on the image frames generated to achieve better accuracy.
4. To test the proposed model on a real-time video dataset and formulate its accuracy.
5. To test the dataset on YTO (YouTube-objects) dataset, to compare our results with the bench-mark results .

Chapter 4

Data Collection, Pre-Processing And Segmentation

4.1 Data Collection:

For the implementation of our thesis, we have used two dataset. We have used the CIFAR-10 dataset for training the model and YTO dataset for testing the system using the concept of transfer learning.

4.1.1 The CIFAR-10 dataset:

The CIFAR-10 dataset basically consists of 60,000 images, of 32*32 pixel, belonging to 10 classes. With 6,000 images per class. By default there are 50,000 training and 10,000 test images.

The dataset is partitioned into five training and one testing group, each with 10,000 pictures. The test bunch contains precisely 1000 randomly– chose pictures from each class. The training batch contain the rest of the pictures in random fashion, yet some training groups may contain a bigger number of pictures from one class than another.

The compress file of CIFAR-10 dataset comprises of five files – data_batch_1, data_batch_2, data_batch_3, data_batch_4 and test_batch. Every one of these document is a python "pickled" object generated with cPickle. Shown below, is a python 3 function which opens a file and then returns a dictionary:

```
Def unpick(file):
    import pickle
    with open(file, ‘rb’) as f:
        dic=pickle.load(f, encoding=’bytes’)
    return dic
```

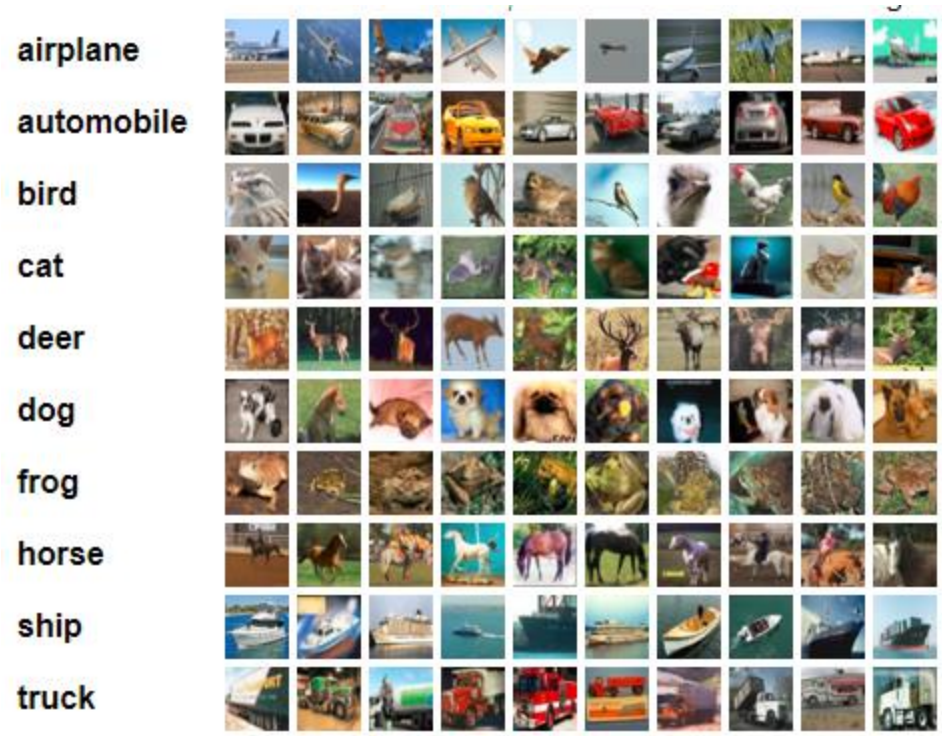

Figure 4.1 shows the classes of the dataset, with 10 random images corresponding to each class:

4.1.2 Youtube-Objects dataset:

The Youtube-Objects dataset is made out of vides gathered from Youtube belonging to ten classes. It contains in the vicinity of 9 and 24 recordings from each class. The time of every video shifts between 30 seconds to 3 minutes. The videos are annotated weakly.

Table 4.1 below shows the videos corresponding to each class and the video-frames associated with each class of objects.

Tab.4.1**:** Videos corresponding to each class of objects.

