N° d'ordre :
N° de série :

PEOPLE'S DEMOCRATIC REPUBLIC OFALGERIA
Ministry of Higher Education and Scientific Research

ECHAHID HAMMA LAKHDAR UNIVERSITY - EL OUED
FACULTY OF EXACT SCIENCES
Computer Science department

End of Study Memory
Presented for the Diploma of

ACADEMIC MASTER

Domain : **Mathematics and Computer Science**
spinneret: **Computer Science**
Speciality : **Artificial Intelligence and Distributed Systems**

Presented by :

- **Riadh BenNaceur**
- **Oussama Ben Abdallah**

Theme

Anomaly Detection By Learning, Case Study: "Road Control"

M.		MCA	President
M.		MAA	Reporter
M.	Mouadh Bali	MAA	Supervisor
M.	Adnane Laib	MAA	Co-Supervisor

University Year: 2019/2020

Acknowledgments

We are grateful to numerous local and global peers who have contributed towards shaping this thesis. we are very much obliged to Prof. Mouadh Bali for his guideline, advice and support during our thesis work.we are very much indebted to Prof. Laib Adnane, for his con-tenuous encouragement and support. He is always ready to help with a smile.we are also thankful to all the professors of the department for their support.we are really thankful to my all friends. our sincere thanks to everyone who has provided us with kind words, a welcome ear, new ideas, useful criticism, or their invaluable time, we are truly indebted. Last, but not the least, we would like to dedicate this thesis to our family, for their love, patience, and understanding.

Oussama Ben Abdallah, in September 28, 2020.

Abstract

Road accidents are a very serious and high priority concern for public health, as statistics refer to the National Center for Prevention and Security, by road during the first nine months of the year 2019, 17,525 [1] traffic accidents were recorded that occurred at the national level, and out of 26,954 victims we found nearly 10 percent of them died. Various risk factors such as speeding, driving under the influence of alcohol, lack of safety equipment, distracted driving, unsafe vehicle, law enforcement and most importantly, insufficient emergency care after a collision. Any delay in detecting and providing emergency care could increase the severity of the accident.

With advancements in the fields of artificial intelligence, machine learning and deep learning, we are able to make our devices smarter and smarter. We have to install traffic cameras in almost every part of the city.

In this research, we propose an AI-based traffic monitoring system that can detect the occurrence of vehicle accidents such as cars, bicycles, etc. in live camera feeds and detect collisions of these moving objects.

For this purpose, we focused on an improved Yolo algorithm capable of detecting incidents in real time, as the model was trained on a custom dataset. It achieved an average accuracy (mAP) of 0.605. This is followed by an object-tracking algorithm that relies on the Lucas–Kanade method to track surveillance footage. The probability of an accident is determined based on anomalies in the roads after they have interfered with or veered other vehicles. The proposed model provides a robust method for detecting incidents with a high accuracy of 94.5 percent on CCTV surveillance footage of public traffic. This framework was evaluated on diverse conditions such as broad daylight, low visibility, rain, hail, and snow using the proposed dataset. This framework was found effective and paves the way to the development of general-purpose vehicular accident detection algorithms in real-time

Keywords— Computer Vision, CCTV Camera, Object Detection, Object Tracking, Anomaly Detection

[1] http://www.aps.dz/societe/96240-accidents-de-la-route-2-557-deces-et-24-397-blesses-lors-des-9-premiers-mois-de-2019 (accessed 27.10.2019)

الملخص

تعتبر حوادث الطرق مصدر قلق شديد الخطورة وذات أولوية عالية للصحة العامة حيث تشير الإحصاءات للمركز الوطني للوقاية والأمن عبر الطرق خلال التسعة أشهر الأولى من العام 2019 تم تسجيل 17525 حادث سير وقعت على مستوى التراب الوطني ومن بين 26954 ضحية وجدنا تقريبا 10 بالمائة منهم قد لقوا حتفهم . نتيجة عوامل الخطر المختلفة مثل السرعة ، والقيادة تحت تأثير الكحول ، وعدم وجود معدات السلامة ، والقيادة المشتتة ، والمركبة غير الآمنة ، وعدم احترام القانون ، والأهم من ذلك عدم كفاية الرعاية الطارئة بعد الاصطدام. أي تأخير في الكشف عن تقديم الرعاية الطارئة يمكن أن يؤدي ذلك إلى زيادة خطورة الحادث.

