Warsaw University of Technology

FACULTY OF
POWER AND AERONAUTICAL ENGINEERING

Institute of Aeronautics and Applied Mechanics

Master's diploma thesis

in the field of study Automatic Control and Robotics
and specialisation Robotics

People tracking in crowded locations using multiple object Tracking (MOT)

Haftom Solomon
student record book number ,317237

thesis supervisor
Andrzej Kordecki(Ph.D.)

WARSAW , 2022

załącznik nr 10 do zarządzenia
nr 46 /2016 Rektora PW

Politechnika Warszawska
Warsaw University of Technology

…….....
miejscowość i data
place and date

……..
imię i nazwisko studenta
name and surname of the student
……..
numer albumu
student record book number
…….……
kierunek studiów
field of study

OŚWIADCZENIE

DECLARATION

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.
Under the penalty of perjury, I hereby certify that I wrote my diploma thesis on my own, under the guidance of the thesis supervisor.

Jednocześnie oświadczam, że:
I also declare that:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- *this diploma thesis does not constitute infringement of copyright following the act of 4 February 1994 on copyright and related rights (Journal of Acts of 2006 no. 90, item 631 with further amendments) or personal rights protected under the civil law,*

- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- *the diploma thesis does not contain data or information acquired in an illegal way,*

- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- *the diploma thesis has never been the basis of any other official proceedings leading to the award of diplomas or professional degrees,*

- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- *all information included in the diploma thesis, derived from printed and electronic sources, has been documented with relevant references in the literature section,*

- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.
- *I am aware of the regulations at Warsaw University of Technology on management of copyright and related rights, industrial property rights and commercialisation.*

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

I certify that the content of the printed version of the diploma thesis, the content of the electronic version of the diploma thesis (on a CD) and the content of the diploma thesis in the Archive of Diploma Theses (APD module) of the USOS system are identical.

.....
czytelny podpis studenta
legible signature of the student

Abstract

Multiple Object Tracking is the most important topic in the field of computer vision.It can be implemented in different advanced technology application such as autonomous driving system, traffic monitoring and analyzing of peoples in different fields such as sport players.

Therefore, in this paper we applied MOT algorithm for tracking of peoples in crowded area with the help of detection system. The model solves MOT problems with the strategy of tracking-by-detection (DBT). The YOLOv4 which is used in our detection algorithm is presented and DeepSORT tracking is also discussed for our tracking algorithm. We use the Kalman filter to predict the locations of existing tracks in the current frame from the previous frame.

To improve the data association of DeepSORT which is solved as linear assignment problem with a generated cost matrix a set of new objects tracking data association cost matrices based on intersection over union and bounding box metrics is proposed. For the evaluation of the MOT by detection the YOLOv4 is used to detect and classify the objects available on images. It is also discussed the differences between YOLOv4 and YOLOv4-tiny, and how the tracking is dependent on the accuracy of detection output. Evaluation metrics for detection is presented based on the input dataset MOT20. we have achieved 97.7% a mAP value for our detection system of YOLOv4.

Keywords:Multiple Object Tracking, kalman filter, YOLOv4,DeepSORT,DBT

Streszczenie

Metody śledzenie wielu obiektów w obrazach (Multiple Object Tracking, MOT) są ważnymi metodami stosowanymi zadaniach wizji komputerowej. Zakres zastosowań algorytmów MOT obejmuje zaawansowane rozwiązania technologiczne do których zaliczamy: autonomiczne pojazdy, systemy monitorowania ruchu drogowego, czy analizy ruchu ludzi w różnych obszarach badań np. ruchu sportowców.

W pracy magisterskiej przedstawiono zastosowanie algorytmu MOT do śledzenie ludzi w zatłoczonych lokacjach z wykorzystaniem algorytmu detekcji, czyli z wykorzystaniem strategii tracking-by-detection (DBT). Algorytm YOLOv4 zastosowano do detekcji ludzi w obrazach, a moduł śledzenie wykorzystywał algorytm DeepSORT. Podstawą działa algorytmu śledzenie jest predykcja lokacji obiektu w obrazie za pomocą filtru Kalmana w klatce obecnej na podstawie danych z poprzedniej klatki obrazu wideo.

W celu poprawy metody asocjacji modułu śledzenia DeepSort, który rozwiązuje zestaw problemów liniowym za pomocą przypisania wykrytych obiektów do wygenerowanej macierzy kosztów bazującej na współczynniku IoU, został zastosowany nowych zestaw danych uczących. Poprawa wyników śledzenia została przeprowadzona przez zmianę ustawień sieci YOLOv4. Różnice pomiędzy przetestowanymi sieciami YOLOv4 i YOLOv4-tiny zostały opisane w pracy wraz z ich wpływem na wyniki śledzenia. W badaniach zastosowano bazę wejściową obrazów wideo MOT20, na której uzyskano wynik mAP wynoszący 97,7%.

Słowa klucze: Multiple Object Tracking, filtr Kalmana, YOLOv4, DeepSORT, DBT

Contents

List of Figures

List of Tables

Acronyms

MOT Multiple object tracking
SOT Single object tracking
MTT Multi target tracking
ANN Artificial neural networks
MAE Mean Absolute Error
MSE Mean Squared Error
RCNN Region-Based Convolutional Neural Networks
FRCNN Fast Region-Based Convolutional Neural Networks
YOLO You only look once
SSD Single shot detector
CNN Convolution Neural Networks
ROI Region of interface
SVM Support vector machine
RPN Region Proposal Network
DBT Detection-based multi-object tracking
IOU Intersection over Union
CVA Cost volume-based association
MFW Motion-guided feature warpe
CSP Cross Stage Partial
SPP Spatial pyramid pooling
PAN Path Aggregation Network
BoF Bag of Freebies
BoS Detection free tracking

Subsequent of Chapters

The following chapters are organized as follow:

chapter 1:-introduces about the theory of MOT and the application,Motivation and goal of the thesis.

Chapter 2:Explains overall the background of Deep learning and how the MOT is applied using the Deep learning detection.And also discusses general theory of artificial neural network (ANN),Activation functions,loss functions ,object detection and their types.

Chapter 3: the work of different latest papers of MOT are reviewed here.DBT,DFT,JDE, SORT,StrongSort,TrackFormer,Byte Track,Patch Track ,Fair MOT,and TraDes.

Chapter 4:describes the detection and object tracking of the model.first ,it will discuss in the detection model of YOLOv4 in detail and analysis by comparing another version of YOLOv4-tiny model. DeepSORT algorithm is used for the tracking of object and is described here in detail.

Chapter 5:the methods and experiments are explained here.starts with the benchmark dataset MOT20 ,the training of the model ,evaluation metrics and results of the detection and tracking outputs.

Chapter 6:concludes the work and discusses the future work.

Chapter 1

Introduction

In our daily basis peoples can move from one place to another in streets, stadiums, shopping malls, concerts, and other crowded public spaces. As security is vital for humans' daily movement, object tracking system plays a great role on managing the security of public crowded places.

Object tracking is one of the most important tasks in computer vision, it is the process of locating moving objects over time in videos. Which is to mean it is the task of estimating the position and other relevant information of moving object in video. Object tracking can be in single object tracking (SOT) and multi-object tracking (MOT) or multi target tracking (MTT). In this paper we will discuss about multi-object tracking (MOT).

MOT has a lot of applications in many disciplines including in sport events for tracking and analyzing players accurately [13]. It is also implemented in monitoring traffic by using trajectory of vehicles [32] and tracking of multi pedestrians in street [39]. In the biomedical field the mobility of cells is considered to assess tissue-repair and predict diseases in early stages [9]. Similarly multi-object tracking is a crucial component in an autonomous driving system as it provides pivotal information to take decisions such as obstacle avoidance, path planning, and intent recognition. [33] [27]

Multi object tracking as its name indicates is the process of locating multiple objects over a sequence of frames (video). The MOT algorithm has to obtain the different states (trace) of the objects in the different images but relating the same object over the different images. MOT can be manipulated in two different approaches based on the number of cameras, which is single camera of MOT and multi camera view of MOT, as the second approach is a bit complex to deal with overlapping and non-overlapping cameras as it creates problems in tracking of objects. Here in this paper, we will consider the MOT with a single camera approach.

1.1 Goal of the thesis

This paper mainly discusses about detection and tracking of peoples in crowded public services using Multi object tracking algorithm. Multiple Object Tracking (MOT) is a subclass of object tracking where the goal is to track not a single but multiple objects from one or multiple classes.when the movement of the target object is too high, a tracking

algorithm may not be able to maintain the track of the object. So that object tracking usually involves the process of object detection. In such a system, the detection and tracking algorithms are independent of each other and the tracking algorithm doesn't have any effect on detection results. In this thesis, we implemented MOT algorithm in which YOLOv4 uses for detecting objects and DeepSORT for finding object trajectories.Mainly it is TBD (tracking based detection) system based on the data provided in a benchmark for Multi-Object Tracking (MOT20) dataset.

Figure 1.1: MOT20-06/000004,output YOLOv4-tiny

(a) detection output **(b)** tracking output

Figure 1.2: sample outputs of detection and tracking

1.2 Motivation

Recently, deep learning has shown a great potential and produced some excellent results towards solving a wide spectrum of computer vision tasks. Multi-object tracking is one of those areas of research which has benefited from the power of deep learning. One such useful MOT case is the development of self-driving cars. A robot driver needs to be constantly tracking the surrounding objects, such as pedestrians, cyclists, and vehicles to avoid collisions of any kind. Other applications are expected to make valuable contributions to other various medical applications, robot-assisted procedures, cardiac surgeries. A surveillance case is one of the main ways that MOT could be implemented, for tracking and detecting people or objects to identify suspicious people around crowded areas. In many sports such as soccer, basketball and hockey, it is valuable to precisely track different players and balls. This helps the audiences to easily visualize the individual motion trajectory of each player .

Despite that MOT can be studied for the research efforts in numerous ways of applications depending on their corresponding problems they solve. The system can be real-time, in which it will be capable of responding to the real world in the desired time. MOT algorithms have different challenges which arise during tracking, occlusion is one of the most common challenges. Occlusion sensitivity allows the user to identify which feature of the object is confusing the network. And some of MOT trackers can be based on detecting or free detecting.Thereby, in this thesis we aim to design a MOT tracking system based on detecting using YOLOv4 detector and DeepSORT as our tracking algorithm based on the dataset challenges of the MOT20.

Chapter 2

Background of Deep learning

2.1 General Theory of Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) are biologically inspired neurons which are configured in computing systems to perform different specific tasks.The tasks can be such as clustering,pattern,classification recognition.For performing the tasks the systems should have to learn or train by processing examples, in other words the systems are not programmed based on algorithms or specific rules.

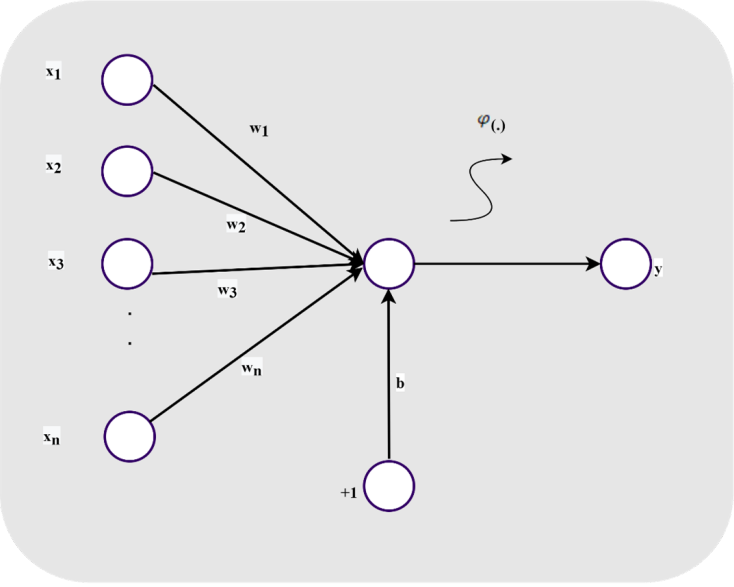


Figure 2.1: perceptron

An Artificial Neural Network in machine learning is an information processing technique that uses the same concept of biological neural networks but not identical to it.In 1958 a psychologist Frank Rosenblatt invented the first Artificial Neural Network called

Perceptron. [36] The perceptron is simple model of neuron that receives a multiple input signals x and computes the weighted w sum of those inputs ,then checks the certain threshold ϕ and outputs the result y.Which output it sends is determined by the activation function and is often chosen to be between 0 and 1 or -1 and 1.neuron mode can be written as equation 2.1.

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b = 1\right) \tag{2.1}$$

But a single perceptron won't be enough to learn complicated systems. Fortunately, we need to expand the single perceptron, to create a multi-layer perceptron model. To build a network of perceptrons, we can connect layers of perceptrons, using a multi-layer perceptron model as shown in figure 2.2.The input layer is responsible for receiving the inputs from external sources in different form of data like *csv* file,.*txt* file or web services etc.Those input data will be transfer for further processing to the subsequent layers of artificial neurons.

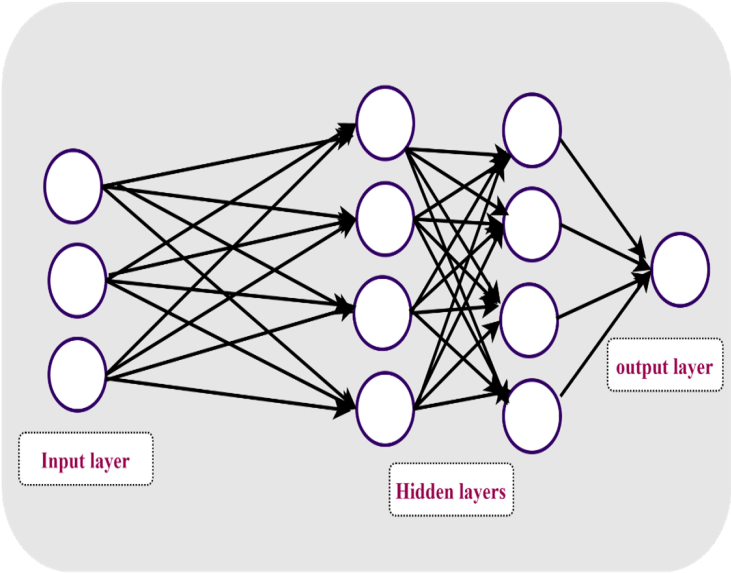


Figure 2.2: Multi-layer ANN

The hidden layers are neural networks intermediate between the input layer and the output layers which process the data by applying non-linear function to them.These layers are the basic component that helps a neural network to learn a very complex tasks and achieve a desired excellent performance.They perform multiple functions at the same time such as data transformation, automatic feature creation, etc.The hidden layers are private to the neural network ,in which they are hidden to the external system.The neural network solution depends on the number of hidden layers, the larger number of hidden layer takes longer time to get the output.