Class	**Videos**	**Shots**	**Frames**
Aero	13	1097	71,327
Bird	16	205	27,532
Boat	17	606	74,501

Car	9	208	14,129
Cat	21	220	42,785
Cow	11	212	29,642
Dog	24	982	82,432
Horse	15	432	70,247
Mbike	14	511	40,604
Train	15	1,034	1,17,890

Thus, the dataset contains a total of 570,000 frames.

4.2 Data-Preprocessing

Firstly, we obtain the video frames from the videos. Here, is a python code snippet that is used to extract image frames from the videos:

```
countVideo = 0
successRate = True
while successRate:
  successRate,image = vidcap.read()
  print ('Read next frame:')
  cv2.imwrite("frame%d.jpg" % countVideo, image)
  countVideo += 1
```

The video frames obtained may differ from each other on various aspects such as pixels, contrast and may also contain factors of noise. Thus, first we resize the image to 32*32 pixels and then we use watershed algorithm that basically segments the image such that it is easier to analyze.

Chapter 5

Implementation & Experimental Results

The prime purpose of using Tensorflow for the implementation of Convolution Neural Network is that it internally uses a computational graph for execution purpose which is indeed more efficient in comparison to the calculations performed by python directly. Tensorflow is even better than numpy as in numpy, at any particular instant of time only a single mathematical operation is known, while in case of Tensorflow, an entire computational graph is executed.

Tensorflow also provides inbuilt mathematical techniques to evaluate gradient values that in-turn results in the optimization of variable values to enhance the accuracy of the model. Tensorflow is basically an open source library provided by google for performing numerical computations using data flow graphs. Graphs consists of nodes and edges. Nodes denote mathematical operations while edges indicate multidimensional data tensors to facilitate communication between the nodes. Tensorflow can also be used for parallel processing using multi-core CPU as well as GPU.

The main highlights which a Tensorflow graph includes to form a computational graph are listed as follows:

1) Providing input to computational graph, a placeholder variables are necessary.
2) Weights and Bias variable values are indeed necessary to enhance the accuracy of the model.
3) The cost measure function that is basically used to show the difference between actual and predicted value.
4) An optimization method that is basically used to update the values of weights and bias.

5.1 How Tensorflow works?

The flow about how optimization takes place through tensorflow library is shown below:

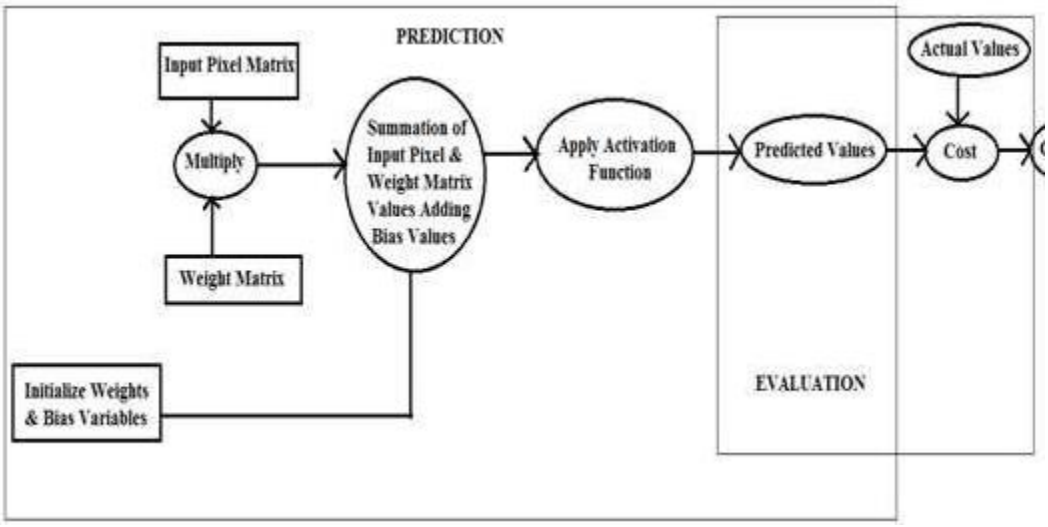

Figure 5.1: How optimization flow occurs

Initially, the input image that basically consists of a matrix of pixel values ranged between 0 to 255 is multiplied by the weight matrix that is generated randomly. After multiplication of the pixel values with the corresponding weights, non-lineraity function ie. Bias is added to these summation of multiplication values. Then an activation function is imposed on this summation so as to reduce the number of computations needed. We may use different activation functions. For example, reLu (Rectified Linear Units) function is an effective function in comparison to tanh function as reLu considers only non-negative values and dumps the negative values. After that the predicted values are evaluated using sigmidal function and fully connected network next to the convolution layer. Then an optimization function such as Adam optimizer, gradient descent optimizer can be used to update the bias and weight values, thereby improving the accuracy.