مع التقدم في مجالات الذكاء الاصطناعي والتعلم الآلي والتعلم العميق ، أصبحنا قادرين على جعل أجهزتنا أكثر ذكاءً. فإلزاما علينا تثبيت كاميرات مراقبة حركة المرور في كل جزء من المدينة تقريبًا.

نقترح في هذا البحث نظامًا لمراقبة حركة المرور قائم على الذكاء الاصطناعي يمكنه اكتشاف وقوع حوادث المركبات مثل السيارات والدراجات وغيرها في موجزات الكاميرا الحية واكتشاف تصادم هذه الأجسام المتحركة .

لهذا الغرض، قمنا بالتركيز على خوارزمية YOLO المحسّنة القادرة على اكتشاف الحوادث في الوقت الفعلي ،حيث تم تدريب النموذج على مجموعة بيانات مخصصة . يحقق متوسط دقة (mAP) يبلغ 0.605 . متبوعًا بعد ذلك بخوارزمية تتبع كائن تعتمد على خوارزمية Lucas-Kanade (LK) لتتبع لقطات المراقبة. يتم تحديد احتمال وقوع الحادث بناءًا على الشذوذ في الطرق بعد تداخلها مع مركبات أخرى او انحرافها . يوفر النموذج المقترح طريقة قوية لاكتشاف الحوادث بدقة مرتفعة بنسبة 94.5 بالمائة على لقطات مراقبة CCTV لحركة المرور العامة.

تم تدريب إطار العمل هذا في ظروف متنوعة مثل ضوء النهار الواسع والرؤية المنخفضة والمطر والبرد والثلج باستخدام مجموعة البيانات المقترحة. تم العثور على إطار العمل هذا فعالاً ويمهد الطريق لتطوير خوارزميات اكتشاف حوادث المركبات للأغراض العامة في الوقت الفعلي .

الكلمات المفتاحية : تعلم الألة ، الشبكات العصبية ، الرؤية الحاسوبية، كشف الشذوذ ،تحديد الأشياء ، تتبع الأشياء.

Résumé

Les accidents de la route sont une préoccupation très grave et hautement prioritaire pour la santé publique, car les statistiques se réfèrent au Centre national de prévention et de sécurité, par route au cours des neuf premiers mois de l'année 2019, 17525 accidents de la route ont été enregistrés survenus au niveau national [2] , et sur 26954 victimes, nous avons constaté que près de 10% personnes ont trouvé la mort. Divers facteurs de risque tels que l'excès de vitesse, la conduite sous l'influence de l'alcool, le manque d'équipement de sécurité, la distraction au volant, le véhicule dangereux, l'application de la loi et, surtout, l'insuffisance des soins d'urgence après une collision. Tout retard dans la détection et la fourniture de soins d'urgence pourrait augmenter la gravité de l'accident.

Grâce aux progrès réalisés dans les domaines de l'intelligence artificielle, de l'apprentissage automatique et de l'apprentissage profond, nous sommes en mesure de rendre nos appareils plus intelligents et plus intelligents. Nous devons installer des caméras de circulation dans presque toutes les parties de la ville.

Dans cette recherche, nous proposons un système de surveillance du trafic basé sur l'IA qui peut détecter la survenue d'accidents de véhicules tels que des voitures, des vélos, etc. dans les flux de caméras en direct et détecter les collisions de ces objets en mouvement.

Pour cela, nous nous sommes concentrés sur un algorithme Yolo amélioré capable de détecter les incidents en temps réel, car le modèle a été formé sur un jeu de données personnalisé et a atteint une précision moyenne (mAP) de 0,605. Ceci est suivi d'un algorithme de suivi d'objets qui s'appuie sur l'algorithme Lux-Knad pour suivre les images de surveillance. La probabilité d'un accident est déterminée en fonction des anomalies des routes après qu'elles ont interféré avec ou viré d'autres véhicules. Le modèle proposé fournit une méthode robuste pour détecter les incidents avec une précision élevée de 94,5% sur les images de surveillance CCTV du trafic public.

Ce cadre a été formé dans des conditions aussi diverses que la lumière du jour, la faible visibilité, la pluie, la grêle et la neige en utilisant l'ensemble de données proposé. Ce cadre s'avère efficace et ouvre la voie au développement d'algorithmes de détection des accidents de véhicules à usage général en temps réel.

Mots clés— Apprentissage automatique (Machine Learning), réseaux de neurones (NN), Détection d'anomalie, CCTV, Vision par ordinateur (Computer Vision).