The last type of layer is the output layer which holds the final result or the output of the problem.The output layer takes the inputs which are passed in from the layers before it and performs the calculations through its neurons and then the output is computed. But in any complex neural network the output layer receives inputs from the previous hidden layers. The most commonly used activation function in a binary classification setting is the logistic sigmoid, while in multi class settings, the softmax is most frequently used.

2.2 Activation Functions

when the input layers receives different data such as .csv file or images .In which in our case the inputs were .txt file.These inputs with their corresponding weights will be calculated to the linear function in which these linear function will be used by the activation function.And then the calculated output of the activation function will pass to the next layer of the network.In most of the deep learning models the activation function are used to transform the nonlinear inputs to get better result for complex neural networks. There are different types of activation functions which are commonly used in neural networks.

2.2.1 Linear activation function

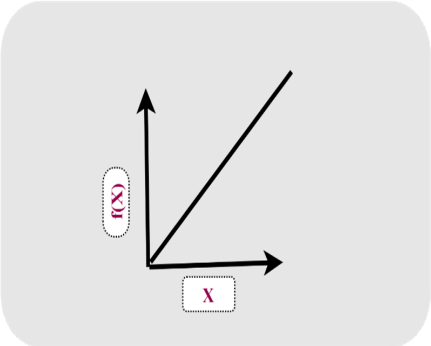


Figure 2.3: linear function

It is a simple straight line (Figure2.3) in which it is directly proportional to the input. Mathematically it can be represented as equation 2.2.

$$f(x) = x \tag{2.2}$$

2.2.2 The Sigmoid function

A function characterized by S-shape curve and has an output ranged in the interval between [0,1] and has an input ranged from large negative number to large positive

number.The sigmoid function $f(x)$ has mathematical shown in equation below, where x is the input.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

Since the range of the sigmoid is between 0 and 1, most of the time this function will help for models which predicts probability as an output.

2.2.3 Rectified Linear Unit (ReLU) function

The ReLu is short piece-wise linear function in which the output will be zero for negative inputs and for positive inputs it will return the input value by itself as shown in equation 2.4.It is the most commonly used activation function [7] in deeplearning models.The ReLu is used in neural networks to prevent the vanishing of gradients .And the ReLu is served instead of sigmoid and tanh in the hidden layers which directs to problem with vanishing of gradients.

$$f(x) = max(0, x) \tag{2.4}$$

2.2.4 Hyperbolic Tangent Function (Tanh)

This function is modifed version of sigmoid function with same S-shaped curve characteristics .The difference between them is the the range interval in the ouputs ,in which the tanh output is modifed to range -1 to 1. Mathematically it cab described as equation 2.5.

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.5}$$

2.3 Loss Functions

The loss function is a neural network characteristics which measures the the difference between the desired output and the output from machine learning model .From loss function the gradients can be calculated to update the weights of the inputs.

In Deep learning Loss function is divided in to two different types : Classification loss and Regression Loss.Classification loss is used in neural network models based on classification neural, where as Regression loss is used in neural networks modeled for regression.In classification neural networks the outputs are listed in a set of probabilities to different categories according to the given input.Example of classification loss are Binary Cross-Entropy, Categorical Cross-Entropy. In regression the model will calculate the prediction according to the corresponding outputs.Examples are Mean Absolute Error,Mean Squared Error [49].

2.4 Object Detection

In this section we will discuss the general view of different techniques the object detector and object tracking as well. Object detection is one of the important areas in deep learning and computer vision which has been determined the numerous applications such

as video surveillance, image segmentation, medical imaging, object tracking and several other technology applications. It is an important computer vision task used to detect instances of visual objects of certain classes (for example, humans, animals, cars, or buildings) in digital images such as photos or video frames.

Object detection is merely to recognize the object with a bounding box in the image or video frame. It is slightly more advanced, as it creates a bounding box around the classified object. where in image classification, we can simply categorize(classify) that an object is in the image or not in terms of the likelihood (Probability). The purpose of object detection is to develop models that provide the most important information needed by computer vision applications: "What objects are where?".

2.4.1 Modes of object detection

The mode of object detection can be classified in two basic approaches:**Machine learning based object detection approach** and **Deep learning-based object detection approach**.There are lots of different types of object detection algorithms which are categorized under these two modes.few of detection algorithms are traditional and some of them are based on modern approaches.

Machine learning based object detection approach

These are traditional based approaches in which most of the time they used computer vision techniques to look the feature of the object.The feature of the object can be identified using color histogram or edges.Then after these feature will be passed to the regression for identifying the location of the object.generally these are not learned system.The most common techniques that used for the extraction feature of the images are listed below:

- Histogram of oriented gradients (HOG) [12]
- Speeded-up robust features (SURF) [1]
- Color histograms [40]
- Haar wavelets [24]
- Local binary patterns (LBP) [19]

Deep learning-based object detection approach

These are a modern approaches based on deep learning developed to solve the limitation of traditional approaches object detection.They used CNN for automatic extraction of complex features.These deep learning based approaches consists:

- RCNN [16]
- SPP-NET [17]
- FRCNN [15]

- FasterRCNN [35]
- G-CNN [31]
- YOLO [34]
- SSD [26]

2.5 Types of object detection

Generally, Object detection is categorized into two main stages figure 2.4 shows different types of object detection classified under those category.

1. Multi-stage object detectors
2. Single-stage object detectors

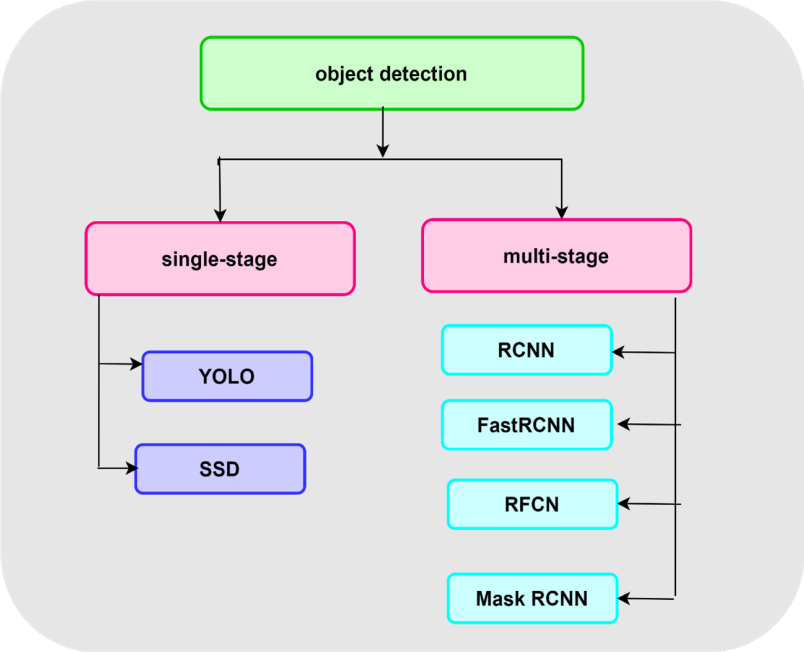


Figure 2.4: Types of object detection

2.5.1 Multi-stage Object detection

The multi-stage object detection has two stages to perform the output of the detection .The first one is generation of regional proposals and the next stage is object classification for each proposal regions ,in addition bounding box for regression for refining proposal can be done. Examples of object detection architectures that are 2 stages oriented include R-CNN [16],Fast-RCNN [15], Faster-RCNN [35], Mask-RCNN and others.

R-CNN

R-CNN is a region based Object Detection Algorithm developed by Girshick,from UC Berkeley in 2014. [16].R-CNN stands for Region-based Convolutional Neural Network. The key concept behind the R-CNN series is region proposals. Region proposals are used to localize objects within an image.It has achieved a mean average precision (mAP) of 53.3% with more than 30% improvement over the previous result on PASCAL VOC 2012 [38].

The Region-based Convolutional Network method (RCNN) is a combination of region proposals with Convolution Neural Networks (CNNs). R-CNN helps in localizing objects with a deep network and training a high-capacity model with only a small quantity of annotated detection data. It achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN has the capability to scale to thousands of object classes without resorting to approximate techniques, including hashing. As we

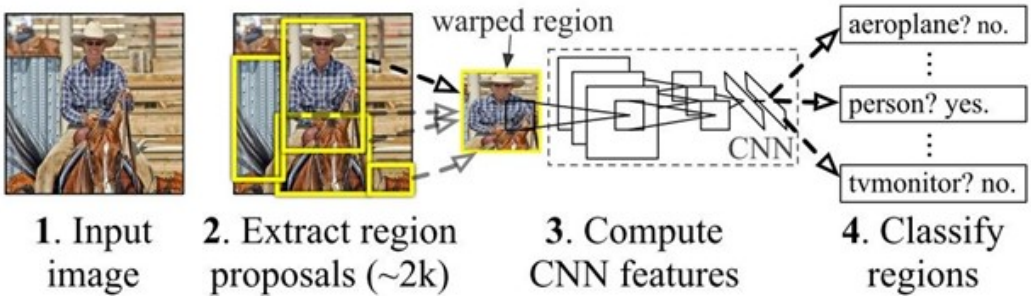


Figure 2.5: R-CNN [16]

can see from the Figure 2.5 R-CNN consists of three modules:Region Proposal, Feature Extractor and Classifier.

1. **regional proposal** R-CNN uses selective search to generate around 2000 proposals for each image.And these groups of regions are combined together to be investigated

2. **feature extractor** each proposed region will be trained by a CNN network and the last layer (4096 features) will be extracted as features so the final output from Feature extractor will be Number of proposed regions x 4096.

3. **Classifier** is to classify the objects inside each regions. To do this a linear SVM model is trained for classification, Specifically one SVM model for each class.

The R-CNN has different limitations that emerged in the model by itself each stage of the model independent components thus,it cannot be trained end-to-end.It requires a large size of storage when it caches the extracted features from the pre-trained CNN.The other main drawback of R-CNN is the model depends on the selective search algorithm which takes a lot of time.As an extension of the R-CNN model, the Fast R-CNN [15] model is proposed to overcome some limitations.

Fast R-CNN

This object detection is introduced to solve the limitation of R-CNN,it introduces a multi-task loss on classification and box regression by proposing a novel CNN architecture named Fast R-CNN. The general architecture of Fast R-CNN is shown in figure 2.6. The model have of a single-stage, compared to the three stages in R-CNN. It just accepts an image as an input and results the class probabilities and bounding boxes of the detected objects.The feature map from the last convolutional layer is fed to an ROI Pooling layer.it is because to extract a fixed-length feature vector from each region proposal.

The extracted feature vector using the ROI Pooling [8] is then transformed to some FC layers. The output of the last FC layer is split into two branches Softmax layer to predict the class scores and FC layer to predict the bounding boxes of the detected object. In R-CNN, each region proposal is fed to the model independently from the other region proposals. The Fast R-CNN [15] is faster than the R-CNN [16] as it shares computations across multiple proposals. Fast R-CNN is generally more accurate and

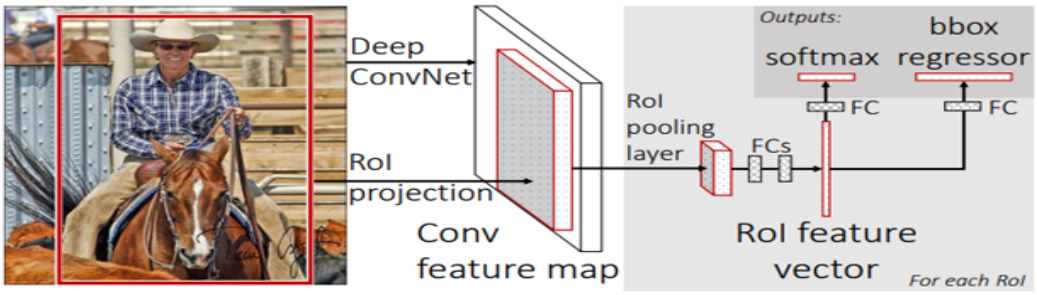


Figure 2.6: Fast R-CNN [16]

shares convolutional layer calculations across all proposals (ROIs) rather than doing the calculations for each proposal independently . To improve the speed of Fast R-CNN an extension of this detection has introduced which is Faster R-CNN.

Faster R-CNN

Faster R-CNN uses a region proposal method to create the sets of regions.Faster R-CNN have an additional CNN for gaining the regional proposal so called regional proposal network, RPN. These regions are then used in concert with a Fast R-CNN [15] model in a single model design. These improvements both reduce the number of region proposals and accelerate the test-time operation of the model to near real-time.The architecture of Faster R-CNN [35] is shown in the figure 2.7. It consists of 2 parts:

1. **Region Proposal Network** Convolutions neural network for proposing regions and the type of object to consider in the region.

2. **Fast R-CNN** Convolution neural network for extracting features from the proposed regions and outputting the bounding box and class labels.

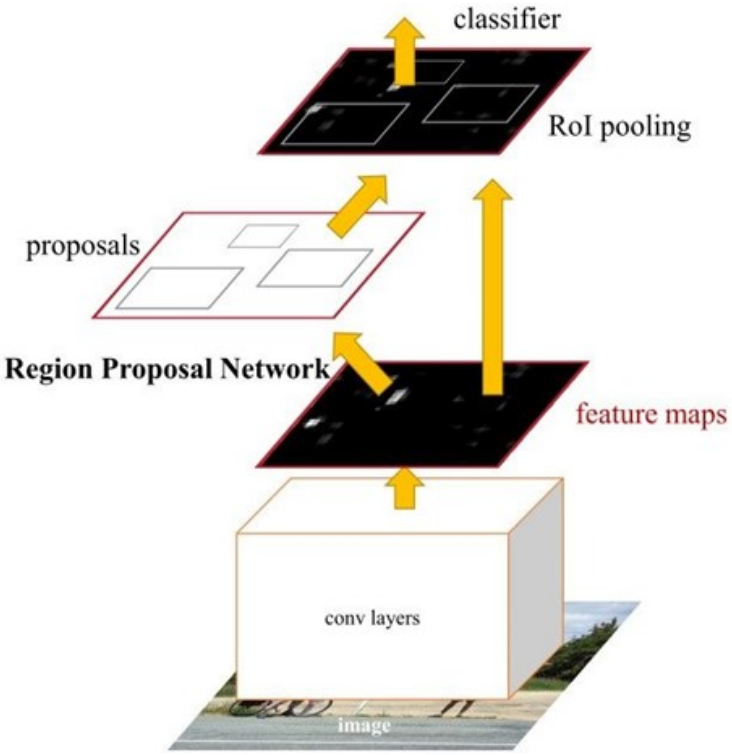


Figure 2.7: Faster R-CNN [35]

Region Proposal Network (RPN): The RPN [15] applies the concept of attention in neural network for generating region proposals.The R-CNN and Fast R-CNN models depend on the Selective Search algorithm for generating region proposals. Each proposal is fed to a pre-trained CNN for classification.RPN takes an image of arbitrary size to generate a set of rectangular object proposals. The important point here is to note that RPN operates on a specific conv layer with the preceding layers shared with the object detection network. In other words, to generate "proposals" for the region where the object lies, a small network is slide over a convolutional feature map that is the output by the last convolutional layer.