5.2 Hardware And Software Requirements

The various hardware and software requirements for the implementation of our proposed framework is described below:

- **CPU Architecture**: x86 64
- **System Memory**: 8-32 GB

- **Operating System**: Windows 7, Ubuntu 14.04
- **Integrated Development Environment**: Pycharm IDE
- **Python**: version 3.6
- **Tensorflow Library**: Open source library of google used for the implementation of Convolution Neural Network.
- **Numpy Library**: Used for performing mathematical operations.

5.3 TensorFlow Installation

In the proposed framework, we have used windows for the installation of Convolution Neural Networks (CNNs). Following are the steps that we followed for installation of python library.

Before installing Tensorflow, the pre-requisites: Python and pip should be installed in your Windows environment.

1) Installation of Python from *https://www.python.org/downloads/*
2) Download Pycharm from *https://www.jetbrains.com/pycharm/download/*
3) Installation of Tensorflow using the following command.
   py -m pip install --upgrade tensorflow

To validate if tensorflow is successfully installed, you can run the following code snippet:

```
import tensorflow as tf;

res = tf.constant("Hi I Am Working..!!")

sess=tf.Session();

print sess.run(res);
```

If tensorflow is successfully installed in your Windows environment, then it will print: "Hi I Am Working..!!".

5.4 Implementation

For the implementation of our proposed framework, we introduce the concept of Transfer Learning. Transfer learning make utilization of the information picked up while tackling one issue and applying it to an alternate however related issue. Transfer learning is basically the ability to transfer knowledge from one domain to another. Here, we have used the ImageNet dataset, CIFAR-10 to train our model. The YTO dataset is basically dataset of videos and videos are basically collection of image frames. Thus, both these datasets are related. Hence, we have used the knowledge gained by training the system with CIFAR-10 dataset to detect the objects in YTO dataset.

5.4.1 Implement Deep Learning Model

Firstly, our CNN model is build consisting of input layer of size 1*28*28; two convolution layers of size 32*24*24 and 32*8*8 respectively. Also, max pool operation is performed corresponding to the two convolution layers having filter size of 32*12*!2 and 32*8*8 respectively. The model consists of two dropout layers of size 32*4*4 and 256 respectively; one fully connected layer and one output layer consisting of 10 neurons corresponding to the ten classes. In case of Convolution Neural Network, the layers are arranged in three dimensions: width, height and depth unlike regular neural network. For grey scale images depth is equal to one and for colored images depth is equal to three: for red, blue and green colors. Here, the neurons in a layer will be connected not to all neurons in the previous layer, rather have connections with a small regions of the layers proceeding it. The final output of the convolution layer network will be a vector array of size equal to the total number of available classes, because by the end of convolution network architecture, input full size image is converted into a single vector of class probabilities that are arranged along the depth dimension. Convolution Neural Network consists of a number of layers. It uses all these layers to compute calculations via small regions of the layer referred as activation map window. The figure 5.2 shows a very general architecture of Convolution Neural Network (CNN):