[2] http://www.aps.dz/societe/96240-accidents-de-la-route-2-557-deces-et-24-397-blesses-lors-des-9-premiers-mois-de-2019 (accessed 27.10.2019)

Contents

Abstract iii
List of Figures ix

GENERAL INTRODUCTION 1

1 ANOMALY DETECTION 3
1.1 Introduction 3
1.2 Anomaly detection definition 3
1.3 Need for Anomaly Detection 3
1.4 Types Of Anomalies 4
1.4.1 Point Anomaly 4
1.4.2 Contextual Anomaly 4
1.4.3 Collective Anomaly 4
1.5 aspects of an anomaly detection problem 4
1.5.1 Nature of Input Data 4
1.5.2 Output of Anomaly Detection 5
1.6 Data Labels of anomaly detection techniques 5
1.6.1 Supervised anomaly detection 5
1.6.2 Semi-supervised anomaly detection 5
1.6.3 Unsupervised anomaly detection 5
1.7 Type of Data 6
1.8 Applications of Anomaly detection 7
1.9 Classification-Based Anomaly Detection techniques 8
1.9.1 Neural Networks Based 8
1.9.2 Bayesian Networks Based 8
1.9.3 Support Vector Machines Based 8
1.9.4 Rule Based 8
1.10 Road Traffic Anomaly Detection 8
1.10.1 Video surveillance and Anomaly detection 8
1.10.2 Road -Accident Detection 9
1.10.3 Technology of anomaly detection in road traffic 9
1.10.4 Surveillance video analysis: relevance in present world 9
1.10.5 Challenges and Possibilities 10
1.11 Conclusion 11

2 STATE OF THE ART 12
2.1 Machine Learning (ML) 13
2.1.1 Types of Learning 13
2.1.2 Categories Of Machine Learning 14
2.2 K-Means Clustering Algorithm 14

2.3 Deep Learning 15
2.4 Artificial neural networks 15
2.4.1 Basic Concepts 15
2.4.2 Multi-layer networks 16
2.4.3 Gradient Descent Methods 16
2.4.4 Back-propagation 16
2.4.5 Cost Function 17
2.4.6 Activation function 17
2.4.7 Regularization 19
2.4.8 Dropout 19
2.4.9 Overfitting 19
2.4.10 Steps for reducing over-fitting 20
2.5 The Convolutional Neural Network (CNN) 21
2.5.1 What is Convolutional neural networks ? 21
2.5.2 CNN architectures 21
2.5.3 Layer of CNNs 24
2.5.4 Filters (Convolution Kernels) 25
2.5.5 Applications of CNNs 26
2.6 Recurrent neural networks 27
2.6.1 Long Short Term Memory(LSTM) 28
2.6.2 Anomaly Detection for Temporal Data using LSTM 29
2.7 Computer Vision 29
2.7.1 Object detection 29
2.7.2 Neural Networks framework and Detection algorithms 31
2.7.3 Object Detection-based Deep Learning 31
2.7.4 Real Time Object Detection 32
2.8 Related work 32
2.8.1 Object detection 32
2.8.2 Object tracking 33
2.8.3 Traffic anomaly detection 33
2.9 Conclusion 34

3 METHODOLOGY AND FRAMEWORK 35
3.1 Overview of traffic accident detection system 35
3.2 Object Detection 35
3.2.1 Comparisons with State-of-the-art Detection Methods 35
3.2.2 You only look once (YOLO) 36
3.2.3 YOLO-V2 37
3.2.4 YOLO-V3 37
3.2.5 Architecture Of YOLO 38
3.2.6 Grid cell 39
3.2.7 Anchor Box 40
3.2.8 Intersection over Union (IoU) 41
3.2.9 Network 42
3.2.10 Loss Function 43
3.2.11 Non-maximal suppression 45

3.2.12 mAP(mean Average Precision) 46
3.2.13 Datasets 46
3.2.13.1 MIO-TCD Dataset 46
3.2.13.2 Urban dataset 47
3.2.13.3 Data preprocessing 47
3.2.14 Training 50
3.2.15 Evaluation Metrics 50
3.3 Object Tracking 53
3.3.1 Optical flow 53
3.3.2 Shi-Tomasi method 55
3.3.3 Lucas-Kanade (LK):Sparse Optical Flow 56
3.3.4 Method 57
3.4 Anomaly Detection(Accident Detection) 57
3.4.1 Method 57
3.4.2 Dataset 59
3.4.3 Training 61
3.4.4 Evaluation 61
3.5 Challenges 62
3.6 Experimental Analysis 62
3.7 Conclusion 62