In summary, instead of slow selective search, proposes RPN to train the bounding box proposal process. And RPN model predicts the probability, location of an object on an anchor.

2.5.2 Single-stage object detections

These models skip the region proposal step in two-stage object detection.single-stage model predict bounding boxes over the images without the region proposal step.Hence this models are faster and structurally simpler than of multi-stage object detection,But they are not good at recognizing irregularly shaped objects or a group of small objects.The most popular one-stage detectors include the YOLO, SSD, and RetinaNet.

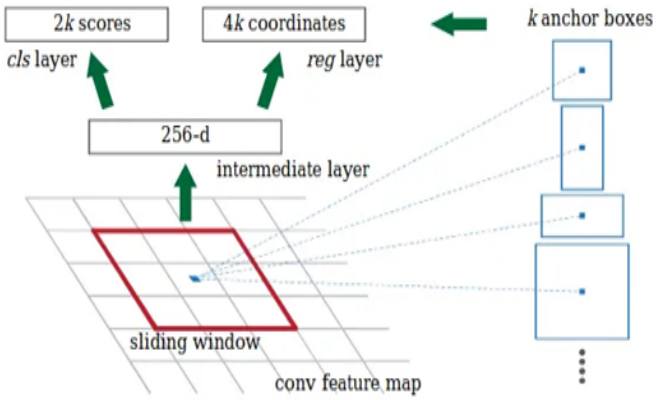


Figure 2.8: RPN in Faster R-CNN [35]

Single-Shot Detector (SSD)

As its name reveals SSD [26] is an object detection model which detects objects in a single pass,in which it saves a lot of time.SSD model produces predictions at different scales from the feature maps of different scales and it separates predictions by aspect ratio. This techniques help SSD to be an end-to-end model.The SSD model consists two parts the backbone model and SSD head. The Backbone model is a typical pre-trained

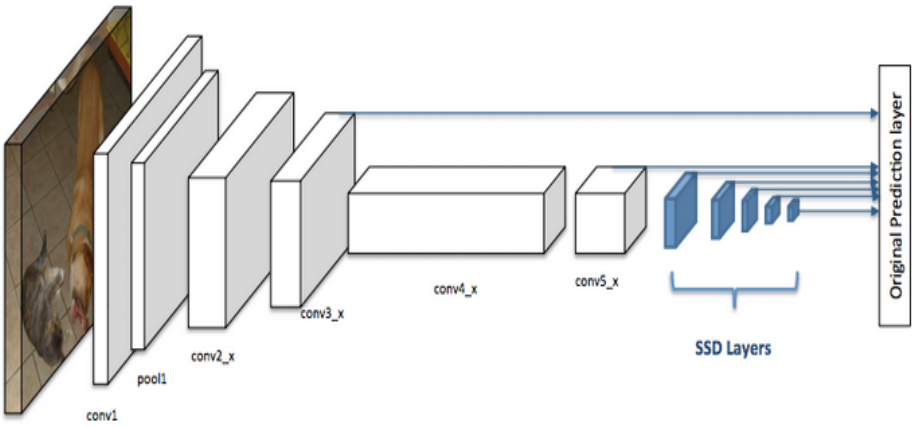


Figure 2.9: Architecture of a convolutional neural network with a SSD detector [26]

image classification network that works as the feature map extractor.In order to get only the extracted feature maps the final image classification layer of the model will be removed.The SSD head is just one or more convolutional layers added to this backbone and this gives us the output as the bounding boxes over the objects.The architecture of a convolutional neural network with a SSD detector is shown in figure 2.9 and the first

few layers (white boxes) are the backbone, the last few layers (blue boxes) represent the SSD head.

Another model which is classified under sing-stage object detector is YOLO family object detector which is described in details in section 4.1.

Chapter 3

State of the art

This section illustrates the related works of Object detection and tracking. There are different algorithms described under both detection and tracking. Multiple object tracking (MOT) concerns identifying objects of interest and tracking their moving trajectories in video sequences. Intuitively, successful MOT algorithms need to be able to handle subtle appearance differences between multiple tracked objects and resolve the ambiguity via other cues, such as motion, when the targets are visually indistinguishable. With the powerful appearance encoding capability of CNN, the tracking by- detection paradigm dominates MOT methods in the past decade [2] [44]. Highly accurate CNN-based object detection is first performed in all frames independently, and then association of these detected objects across frames is performed to establish tracks of consistent object IDs. Different research used different paradigm of tracking.

3.1 Detection-based tracking paradigm (DBT)

One major paradigm in MOT is tracking-by-detection, where the MOT systems first obtain detections for each frame and then associate them across frames to form tracks. In DBT, objects are at first localized in each frame and then object hypotheses are linked into trajectories. [28] Since the object detection is a standalone step in the tracking process, one benefit of tracking-by-detection methods is the flexibility to pair different object detection models with different association strategies, thus be benefited directly from advancement in the area of object detection. [5] The consecutive video frames are given to a pretrained object detector that gives detection hypothesis which in turn is used to form tracking trajectories.Figure 3.1 shows the flow of DBT.

It is more popular because new objects are detected and disappearing objects are terminated automatically. In these approaches, the tracker is used for the failure cases of object detection. In another approach, object detector is run for every n frame and the remaining predictions are done using the tracker.

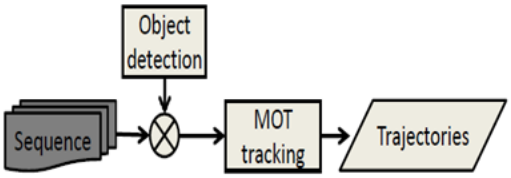


Figure 3.1: procedure flow of DBT [29]

3.2 Detection free tracking paradigm (DFT)

This paradigm doesn't rely on the former object detector for providing object hypotheses. As shown in Figure 3.2 DFT requires manual initialization of a fixed number of objects in the first frame, then localizes these objects in subsequent frames DBT is more popular since it can handle the occurrence of new objects and disappearance of existing objects naturally. DFT requires manual initialization of each object, thus it cannot deal with the case when new objects appearing the middle of frames. However, it is free of pre-trained object detectors.

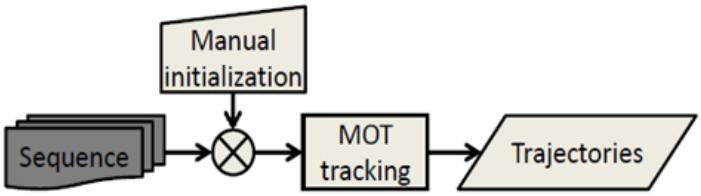


Figure 3.2: procedure flow of DfT [29]

According to the processing mode of tracking MOT can be classified as **online tracking** and **offline tracking**. online tracking uses only the actual or past frames to identify or re-identify the detect objects (real-time processing), and offline tracking can use all the sequence of frames, past, present, and future frames (posterior processing).

3.3 Joint Detection and Embedding (JDE)

Tracking-by-detection paradigm requires two components: detection and embedding (Re-ID) extractor. And this method by convenience is called as Separate Detection and Embedding (**SDE**), in which bounding boxes detected from the new-coming frame are allocated to the existing trajectories.

The methods of SDE brings critical challenges in building a real-time MOT system, which

is essential issue for implementation and practice. To resolve this challenge researchers introduced a new paradigm called Joint Detection and Embedding (**JDE**) [43]which attempt to jointly learns the detector and embedding model in a single shot deep.

Joint Detection and Embedding (**JDE**) is a single-shot detector (**SSD**) designed to solve a multi-task learning problem. This single-shot detector is designed to solve a multi-task learning problem, i.e., anchor classification, bounding box regression and embedding learning.

JDE uses base architecture so called Feature Pyramid Network (FPN) [43] as feature extractor. FPN is a feature extractor that takes a single-scale image of an arbitrary size as input, and outputs proportionally sized feature maps at multiple levels, in a fully convolutional fashion. The construction of FPN pyramid(figure 3.3) is composed of a bottom-up pathway and a top-down pathway. The bottom-up pathway is the usual convolutional network for feature extraction. As we go up, the spatial resolution decreases.

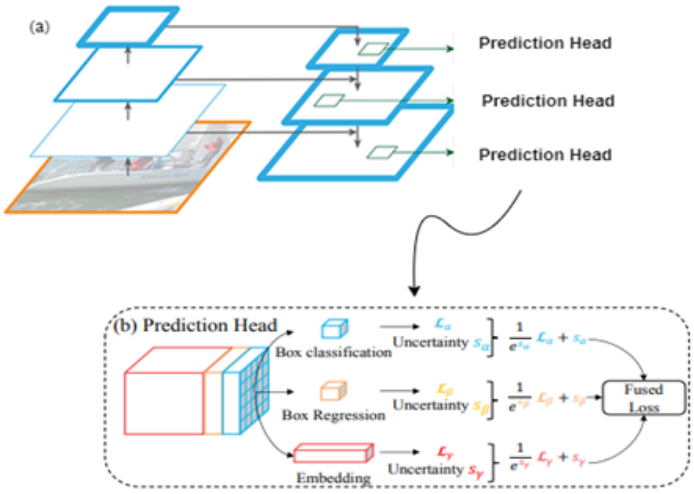


Figure 3.3: (a) FPN network Architecture (b)Prediction head [43] and [25]

To achieve object tracking, besides bounding boxes, and classes, the JDE model also outputs appearance embedding vectors when processing the frames. These appearance embeddings are compared to embeddings of previously detected objects using an affinity matrix. Finally, the Hungarian algorithm and Kalman filter are used for smoothing out the trajectories and predicting the locations of previously detected objects in the current frame.

3.4 SORT

This Simple Online and Real-Time Tracking (SORT) [2] algorithm is most common object tracking which is used to track objects at real-time and uses three basic methods which is object detection, estimation model and data association of objects for achieving the output. Using the Kalman Filter method, it is estimated where a detected object will be in the next image. The method uses the Hungarian algorithm to accurately associate detected objects (by an object detector) with objects that are being tracked.

3.4.1 Object Detection

In order to follow the tracking of the object, first the object should have to be detected which can be any deep learning detector. Most common utilized is **Faster R-CNN (VGG16).**

3.4.2 Estimation model

To further improve tracking accuracy, it is necessary to introduce the motion information of targets. As the velocity of targets in adjacent video frames is relatively stable, The Kalman filter [22] is used to predict the motion state of targets in the next frame. Firstly, by initialize a new motion state model based on the detection results for targets that first appear in the video. The motion state model is defined as:

$$X = [u, v, s, r, \dot{v}, \dot{u}, \dot{s}]^T \tag{3.1}$$

Here, u and v represent the horizontal and vertical pixel position of the center of the target, while the s and r scales represent the scale (area) and aspect ratio of the target's bounding box, respectively $\dot{v}, \dot{u}$ represents the horizontal and vertical velocity of the targets. When a detection is associated to a target, the detected bounding box is used to update the target state where the velocity components are solved optimally via a Kalman filter framework [2]

3.4.3 Data Association model

This module receives, as input, N detected bounding boxes and M predicted bounding boxes (acquired from their respective KF). The module formulates a linear assignment problem by computing a cost matrix between each detected bounding box and all predicted bounding boxes(respectively $D_i = i\varepsilon\{1, ..N\} and\ P_i = i\varepsilon\{1, ..M\}$) with the Intersection over Union (**IoU**) as metric

$$IoU(D, P) = \begin{pmatrix} iou(D_1, P_1) & iou(D_1, P_2) & \cdots & iou(D_1, P_M) \\ iou(D_2, P_1) & iou(D_2, P_2) & \cdots & iou(D_2, P_M) \\ \vdots & \vdots & \ddots & \vdots \\ iou(D_N, P_1) & iou(D_N, P_2) & \cdots & iou(D_N, P_M) \end{pmatrix} \tag{3.2}$$

where the IoU between a detected bounding box and a predicted bounding box is given by

$$iou(D_i, P_i) = \frac{D_i \cap P_i}{D_i \cup P_i} \tag{3.3}$$

After computing the cost matrix, the Hungarian algorithm [23] is used to associate the bounding boxes. a minimum IOU is imposed to reject assignments where the detection to target overlap is less than $(IOU)_{min}$.
Object association is made by looking at the intersection of previously detected and predicted objects. For the same object, the intersection is tried to be maximized. If the tracked object is not found in the next image, detected incorrectly or is not visible for any reason, the object is considered to have left the image. When the object starts to appear again, a new number is assigned.
The SORT algorithm gives a good result in terms of speed and success compared to many real-time multi-object tracking algorithms used. SORT algorithm gives results at 260 Hz on a computer with 16 GB RAM, Intel i7 2.5GHz single core specifications. [2] When object tracking using SORT algorithm, it may not give successful results in cases where objects collide, or the object does not appear in the next image. In these cases, the Deep SORT algorithm has been proposed to increase the success.

3.5 StrongSORT

This method is among mostly used object tracking framework which is an extension to DeepSORT. According to the paper [11] StrongSORT can be designed by equipping the DeepSORT with advanced components in different aspects and can achieve also new SOTA in datasets MOT17 and MOT20.The improvements are based on two appearance-free algorithms, AFLink and Gaussian-smoothed interpolation algorithm (GSI), which can be plugged into various trackers to improve their performance by a large margin. As StrongSORT is the extension of DeepSORT it is vital to look how the DeepSORT works.

3.5.1 Summary on DeepSORT

The DeepSORT is the extension of SORT which modifies the problem in SORT. The problem with SORT is the frequent ID switches as sort uses a simple motion model and does not handle occluded tracks well. Deep sort uses the appearance features to track objects through longer periods of occlusion. The cost only consists of appearance metrics, although bounding box distance is used as a gating process.
The DeepSORT has two branch of framework which is the appearance branch and motion branch as shown in Figure 3.4 .The appearance feature a simple CNN is used for as deep appearance descriptor which is used on application extracting appearance features. Like SORT Kalman filter is crucial component in DeepSORT with constant velocity motion and linear observation model, where bounding box is direct observations of the object state.

By having new bounding boxes tracked from the Kalman filter, the next problem lies in associating new detections with the new predictions. To incorporate motion information

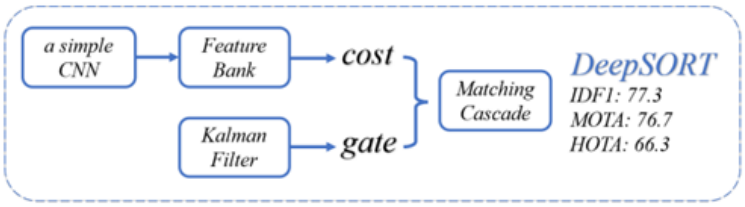


Figure 3.4: (a) Framework for DeepSORT [11]

Authors, use the (squared) Mahalanobis [44]distance between predicted Kalman states and newly arrived measurements. As new detections come, the smallest cosine distance between the feature bank R_i of the i^{th} tracklet and the feature f_i of the j^{th}

$$d(i,j) = min\{1 - f_k^T f_k^{(i)} | f_k^{(i)} \varepsilon R_i\} \tag{3.4}$$

Then, the matching cascade algorithm is proposed to solve the association task as a series of subproblems instead of a global assignment problem

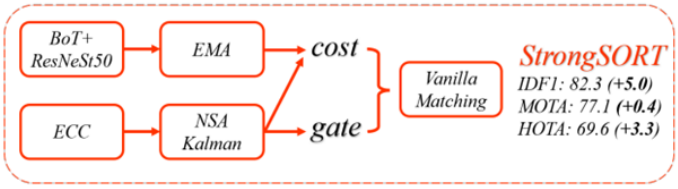


Figure 3.5: Framework of StrongSORT [11]

Then the difference of DeepSORT and StrongSORT is explained in the figure 3.5 in which :-

- **ECC** : a compensation method for camera motion
- **EMA** : Linear update of embedding.
- **NSA Kalman**: Kalman filter to remove noise
- **Additional arrows** : Add a movement distance limit to the embedding metric matrix

3.6 TrackFormer

This method of MOT tracking introduces the **tracking-by-attention** [30] paradigm which uses attention for data association and jointly performs the tracking and detection. TrackFormer is an end-to-end trainable Transformer [41] encoder-decoder architecture. It encodes frame-level features from a convolutional neural network (CNN) and decodes queries into bounding boxes associated with identities. ResNet-50 is used as common CNN backbone for frame level featuring.

3.6.1 ResNet-50

ResNet stands for Residual Network. It is an innovative neural network that was first introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their 2015 computer vision research paper titled 'Deep Residual Learning for Image Recognition [18] . ResNet-50 is a variant of ResNet with 50 neural network layers deep.

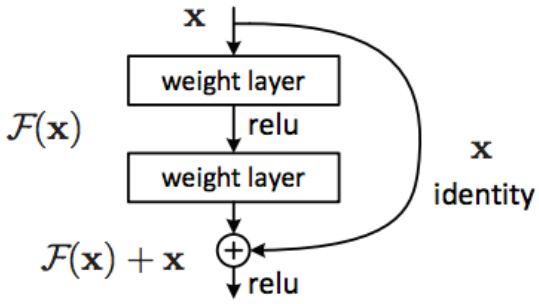


Figure 3.6: Skip Connection

Convolutional Neural Networks have a major disadvantage 'Vanishing Gradient Problem'. During back-propagation, the value of gradient decreases significantly, thus hardly any change comes to weights. To overcome this,ResNet is used. It makes use of *"skip connection"* which is shown in Figure 3.6. In other words, the skip connections add the outputs from previous layers to the outputs of stacked layers, making it possible to train much deeper networks than previously possible.

3.6.2 TrackFormer training

According to the computer vision paper [30]TrackFormer requires a frame-to-frame tracking training. A TrackFormer working principles is summarized in Figure 3.7.At frame t = 0, the decoder transforms N_{object} object queries (white) to output embeddings either initializing new autoregressive track queries or predicting the background class (crossed). On subsequent frames, the decoder processes the joint set of $N_{object} + N_{track}$ queries to follow or remove (blue) existing tracks as well as initialize new tracks (purple).

3.7 Byte Track

Most object detection algorithms ignore the bounding boxes with low scores like occluded objects which gives low accuracy in the trajectory of the object. But according to the method of ByteTrack [47] ignoring the low scores will decrease the efficiency of the tracking model. ByteTrack associates every detec-tion of box instead of considering only the high score ones. ByteTrack solves this problem by using a motion model that manages a queue called tracklets to store objects being tracked and performs tracking and matching between bounding boxes with low score values

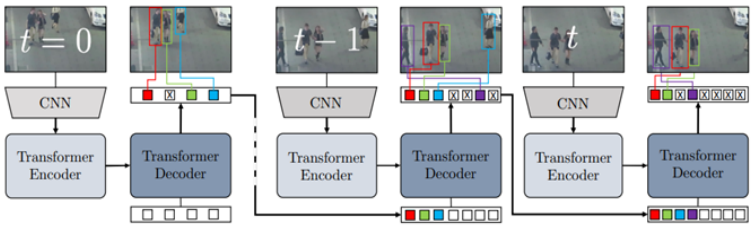


Figure 3.7: TrackFormer [30]

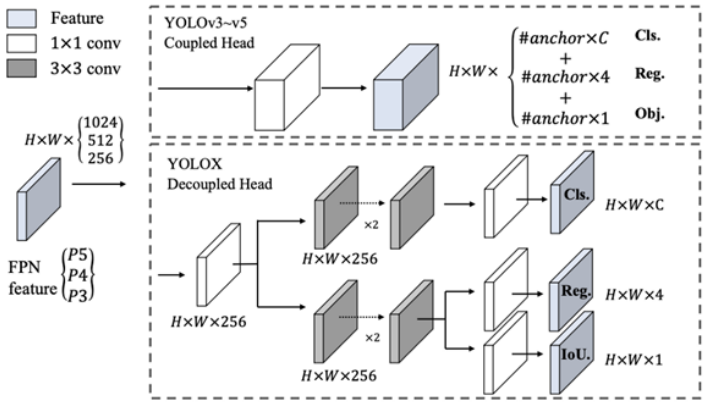


Figure 3.8: difference between YOLOv3 head and the YOLOX decoupled head [14]

The algorithm for ByteTrack is shown Figure 3.9. First of all, the low score detection and high score detection boxes will be separated by using the threshold detection value. The Kalman filter will be used to predict the new locations of current frame of each track then they are matched with high score detected bounding boxes using motion similarity. With motion similarity again, the score is computed by Interaction over Union **(IoU)**, which indicates the amount of overlap between objects.
Next the algorithm will perform the second association between the low score detection boxes and uses **(IoU)** for the Similarity in the second association because the low score detection boxes usually contain severe occlusion or motion blur and appearance.

To obtain the detection boxes ByteTrack used a high-performance detector based on YOLOX [14] which is trained on the MOT19 and MOT20 datasets. YOLOX is the latest anchor-free version of YO-LO models, pushing the limit in terms of speed and accuracy. The biggest modeling changes include the removal of box anchors (improves the portability of the model to edge devices) and the decoupling of the YOLO detection head into separate feature channels for box classification and box regression. Figure 3.8 shows the difference between the coupled head used in YOLOv3 to YOLOv5 and decoupled head model in YOLOX.

In coupled head the *regression* and *classification* are performed simultaneously which results to reduce the accuracy of the model. To improve this YOLOX is used decoupled

Algorithm 1: Pseudo-code of BYTE.

Input: A video sequence V; object detector Det; detection score threshold τ
Output: Tracks $\mathcal{T}$ of the video

Initialization: $\mathcal{T} \leftarrow \emptyset$
for *frame* f_k *in* V **do**
  /* Figure 2(a) */
  /* predict detection boxes & scores */
  $\mathcal{D}_k \leftarrow \text{Det}(f_k)$
  $\mathcal{D}_{high} \leftarrow \emptyset$
  $\mathcal{D}_{low} \leftarrow \emptyset$
  for d *in* $\mathcal{D}_k$ **do**
    if $d.score > \tau$ **then**
      $\mathcal{D}_{high} \leftarrow \mathcal{D}_{high} \cup \{d\}$
    end
    else
      $\mathcal{D}_{low} \leftarrow \mathcal{D}_{low} \cup \{d\}$
    end
  end
  /* predict new locations of tracks */
  for t *in* $\mathcal{T}$ **do**
    $t \leftarrow \text{KalmanFilter}(t)$
  end
  /* Figure 2(b) */
  /* first association */
  Associate $\mathcal{T}$ and $\mathcal{D}_{high}$ using Similarity#1
  $\mathcal{D}_{remain} \leftarrow$ remaining object boxes from $\mathcal{D}_{high}$
  $\mathcal{T}_{remain} \leftarrow$ remaining tracks from $\mathcal{T}$
  /* Figure 2(c) */
  /* second association */
  Associate $\mathcal{T}_{remain}$ and $\mathcal{D}_{low}$ using similarity#2
  $\mathcal{T}_{re-remain} \leftarrow$ remaining tracks from $\mathcal{T}_{remain}$
  /* delete unmatched tracks */
  $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{T}_{re-remain}$
  /* initialize new tracks */
  for d *in* $\mathcal{D}_{remain}$ **do**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{d\}$
  end
end
Return: $\mathcal{T}$

Figure 3.9: Algorithm for ByteTrack [47]

head for regression and classification. For each level of the FPN feature, first, a 1×1 conv layer is applied to reduce the feature channels to 256. Then two parallel branches are added with two 3×3 conv layers, for classification and localization tasks, respectively. Finally, IoU branch is added on the regression branch.

3.8 PatchTrack

Object appearance and object motion are used as part of the detection association strategy for this method. Object motion and object appearance are the most common used for information in multiple object tracking.
PatchTrack [5]is Transformer-based [41]joint-detection-and-tracking system that predicts tracks using patches of the current frame of interest. Kalman filter is used obtain track candidates in the current frame from existing tracks in the previous frame and crop the current frame using the bounding box of these candidates to get patches.

3.8.1 Transformer Architecture

The paper 'Attention Is All You Need' [41] introduces a novel architecture called Transformer. A Trans-former is a deep learning model that adopts the mechanism of attention, deferentially weighing the significance of each part of the input data. A transformer is built using an encoder and decoder and both are comprised of modules that can stack onto the top of each other multiple times. The architecture of Trans-former is explained in Figure 3.10
The transformer has the following basic components for performing the tasks: -

- An **Encoder** that encodes an input sequence into state representation vectors. The Encoder stack is composed of a stack of 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network.
- An **Attention** mechanism that enables the Transformer model to focus on the right aspects of the sequential input stream. This is used repeatedly within both the encoder and the decoder to help them contextualize the input data.
- A Decoder that decodes the state representation vector to generate the target output sequence

3.9 FairMOT

FairMOT [48] is a model for multi-object tracking which consists of two homogeneous branches to predict pixel-wise objectness scores and re-ID features. The achieved fairness between the tasks is used to achieve high levels of detection and tracking accuracy.
Most methods of MOT like [2] [44]treated the detection and re-ID feature differently, whereas the FairMOT treat equally both branches. The approach is developed by addressing the problems based on three factors (Failures from previous methods) and presenting an effective solution. The first factor is based on Anchors, in which Anchors

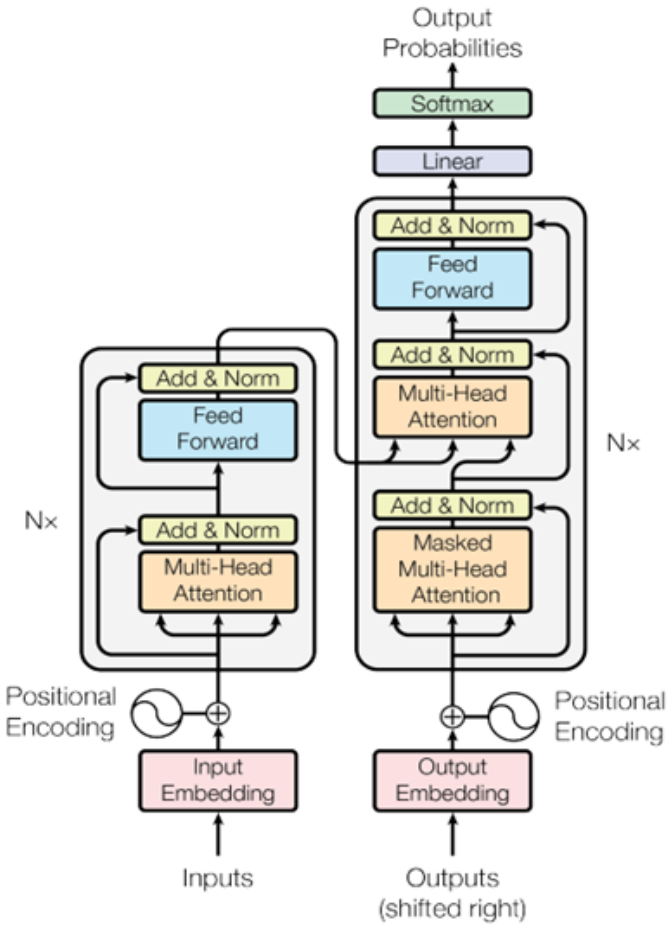


Figure 3.10: Transformer architecture [41]

are designed for detecting objects not suitable for extracting re-ID features. The second failure is caused by feature sharing, in which FairMOT considers the re-ID feature and detection features are totally different tasks. The third failure is dimension of re-ID features is usually as high 512 or 1024 which is much larger than the object detection. FairMOT suggests that the dimension size of re-ID features should be smaller for better achieving in accuracy and efficiency of tracking. The FairMOT network structure consists of the two homogeneous branches which is detection and re-ID features is shown in Figure 3.11

The detection branch is built on top of the CenterNet [50] .CenterNet is an anchorless object detection architecture based on the insight that box predictions can be sorted for relevance based on the location of their centers, rather than their overlap with the object. CenterNet infers a heatmap of the object's center coordinates, the offset of the center coordinates, and the object's size which is shown in Figure 3.11 .The heatmap head [48] is responsible for estimating the location of the object centers and the offset head is used to recover from the discretization error caused due to the down sampling of the input. After

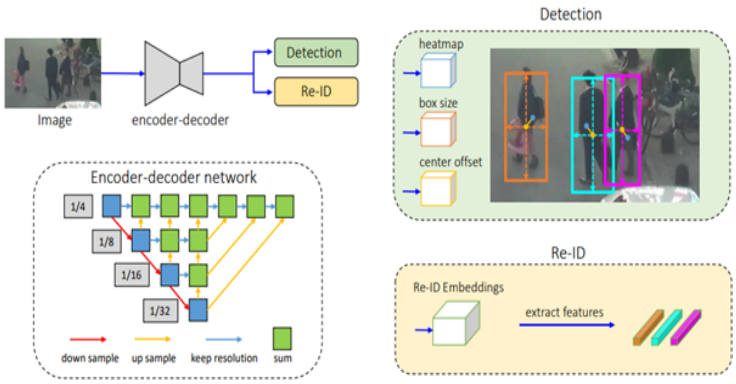


Figure 3.11: overview of FairMOT [48]

the prediction of the center points, these coordinates must map to a higher dimensional input image. The box size is used for the prediction of the dimensions of the boxes size width and height. FairMOT uses ResNet-34 as backbone for extracting features.

3.10 TraDeS

TraDeS (TRAck to Detect and Segment) [45] is an online joint detection and tracking model which integrates tracking cues into detection. Most online MOT methods [2] [44] work their detection and track-ing solely without connecting the tracking to the input of detection ,whereas TraDes introduces integration of tracking cues into detection and perform detection and tracking in efficient way.
The feature map in TraDes can represent either an object center or background region. TraDes incorporates the tracking into detection and designed re-ID learning scheme. The authors introduce two models *a cost volume-based association module(CVA)* and *a motion-guided feature warper (MFW) module.*

- **cost volume-based association (CVA)**: -the cost volume was applied for optical flow estimations and depth estimations in associating pixels between two frames. CVA helps for extracting re-ID embedding features and constructs cost volume for matching similarities between the embedding's frames. The cost volume will have the offset for tracking which are the potential object centers in both frames.
- **motion-guided feature warper (MFW)**: - by taking the tracking offsets from volume cost and uses them as motion cues to propagate object features from the previous frames to the current one [20] and finally uses to drive detection and segmentation.

Chapter 4

Detection and Tracking of model

As our approach for our MOT algorithm is based on the strategy tracking-by-detection, an object detector which is fully trained from our dataset MOT20 is employed to detect and classify peoples (" person") from other objects in terms of bounding box. Next, the program would perform data association by connecting the detection responses to corresponding tracklets. Finally, an output database file is generated, which includes details of the tracking information for each object such as object ID and positions at every frame. In this section we will discuss techniques and how our object detector and object tracking work in details.

4.1 YOLOv4

The original YOLO (You Only Look Once) was written by Joseph Redmon (now retired from computer vision research) in a custom framework called Darknet. [34] Darknet is a high-performance open source framework for implementing neural network written in C and CUDA. YOLO was the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network. YOLOv4 uses CSPDarknet53 as a Backbone, SPP (Spatial pyramid pooling) and PAN (Path Aggregation Network) as "Neck" and YOLOv3 as "Head" as shown in figure 4.1. [4]

4.1.1 Backbone

It's a deep neural network composed mainly of convolution layers. The main objective of the backbone is to extract the essential features, the selection of the backbone is a key step it will improve the performance of object detection. The authors initially considered CSPResNext50, CSPDarknet53 and EfficientNet-B3 as the backbone networks. Finally, after a lot of testing and experimental results they chose CSPDarknet53 CNN. [4] The CSPResNext50 and the CSPDarknet53 are both based on DenseNet. DenseNet was designed to connect layers in convolutional neural networks to alleviate the vanishing gradient problem (it is hard to backprop loss signals through a very deep network), to bolster feature propagation, encourage the network to reuse features, and reduce the

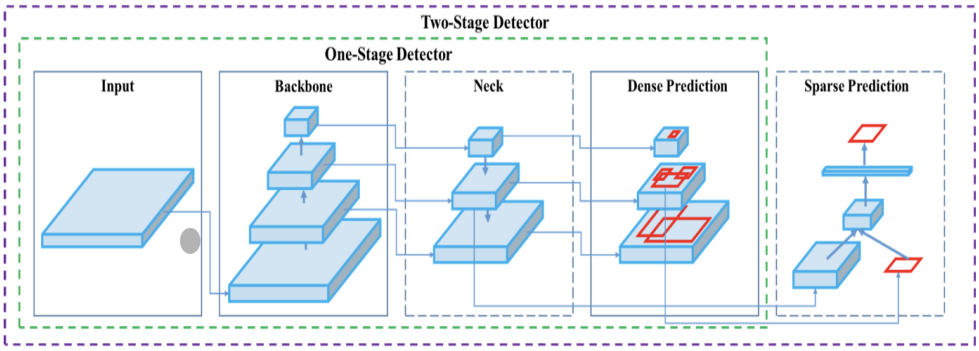


Figure 4.1: object detector [4]

number of network parameters. [42]

CSPDarknet53

YOLOv4 utilizes the CSP connections above with the Darknet-53 below as the backbone in feature extraction. CSPDarkNet53 consists of two blocks ,Convolutional Base Layer,Cross Stage Partial (CSP) Block.Cross Stage Partial strategy splits the feature map in the base layer into two parts and merges them with the help of Cross-stage hierarchy. The CSP block, which is stacked next to the Convolutional Base layer, divides the input into two halves, one half will be sent through the dense block, while the other half will be routed directly to the next step without any processing. [20]

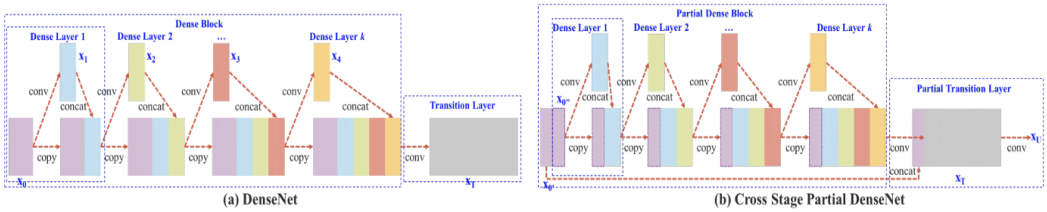


Figure 4.2: (a) DenseNet and (b) our proposed Cross Stage Partial DenseNet (CSPDenseNet) [20]

4.1.2 Neck

The essential role of the neck is to collect feature maps from different stages of backbone then mixes and combines them to prepare them for the next step. Usually, a neck is composed of several bottom-up paths and several top-down paths.

SPP (spatial pyramid pooling layer)

SPP applies a slightly different strategy in detecting objects of different scales. It replaces the last pooling layer (after the last convolutional layer) with a spatial pyramid pooling layer. Spatial Pyramid Pooling Layer will allow us to generate fixed size features whatever the size of our feature maps. To generate a fixed size, it will use pooling layers like Max Pooling and generate different representations of our feature maps. [42]

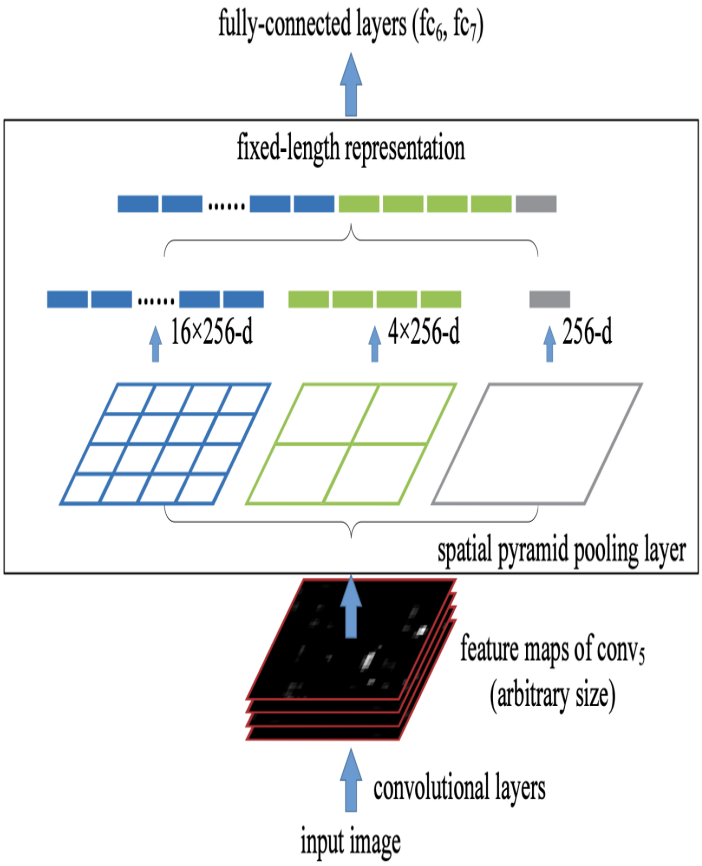


Figure 4.3: SPP [42]

Path Aggregation Network (PAN)

YOLOv4 uses a modified path aggregation network, mainly as a design improvement in-order to make it more suitable for training on a single GPU. The main role of PANet is to improve the process of instance segmentation by keeping the spatial information which in turn helps in proper localization of pixels for mask prediction. *Bottom-up Path Augmentation*, Adaptive Feature Pooling & *Fully Connected Fusion* are the important properties that make them so accurate for mask prediction. [17]In the original implementation of

PaNet, the current layer and information from a previous layer is added together to form a new vector. In the YoloV4 implementation, a modified version is used where the new vector is created by concatenating the input and the vector from a previous layer.

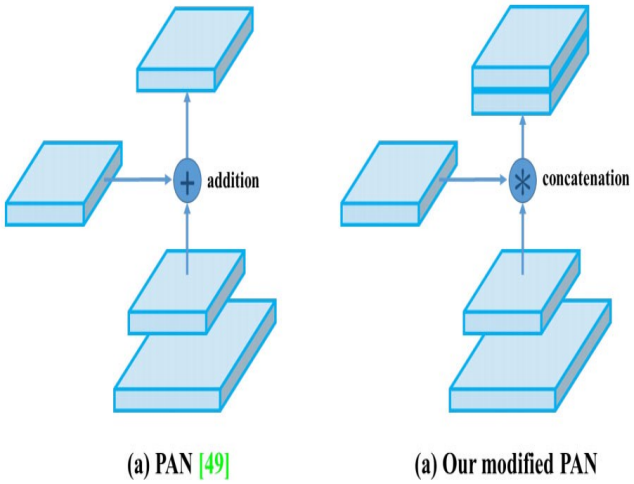


Figure 4.4: (SPP [4]

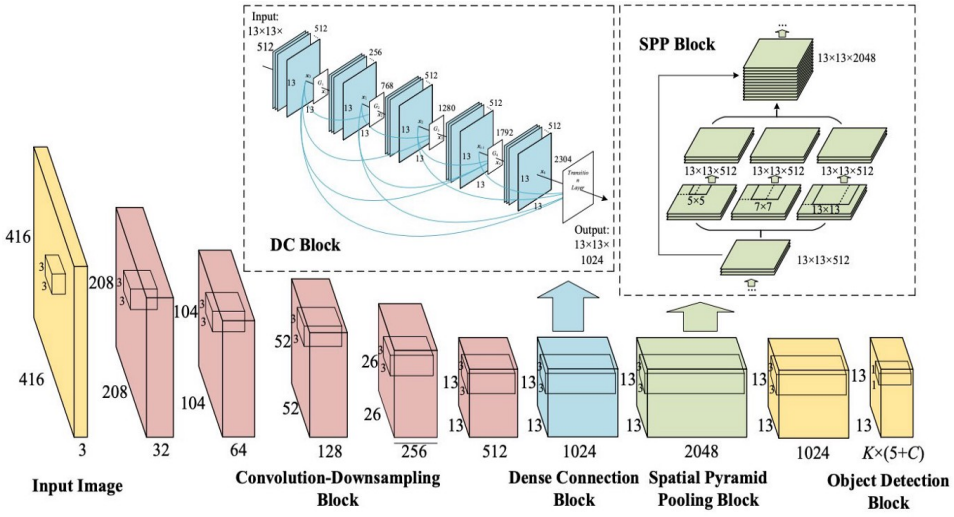


Figure 4.5: CSPDarknet53 [20]

4.1.3 Head (detector)

The role of the head in the case of a one stage detector is to perform dense prediction. The dense prediction is the final prediction which is composed of a vector containing the

coordinates of the predicted bounding box (center, height, width), the confidence score of the prediction and the label. YOLOv4 deploys YOLOv3 as head for detection with the anchor-based detection steps and three levels of detection granularity. Two new terms were introduced by the authors called Bag of Freebies (BoF) & Bag of Specials (BoS). [4]

Bag of Freebies (BoF): They improve performance and accuracy of the network without adding to the inference time. The Bag of Freebies for backbone include CutMix and Mosaic data augmentation, DropBlock regularization and Class label smoothing. And the Bag of Freebies for detector are CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, eliminate grid sensitivity, using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyper-parameters, Random training shapes. [4]
Bag of Specials (BoS): These strategies add marginal increases to inference time but significantly increase performance. BoS for backbone are Mish activation, Cross-stage partial connections (CSP), Multiinput weighted residual connections (MiWRC). [4]BoS for detectors are Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS.

The network of five laminated convolution-pooling blocks, a dense connection block with four dense units, a spatial pyramid pooling block with three maxpooling layers and a multi-scale object detection block is shown in Figure 4.5.

4.2 DeepSORT

In object detection, we detect an object in frames and put bounding boxes on the detected object (person in our case) and it processes each frame independently and identifies numerous objects in that particular frame. Note that the job of detector ends here. Now after our YOLOv4 detector detects the object an abject tracker DeepSORT needs to track across the subsequent frames (with the help of a unique ID).

As it explained in section 3.5.1 DeepSORT is the extension of the SORT algorithm. DeepSort is an improvement based on Sort target tracking. It introduces a deep learning model to extract the appearance features of the target for nearest neighbor matching in the process of real-time target tracking. The SORT algorithm uses a simple Kalman filter to deal with the correlation of frame-by-frame data and uses the Hungarian algorithm to measure the correlation. This algorithm has achieved good performance at a high frame rate. However, since SORT ignores the appearance feature of the detected target, it will be accurate only when the uncertainty of target state estimation is low. In addition, in order to improve the tracking efficiency, SORT deletes the target that has not been matched in a continuous frame, but this causes the problem of ID switching, that is, the ID assigned to the target is easy to change constantly. Therefore, DeepSORT adds appearance information and borrows the ReID model to extract appearance features, reducing the number of ID switches by 45%. [46] DeepSORT also turns SORT's matching mechanism based on IoU cost matrix into a mechanism of Matching Cascade and IoU matching. Specifically, the core idea of Matching Cascade is to give greater priority to

track matching to the targets that appear more frequently in the long-term occluded targets. The general flow of MOT is generalized as figure 4.6.

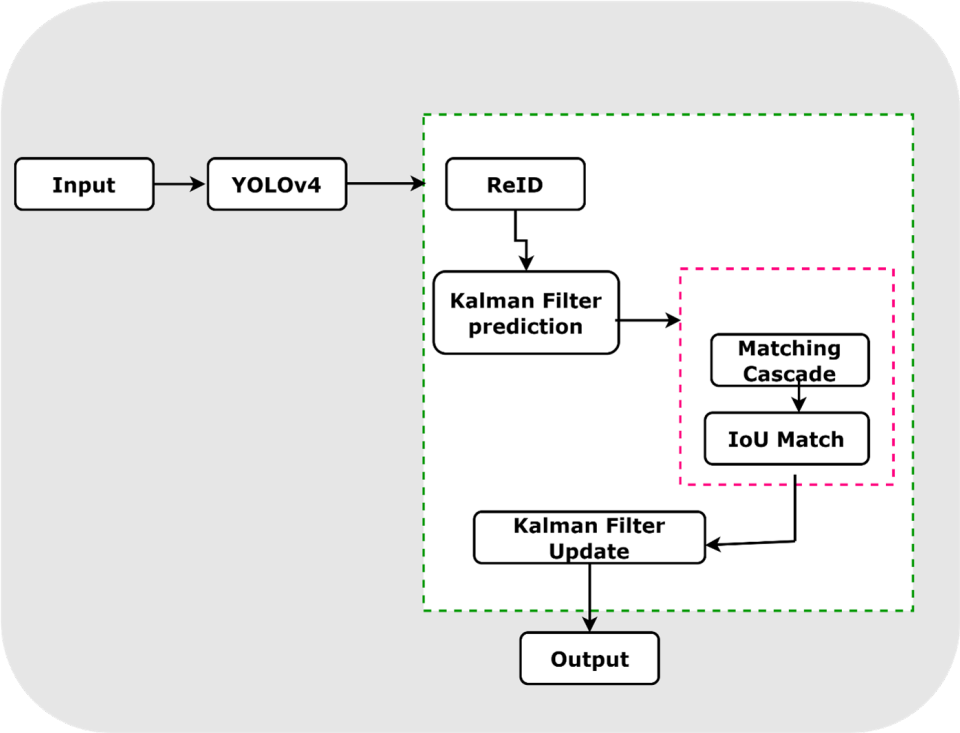


Figure 4.6: Flow of MOT

4.2.1 Kalman Filter

The kalman filter has role on determining the prediction the next frames during tracking of objects.It is mathematical equations that provides the effect solution to estimate the state of the process by minimizing the mean square error. [22] [37] The filter is very powerful in several aspects: it supports estimates of the past and present states, same future.

In case of object tracking system [44] [12] [37] the kalman filter grabs information during the movement of the person and by using this information it will predict the state of the moving object for the next frame.For this the measurement vector (position in x, in y, width and height of the object) uses as an input The state vector is defined by the initial position, the width, and the length of the target frame in the 8-dimensional state space [12] (u, v, γ, h, $\dot{x}$, $\dot{y}$, $\dot{\gamma}$, $\dot{h}$), frame center (u, v), aspect ratio γ ,Height h and their respective velocities in the image coordinate system. The uniform motion model is still used here, and (u, v, γ, h) As a direct observation of the object state.

$$X = [u, v, s, r, \dot{v}, \dot{u}, \dot{s}]^T \tag{4.1}$$

4.2.2 Process to estimate

The mean value and covariance are the two states of the target that need to be estimated, the mean contains the position and speed information of the target 8-dimensional vector. whereas the covariance indicates the uncertainty of the estimated state.

In the prediction phase Kalman filter estimates the state X_k of discrete process by using the last moment$k-1$ a posterior estimate passes to the state transition matrix (F) and gets current moment a prior estimation state (k).

$$X_k = F \times X_{k-1} \tag{4.2}$$

$$\begin{bmatrix} u \\ v \\ \gamma \\ h \\ \dot{x} \\ \dot{y} \\ \dot{\gamma} \\ \dot{h} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ \gamma \\ h \\ \dot{x} \\ \dot{y} \\ \dot{\gamma} \\ \dot{h} \end{bmatrix}_{k-1} \tag{4.3}$$

The next step is to calculate the prior estimated covariance of the current time k using the posterior estimated covariance of the previous time $k-1$.

$$p_k^- = FP_{k-1}F^T + Q \tag{4.4}$$

In update phase of kalman filter a prior estimated mean x and covariance p at the current time are estimated by a posterior estimated mean and variance at the previous time.

1. During the update phase of the Kalman filter the Kalman gain matrix is calculated by using a prior estimation covariance matrix P , observation matrix H and measurement state covariance matrix R.
$$K = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{4.5}$$
2. Then project the a prior estimated value x of the Kalman filter into the measurement space through the observation matrix H, and calculate the residual y with the measured value z.
$$y = Z_k - HP_k^- \tag{4.6}$$
which is shown as in equation 4.7

$$\begin{bmatrix} u \\ v \\ \gamma \\ h \end{bmatrix} = \begin{bmatrix} u \\ v \\ \gamma \\ h \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ \gamma \\ h \\ \dot{x} \\ \dot{y} \\ \dot{\gamma} \\ \dot{h} \end{bmatrix} \tag{4.7}$$

3. the predicted value and measured value of Kalman filter are fused according to the proportion of Kalman gain to obtain the a posterior estimated value x.

$$x_k = x_k^- + K_y \tag{4.8}$$

4. calculating a posterior estimation covariance of Kalman filter .

$$p_k = (I - KH)P_k^- \tag{4.9}$$

With K presents the gain of the Kalman filter at the moment k, X_k the state estimated and predicted at the moment k, P_k is the prediction covariance matrix at time k.These three equations (4.5), (4.8) and (4.9) present the output parameters of the Kalman filter. [37]

4.2.3 Hungarian Algorithm

A conventional way to solve the association between the predicted Kalman states and newly arrived measurements is to build an assignment problem that can be solved using the Hungarian algorithm.During tracking to perform the correlation matching between detector and tracking track on the prior estimate of the object target position and assign the same identity ID to the object with the same target. [12] Two indicators to achieve correlation are:

1. **Mahalanobis distance matching**:Mahalanobis distance [6], also known as covariance distance, is an effective method to calculate the similarity between two unknown sample sets, which measures the matching degree of prediction and detection. In order to integrate the motion information of the object, the (square) Mahalanobis distance between the predicted state and the measured state is used.

$$d_{(i,j)}^{(1)} = (d_j - y_i)^T S_i^{-1} (d_j - y_i) \tag{4.10}$$

   Where, d and y represent measurement distribution and prediction distribution respectively; S is the covariance matrix between two distributions. It is a real symmetric positive definite matrix.

2. **Appearance feature matching**:when the uncertainty of object motion state is relatively low, using Mahalanobis distance is indeed a good choice [6]. Because the Kalman filter uses the uniform motion model, it can only provide a relatively rough linear estimation of the moving position of the object. When the object accelerates or decelerates suddenly, the distance between the prediction frame and the detection frame of the tracker will become relatively far. At this time, only using the Mahalanobis distance will become very inaccurate. First, d_j is detected based on each bounding box, and the appearance descriptor r_i is calculated. Secondly, according to each track k, a library of appearance descriptors is established, and the feature vectors successfully associated with 100 frames before and after are saved.Finally,the minimum cosine distance between the i^{th} and track $r_k^{(i)}$ and j^{th}

detector r_j^T were calculated.The calculation of appearance features is shown as follows.

$$d^{(2)}(i,j) = min\left\{1 - r_j^T r_k^{(i)} \middle| r_k^{(i)} \in R_i\right\} \tag{4.11}$$

The appearance matching and mathalanobis distance matching are fused as the final metric of association matching as shown in equation 4.12

$$c_{i,j} = \lambda d_{(i,j)}^{(1)} + (1-\lambda) d_{(i,j)}^{(2)} \tag{4.12}$$

Cosine distance is used to measure the apparent features of Kalman prediction and the corresponding apparent features of detector, so as to predict ID more accurately. Only using motion information for matching in SORT will lead to serious ID Switch.The introduction of appearance model + cascade matching can alleviate this problem. [6]

4.2.4 Cascade matching

When the object is occluded for a long time, Kalman filter cannot accurately predict the target state. Therefore, the probability quality is dispersed in the state space, and the peak of observation likelihood is reduced. However, when two trajectories compete for the same detection result, the Mahalanobis distance will bring greater uncertainty. Because it effectively reduces the distance between the standard deviation of any detection and the mean value of the projected trajectory and may lead to increased trajectory fragmentation and unstable trajectory. Therefore, Deep Sort introduces cascade matching to give priority to targets that appear more frequently.

4.2.5 IOU matching

This stage occurs after cascade matching. The matched tracker objects are unconfirmed tracks and tracks with age 1 in the previous round of cascade matching failure This helps to reduce the probability of appearance changes caused by sudden occlusion in a frame.

Chapter 5

Methods and Result of Experiments

This chapter aims on the methods that we used during the detection and tracking of the object and provide different conducted methods and experiments on training datasets, implementations, Evaluations of metrics and discussion. To begin with we introduced first the dataset section and how we organize the dataset for training to our detection system. We will discuss how our object detect YOLOv4 [34] can be trained with dataset multiple objects tracking benchmark (MOT20). we will also cover the conventional metrics that are used to rank trackers. The tracking of persons from crowded area using the DeepSORT algorithms based on the result of our detection system will be discussed under this section. In summary the methods that we use to achieve our output is explained in the following figure

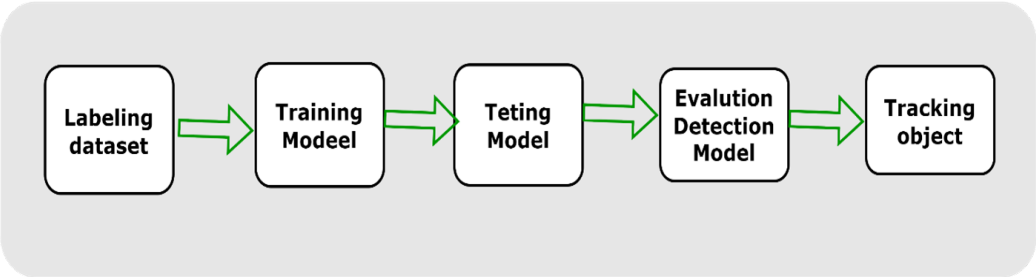


Figure 5.1: methodology of work

5.1 MOT Challenge

MOTChallenge provides carefully annotated datasets and clear metrics to evaluate the performance of tracking algorithms and pedestrian detectors. [10] we choose this MOTChallenge as our dataset to our training model because all sequences have been carefully

selected and annotated and this benchmark addresses of crowded scenes in which very density around 246 pedestrians per frame.Table 5.1 and 5.2 shows the summary of compiled a total of 8 sequences which is 4 for training and 4 for testing.

Table 5.1: overview of training sequences MOT20 [10]

Training sequences							
Name	FPS	Resolution	Length	Tracks	Boxes	Density	Description
MOT20-01	25	1920x1080	429	74	19870	46.32	indoor
MOT20-02	25	1920x1080	2,782	270	154,742	55.62	indoor
MOT20-03	25	1173x880	2,405	702	313,658	130.42	outdoor,night
MOT20-05	25	1654x1080	3,315	1,169	646,344	194.98	outdoor,night
Total training			**8,931**	**2,215**	**1,134,614**	**127.04**	

Table 5.2: overview of testing sequences MOT20 [10]

Testing sequences							
Name	FPS	Resolution	Length	Tracks	Boxes	Density	Description
MOT20-04	25	1545x1080	2,080	669	274,084	131.77	outdoor,night
MOT20-06	25	1920x734	1,008	271	132,757	131.70	outdoor,night
MOT20-07	25	1920x1080	585	111	33,101	56.58	indoor
MOT20-08	25	1920x734	806	191	77,484	96.13	outdoor,day
Total training			**4,479**	**1,242**	**517,426**	**115.52**	

The crowded scenes in figure 5.2 are sample images taken from the MOT20 training sequences. We trained a YOLOv4 [34] and YOLOv4-tiny [21] models with backbone of Darknet-53 on the MOT20 training sequences. In order to learn from the input dataset our model should have same data format of with MOT20 training sequences. Since labeling highlights data features or properties, characteristics, or classifications that can be analyzed for patterns that help prediction of the target. having these we use a MATLAB software for annotating our data from MOT20 training sequences. We use the bounding boxes as our annotation of method. All 8931 training images were converted into YOLOv4 data format annotation. The ground truth file of MOT20 benchmark has the file format as shown below.

<frame>, <id>, <bbleft>, <bbtop>, <width>, <height>, <conf>, <x>, <y>, <z>

The format of the annotations is described in detail in the table. 5.3 Hence, in order to perform the training of our model (YOLOv4) we changed the MOT data format to YOLO format annotation.The YOLO format consists of object class, object coordinates, height, and width as the following format

<object-class>, <x>, <y>, <width>, <height>

In which

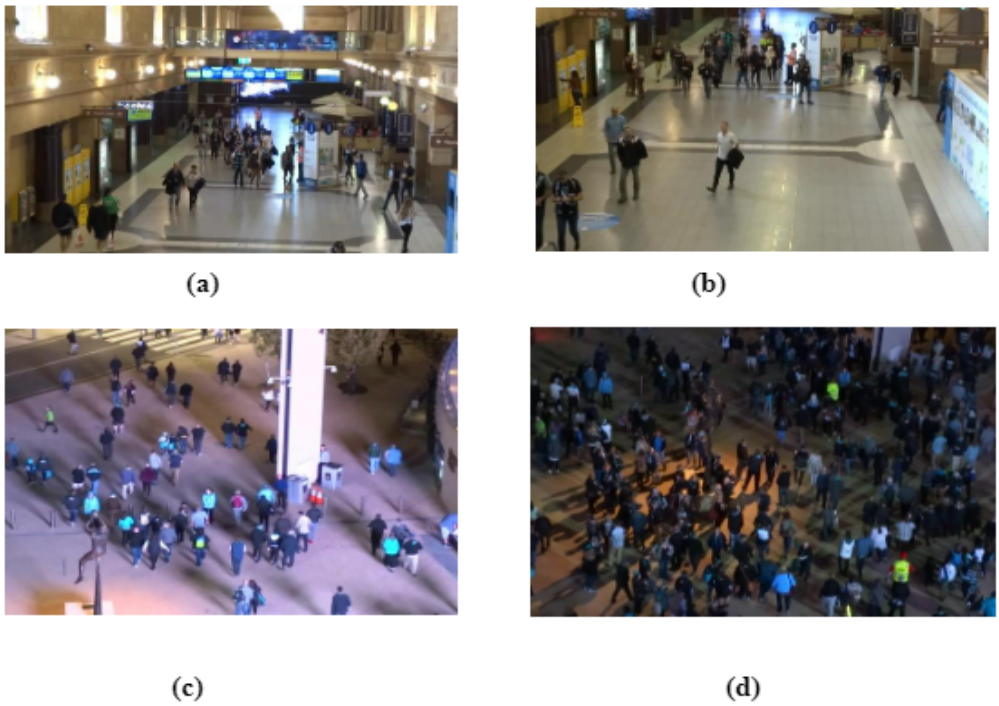


Figure 5.2: sample images from: a) MOT20-01 b) MOT20-02 c) MOT20-03 d) MOT20-05 [10]

Table 5.3: Data fomat annotation GT files

position	Name	description
1	frame	frame number
2	id	Identity number
3	bbleft	Bounding box left
4	bbtop	Bounding box right
5	width	Bounding box width
6	height	Bounding box height
7	conf	Confidence score
8	x	class
9	y	visibility

- *<object-class>*:-integer number of object from 0 to (classes-1)
- *<width>* *<height>*:- float values relative to width and height of image, it can be equal from (0.0 to 1.0].
- *<x>* *<y>*:-are center of rectangle (are not top-left corner).

Figure 5.3 shows the annotation three sample files taken from MOT20-01 training sequences which is arranged to the YOLO format. In this format a.*txt* file is generated

```
0 0.140104 0.876852 0.072917 0.248148
0 0.213021 0.877315 0.064583 0.247222
0 0.234896 0.850463 0.054167 0.239815
```

Figure 5.3: sample annotation of file in YOLO data format

with the same name for each image file in the same directory. Each *.txt* contains the annotation for the corresponding image files. the output scenes in figure 5.4 are some of the output images after annotated with bounding boxes.

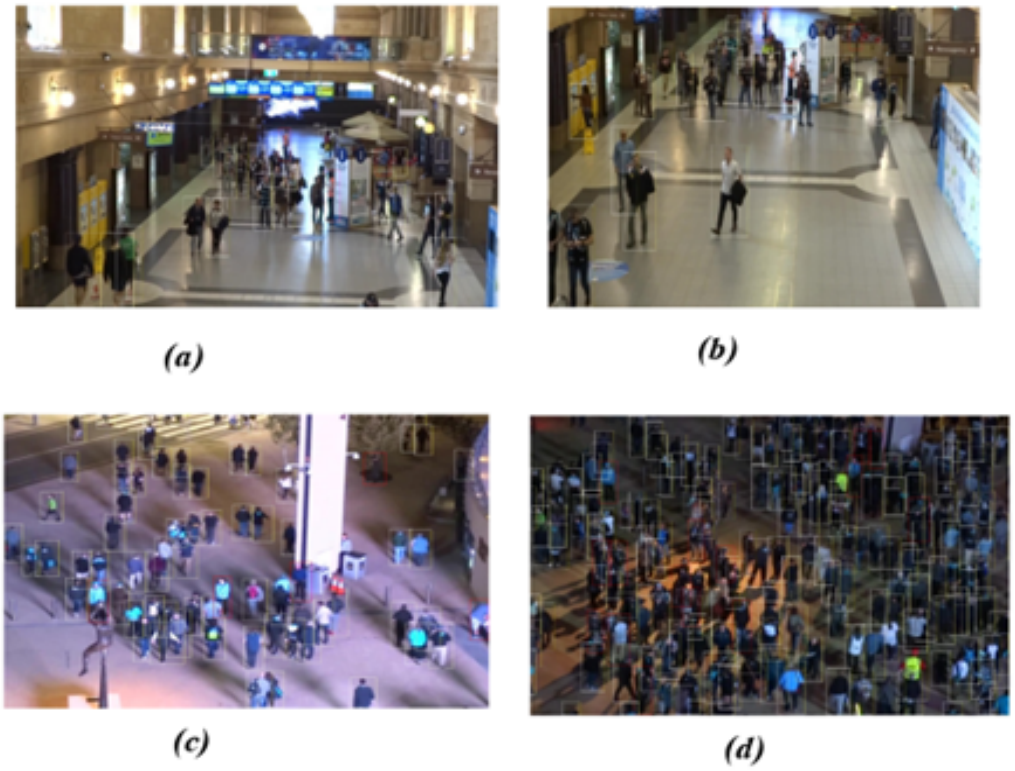


Figure 5.4: annotation output sample from a) MOT20-01 b) MOT20-02 c) MOT20-03 d) MOT20-05

5.2 Training of the model

After labeling our dataset MOT20 [10] we trained our object detection system YOLOv4. For training, we use google Colab GPU a NVIDIA-SMI, CUDA *Version: 11.2*. The object detector is composed of two parts a backbone which is pre-trained on ImageNet and a head for prediction a class and bounding boxes of the object. The YOLOv4 uses Darknet as backbone. The original version of Darknet was written by Joseph Redmon, but his "pjreddie" repo was abandoned many years ago and should not be used. It hasn't received any significant updates since 2016.Instead we use the current version of the darknet modified by AlexAB implementation [3] [4] To perform for the training process the procedures in figure 5.5 were used during the training of the YOLOv4 and YOLOV4-tiny model. Our detection algorithm is based on YOLOv4 and YOLOv4-tiny

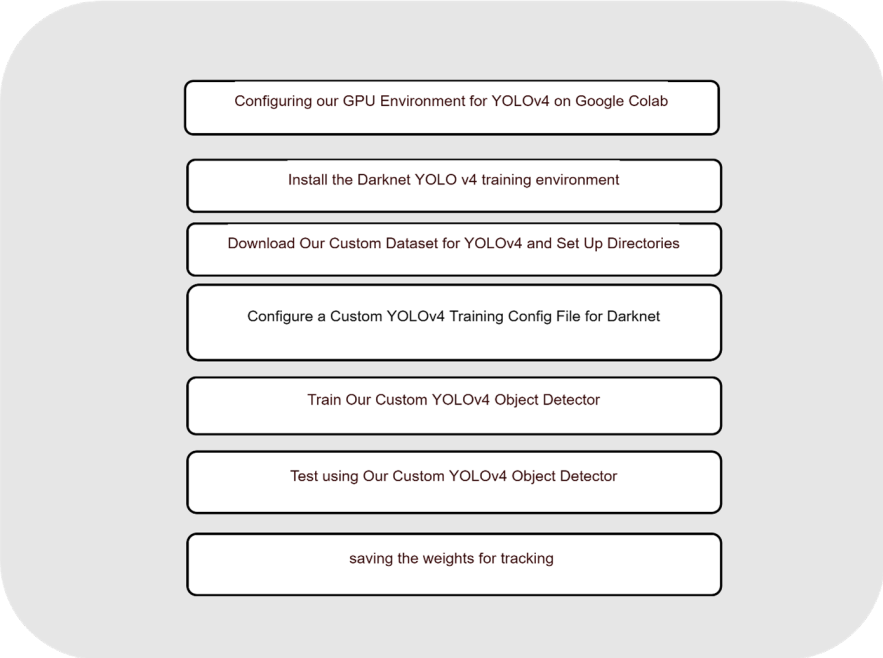


Figure 5.5: Flow of training the network model

[21]. YOLOV4-tiny is a compressed version of YOLOV4 [34]. Its main objective is to make the network structure simpler and reduce parameters. Then by cloning from the original repo of darknet, pretrained weights for YOLOv4 and YOLOv4-tiny and then by importing our dataset the training will be performed. To train a custom network there are several factors that determines how much memory is needed for GPU to train the network. the following configuration was made in our custom configuration file of the training. In our case it is not possible to feed all the training data into an algorithm in one pass. This is due to the size of the dataset and memory limitations of the compute instance used for training. There are some terminologies required to better understand how data is best broken into smaller pieces.

- **Batch size:**the batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. Our batch size was arranged to be 64. Which is 64 images will be loaded for one iteration.The number of iterations to complete one epoch was 6000 in another word 6000 batches were needed to complete the training for one epoch.
- **Network size:** the size of the network was chosen width=416 height=416.
- **Class:** since we have only one class of object "person" the number of filters is 18. we change the filters (255) to filter = (classes +5) x3 in the 3 convolutional layers before each yolo layer.
- **Subdivision:**Split batch into 32 mini-batches so 64/32 = 2 images per mini-batch and these 2 images are sent for processing. This process will be performed 32 times until the batch is completed and a new iteration will start with 64 new images. Since the entire dataset (**one epoch**) is too big to feed to the to the neural network at once we divide it in several smaller batches.

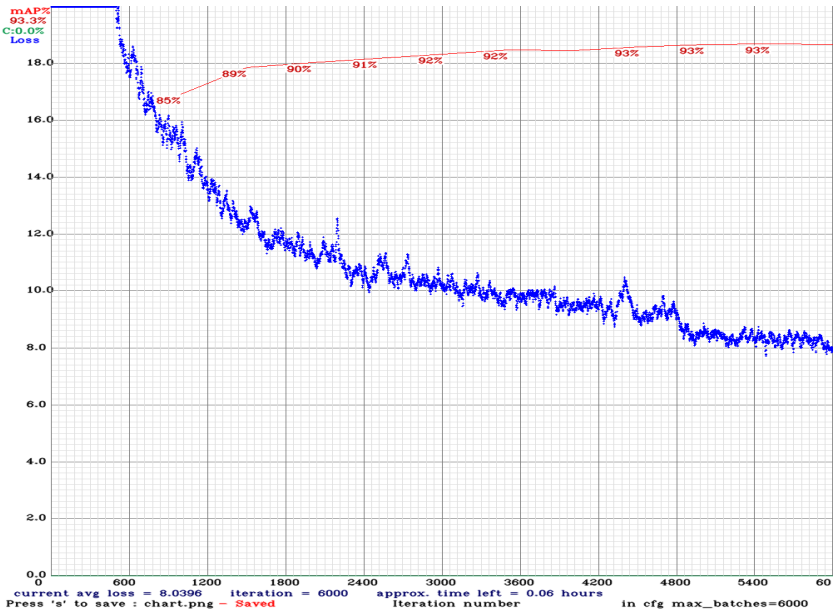


Figure 5.6: mAP and the average loss of YOLOv4-tiny

The primary difference between YOLOv4 tiny and YOLOv4 [34] is that the network size is dramatically reduced. The number of convolutional layers in the darknet backbone are compressed. The number of YOLO layers are two instead of three and there are fewer anchor boxes for prediction. In the case of configuring the YOLOv4-tiny configuration file, the parameters are almost the same with YOLOv4 except for the batches were split into 16 mini-bathes which is 4 images per mini-batch and these 4 images will pass to the process. This will speed up the training process as this loads more images per iteration.

The YOLOv4-tiny has only two YOLO heads as opposed to three in YOLOv4 and it has been trained from 29 pre-trained convolutional layers as opposed to YOLOv4 which has been trained from 137 pre-trained convolutional layers.

We trained the network for 6000 iterations for both YOLOv4 and YOLOv4 tiny on MOT20 training sequences. The training process took around 19' hours for the network YOLOv4 and around 5 hours for the YOLOv4-tiny. To compare the performance of the YOLOv4 and YOLOv4-tiny network we trained and tested both models on the object detection dataset(MOT20) [10] and evaluated for performance metrics. The comparative results for performance metrics achieved by YOLOv4 and YOLOv4- tiny [21] networks are described by the metrics evaluations.

When the training of YOLOv4 tiny model is performed a chart which shows the average loss and mAP values were generated, the chart also displays the average loss and the number of iterations of the dataset from 0 to 6000 which is shown in the figure 5.6.To generate a chart, the training should not be interrupted till the end of iteration. whereas this was not possible during our training of YOLOv4 [34] model as we could not train with out interrupting because of the availability of GPU in google Colab is limited in hours. The chart in figure 5.7 only shows the coordinates of the mAP and iteration which it was not able to display the decrease of average loss.

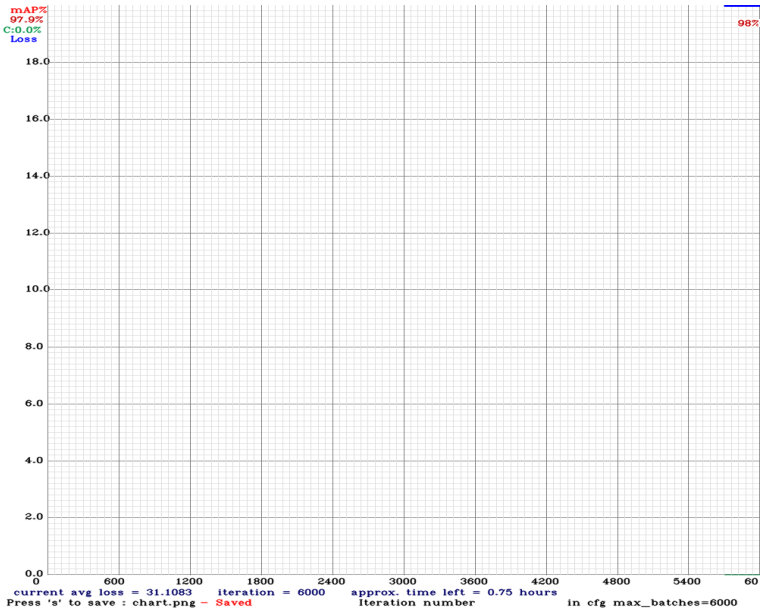


Figure 5.7: mAP and the average loss of YOLOv4

5.3 Evaluation metric

In order to check the performance of our object detect there are different commonly used evaluation metrics. Detection task metrics focus on the precision of the bounding

box, how confident are the detections, and how many objects it misses or wrong detections. The evaluations were made for both networks YOLOv4 and YOLOv4-tiny and the following metrics were evaluated for the detections of person.

5.3.1 Intersection Over Union (IoU)

This metrics helps for comparing the ground truth bounding boxes versus the predict bounding boxes.The model will predict multiple bounding boxes for each corresponding detected objects using the threshold value.The threshold value in our training was 50%, which means that if the IoU between the ground truth bounding boxes and the predicted boxes is 0.5, then we conclude that there is an object inside the box. The average of IoU for YOLOv4 [34] and YOLOv4-tiny was computed 76.41% and 70.47% respectively.The comparative output of metrics values for AP,precision and the average of IoU for both YOLOv4 and YOLOv4-tiny is described in bar graph as shown in figure 5.8.

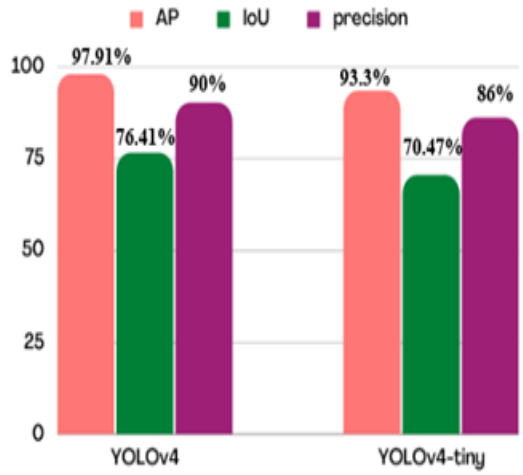


Figure 5.8: comparing of precision ,IoU and AP of the models

5.3.2 True Positive (TP) ,False Positive (FP) and False Negative (FN)

These are the most common metrics quantifying the performance of object detections. TP measures detection with IoU greater than or equal to the threshold while FP detection with IoU less than threshold. FN means the misses of the prediction with respect to the ground truth. Either metric is counted when the IoU is less than 0.5.The performance of the object detection based on those metrics is described in the table 5.4

Table 5.4: results for TP,FP and FN

Model	TP	FP	FN
YOLOv4	90576	10052	2661
YOLOv4-tiny	85515	13802	7661

5.3.3 Precision (Precision), Recall (Recall and F1 score)

Precision is defined as the number of true positives divided by the sum of true positives and false positives.

$$precision = \frac{TP}{TP + FP} \tag{5.1}$$

Recall is defined as the number of true positives divided by the sum of true positives and false negatives (note that the sum is just the number of ground-truths, so there's no need to count the number of false negatives).

$$Recall = \frac{TP}{number\,of\,ground - truth\,bounding\,boxes} \tag{5.2}$$

F1 score: is a metric that combines recall and precision into a single score by calculating the harmonic mean of precision and recall.

$$F1score = \frac{2 * TP}{2 * TP + FP + FN} \tag{5.3}$$

Table 5.5 shows the computed of the precision , recall and precision the network models that was achieved during the training of crowded persons.

Table 5.5: results for precision,recall and F1-score

Model	Precision	Recall	F1-Score
YOLOv4	0.90	0.97	0.93
YOLOv4-tiny	0.86	0.92	0.89

5.3.4 Average Precision (AP)

Average precision computes the average precision value for recall value over 0 to 1. Different datasets require different AP criteria.The average precision for YOLOv4 and YOLOv4-tiny model were computed to be 97.91% and 93.30% respectively.

5.3.5 Mean Average Precision(mAP)

Mean average precision is an extension of Average precision. In Average precision, we only calculate individual objects but in mAP, it gives the precision for the entire model. To find the percentage correct predictions in the model we are using mAP. We use mAP

iou=0.5 which represents the model has used 0.5 threshold value to remove unnecessary bounding boxes, it is the standard threshold value for most of the models. The result of mAP for both model yolov4 and YOLOv4-tiny is shown in table .The result is same with average precision because our class of object is only one "person".

Table 5.6: results for mAP

Model	**mAP(%)**
YOLOv4	97.91
YOLOv4-tiny	93.3

The average loss was also decreasing during training of our model, the average loss for yolov4 till the last iteration reach was 29.58 which is to mean the mAP after this iteration remains almost constant. While for the yolov4-tiny was 8.9. Summarily, YOLOv4-tiny is faster in training and detection than YOLOv4. For real-time object detection, YOLOv4-tiny is the better option when compared with YOLOv4 as faster inference time. However, YOLOv4 have high accuracy of detection than YOLOV4-tiny.

5.4 Results

5.4.1 Detection output

Here are some detection outputs found from both YOLOv4 and YOLOV4-tiny [21] model. The images are sample taken from test MOT20 sequences.

Figure 5.9: MOT20-04/000001,output YOLOv4

Figure 5.10: MOT20-04/000001,output YOLOv4-tiny

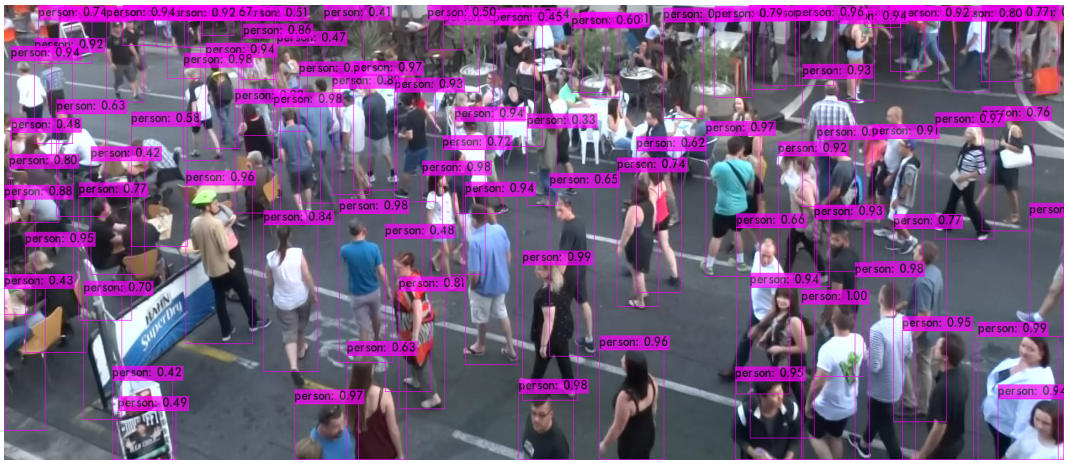
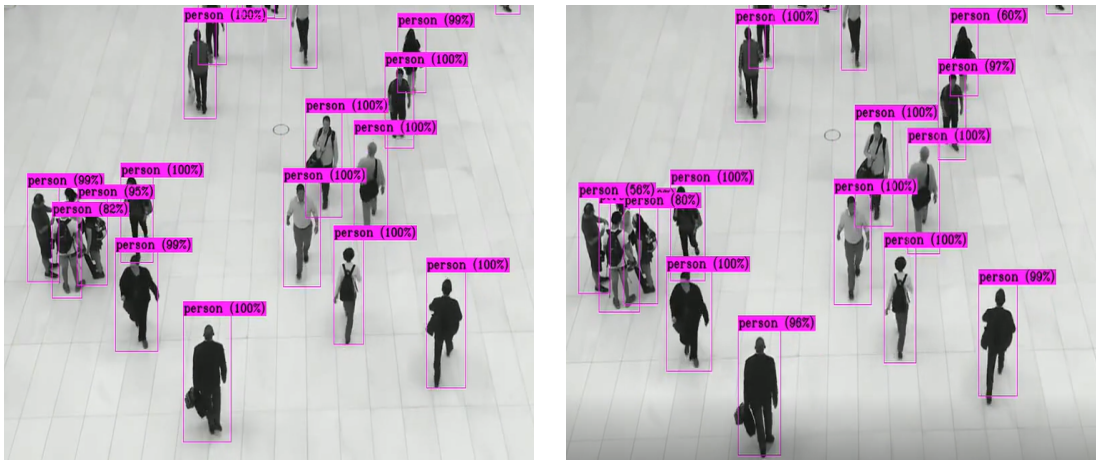


Figure 5.11: MOT20-06/000002,output YOLOv4

Figure 5.12: MOT20-06/000002,output YOLOv4-tiny

The detection experiment is also tested on sample video ,which is video from crowded area ,the model achieves to detect all the moving persons in that area. figure 5.13a shows an output from yolov4 and figure 5.13b is from yolov4-tiny [21].we can see that the output from YOLOv4 is more accurate than YOLOv4-tiny.



(a) output from yolov4 **(b)** output from yolov4-tiny

Figure 5.13: detection output from video

5.4.2 Tracking output

Unfortunately our tracking is based the yolov4 only, as the yolov4-tiny [21] detection was less accurate it was not able to get the result of tracking based on the yolov4-tiny. but the tracking outputs based on YOLOv4 detection which is tested with the same video we used for detection in section 5.4.1 .The result for tracking the objects(in our case "person") was done using the DeepSORT algorithm in which best detected object will make our our tracker to be good and assign ID for each person as shown in figure 5.14.

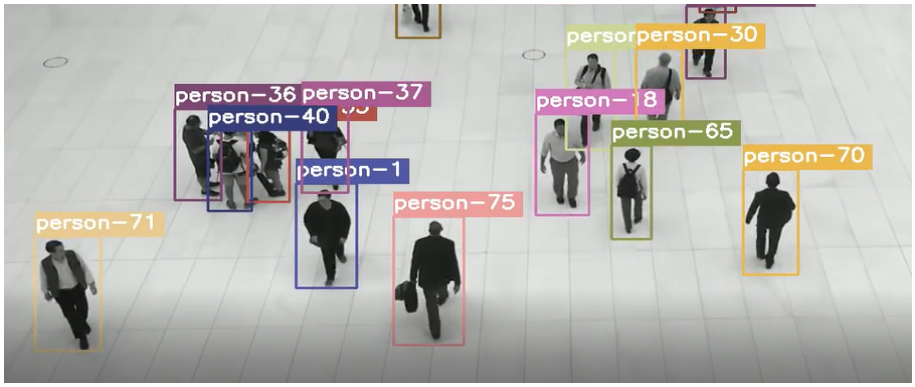


Figure 5.14: output of tracker

(a) previous frame **(b)** next frame

Figure 5.15: ID switching of persons

But in figure 5.15 we can see that there is limitation of switching of ID of the persons.in the first figure 5.15a the ID assigned to the object was *57* but in next frame in figure 5.15b the ID is changed to be *65*.In which this will affect on decreasing the quality measurements of MOT metrics MOTA ,since this dependes in swithcing of ID of the objects.

Chapter 6

Conclusion

We presented in this thesis a multiple object tracking of peoples in crowded area based on DBT (detection-based tracking). We choose YOLOv4 as our detector which will feed its output to our tracking algorithm which is DeepSORT.

To apply MOT application on the required target of objects, we firstly trained the YOLOv4 model on the MOT20 dataset. In the second we retrained into another model YOLOv4-tiny to analyze the influence of MOT by varying of detector. YOLOV4-tiny is compressed version of YOLOv4 with less layer network than of YOLOv4.We find that the YOLOv4 is more accurate than YOLOv4, but less in speed. To be more precise the model type YOLOv4-tiny is better suited for real-time video because of its time performance in detecting objects while model YOLOv4 is more suitable in cases where higher levels of accuracy are required.

Due to its suitability for real-time tracking and robustness, Deep SORT was selected as the tracking algorithm in this thesis for tracking of peoples in real time. The DeepSORT is a tracking algorithm which tracks objects not only based on the velocity and motion of the object but also the appearance of the object. The Kalman filter is used to predict the motion state of targets in next frame. And Hungarian algorithm is used to associate bounding boxes.

In the experiment we calculated the detection metric evaluation for the standard results. We use mAP (@Iou=0.5) which represents the model has used 0.5 threshold value to remove unnecessary bounding boxes. The mAP for YOLOv4 was achieved 97.91% where as YOLOv4 was 93.3% which shows less accurate than of YOLOv4.

In the future application we will recommend to improve the performance of real time people detection and tracking in crowded using the most recent version of YOLO algorithm and to get better MOT metrics.

Bibliography

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[3] Alexey Bochkovskiy. Yolov4 - neural networks for object detection (windows and linux version of darknet. https://github.com/AlexeyAB/darknet, May 2020.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[5] Xiaotong Chen, Seyed Mehdi Iranmanesh, and Kuo-Chin Lien. Patchtrack: Multiple object tracking using frame patches. *arXiv preprint arXiv:2201.00080*, 2022.

[6] Xuewen Chen, Yuanpeng Jia, Xiaoqi Tong, and Zirou Li. Research on pedestrian detection and deepsort tracking in front of intelligent vehicle based on deep learning. *Sustainability*, 14(15):9281, 2022.

[7] Forrest Ray Cord. Trauma informed practices at the middle school level. 2020.

[8] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 29, 2016.

[9] Olivier Debeir, Philippe Van Ham, Robert Kiss, and Christine Decaestecker. Tracking of migrating cells under phase-contrast video microscopy with combined mean-shift processes. *IEEE transactions on medical imaging*, 24(6):697–711, 2005.

[10] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*, 2020.

[11] Yunhao Du, Yang Song, Bo Yang, and Yanyun Zhao. Strongsort: Make deepsort great again. *arXiv preprint arXiv:2202.13514*, 2022.

[12] Chao Duan and Xingxing Li. Multi-target tracking based on deep sort in traffic scene. In *Journal of Physics: Conference Series*, volume 1952, page 022074. IOP Publishing, 2021.

[13] Xubo Fu, Kun Zhang, Changgang Wang, and Chao Fan. Multiple player tracking in basketball court videos. *Journal of Real-Time Image Processing*, 17(6):1811–1828, 2020.

[14] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Marko Heikkilä, Matti Pietikäinen, and Cordelia Schmid. Description of interest regions with local binary patterns. *Pattern recognition*, 42(3):425–436, 2009.

[20] Zhanchao Huang, Jianlin Wang, Xuesong Fu, Tao Yu, Yongqi Guo, and Rutong Wang. Dc-spp-yolo: Dense connection and spatial pyramid pooling based yolo for object detection. *Information Sciences*, 522:241–258, 2020.

[21] Zicong Jiang, Liquan Zhao, Shuaiyang Li, and Yanfei Jia. Real-time object detection method for embedded devices. In *computer vision and pattern recognition*, 2020.

[22] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[23] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[24] Ülo Lepik and Helle Hein. Haar wavelets. In *Haar Wavelets*, pages 7–20. Springer, 2014.

[25] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[27] Chenxu Luo, Xiaodong Yang, and Alan Yuille. Exploring simple 3d multi-object tracking for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10488–10497, 2021.

[28] Wenhan Luo. Generic multiple object tracking. 2015.

[29] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[30] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8844–8854, 2022.

[31] Mahyar Najibi, Mohammad Rastegari, and Larry S Davis. G-cnn: an iterative grid

based object detector. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2369–2377, 2016.

[32] Mehmet Ocakli and Mubeccel Demirekler. Video tracker system for traffic monitoring and analysis. In *2007 IEEE 15th Signal Processing and Communications Applications*, pages 1–4. IEEE, 2007.

[33] Akshay Rangesh and Mohan Manubhai Trivedi. No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars. *IEEE Transactions on Intelligent Vehicles*, 4(4):588–599, 2019.

[34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[36] F. Rosenblatt. The perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[37] Afef Salhi, Fahmi Ghozzi, and Ahmed Fakhfakh. Estimation for motion in tracking and detection objects with kalman filter. In *Dynamic Data Assimilation-Beating the Uncertainties*. IntechOpen, 2020.

[38] Suyash Shetty. Application of convolutional neural network for image classification on pascal voc challenge 2012 dataset. *arXiv preprint arXiv:1607.03785*, 2016.

[39] Daniel Stadler and Jurgen Beyerer. Improving multiple pedestrian tracking by track management and occlusion handling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10958–10967, 2021.

[40] Michael J Swain and Dana H Ballard. Indexing via color histograms. In *Active perception and robot vision*, pages 261–273. Springer, 1992.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[42] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[43] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *European Conference on Computer Vision*, pages 107–122. Springer, 2020.

[44] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[45] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12352–12361, 2021.

[46] Feng Yang, Xingle Zhang, and Bo Liu. Video object tracking based on yolov7 and

deepsort. *arXiv preprint arXiv:2207.12202*, 2022.

[47] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. *arXiv preprint arXiv:2110.06864*, 2021.

[48] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021.

[49] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016.

[50] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.