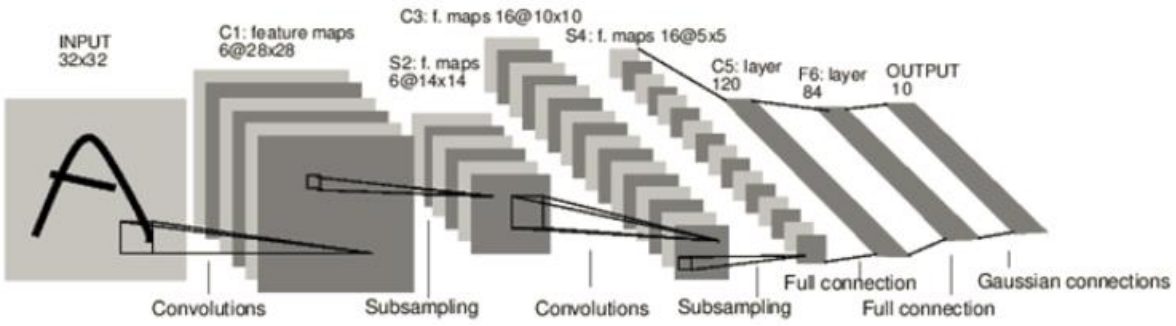

Figure 5.2: General Architecture of Convolution Neural Network

5.4.2 Training Phase

In the training phase, our model is trained using CIFAR-10 dataset, consisting of 60k images of 32*32*3 dimension corresponding to ten object classes with 6000 images per class. During the training phase, the backpropagation process can be separated into four distinct segments: forward pass, loss function, backward pass and loss function. Further the loss function can be defined in terms of Mean from each class Squared Error (MSE)

$$MSE = 1/2 \text{ (actual-predicted)}^2 \quad (1)$$

The loss will be very high for the first few of training images. The prime objective of this phase is to reach a point where the predicted label which is the output of the convolution network coherent with training label. The table below shows the training loss as the number of epoch increases.

Tab.5.1: Values of Training Loss & Validation Loss as the number of epochs increase.

Epoch	**Training Loss**	**Validation Loss**
1	2.423949	1.675540
5	1.158913	1.112673
10	0.944817	0.911486
20	0.762596	0.741700
40	0.578806	0.554007

80	0.345974	0.326073
100	0.267752	0.271636

Thus, as it is clear from the above table that the training loss decreases as the number of epoch increases. The graph below shows the nature or behavior of this decrease:

Fig. 5.3: Graph representing the nature of decrease of the training loss and validation loss as the number of epochs increase.

5.5 Results & Discussion

Now, we test our system trained on CIFAR-10 dataset with:

1) Real-time video dataset

2) YTO dataset

5.5.1 Real -Time video dataset:

We created our own dataset by collecting videos of the objects and dividing the video into image frames. Then we used the watershed algorithm to enhance the distinction, contrast and features of the image. Figure 5.4 shows the steps followed to create and pre-process our video dataset.

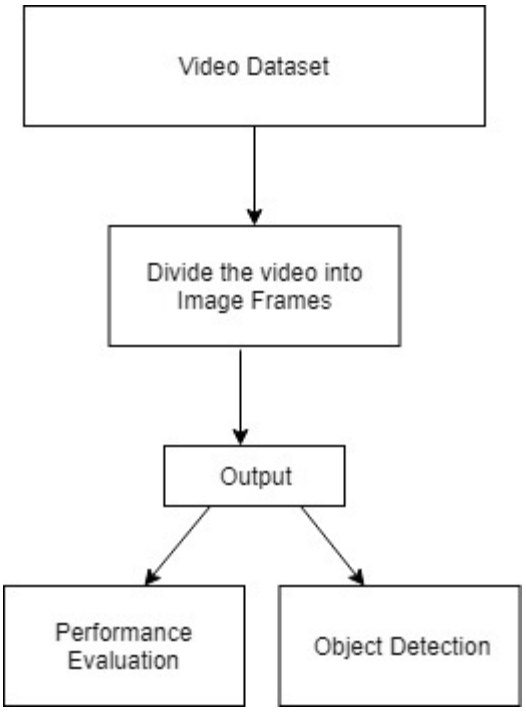

Figure 5.4: Methodology followed for our own video dataset

Tab. 5.2: Shows the accuracy of our dataset:

Object Class	**Total Video Frames**	**Correctly Detected Frames**	**Incorrectly Detected Frames**	**Accuracy (in%)**
Airplane	1,737	1,605	132	92.40
Bird	2,232	1,881	351	84.27
Deer	1,567	1,381	365	88.13
Cat	1,345	1,115	230	82.89
Frog	1,587	1,387	200	87.39
Ship	1,109	871	238	78.53
Truck	1,220	1,016	204	83.27
Horse	1,476	1,345	131	91.11

Thus the overall accuracy of the system on our dataset is 85.99 %. The figure 5.5 shows the accuracy graph of our system on our dataset:

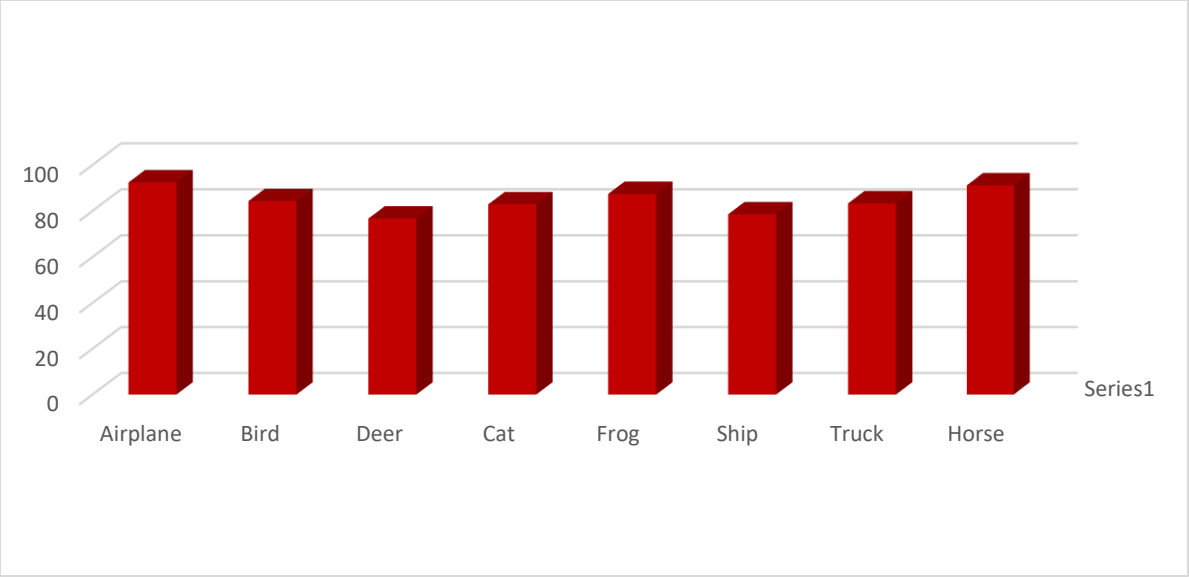

Figure 5.5: Accuracy graph on our video dataset

YTO Dataset:

Here, we assess our suggested framework on the pre-processed video frames obtained from the YTO dataset. In contrast to the VID dataset, the YTO dataset is weakly annotated. There are six classes that are common in CIFAR-10 and YTO dataset. Hence we evaluate the performance on these classes as shown below.

Tab.5.3: Accuracy Analysis of each class of object.

Object Class	**Total Video Frames**	**Correctly Detected Frames**	**Incorrectly Detected Frames**	**Accuracy**
Airplane	71,327	43,296	28,031	60.7
Bird	27,532	17,189	10,343	62.4
Cat	42,785	23,745	19,040	55.4
Dog	82,433	52,675	29,758	63.9
Horse	70,247	47,346	22,901	67.4
Automobile (car + Mbike + Train)	1,72,624	1,05,818	66,806	61.3

Thus, as depicted in the above figure, our proposed framework has an average accuracy of **61.85%.**

Accuracy = (Correctly Detected Frames / Total Frames) *100 (2)

To illustrate the effectiveness and efficiency of our proposed system, we evaluated the results obtained from YTO dataset directly with state-of-art results as shown below.

Tab.5.4: Comparison of Results of Our Proposed System with State-Of-The-Art Results.

Method	Airplane	Bird	Cat	Dog	Horse	Average
Prest et al.[1]	51.7	17.5	22.3	13.5	26.7	26.3
Jouline et al. [2]	25.1	31.2	38.5	33.9	35.6	32.8
Kwak et al. [3]	56.5	66.4	39.9	50.4	56.3	53.9
Proposed System	60.7	62.4	55.4	63.9	67.4	61.96

Now, comparing graphically our proposed system accuracy with the above results:

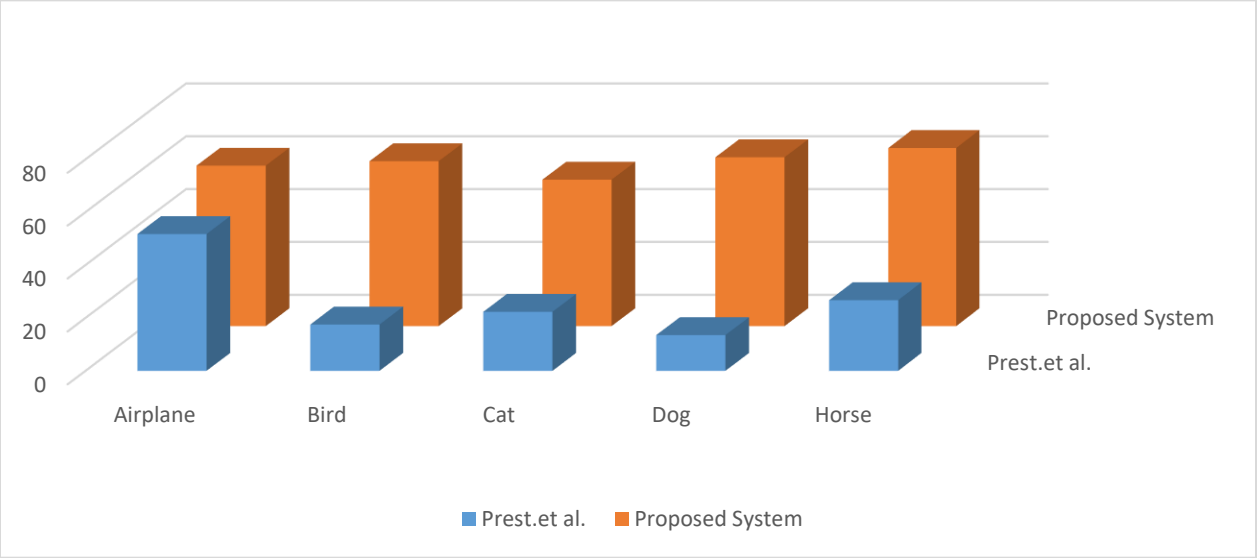

Fig. 5.6: Comparison of Accuracy of our Proposed system with Prest et al. [1] method

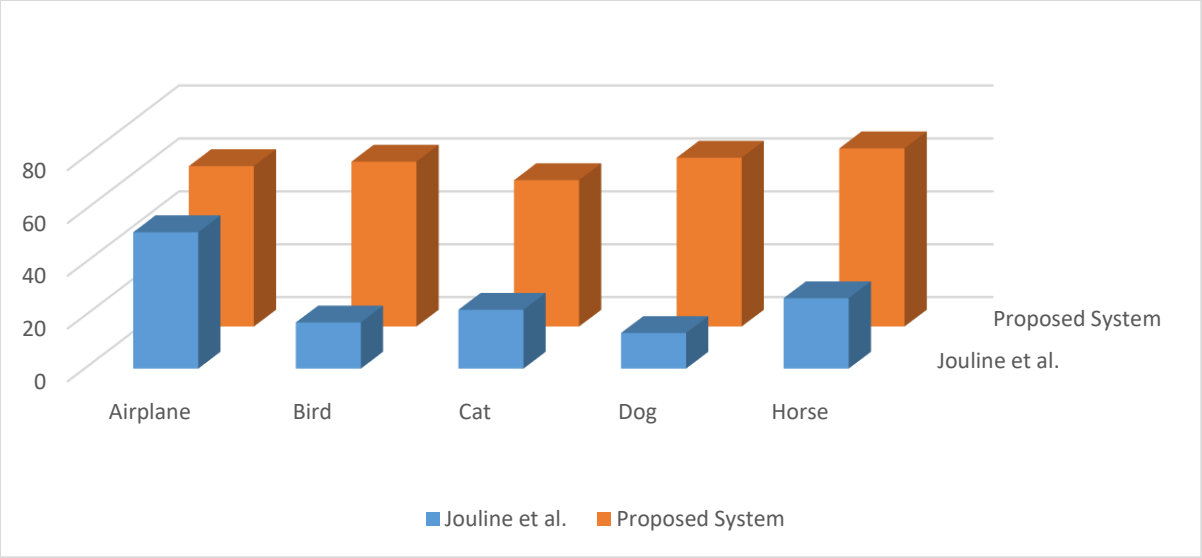

Fig. 5.7: Comparison of Accuracy of our Proposed system with Jouline et al. [2] method

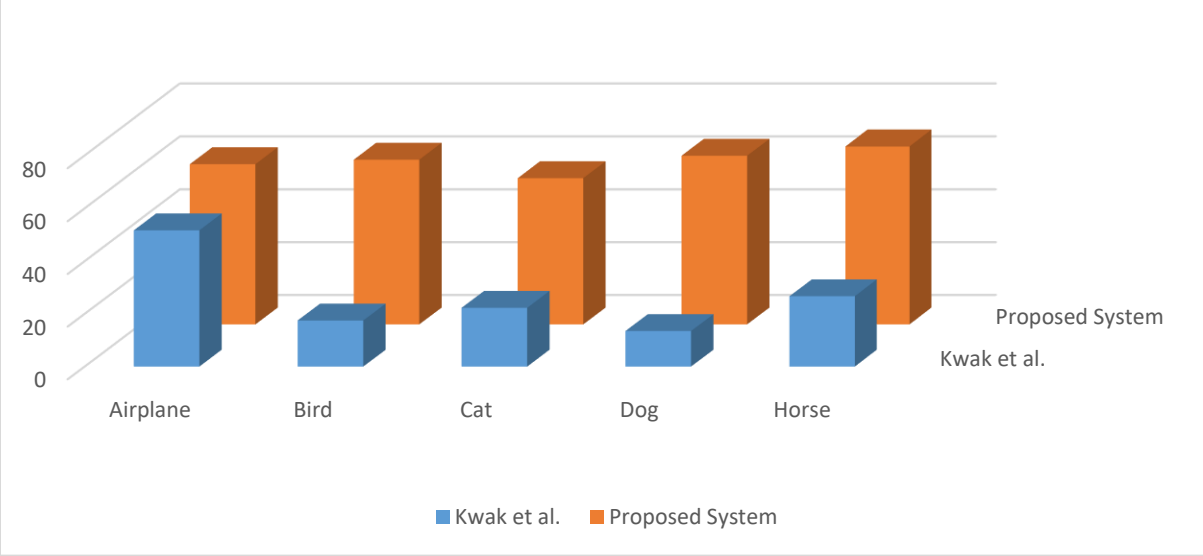

Fig.5.8. Comparison of Accuracy of our Proposed system with Kwak et al. [3] method

Chapter 6

Conclusion And Future Work

6.1 Introduction

In this work, we proposed a framework for detection of objects in real-time video using the concept of transfer learning. The model used to train the system is a Deep Neural Network. The whole concept of how deep neural network learns to recognize the pattern is coined as Deep Learning- which is basically a sub-branch of machine learning. Deep Learning is primarily based on Artificial Neural Network with a concept to mimic the functioning of human brain – visual cortex. Convolution Neural Network is a class of feed forward neural network that has proved its efficiency in analyzing visual patterns.

We have also introduced the concept of transfer learning that has been proved to be quite efficient for increasing the efficiency. Transfer learning is basically a method where a model developed for a task is used as initial point for a model on another task. Likewise, here we train our system with the ImageNet CIFAR-10 dataset and then using this trained model to detect objects in video dataset – YTO dataset.

Here, we used Convolution Neural Network (CNN) to train our system on CIFAR-10 ImageNet dataset and use this trained system to detect objects in YTO dataset. Our proposed framework has an accuracy of 85.99% on our own dataset and an accuracy of 61.96 on YTO dataset in comparison to the baseline results.

6.2 Summary Of Contribution

Following are the summary of contributions which are formulated in this work:

- Google's open source library: TensorFlow, it is cost effective to use.
- Tensorflow basically provides many in-built methods to solve complex mathematical operations and uses a computational graph for evaluation.
- Numpy and matplot libraries for mathematical operations and graph plotting.
- Tensorflow also provides GPU support for parallel computations.

6.3 Future Scope

Here, in this thesis, we have only used the watershed algorithm for image processing and we have achieved an accuracy of 85.99 % on our video dataset and 84.81 % on YTO dataset. In future, we can incorporate efficient image processing techniques to enhance the accuracy further. We can adopt techniques to segment the images and thereby increasing the accuracy of our system.

Here, we trained our system with 100 epoch and using two convolution layers and one fully connected layer and thus had a training loss of 0.26. We can reduce it further by using three convolution layers and one fully connected layer.

References

[1] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. *CVPR*, 2012.

[2] A. Joulin, K. Tang, and L. Fei-Fei. Efficient Image and Video Co-localization with Frank-Wolfe Algorithm. *ECCV*, 2014.

[3] S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid. Unsupervised Object Discovery and Tracking in Video Collections. *ICCV*, 2015.

[4] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. *CVPR*, 2008.

[5] Kai Kang, Wanli Ouyang, Hongsheng Li, Xiogang Wang. Object Detection from Video Tubelets with Convolution Neural Network. *CVPR*, 2016

[6] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully Convolution Neural Network. *ICCV*, 2015

[7] G. Yu and J. Yuan. Fast action proposals for human action detection and search. *CVPR*, 2015.

[8] W. Yang, W. Ouyang, H. Li, and X. Wang. End-to-end learning of deformable mixture of parts and deep convolution neural networks for human pose estimation. *CVPR*, 2006.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with Deep Convolution Neural Networks. *NIPS, pages 1097-1105*, 2012.

[10] J. Long, E. Shelhamer, and T. Darrell. Fully Convolution networks for semantic segmentation. *CVPR*, 2015.

[11] H. Nam and B. Han. Learning multi-domain convolution neural networks for visual tracking. *arXiv: 1510.07945*, 2015.

[12] K. Simonyan and A. Zisserman. Very deep convolution networks for large-scale image recognistion. *ICLR*, 2014.

[13] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. *CVPR*, 2014.

[14] K. Kang, and X. Wang. Fully convolution neural networks for crowd segmentation. *arXiv preprint arXiv*: 1411.4464, 2014.

[15] G. Yu and J. Yuan. Fast action proposals for human action detection and search. *CVPR*, 2015.

[16] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolution neural networks. *CVPR*, 2015.

[17] A. Papazoglu and V. Ferrari. Fast Object Segmentation in Unconstructed Video. *ICCV*, 2013.

[18] W. Ouyang, X. Wang, C.Zhang, and X. Yang. Factors in fine-tuning deep model for object detection with long-tail. *CVPR*, 2016

[19] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. DeepID-net: Deformable deep convolution neural networks for object detection. *CVPR*, 2015.

[20] https://www.cs.toronto.edu/~kriz/cifar.html [accessed: 20/5/2018].

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich fea- ture hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[22] T. Deselaers, B. Alexe, and V. Ferrari. Localizing Objects While Learning Their Appearance. *ECCV*, 2010.

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CVPR, 2015

[26] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. DeepID-net: Deformable deep convolutional neural networks for object detection. CVPR, 2015.

[27] R. Girshick. Fast r-cnn. ICCV, 2015.

[28] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. NIPS, 2015.

[29] A. Joulin, K. Tang, and L. Fei-Fei. Efficient Image and Video Co-localization with Frank-Wolfe Algorithm. ECCV, 2014

[30] S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid. Unsupervised Object Discovery and Tracking in Video Collections. ICCV, 2015

[31] T. Deselaers, B. Alexe, and V. Ferrari. Localizing Objects While Learning Their Appearance. ECCV, 2010.

[32] H. Possegger, T. Mauthner, P. M. Roth, and H. Bischof. Occlusion geodesics for online multi-object tracking. CVPR, 2014.

[33] Y. Li, J. Zhu, and S. C. Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. CVPR, 2015.

[34] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. CVPR, 2015.

[35] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. arXiv:1510.07945, 2015.

[36] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. ICCV, 2015

[37] W. Yang, W. Ouyang, H. Li, and X. Wang. End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. CVPR, 2016.

[38] Jing Shao, Chen Change Loy, Kai Kang, Xiogang Wang. Slicing Convolotuion Neural Network for Crowd Video Understanding. IEEE Conference on Computer Vison and Pattern Recognition, 2016

List of Publications

1. Pranav Kapur, Parteek Bhatia “Object Detection in Video Based on Transfer Learning Using Convolution Neural Network”, International Conference on New Trends in Engineering & Technology, to be held in GRT institute of Engineering & Technology, Tirupathi Highway, Chennai.
   [Accepted]

Thesis

ORIGINALITY REPORT

9%	4%	8%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ee.cuhk.edu.hk Internet Source	3%
2	www.cs.toronto.edu Internet Source	1%
3	Fatemeh Sadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, Jose M. Alvarez. "Bringing Background into the Foreground: Making All Classes Equal in Weakly-Supervised Video Semantic Segmentation", 2017 IEEE International Conference on Computer Vision (ICCV), 2017 Publication	1%
4	Ruchira. Ajay Jadhav, Roopa Ashok Thorat. "Computer aided breast cancer analysis and detection using statistical features and neural networks", Proceedings of the International Conference on Advances in Computing, Communication and Control - ICAC3 '09, 2009	<1%