General Conclusion and Future Scope **63**

References **64**

A Original Project Code **69**
A.1 Object detection pseudo code 69
A.2 Loss function 71
A.3 Object Tracking pseudo code 73
A.4 Anomaly detection pseudo code 76

List of Figures

1.1 Taxonomy of anomaly detection 4
1.2 Generic framework for network anomaly detection.From [4] 6
1.3 Block diagram of the Nature of data.From [16] 7
1.4 Video traffic surveillance system 9

2.1 Structure of a biological neuron. 15
2.2 architecture of an artificial neuron 16
2.3 A fully-connected multi-layer neural network 16
2.4 Neural network pipeline 17
2.5 Commonly used activation functions 19
2.6 Dropout Neural Net Model 19
2.7 Schematic diagram of CNN structure.Figure from: [18] 22
2.8 Schematic diagram of the AlexNet structure.Figure from: [18] 22
2.9 VGG-16 neural network architecture Figure From :[51]. 23
2.10 Comparison of network depth and accuracy on state-of-the-art methods [35]. 24
2.11 Schematic diagram of pooling layer calculation process (max-pooling layer). 25
2.12 The kernel is moved over the image performing the convolution as it goes. 25
2.13 A step in the Convolution Process. 26
2.14 A standard RNN. The left hand side of the figure is a standard RNN.The state vector in the hidden units is denoted by A. On the right hand side is the same network unfolded in time to depict how the state is built over time. Image adapted from [70]. 28
2.15 The structure of the Long Short-Term Memory (LSTM) 29
2.16 A comparison between the two main one-stage object detection networks: SSD and YOLO. Figure from:[49] 30
2.17 An example of SSD default boxes. 31
2.18 vehicles detection-based deep learning (Yolo and SSD). 32

3.1 Block diagram of the our Methodology 35
3.2 Detection results between different methods on UA-DETRAC dataset. In addition to YOLO3, DP-SSD512 outperforms all other methods on accuracy in real-time, and YOLO2 achieves the highest speed of 64.65 FPS. [86] 36
3.3 The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence. Figure from:[35] 37
3.4 YOLOv3 network structure. Figure from [87] 38
3.5 The Network Architecture Of YOLO. 39

3.6 The YOLO detection model divides the input image into an S×S grid, and for each grid cell it predicts B bounding boxes,confidence for those boxes, and C class probabilities. Figure from: [26] 39
3.7 Yellow box is grid cell, blue box is the prediction box 40
3.8 Anchor boxes. 40
3.9 iou by the number of anchors And Anchors. 41
3.10 Illustration depicting the definitions of intersection and union. 41
3.11 Intersection over Union (IOU) calculation diagram. 42
3.12 Flowchart of YOLO network 42
3.13 Bounding boxes with dimension priors and location prediction.[62] 43
3.14 Non-Max Suppression From [12] 45
3.15 Sample images from the MIO-TCD dataset.[1] 47
3.16 flipping image 48
3.17 Adding noise 48
3.18 Sheared image 49
3.19 translated image 49
3.20 scaling image 49
3.21 RGB image transformed to HSV color space 50
3.22 training losses 50
3.23 Evaluation Metrics 51
3.24 Precision-Recall curve of each category 53
3.25 Optical flow problem 54
3.26 good Features To Track 55
3.27 Optical flow is estimated for the black pixels 56
3.28 Object tracking method 57
3.29 Network 58
3.30 VGG16(without Fully Connected layers) 58
3.31 label program 59
3.32 Car crash detection with the proposed method 60
3.33 label program 60
3.34 Frame (On the left crop1 and on the right crop2). 61
3.35 distributed frames 61

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
BP	Back Propagation
CNN	Convolutional Neural Network
CCTV	Closed Circuit TeleVision
CV	Computer Vision
COCO	Common Objects in Context
DL	Deep Learing
DNN	Deep Neural Network
ML	Machine Learing
NN	Neural Network
FC	Fully Connected (layer or network)
Fast R-CNN	Fast Region CNN
ResNet	Residual Neural Network
RL	Reinforcement Learning
ReLU	Rectied Linear Unit
RNN	Recurrent neural network
Faster R-CNN	Faster Region CNN
SVM	Support Vector Machine
YOLO	You Look Only Once - Object Detector
LSTM	Long short-term memory
KLT	Kanade-Lucas-Tomas
mAP	mean Average Precision
BCE	Binary Cross Entropy
HSV	Hue, Saturation and Value

GENERAL INTRODUCTION

Pervasive use of CCTV cameras in public and private places has laid the foundation for development of various automated systems for intelligent visual monitoring. Numerous tasks such as anomaly detection, object tracking, Facial Recognition etc. play a significant role in ensuring secure and intelligent transportation. Where we can say that the detection of anomalies is an important problem that has been well-studied within diverse research areas and application domains such as attacks or large data transfers in IP networks.

Anomaly detection is one of the most challenging and long standing problems in computer vision, Since the underlying building block of a typical anomaly detection is learning . More specifically, automatic detection of anomalous events in road/traffic videos can have multiple applications such as traffic rules violation detection, accidents/suspicious movements analysis, etc.

Anomaly usually means identification of events that significantly deviate from regular/normal behavior or process of identifying unexpected items or events in datasets, which differ from the norm. However, the meaning of anomaly may change , For example, driving a car on the road is normal but stalled car on highway is considered to be anomaly. Furthermore, the non-moving cars stationed in parking area does not constitute anomalous behavior. Similarly, the vehicles stopped near traffic lights are normal behavior when it is red but anomaly when it is green.

Anomaly detection in videos is a challenging problem due to sparse occurrence of anomalous events, inconsistent behavior of different type of anomalies. In this thesis, and with focusing primarily on roads, we aim to road anomalies detection on videos surveillance . Thus, the proposed system detects the anomaly on video using computer vision based deep learning technique CNN (Convolution Neural Network).

The aim of this work is to detect anomalous vehicles in traffic surveillance videos like accidents and abnormal objects.We propose and evaluate a framework for detecting abnormal vehicle behavior using traffic camera. The framework uses method for object detection in video frames, then tracks vehicle location through successive frames, and finally performs anomaly detection vehicle behavior. To achieve these goals two asked questions are established, in extension to related works.
® What is a practical solution to real time detection of anomaly .in our case "accident detection in road " ?
® How can hyper parameter optimisation, transfer learning and dataset composition be used to improve the performance in terms of speed and accuracy of a the proposed solution?

This work is divided into four chapters. Following this General introduction :

® **This part is about the first objective of the thesis:** Anomaly detection. In this part, we start by a the concept of anomaly detection. This chapter will first address anomaly detection in its global aspects. Then, Afterwards, a focus will be made on particular techniques used for anomaly detection.

® **Chapter 2** provides a brief introduction into the important concepts of machine learning and deep learning ,neural networks as a part of machine learning. In this thesis, with a focus on convolutional neural networks

® **Chapter 3:** In this chapter, theoretical knowledge required for understanding the methods discussed in the chapter 3 has been provided. Details about machine learning, neural networks, and computer vision have been discussed, followed by the explanation of the Algorithm of the Object detection and object Tracking .

® **Chapter 4 :** In this chapter, we begin discussing the experimental part of the thesis. First, we will discuss selection for methods and datasets. Then we will describe the selected methods, their parameters and the selected datasets. Finally, we will discuss postprocessing and evaluation. The implementation of the methods is discussed in the following chapter.

® Lastly, concludes the work of this work and explains briefly all the steps,and discuss the future work in this research area.

Chapter 1

ANOMALY DETECTION

1.1 Introduction

The question of anomaly detection has interested researchers in different domains for more than a century. Because it is a very large and complex field .The question has taken many names such as “anomaly detection”, “outlier detection”, “novelty detection”,” Abnormal Event detection ” There is no universally accepted definition of anomaly or outlier. Nevertheless, many definitions depending on the context have been proposed. to detecting anomalous behavior in road traffic is traffic accident, traffic jam, and red-light violation etc ,All of this is called an anomaly .

Therefore, In this chapter , we present the state of the art on anomaly detection and clarifies the related concepts of anomaly detection, e.g., outlier detection and novelty detection. Then, diverse aspects of anomaly detection, for example, the types of anomalies and the general methods, are further discussed. With the related concepts and taxonomy being addressed.

1.2 Anomaly detection definition

There are many definitions to detect anomalies :

As explained in [16], anomaly detection can be defined as the ability of finding patterns which differ from an expected nature.

Anomaly detection(also known as outlier detection) is an important data analysis task[3].It is an important tool to detect abnormalities in many different domains including Traffic Accident Detection , financial fraud detection, computer network intrusion, human behavioural analysis . and they are significant because they indicate significant but rare events[3].

1.3 Need for Anomaly Detection

Earlier, anomalies seldom occur. If the anomaly is not detected and rightful actions are not taken, soon the consequences may prove to be costly in situations like Network intrusion,

change in log patterns, data leak, fraud transactions, Insider trading and many more. Just imagine the loss that could incur if any of the lists mentioned above occurs.

1.4 Types Of Anomalies

Anomalies can be classified into following three categories:

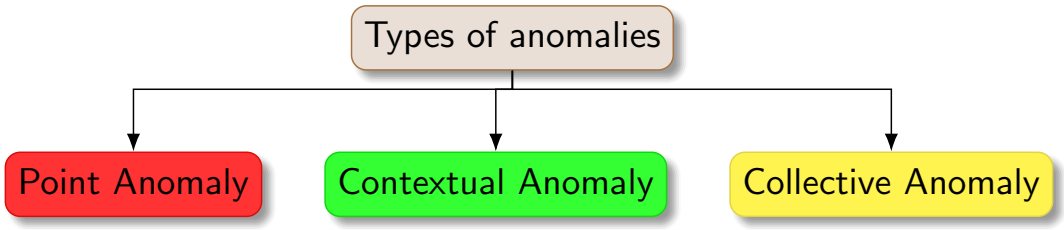


Figure 1.1: Taxonomy of anomaly detection

1.4.1 Point Anomaly

are individual samples of data having an abnormal value. For instance, a server, overloaded for a short time span, may have an abnormally long average response time.[53]

1.4.2 Contextual Anomaly

When a data instance behaves anomalously in a particular context, it is termed a contextual or conditional anomaly. For instance, the drop of the cell traffic is normal during scheduled maintenance operations and anomalous otherwise.[4]

1.4.3 Collective Anomaly

collective anomalies, are collections of data samples that are abnormal as a whole. However, the individual samples are not necessarily anomalous. This case may occur in sequential data and more precisely in time series. This could be the case of a wrongly configured probe having a persistent traffic loss[53]

1.5 aspects of an anomaly detection problem

1.5.1 Nature of Input Data

A primary part of any anomaly detection is the nature of the input data. The input data can be seen as a set of attributes. The attributes can be of different kinds such as categorical binary or continuous. Each data might has just one attribute or multiple attributes. Furthermore, the attributes of each data stances may be the same or different types.

1.5.2 Output of Anomaly Detection

One important issue for anomaly detection is how anomalies are represented as output which, generally, is in one of the two following ways (Chandola et al., 2009).

1. **Anomaly Scores** many anomaly detection algorithms output a score qualifying the level of "outlierness" of each data point. this kind of output can contain variety of parameters related to the data point.[17]
2. **Binary/label** According to these techniques, outputs are considered in a binary fashion, either anomalous or normal.

1.6 Data Labels of anomaly detection techniques

Three broad categories of anomaly detection techniques exist:

1.6.1 Supervised anomaly detection

is to build a predictive model for normal vs. anomaly classes. Supervised Anomaly Detection learns the separating boundary from a set of annotated data instances (training) and then, classify a test instance into either normal or anomalous classes with the learned model (testing).[16]

1.6.2 Semi-supervised anomaly detection

The training here only contains instances for the normal class. Any thing that can not be characterized as normal is thus marked as anomalous.[27]

1.6.3 Unsupervised anomaly detection

No training set is available nor is it needed.[27]

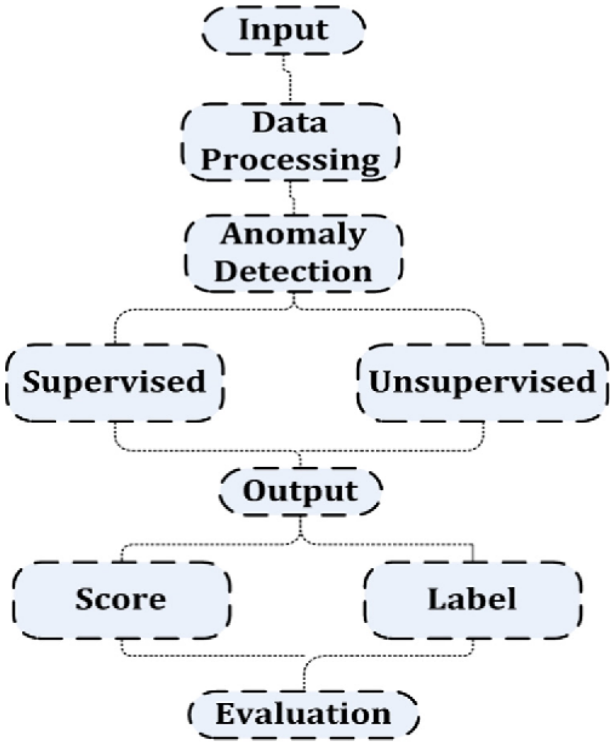


Figure 1.2: Generic framework for network anomaly detection.From [4]

Figure 1.2 displays a generic framework for network anomaly detection. The input data requires processing because the data are of different types. Processing techniques are based on the individual anomaly detection techniques. Then, the anomaly detection techniques (broadly categorized in two: supervised and unsupervised) are applied on the data. For evaluation of the output, either scores or labels are used (discussed in Section 1.5.2).

1.7 Type of Data

The fundamental key part of any anomaly detection method is the idea of information as you taking as input. Refer figure 1.3.

In general, data instances can be related to each other. Some examples are sequence data, spatial data, and graph data. In sequence data, the data instances are linearly ordered, e.g., time-series data, genome sequences, protein sequences. In spatial data, each data instance is related to its neighboring instances, e.g., vehicular traffic data, ecological data. When the spatial data has a temporal (sequential) component it is referred to as spatio-temporal data, e.g., climate data. In graph data, data instances are represented as vertices in a graph and are connected to other vertices with edges.[16]

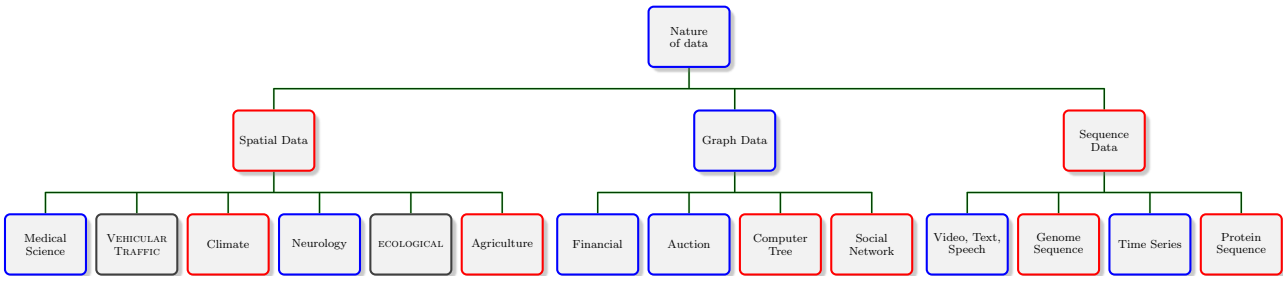


Figure 1.3: Block diagram of the Nature of data.From [16]

A primary data requirement for collective anomaly detection, is the presence of relationship between data instances. Three types of relations that have been exploited most frequently are sequential, spatial, and graphs:

Spatial: These techniques work with spatial data and find connected sub regions within the data as anomalies (also referred to as spatial anomalies).The data has spatial attributes, which define the location of a data instance .[15]

Graph Anomaly Detection Techniques: These techniques work with graph data and find connected sub graphs within the data as anomalies (also referred to as graph anomalies). Anomaly detection techniques have been applied to graph data in [58][Noble and Cook 2003].

Sequential: These techniques work with sequential data and find sub sequences as anomalies .The data is sequential, i.e., the contextual attributes of a data instance is its position in the sequence. [29]

Time-series data has been extensively explored in the contextual anomaly detection category.[29]

1.8 Applications of Anomaly detection

Anomaly detection is widely applied in research and in practice. These applications include network intrusion detection, fraud detection, fault detection in safety systems, novelty detection, and enemy activities surveillance (military use) (in [16]). and It is among these applications :

- network intersection detection
- Fraud detection
- Sensor Networks Anomaly detection
- internet of things (IOT Big-Data Anomaly detection)
- video surveillance anomaly detection
- Medical Anomaly detection
- cyber intrusion detection

1.9 Classification-Based Anomaly Detection techniques

Classification based anomaly detection techniques operate under the following general assumption:

1.9.1 Neural Networks Based

The neural network has been recognized as a promising technique for anomaly detection because the intrusion detector should ideally recognize

A basic multi-class anomaly detection technique using neural networks operates in two steps. First, a neural network is trained on the normal training data to learn the different normal classes. Second, each test instance is provided as an input to the neural network.[16]

1.9.2 Bayesian Networks Based

Bayesian networks has been used for anomaly detection in the multi-class setting. This technique is generally used for intrusion detection in combination with statistical schemes, a procedure that yields several advantages, including the capability of encoding inter dependencies between variables and of predicting events[40]

1.9.3 Support Vector Machines Based

Support Vector Machines (SVMs) has been used to anomaly detection in the one-class setting and gain big success. Such techniques use one class learning techniques for SVM and learn a region that includes the training data instances.[16]

1.9.4 Rule Based

Rule-based anomaly detection techniques learn rules that capture the normal behavior of a system. A test instance that is not covered by any such rule is considered as ananomaly. Rule-based techniques have been applied in multi-class as well as one-class settings.

1.10 Road Traffic Anomaly Detection

1.10.1 Video surveillance and Anomaly detection

Surveillance (closed circuit television) CCTV cameras are increasingly being used in public places like intersections, banks. to increase public safety. However, anomaly detection in traffic videos, such asover speeding, collision, red-light violation, etc. has been a challenging task due to the lack of annotated data and great diversities of scenes. However, it is worthy to explore robust computer vision methods to solve this problem with the increasing use of cars and the importance of traffic safety.

1.10.2 Road -Accident Detection

The death rate caused by road accidents is gradually increasing. One possible way to help avoid death on road is to get timely help to the victims. The smart surveillance system can detect road accidents in real-time.

1.10.3 Technology of anomaly detection in road traffic

The surveillance video based anomaly detection system consists of components of road region definition, traffic rule definition, detection and tracking for vehicle and non-vehicle objects.

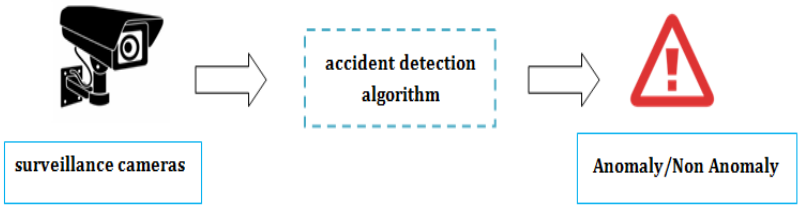


Figure 1.4: Video traffic surveillance system

The video is input from surveillance camera, which should be installed in a high place in order to watch multiple sections at a cross, as shown in Figure ??. In this way, one camera can monitor all the directions at a cross.[9]

1.10.4 Surveillance video analysis: relevance in present world

Analysis of surveillance videos involves a series of modules like object recognition, action recognition and classification of identified actions into categories like anomalous or normal [73]. The main objectives identified which illustrate the relevance of the topic are listed out below.

- Continuous monitoring of videos is difficult and tiresome for humans.
- Intelligent surveillance video analysis is a solution to laborious human task.
- Maximum accuracy is needed in object identification and action recognition.
- Prediction of certain movement or action or violence is highly useful in emergency situation like incident.
- Availability of huge data in video forms.

1.10.5 Challenges and Possibilities

Some of the stringent challenges on video-based anomaly detection are:

(a) **Illumination**

Even though a handful of anomaly detection methods have already been proposed, the number methods that can handle illumination variations, are limited. This is due to the incapabilities of illuminations agnostic feature extraction from the videos.

(b) **Pose and Perspective**

Often camera angles focusing on the surveillance area can have substantial impact on the performance of anomaly detection as the appearance of vehicle may change depending on its distance from the cameras. Though object detection accuracy has increased manifolds using deep neural network based methods, still there are challenges in tracking smaller objects. Humans can detect objects at different poses with ease, while machine learning may face difficulties in detecting and tracking the same object under pose variations.

(c) **Heterogeneous object handling**

Anomaly detection frameworks are largely based on modeling the scene and its entities. However, modeling heterogeneous objects in a scene or learning the movement of heterogeneous objects in a scene can be difficult at times.

(d) **Sparse vs. Dense**

The methods used for detecting anomalies in sparse and dense conditions are different. