# TRAFFIC VIDEO ANALYSIS

**MIDHILESH E**
**Roll No. 16PT17**

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

**FIVE YEAR INTEGRATED**
**M.Sc. THEORETICAL COMPUTER SCIENCE**
OF ANNA UNIVERSITY


**NOVEMBER 2019**

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES

**PSG COLLEGE OF TECHNOLOGY**
(Autonomous Institution)
**COIMBATORE – 641 004.**

**PSG COLLEGE OF TECHNOLOGY**
(Autonomous Institution)
**COIMBATORE – 641 004.**

**Seventh Semester**
**Project work**

# TRAFFIC VIDEO ANALYSIS

Bona fide record of work done by

**MIDHILESH E**
**Roll No. 16PT17**

Submitted in partial fulfillment of the requirements for the degree of

**FIVE YEAR INTEGRATED**
**M.Sc. THEORETICAL COMPUTER SCIENCE**
Of Anna University

**NOVEMBER 2019**

| | |
|---|---|
| _____ | _____ |
| **Academic Guide** | **Head of the Department** |

Submitted for the Viva-Voce Examination held on _____

| | |
|---|---|
| _____ | _____ |
| **Internal Examiner** | **External Examiner** |

# Contents

# ACKNOWLEDGEMENT

I am extremely grateful to **Dr. K. Prakasan**, Principal In Charge, PSG College of Technology, Coimbatore for giving me this opportunity to do this project at International Institute Of Informational Technology, Bengaluru.

I am very much thankful to **Dr. R. Nadarajan**, Professor and Head, Department of Applied Mathematics and Computational Sciences, PSG College of Technology, Coimbatore, for his continual support and guidance throughout the project tenure.

I owe my deep sense of gratitude and gratefulness to my internal guide, **Dr. A. Kaja Mohideen**, Assistant Professor, Department of Applied Mathematics and Computational Sciences, PSG College of Technology, Coimbatore, for his exemplary guidance, ardent motivation and constant support throughout this project.

I am deeply indebted to my external guide **Dr. Dinesh Babu Jayagopi**, Assistant Professor, International Institute Of Information Technology, Bengaluru, for his constant motivation, guidance, continuous support and help in various aspects to complete this project work.

I am indebted and thankful to my external guide **Mr. Ashish Sardana**, Deep Learning Solutions Architect, NVIDIA, Bengaluru, for his constant motivation, guidance, continuous support and help in various aspects to complete this project work.

Finally, I express my gratefulness to all the staff members of the Department of Applied Mathematics and Computational Sciences, PSG College of Technology, Coimbatore, and my family and friends for their encouragement and support.

# SYNOPSIS

The problem of analysing the huge traffic in the Electronic City can be automated using traffic videos. The problem of analysing traffic videos can be divided into four subproblems namely counting unique vehicles, predicting the speed of those vehicles, detecting the number plates and analysing whether a vehicle is violating the traffic rules.

The primary task for traffic video analysis is tracking vehicles in the video and it is solved using deep learning techniques which is the current state-of-art in computer vision. Simple Online Realtime Tracker (SORT), a deep learning model for tracking vehicles is implemented in this project. The performance of tracking vehicles is further improved by implementing Deep Multi-Object Tracker (DeepMOT), an another deep learning model which outperforms current multi-object trackering models with greater accuracy and precision.

# CHAPTER 1

# INTRODUCTION

Computer Vision, according to Wikipedia is an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos. It seeks to automate tasks that the human visual system can do.

Traffic video analysis comes under the field of computer vision which aims in developing a mathematical model for analysing traffic videos which performs at a high frame per second (fps), invariant of light and climatic condition and produces high accuracy, precision. Analysing traffic video can be solved using traditional image processing techniques like contour detection, template matching which run with high fps but does not satisfies other conditions mentioned above.

Deep Learning along with cloud computing and edge computing on the other hand works extremely well in all the above conditions. After the discovery of Convolutional Neural Network by Dr. Yann Le Cun, all computer vision problems have achieved state-of-art and servs in realtime applications.

## 1.1 LITERATURE REVIEW

Samira Ebrahimi Kahou et.al [2] proposed a soft attention based model for tracking objects by finding where to look for objects in the image. Alex Bewley et.al [1] proposed a model which comes under detection based tracker, where hungarian algorithm is used for assigning predicted position of objects using kalman filters and detected poitions of object using any deep object detectors. Yihong Xu et.al [3] proposed a model which accounts appearence metrics for object tracking and aims in improving accuracy and precision during tracking. Bo Li et.al [4] proposed a single object tracker model based on Siamese network.

# CHAPTER 2

# SYSTEM CONFIGURATION

## 2.1 HARDWARE CONFIGURATION

- **Processor** - Intel core i9-7000x CPU @ 3.30GHz x 20
- **GPU** - NVIDIA Quadro RTX 8000
- **Main Memory** - 64 GB
- **Secondary Memory** - 2TB

## 2.2 SOFTWARE CONFIGURATION

- **Operating System** - Ubuntu 18.04
- **Environment** - NVIDIA Docker
- **Programming Language** - Python 3.5
- **GPU Language** - CUDA 10.1.1
- **Deep Learning Framework** - Pytorch 1.1

# CHAPTER 3

# FUNDAMENTALS OF OBJECT DETECTION AND TRACKING

## 3.1 OBJECT DETECTION

Object Detection is the combination of Object Localization and Object Classfication. In Object Classification, the aim is to classify the given image into some class. In Object Localization, the aim is to tell the exact location of an object. For locating an object bounding box is used. Thus object detection means locating objects and classifying it.

### 3.1.1 Bounding Box

Bounding box represents the position of an object in the image. Bounding box can be defined using top left (x,y) co-ordinate, width and height of a box.

### 3.1.2 Intersection of Union (IoU)

In Object Detetion, there is a specific metric for determining the bounding box accuracy which is called as Intersection of Union (IoU). A bounding box is accurate when it's IoU value is near 0 and it is worse when it's value is near 1 with respect to ground truth bounding box. Figure 3.1 shows the formula for computing IoU.

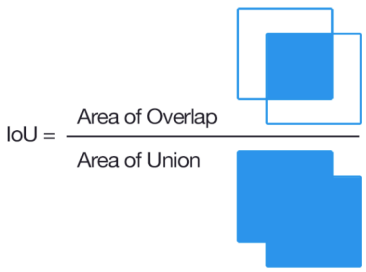


**Figure 3.1: Intersection of Union**

## 3.2 OBJECT TRACKING

Object Tracking means associating similar objects over consecutive frames in a video. It can be classified into two types:

- Single Object Tracking
- Multi Object Tracking

### 3.2.1 Single Object Tracking

Single Object Tracking (SOT) means associating one particular object (target object) over consecutive frames in a video. This can be performed by learning a discriminative model that is capabale of classifying whether a given object is the target object or not. Examples of single object trackers are listed below:

- MOSSE tracker
- GOTURN tracker
- SiameseRPN tracker

### 3.2.2 Multi Object Tracking

The task of Multi Object Tracking (MOT) includes locating multiple objects, maintaining their identities and yielding individual trajectories given an input video. This can be divided two types as follows:

- Detection-Based Tracking
- Detection-Free Tracking

### Detection-Based Tracking

In this approach there are two models one for detecting objects in the image and the other for tracking the detected objects. The main problem of this approach is to associate the detections of the same object in two consecutive frame as this combinatorial problem and this association should be global. In Figure 3.2, detection based tracking is represented diagrammatically.

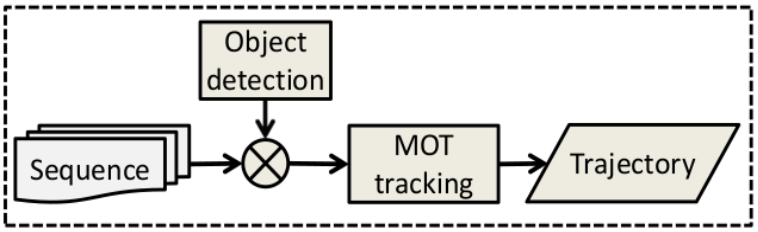


**Figure 3.2: Pictorial representation of Detection-Based Tracking**

### Detection-Free Tracking

In this approach there is only one model for tracking objects, which are chosen by humans. The main problem of this approach is humans need to start tracking, hence the process cannot be automated. In Figure 3.3, detection based tracking is represented diagrammatically.

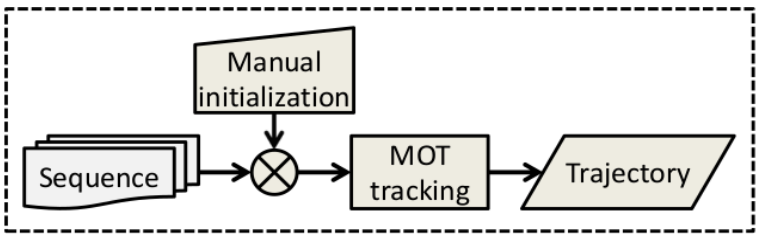


**Figure 3.3: Pictorial representation of Detection-Free Tracking**

## 3.3 DEEP LEARNING CONCEPTS

### 3.3.1 Convolutional Neural Networks (CNN)

It is a class of Neural Networks, where both feature extractor and a classifier are combained. Figure 3.4, shows the pictorial representation of Convolutional Neural Network. The output of CNN is called **feature map**.

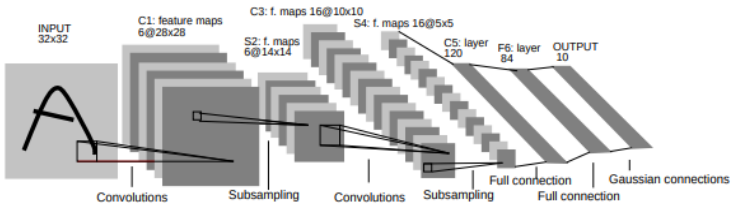


**Figure 3.4: Architecture of LeNet-5, a Convolutional Neural Network**

#### Convolutions

It is a mathematical operations where dot product between the section of an input image and the weight matrix (**kernel**) is performed. This operation is performed all over every seperate section of an image by sliding over kernel. The amount of sliding is determined by **strides**. Convolution is capable of extracting salient features from the input image.

#### Subsampling (Pooling)

Replacing a section of input either by the maximum value (Max Pooling) or by the average (Average Pooling) of that section. This helps in reducing the dimension of the input (downsampling the image).

#### Fully connection

It is a multi-layer perceptron capable of discrimating the input features extracted by series of Convolutions and Subsampling operations.

### 3.3.2 Bidirectional Recurrent Neural Networks

Bidirectional Recurrent Neural Network is basically putting two independent Recurrent Neural Network together. The input is fed in normal order for one network and in reverse order for the other network. The output of the two networks are concatenated at each time steps. This structure allows the network to have both the forward (future) and backward (past) information at each time steps. The figure 3.5 shows the pictorial representation of Bidirectional Recurrent Neural Network.

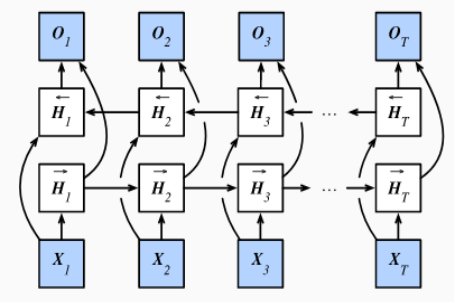


**Figure 3.5: Bidirectional Recurrent Neural Network**

### 3.3.3 Loss function

Loss fuction is defined based on the model and task. By finding the minimum point for the loss function the model learns optimal parameters for the given task. (eg., cross entropy loss). Loss function should be differentiable.

**Cross-entropy loss** (3.2) is defined below where p is model predicted probability for a class to occur and y $\epsilon$ {-1,+1} specifiying ground truth classes.

$$CE(p,y) = \begin{cases} -\log(p) & if\ y = 1 \\ -\log(1-p) & otherwise \end{cases}, \; p_t = \begin{cases} p & if\ y = 1 \\ 1-p & otherwise \end{cases} \tag{3.1}$$

$$CE(p,y) = CE(p_t) = -\log(p_t) \tag{3.2}$$

# CHAPTER 4

# SIMPLE ONLINE REALTIME TRACKER

Simple Online Realtime Tracker developed by Alex Bewley et.al [1] comes under Detection-Based Tracking methodology. For object detection, instead of **FasterRCNN** mentioned by Alex Bewley et.al [1], **YOLOv3** has been used, **kalman Filter** for object tracking and **hungarian algorithm** for data-track association. This algorithm runs at 130fps due to its simplicity.

## 4.1 KALMAN FILTER

Kalman Filter is an iterative mathematical process that uses a set of equations and consecutive data inputs to quickly estimate the true value, position, velocity. It assumes all the information in the form of **gaussian distribution**. It also predicts the value at near future with some degree of uncertainity.

For example predicting the location of a car using GPS measurement. Since GPS measurement contains some degree of error, kalman filter is used to predict the true position with very less error using **bayesian update** where the knowledge obtained by kalman filter from the last updation used to predict the current position ($\widehat{x_k}$) which acts as prior distribution and the GPS measurement ($y_k$) acts as likelihood to predict the posterior distribution whose mean gives the position of car ($\hat{x}_k$).

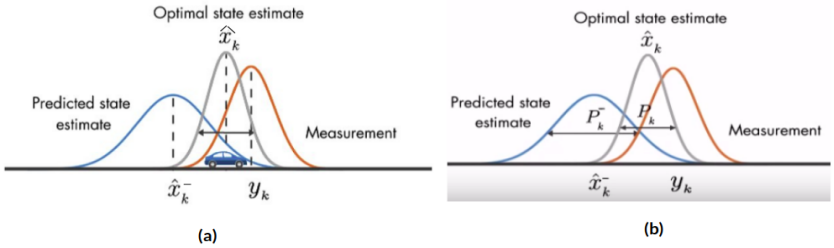


**Figure 4.1: Working of kalman filter (a) labelled without process covariance matrix and (b) with process covariance matrix**

In the context of object tracking, kalman filter tries to predict an object's position at next frame with increased uncertainity and assuming that object moves in uniform velocity.

### 4.1.1 State Matrix

It contains the target variable along with the variables necessary to make prediction. For example for predicting the position of an object using it's velocity. Here position of an object is th target value and the velocity is the variable necessary to make prediction.

In object tracking an object is represented in a 2D space, where x represents position in x direction, $\bar{x}$ represents velocity in x direction, y represents position in y direction and $\bar{y}$ represents velocity in y.

$$X = \begin{bmatrix} x & \bar{x} & y & \bar{y} \end{bmatrix} \tag{4.1}$$

### 4.1.2 Process Covariance Matrix

In Figure 4.1 (b) $P_k$, $\overline{P_k}$ denotes Process Covariance Matrix. Since, state matrix is one dimension $P_k$, $\overline{P_k}$ is basically the standard deviation but if state matrix is of higher dimension, then $P_k$, $\overline{P_k}$ is basically a covariance matrix for variables in the state matrix.

In Figure 4.2 (a) represents the Process Covariance Matrix for state matrix consist of position and velocity of an object after current updation $P_k$ and before updation $\overline{P_k}$. In Figure 4.2 (a) we can find a linear relationship between position and velocity with the assumption of constant velocity we can predict the position of the object for the next frame.

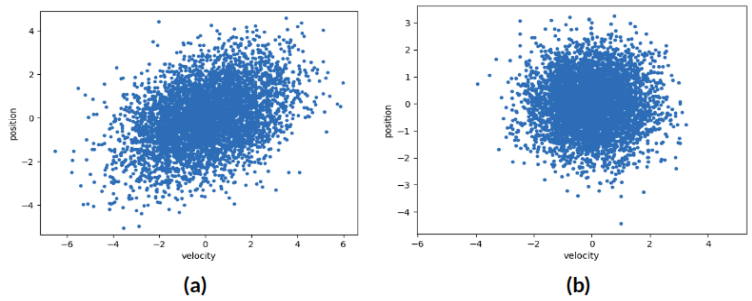


**Figure 4.2: (a) Scatter plot of covariance matrix after updation ($P_k$) and (b) before updation ($\overline{P_k}$)**

### 4.1.3 Kalman Gain

Kalman Gain ($K_k$) gives the measure that tells how much to trust the measured value and predicted value from previous state of the kalman filter. It lies between 0 and 1. It minimizes the posterior error covariance.

If Kalman Gain = 0, it means measured value is accurate and the predicted value is unstable.

If Kalman Gain = 1, it means measure value is unstable and the predicted value is stable.

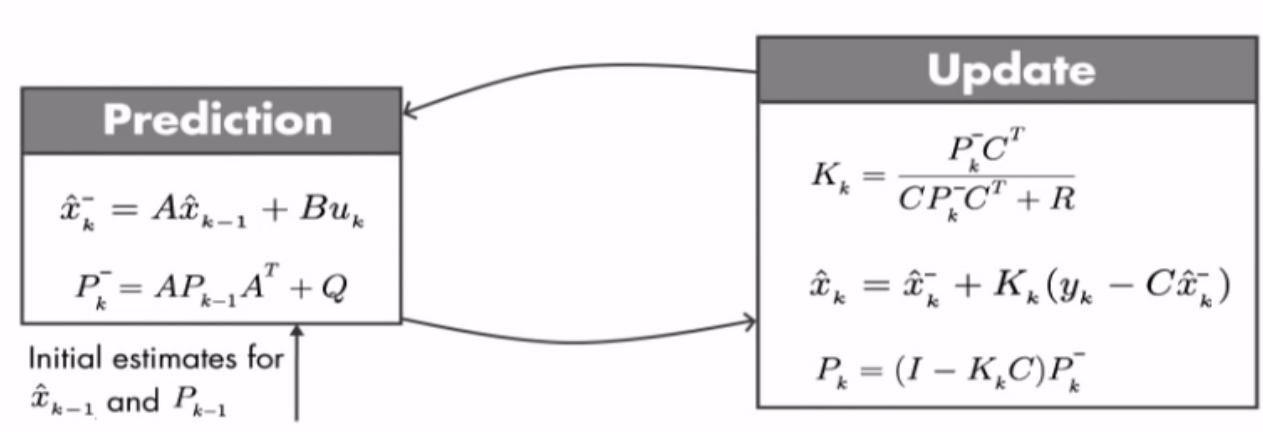


**Figure 4.3: Work flow of kalman filter**

Figure 4.3 Flow chart representing how kalman filters work. It consist of prediction and update part. Initial estimate can be any random value and kalman filter guarantees the convergence to true value.

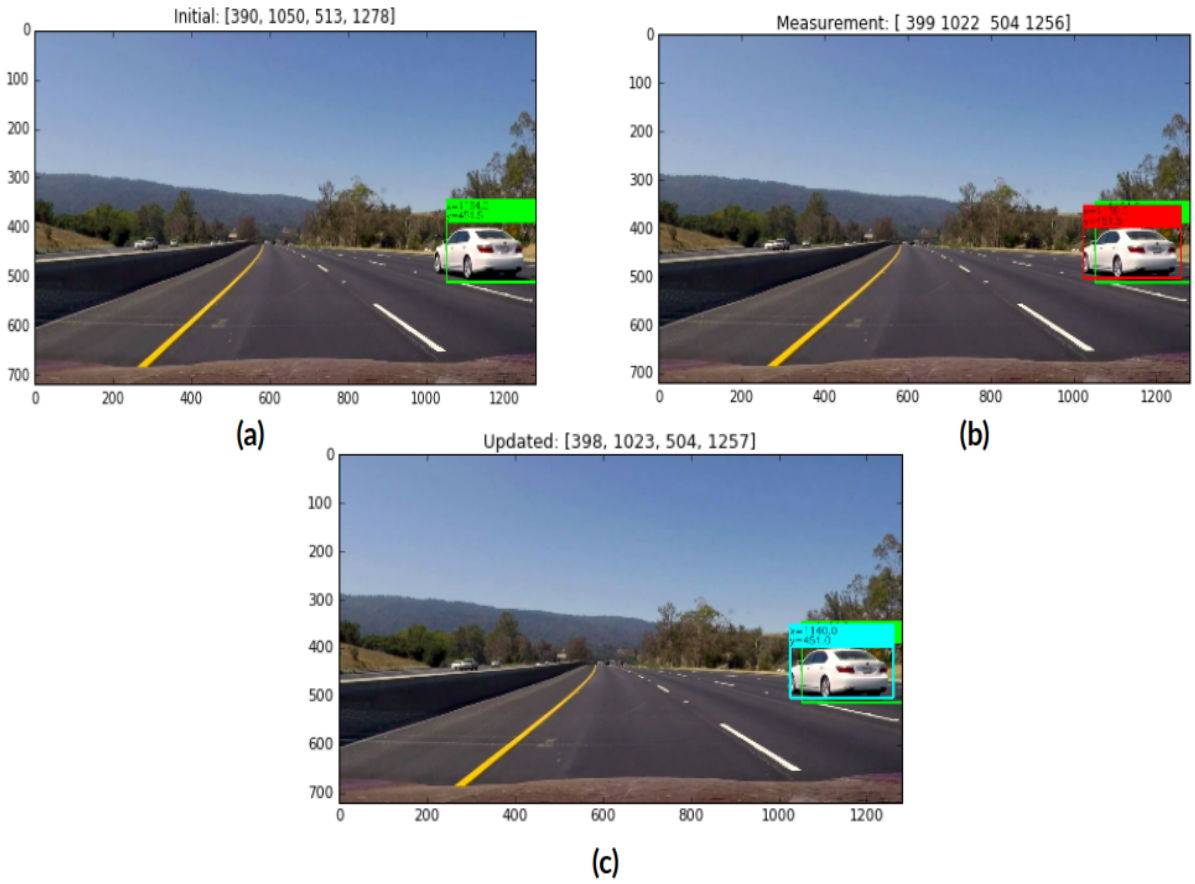


**Figure 4.4: (a),(b) Input for kalman filter and (c) Output of kalman filter**

Figure 4.4 shows how kalman filters tries to predict the actual position of the car (white color bounding box) using the initial estimate (yellow color bounding box) and measurement (red color bounding box). Here the state matrix contains the top left (x,y) and bottom right (x,y) coordinate along with their velocities but only the position values are showed in the image.

## 4.2 YOLOv3- YOU ONLY LOOK ONCE

YOLOv3 [6] is an object detection algorithm where the authors of YOLOv3 reframed it as single regression problem which means bounding box co-oridnates and class probabilities are predicted from the image using a single CNN network.

**Advantages of using YOLOv3:**

- Extremely fast. It runs at 45 fps on Tital X GPU with no batch processing.
- Reasons globally about the image during prediction. Hence, it makes half the number of background erros (predicting background as object) compared to Fast R-CNN.
- Learns generalizable representation of objects.

**Advancement made in YOLOv3**

Applying anchor boxes for predicting bounding boxes rather than using fully connected layers. Hence for using anchor boxes the basic CNN model is converted into Fully Convolutional Neural Network. Along with FCNN, Batch Normalization, increasing the resolution of the image being trained and increasing the depth of the network to 53 layers (**Darknet-53**) are used.

### 4.2.1 Fully Convolutional Neural Network (FCN)

A brief of Convolutional Neural Network (CNN) is seen in 2nd chapter. In CNN, fully connected layers are used for predicting continuous values and class labels which makes the whole CNN model to have large number of parameters. Hence, the fully connected layers are removed instead replaced by convolutional layers with 1×1 kernels which reduces the channels in the feature map and acts similar to fully connected layers with less number of parameters.

### 4.2.2 Working Of YOLOv3

Every image is divided into $13 \times 13$ cells and each cell contains B bounding boxes and it predicts value in $B \times (5 + C)$ shape, which means for each bounding box along with the coordinates, width and height measurement, it also contains an **object score** representing whether it contains any object or not in the cell and C represents the **class probability** (probability representing the occurence of each classes in that cell). In figure 4.5 represents the pictorial representation of yolov3 output.

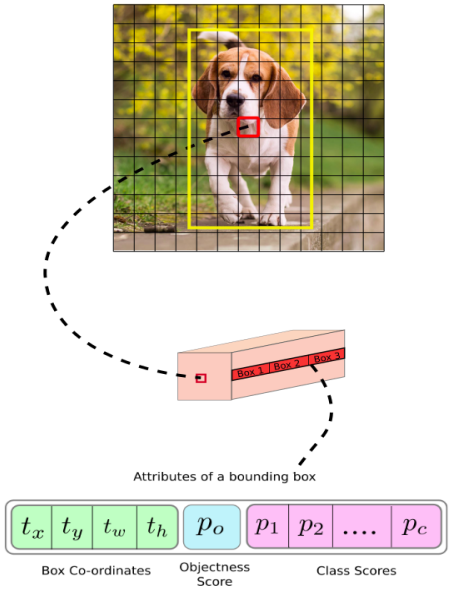


**Figure 4.5: output of YOLOv3 without post-processing**

In Figure 4.5, the red cell predicts three bounding boxes. Which one among the three bounding boxes assigned to the dog's ground truth image?. To solve this problem the authors of YOLOv3 have used anchor boxes.

### 4.2.3 Anchor Boxes

Anchor Boxes are the predefined bounding boxes with different scale and aspect ratio using these to predict the actual bounding boxes. In yolov3, it uses 10 anchor boxes which means every cell 10 bounding boxes will be predicted. Anchor boxes solves the issue of chosing which bounding box among the 10 boxes by chosing the box which has maximum IoU (Intersection of Union) between the anchor boxes and the ground truth. It also helps in predicting multiple objects in a single cell.

**Predicting bounding box using anchor boxes**

$$b_x = \sigma(t_x) + c_x \tag{4.2}$$

$$b_y = \sigma(t_y) + c_y \tag{4.3}$$

$$b_w = p_w e^{t_w} \tag{4.4}$$

$$b_h = p_h e^{t_h} \tag{4.5}$$

where ($b_x$, $b_y$) is the predicted bounding box (x,y) center coordinate with respect to the image, $b_w$ is the predicted bounding box width, $b_h$ is the predicted bounding box height. ($c_x$, $c_y$) represents the offset of the top left coordinate of the cell from the image. The network predicts ($t_x$,$t_y$) the center of bounding box with respect to the cell. Since ($t_x$, $t_y$) may sometimes takes value that causes the center to lie outside the cell the authors have uses **sigmoid** function to scale between 0 and 1. $p_w$, $p_h$ represents the width and height of anchor boxes respectively.

### 4.2.4 Training

#### Loss Function

Loss functions are used to train a model for a specific task. It represents how much the predicted value deviates from the actual value. By minimizing the loss function the model learns the better parameters to predict values with high accuracy. The following equation represents the loss function used in YOLOv3.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_{ij} - \hat{x_{ij}})^2 + (y_{ij} - \hat{y_{ij}})^2] \tag{4.6}$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_{ij}} - \sqrt{\hat{w_{ij}}})^2 + (\sqrt{h_{ij}} - \sqrt{\hat{h_{ij}}})^2] \tag{4.7}$$

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (c_{ij} - \hat{c_{ij}})^2 \tag{4.8}$$

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (c_{ij} - \hat{c_{ij}})^2 \tag{4.9}$$

$$\sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \ \epsilon \ classes} (p_i(c) - \hat{p}_i(c))^2 \tag{4.10}$$

where the equation (4.6) and (4.7) are the **localization loss**, equation (4.8) and (4.9) are the **confidence loss** and equation (4.10) is **classification loss**. Here (x,y) represents the top left coordinate of bounding boxes, (w,h) represents the width and height of the bounding boxes, c represents the object score and $p(c)$ represents the probability of each classes.

## 4.3 HUNGARIAN ALGORITHM

It is an optimization algorithm for assignment problems. The time complexity for hungarian algorithm is $O(n^3)$. This algorithm determines an optimal assignment of row and columns and the output is a zero-one matrix where one represents that respective row element is assigned to that respective column element and zero no assignment and the input is some cost value.

Simple Online Realtime Tracker [1] authors have used hungarian algorithm for matching the predicting bounding boxes (represented by **P**) using kalman filters for frame t+1 by updating the kalman filters at frame t and the detected bounding boxes at frame t+1 (represented by **D**) and the input is the IOU measure between every pair of bounding box in P, D vector. The following table 4.1 represents the input for the hungarian algorithm.

**Table 4.1: Input matrix for hungarian algorithm**

| Detection $Frame_t$/ Predicted $Frame_t$ | P1 | P2 | P3 |
|---|---|---|---|
| D1 | IOU=0 | IOU=0 | IOU=0 |
| D2 | IOU=0.56 | IOU=0 | IOU=0 |
| D3 | IOU=0 | IOU=0.77 | IOU=0 |

After processing the input, ouput of the hungarian algorithm is represented in the table 4.2 which tells that the bounding box P1 which was predicted from frame t which had some 'id' at frame t and the bounding box D2 are very close to each other. Hence, these two objects may be same and so we can assign the same 'id' for D2 bounding box.

**Table 4.2: Output matrix of hungarian algorithm (optimal assignment)**

| Detection $Frame_t$/ Predicted $Frame_t$ | P1 | P2 | P3 |
|---|---|---|---|
| D1 | 0 | 0 | 0 |
| D2 | 1 | 0 | 0 |
| D3 | 0 | 1 | 0 |

## 4.4 BIRTH AND DEATH PROCESS

Simple Online Realtime Tracker [1] authors tried to overcome the occlusion problem (tracking similar object even it is reappears after occlution) by making kalman filter predict the location of that object. This prediction after few frames leads to provide bounding box with large amount of uncertaininty. If an object continues to be occluded till some 'k' frames, the algorithm makes an assumption that the object is not present in the video and that 'id' is removed. If the same object appears at k+1 frame it is assigned to a new 'id'.

## 4.5 FLOW CHART REPRESENTATION

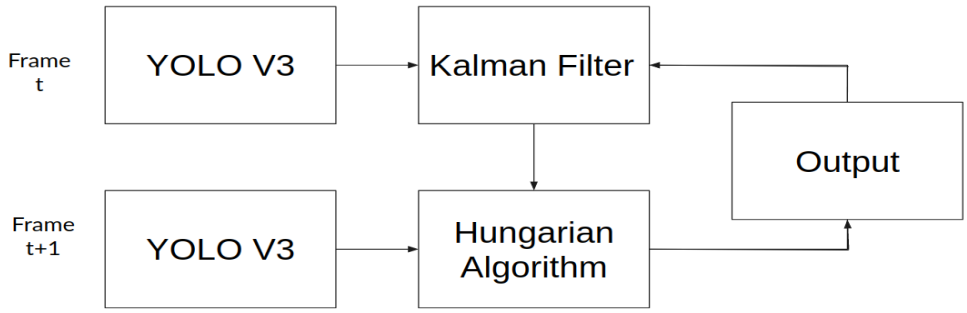


**Figure 4.6: Simple Online Realtime Tracker algorithm work flow**

## 4.6 RESULTS AND DISCUSSIONS

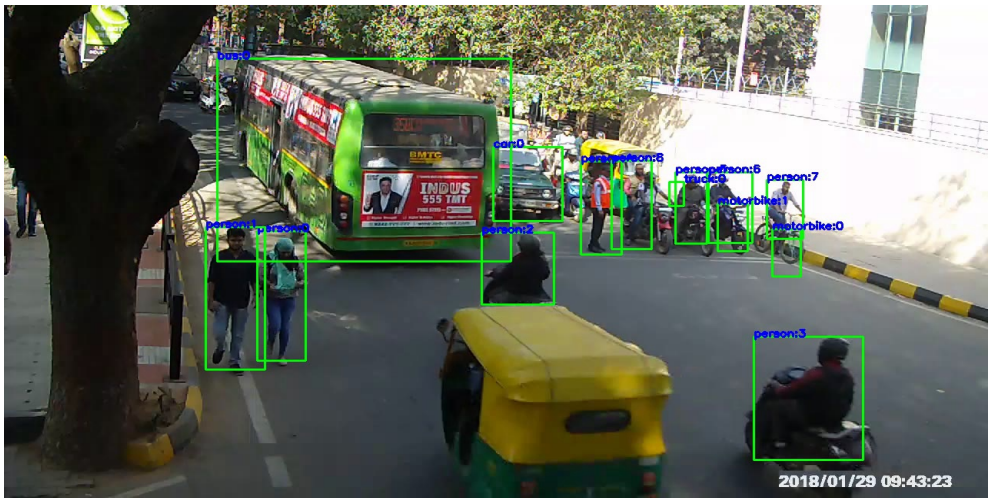


**Figure 4.7: Input video (frame 1)**

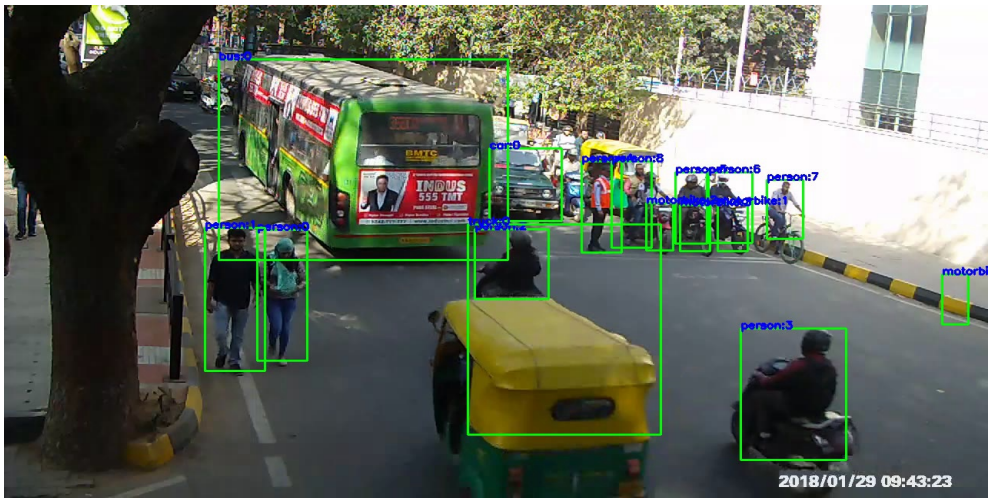


**Figure 4.8: Input video (frame 2)**

The images 4.7 and 4.8, shows the output of SORT algorithm where the similar objects are tracked in the frame 1 and frame 2. The performance of Simple Online Realtime Tracker is better on traffic videos when there is less traffic density.

# CHAPTER 5

# HIGH PERFORMANCE VISUAL TRACKING WITH SIAMESE REGION PROPOSAL NETWORK

Bo li et.al, the authors of High performance visual tracking with siamese region proposal network have used Siamese network and Region Proposal Network with one shot learning method for tracking single object. This model comes under Detection-free tracking.

## 5.1 SIAMESE NETWORK

Siamese network is built using Convolutional Neural Network for measuring how similar the given two images in the range of 0 to 1, where 1 means no similarity and 0 means highly similar.

### 5.1.1 Real time problem

Let's take face recognition application for a company where the aim is to tell a person belongs to that company or not given that person's image. The following are the problems present in this application:

- We cannot expect large amount images for every employers faces to train a model.
- If a new person enters or a person leaves a company the whole network need to retrained.

The solution for the above two problems is **One-shot learning** which will be discussed below.

### 5.1.2 One-shot learning

One-shot learning is a method where the model needs to make correct prediction (whether two images similar or not) given only a very few images (approximate 5) for each classes. This can be achieved directly by addressing the domain specific features or inference procedures which posses highly discriminative properties of target task. One-shot learning can be done in two ways:

- Generative models
- Discrimative embeddings

#### Generative models

Let's take an example for animal classification, where we are given 5 types of animal images among them one animal contains only 5 images and other contains large amount of images. In the paper [8], the authors have formulated an approach, where a model is build with these 4 classes and this representation (model parameters) is termed as prior, features obtained from the $5^{th}$ class images are termed as likelihood and using **bayer's theorem** posterior pobability is computed this acts as a new model. For the problem of conjugate prior they have developed a new way to find the posterior probability.

#### Discrimative embeddings

In this approach, an embedding space is generated where similar images are grouped together and the dissimilar objects are seperated apart to great extent. All the images are converted to this embedding space and using distance measure (nearest neighbour) the output is computed.

### 5.1.3 Architecture of the Siamese network

The following figures 5.1, 5.2 shows the pictorial representation of Siamese network where the convolutional layers and fully connected layers are identical in two figure. Thus two images are fed into same network parallely.

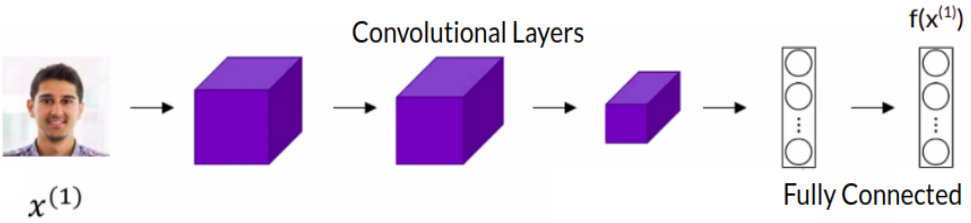


**Figure 5.1: Input1 for Siamese network**

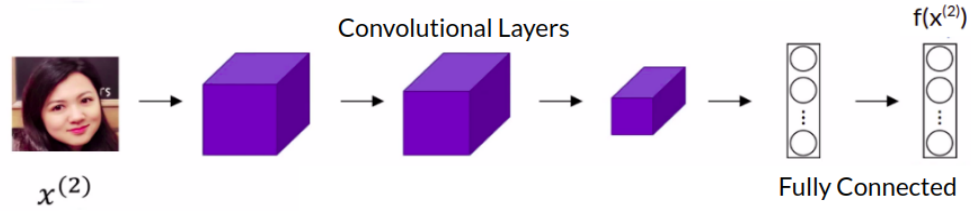


**Figure 5.2: Input2 for Siamese network**

Here the elements $x^{(1)}$ present in the image space is transformed by Siamese network into embedding space $f(x^{(1)})$ similarly for $x^{(2)}$ in the embedding space the distance between these two points are computed. The distance equation is as follows:

$$d(x^{(1)}, x^{(2)}) = ||f(x^{(1)}) - f(x^{(2)})||^2 \tag{5.1}$$

The output is computed by applying sigmoid function over the distance which gives output over the range of 0 to 1.

$$output = \sigma(d(x^{(1)}, x^{(2)})) \tag{5.2}$$

In the figure 5.3 (a) the MNIST images and 5.3 (b) the MNIST images after transformed by siamese network are applied T-SNE dimensionality reduction from higher dimension to 2 dimension and scatter plotted.

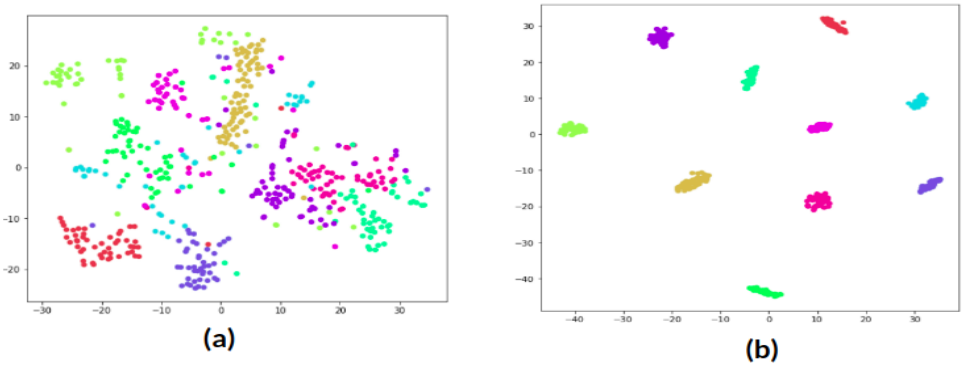


**Figure 5.3: (a) MNIST image space (b) MNIST embedding space**

## 5.2 REGION PROPOSAL NETWORK (RPN)

Object Detection networks depends on region proposal algorithm to hypothesize the object locations. It tires to predict the location of an object in the latent space (transformed image space). An RPN it is a fully convolutional network that simultaniously predicts the object bounds and the object score at each position. The RPN is trained end-to-end to generate high quality region proposals

### 5.2.1 How Region Proposal Network works?

- The input image goes through convolutional network and outputs the feature map on the last convolutional layer which is shown in figure 5.4.

- A sliding windows is run spatially accross the feature map. The size of the sliding window is $n \times n$ (here $3 \times 3$). For each silding window a set of 9 anchors are generated (shown in figure 5.5) which has same center ($x_a$,$y_a$) but with 3 different aspect ratio and 3 different scale. For each anchors a value $p^*$ is computed which indicates overlap between anchor box and the ground truth.


**Figure 5.4: Image representing the output of the CNN**

$$p^* = \begin{cases} 1 & IOU > 0.7 \\ -1 & IOU < 0.3 \\ 0 & otherwise \end{cases} \tag{5.3}$$

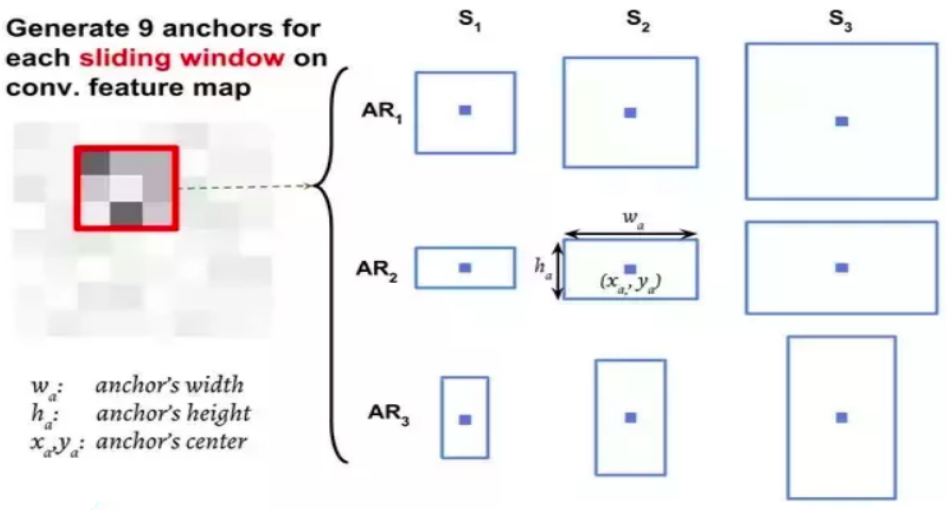


**Figure 5.5: Sample anchor boxes for feature map**

- Finally, the $3 \times 3$ spatial features is fed into a small network which has two task. One is regression which outputs (x,y,w,h) of the bounding box and the other is classification which outputs **p** indicating whether the predicted box contains object or not. The loss function for training this network is given below.

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{5.4}$$

$$t = [(x - x_a)/w_a, (y - y_a)/h_a, \log w/w_a, \log h/h_a] \tag{5.5}$$

$$t^* = [(x^* - x_a)/w_a, (y^* - y_a)/h_a, \log w^*/w_a, \log h^*/h_a] \tag{5.6}$$

## 5.3 ONE-SHOT LEARNING IN SIAMESERPN

In SiameseRPN [7], the authors have used discriminative version of oneshot learning where they use the concept of meta-learning (learning to learn). Learnet [9] is the first work that utilizes learning to learn approach to track object where a network called learnet that predicts the parameters for the other network which tries to tell whether the given two images are similar or not from two different frames for tracking the object (i.e associating these images same id).

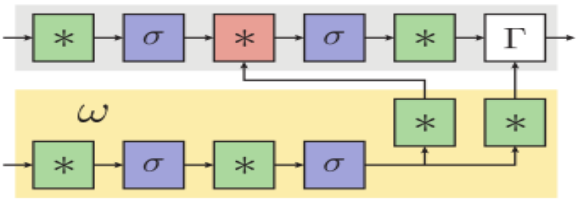


**Figure 5.6: Learnet architecture**

In Figure 5.7, $\Gamma$ is the first network responsible for classification , $\omega$ is the learnet which predicts the parameters for $\Gamma$ network and $\alpha$ is a maxpooling operation.

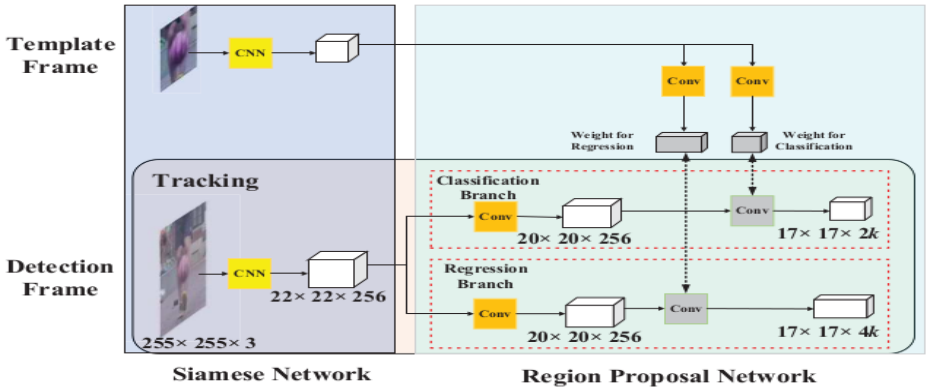


**Figure 5.7: SiameseRPN Framework with Oneshot Learning**

## 5.4 SIAMESERPN FRAMEWORK

The figure 5.7 represents the workflow of SiameseRPN. It contains two branch finding the similarity between tow inputs. These branched are detailed below:

- **Template Branch**

  Template Branch recieves target patch from the historical frame as input (target image) which is denoted as z.

- **Detection Branch**

  Detection Branch recieves the current image or search image (x) as input where the traget patch (z) passed to template branch is to be searched over x and localize using bounding box.

### 5.4.1 Modified Siamese network in SiameseRPN

These two branches contains identical CNN network and they share same parameters which is basically Siamese network. Conventionally, the output of this siamese network is represented as $\varphi(x)$ (detection branch), $\varphi(z)$ (template branch).

### 5.4.2 Region Proposal Network in SiameseRPN

The Region Proposal Network in SiameseRPN consist of two sections for predicting the bounding box of target image in search image.

- **Supervision section**

  The supervision section has two branches.

  – classification branch - foreground and background classification.

  – Regression branch - predicting the bounding box coordinate

  If there are k anchor boxes the network predicts output of shape 2k for classification branch and 4k for regression branch

- **Pair-wise correlation branch**

  The pair-wise corelation section first increases the channels of $\varphi(z)$ to two branches $[\varphi(z)]_{cls}$ and $[\varphi(z)]_{reg}$ which contains 2k and 4k times channel by CNN network similarly for $\varphi(x)$. Thus the output of this section as follows.

$$A^{cls}_{w\times h\times 2k} = [\varphi(z)]_{cls} \star [\varphi(x)]_{cls} \tag{5.7}$$

$$A^{reg}_{w\times h\times 2k} = [\varphi(z)]_{reg} \star [\varphi(x)]_{reg} \tag{5.8}$$

  The reason for this correlation operation is obtained from SiameseFC (Siamese Fully Connected) architecture, where the last layer output of siamese network for two input images (search and traget image) are convoluted (target image output is convoluted over search image output). After this operation, the region where the target image is present on search image has higher peak than other areas.

## 5.5 TRAINING

### 5.5.1 Loss function

Let $(A_x,\ A_y)$,$(A_w,\ A_h)$ denote the center of anchor box and shape of anchor box (width and height) respectively and let $(T_x,\ T_y)$,$(T_w,\ T_h)$ denote the center of ground-truth box and shape of ground-truth box (width and height) respectively.

$$\begin{aligned}
\delta[0] &= \frac{T_x - A_x}{A_w} \\
\delta[1] &= \frac{T_y - A_y}{A_h} \\
\delta[2] &= \ln \frac{T_w}{A_w} \\
\delta[3] &= \ln \frac{T_h}{A_h}
\end{aligned} \tag{5.9}$$

$$smooth_{L1}(x, \sigma) = \begin{cases} 0.5\sigma^2 x^2 & |x| < \dfrac{1}{\sigma^2} \\ |x| - \dfrac{1}{\sigma^2} & |x| \geq \dfrac{1}{\sigma^2} \end{cases} \tag{5.10}$$

$$L_{reg} = \sum_{i=0}^{3} smooth_{L1}(\delta[i], \sigma) \tag{5.11}$$

$L_{cls}$ is the cross-entropy loss

$$loss = L_{cls} + L_{reg} \tag{5.12}$$

The equation 5.12 represents the whole loss function for siameseRPN model which is a differentiable function. We can minimize this loss function by using optimizers namely RMSprop or Adam's optimizer.

# CHAPTER 6

# DEEPMOT: A DIFFERENTIABLE FRAMEWORK FOR TRAINING MULTIPLE OBJECT TRACKERS

DeepMOT [3] tries to overcome the problem faced in Simple Online Realtime Tracker paper including occlusion handling, considering deep appearence metrics for object tracking. It is an end-to-end trainable framework. This also comes under detection-based tracking.

**Importance features in DeepMOT**

The authors of DeepMOT have concentrated on few area which are as follows:

- The paper proposes an approximation of hungarian algorithm which can be differentiable. The reason for this development is the traditional hungarian algorithm is not differentiable and cannot be trained.
- The paper focuses on improving Multi-Object Tracking Accuracy (MOTA). MOTA is a measure that represents how much the trajectory is accurate or only the realobject should be captured (no missed objects) by the bounding box and each trajectory should always have a unique and consistant identity throught the time.
- The paper also focuses on improving Multi-Object Tracking Precision (MOTP). MOTP is a measure that tells how much that the trajectory is precise i.e each bouding boxes should enclose its associated object well.

## 6.1 DEEPMOT FRAMEWORK

For tracking multiple objects in the video at time $t$, several single object trackers were run parallely for tracking multiple objects providing $N_t$ estimated bounding boxes $[X_{t1},...X_{tN_t}]$ where $X_{tn} \epsilon \mathbb{R}^4$, $\forall$ t,n. Similarly there are $M_t$ ground truth bounding boxes $[O_{t1},...O_{tM_t}]$ where $O_{tn} \epsilon \mathbb{R}^4$, $\forall$ t,m. The pair wise distance between estimated bounding box and ground truth bounding box is stored in distance matrix $D_t \epsilon \mathbb{R}^{N_t \times M_t}$. The proposed deep hungarian network is then used to estimate soft assignment $A_t \epsilon [0,1]^{N_t \times M_t}$.

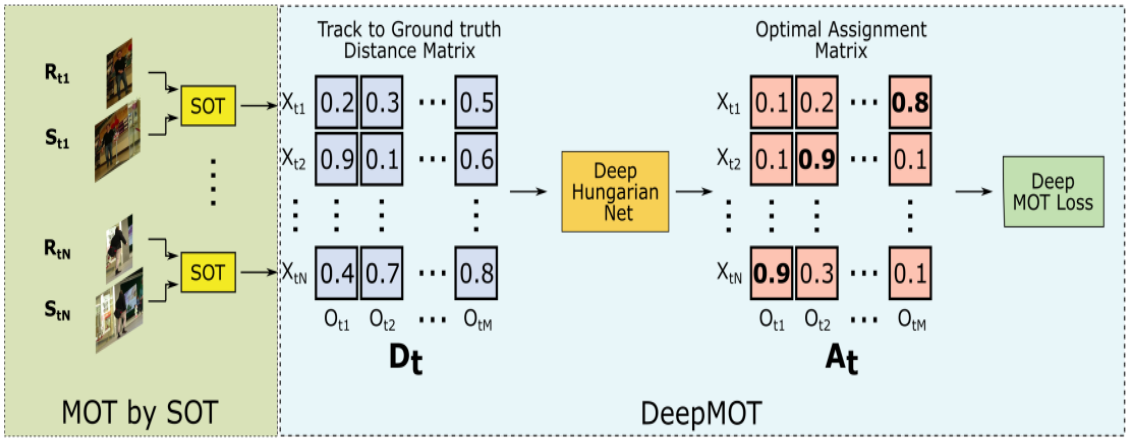


**Figure 6.1: DeepMOT framework**

The figure 6.1 represents the DeepMOT framework. Deep Hungarian Net and DeepMOT loss are discussed in the following sections. For DeepMOT loss the expected value for MOTA and MOTP are calculated using an approximation of MOTA and MOTP function which are differentable.

### 6.1.1 MOT by SOT

For tracking objects we run several single object trackers (e.g., **SiameseRPN**) parallely. At time t for each tracker object n, both reference image $R_{tn} \epsilon \mathbb{R}^{H \times W \times C}$ which is appearence information of target to track and search image $S_{tn} \epsilon \mathbb{R}^{H' \times W' \times C'}$ are used. Every SOT will predict the position of target image position in search image denoted as $X_{tn}$. Formally it can be represented as

$$X_{tn} = SOT(R_{tn}, S_{tn}, W_{SOT}), \forall n \tag{6.1}$$

The distance matrix can be formulated as below equation where L2 is the L2 norm. The reason for using L2 norm because it is differentable.

$$D_{tnm} = \frac{L2(X_{tn}, O_{tm}) + (1 - IOU(X_{tn}, O_{tm}))}{2} \tag{6.2}$$

### 6.1.2 DHN: Deep Hungarian Network

The aim of the Deep Hungarian Network is to compute an optimal softassignment $A_t$ given the distance matrix $D_t$. $A_t$ is a binary matrix. If the track n is assigned to ground truth m then $A_{tnm}$=1 and 0 othewise.

**Advantage of DHN over original hungarian**

- The number of estimated and ground truth bounding boxes may not be the same.
- Beyond a certain distance threshold $\tau_d$ , assignments should not be performed.
- It is differentiable. Hence the network can be trained on the dataset.

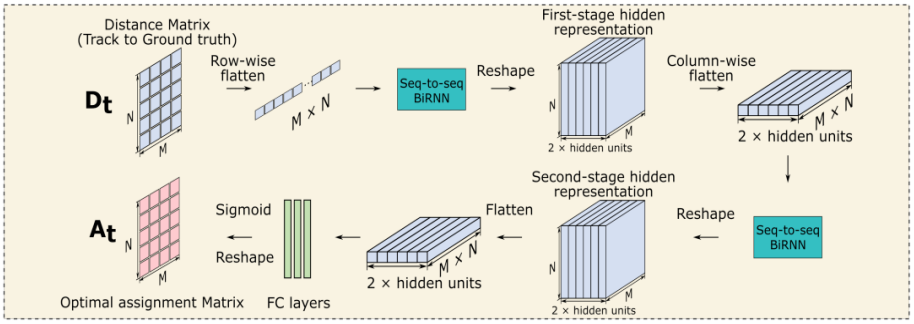


**Figure 6.2: Deep Hungarian Network**

The figure 6.3 represents the DHN network, where the input $D_t$ is made a series of transformations and $A_t$ is obtained. In DHN, the authors have used Bidirectional Recurrent Neural Network as an approximation for hungarian algorithm. The reason for using BiRNN is, it takes decision based on both past and future information as the assignment should be made global. Thus the differentable funtion can be written as follows:

$$A_t = DHN(D_t, W_{DHN}) \tag{6.3}$$

### 6.1.3 Approximation of MOTP and MOTA metrics

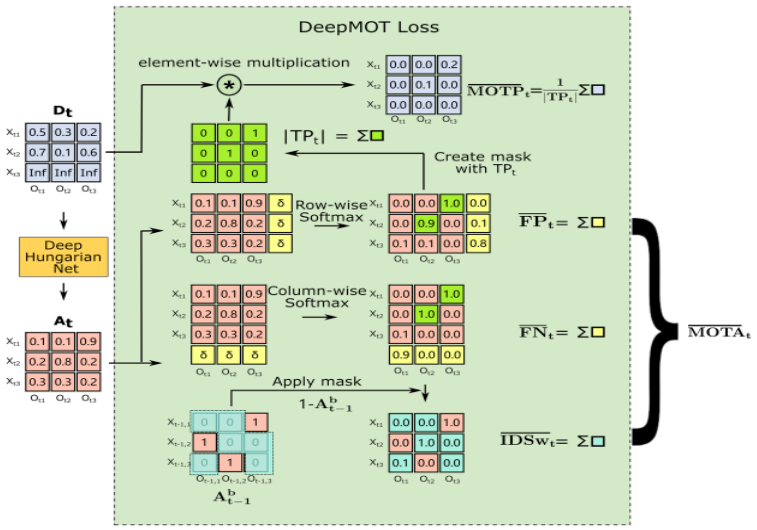


**Figure 6.3: Deep Hungarian Network**

**Actual MOTA**

$$MOTA = 1 - \frac{\sum_t (FP_t + FN_t + IDS_{wt})}{\sum_t (M_t)} \tag{6.4}$$

where FP indicated False Positive, FN indicated False Negative and IDS (Identity switch) labelling same object in two different frames with two different labels.

**How to calculate FP, FN, IDS?**

At time t, if a track is matched to the gound truth identity it is considered as True Positive (TP) else it is considered as False Positive and a missed ground truth is considered as False Negative (FN). The matching criteria is if IoU measure between two bounding box is less than 0.5. For a track marked as TP at time t and at the most previous time step if it assigned to different ground truth identity it is counted as identity switch (IDS).

**Approximate MOTA**

$$\begin{aligned} \overline{FN_t} &= \sum_m A^r_{t(N_t+1)m} \\ \overline{FP_t} &= \sum_m A^c_{tn(M_t+1)m} \\ \overline{IDs_{wt}} &= \sum_{n,m}^{N_t,M_t} (1 - A_{(t-1)nm}) A^r_{tnm} \end{aligned} \tag{6.5}$$

From the figure 6.3, for calculating the equation (6.5) the values $A^c$, $A^r$ matrixes over the yellow region are summed. These matrixes are formed by approximating idea of calculating False Negative, False Positive, Identity switiching by applying row-wise softmax and column-wise softmax because for calculating False Positive and False Negative it is necessary to look row-wise values and column-wise values respectively also to make it differentiable softmax is used.

$$\overline{MOTA_t} = 1 - \frac{\overline{FP_t} + \overline{FN_t} + \overline{IDS_{wt}}}{M_t} \tag{6.6}$$

**Actual MOTP**

$$MOTP = \frac{\sum_t \sum_{A^*_{tnm}=1} D_{tnm}}{\sum_t TP_t} \tag{6.7}$$

where D is the distance matrix. The idea behind the equation (6.7) is to sum the distance between the two matched bounding boxes (i.e distance between two bounding box of same object in two consecutive frames) and normalizing this value.

**Approximate MOTP**

$$\overline{MOTP_t} = \frac{\sum_{nm} A^b_{tnm} D_{tnm}}{|TP_t|} \tag{6.8}$$

**Total loss for DeepMOT**

$$\mathcal{L}_{DeepMOT} = 1 - \overline{MOTA_t} + \lambda \overline{MOTP_t} \tag{6.9}$$

The main objective is to minimize the loss function ($\mathcal{L}_{DeepMOT}$). Thus for minimizing $\mathcal{L}_{DeepMOT}$ it is necessary to maximize $\overline{MOTA_t}$, which means reducing False Positive, False Negative, Identity Switch and minimize $\overline{MOTP_t}$ which means minimizing sum of distance measure. By minimizing the sum of distance measure means decreasing the IoU value which inturn leads to predict bounding box which are more precise.

## 6.2 FOCAL LOSS

The Focal loss is designed to address the one-stage object detection scenerio (YOLOv3 is a one-stage object detector where there is no region proposal network only a single pass is required to detect objects). For this class of models there are extreme imblance between foreground and background classes during training (1:1000) as shown the figure 7.1.

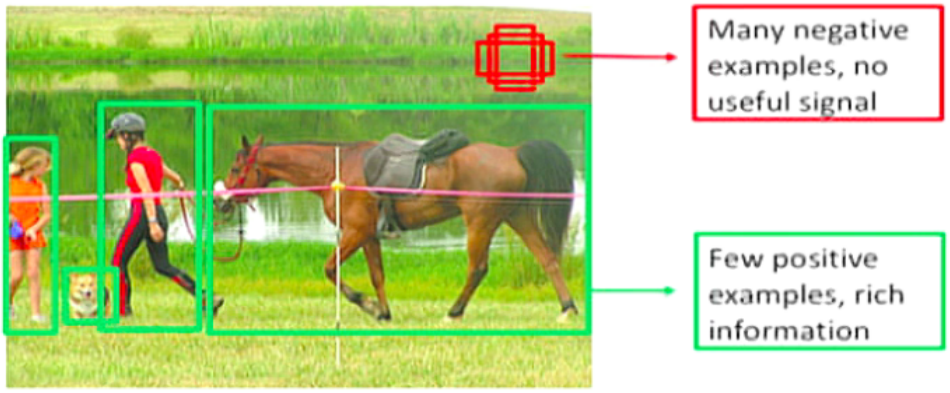


**Figure 6.4: class imbalance (more negative and less positive)**

By modifying the cross entropy loss focal loss can be obtained. Focal loss basically gives more weight for foreground class and less weight for background class. Hence, during training, if the model predicts a foreground wrongly it will be penalized more than wrong prediction of background class.

$$FL(p_t) = -\alpha(1-p_t)^{\gamma}\log(p_t) \tag{6.10}$$

where $\alpha \epsilon [0,1]$ for foreground and 1-$\alpha$ for background classes and $\gamma \epsilon [0,5]$.

## 6.3 TRAINING DEEPMOT

- Apply object detection algorithm over the dataset and generate detected bounding boxes.
- Generate augmented distance matix (M1) using detected bounding box and ground truth bounding box given along the dataset with distance formula (6.2).
- Create an optimal assignment matrix (M2) for the distance matrix (M1) using traditional hungarian algorithm.
- Train DHN network using input data as augmented distance matrix (M1) and output as optimal assignment matix (M2).
- Since M2 matrix contains more 0 entries than 1 the authors used focal loss for this class imblance with $\gamma$=2 and RMSprop as optimizer
- With the trained DHN network train the SiameseRPN using DeepMOT loss. The traditional SiameseRPN loss (5.9) is not used for training DeepMOT.

## 6.4 INFERENCE

For inference Deep Hungarian Network is removed only the SiameseRPN and Deep Affinity Network, an another object detector is used which was trained on MOT2015 dataset and it uses VGGNet architecture.

## 6.5 RESULTS AND DISCUSSIONS

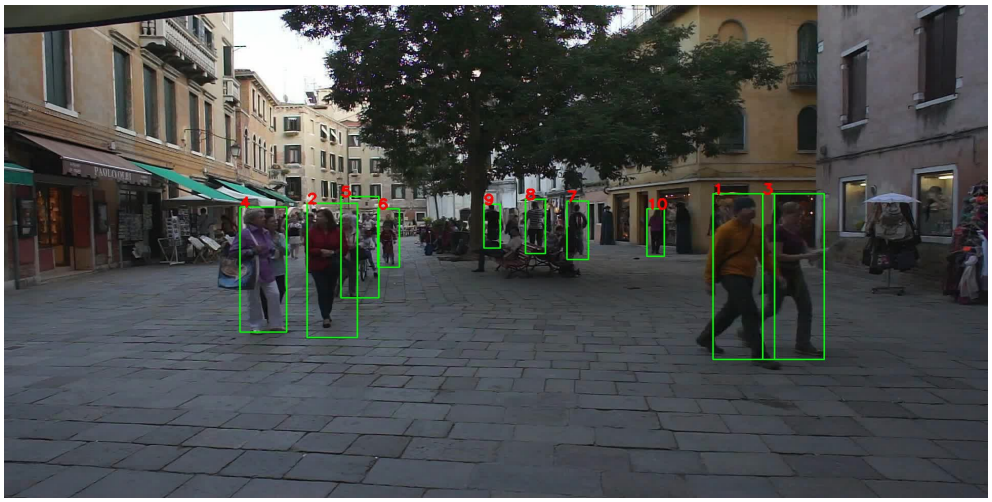

**Figure 6.5: Input video (frame 1)**

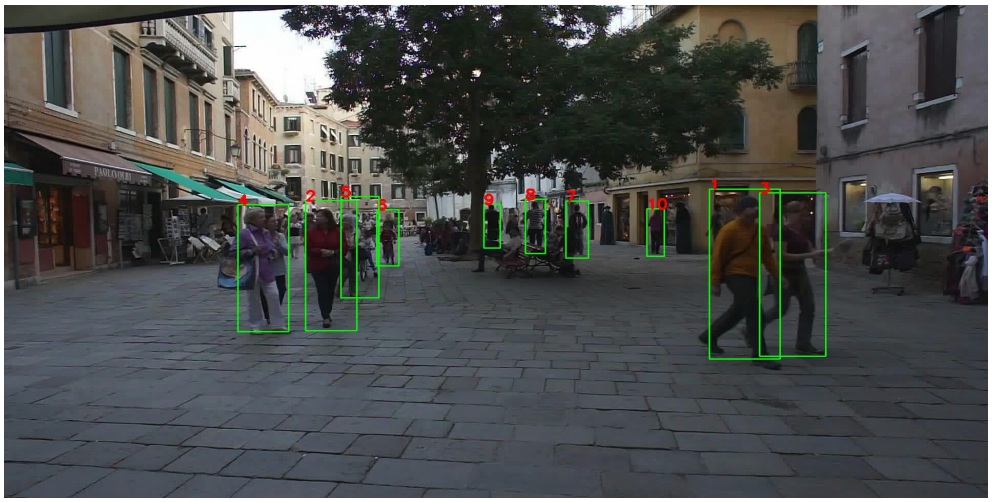

**Figure 6.6: Input video (frame 2)**

Figures 6.5, 6.6 shows the output of DeepMOT algorithm for tracking pedestrains in MOT2017 dataset. From the output DeepMOT it performs better than Simple Online Realtime Tracker model on MOT2017 dataset. DeepMOT performs with high precision since it bounds objects precisely.

# CHAPTER 7

# ABOUT THE DATASET

The dataset used for this project is **IDD dataset**. It si an Indian traffic video dataset collected from Bangalore by IIIT Hyderabad. The dataset containes the images and the annotations which contains the bounding box information. The dataset is 22GB is size. The following images shows the sample images from the dataset along with their annotations.

**Figure 7.1: Sample images in the dataset along with bounding boxes**

# CHAPTER 8

# CONCLUSION AND FUTURE WORK

A detailed study on the problem of traffic video analysis is performed. Deep learning version of object detection and object tracking are developed. An improvement in Simple Online Realtime Tracker algorithm has been done by using YOLOv3 for object detection. The problem of tracking vehicles is planned to improve in the following ways.

- Training DeepMOT model on IDD dataset and check it's performance over Electronic City traffic videos.
- Deploying the trained model on NVIDIA Jetson T2, an edge computing device for realtime usage.

# Bibliography

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos and Ben Upcroft, "Simple Online Realtime Tracker", arXiv preprint arXiv:1602.00763v2, 2017.

[2] Samira Ebrahimi Kahou, Vincent Michalski and Roland Memisevic, "RATM:Recurrent Attentive Tracking Model", arXiv preprint arXiv:1510.08660, 2015

[3] Yihong Xu, Yutong, Ban Xavier,Alameda-Pineda and Radu Horaud, "DeepMOT: A Differentiable Framework for Training Multiple Object Trackers", arXiv preprint arXiv:1906.06618, 2019.

[4] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu and Xiaolin Hu, "High Performance Visual Tracking with Siamese Region Proposal Network", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[5] ShiJie Sun, Naveed Akhtar, HuanSheng Song, Ajmal Mian and Mubarak Shah, "Deep Affinity Network for Multiple Object Tracking", arXiv preprint arXiv:1810.11780, ACM, 2018.

[6] Joseph Redmon, Ali Farhadi, "YOLO9000: Better, Faster, Stronger ", CVPR 2017.

[7] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, Xiaolin Hu, "High Performance Visual Tracking with Siamese Region Proposal Network", CVPR 2018.

[8] Li Fei-Fei, Rob Fergus, Pietro Perona, "A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories", ICCV 2003.

[9] Luca Bertinetto, Joo F. Henriques, Jack Valmadre, Philip H. S. Torr, Andrea Vedaldi, "Learning feed-forward one-shot learners", NIPS 2016.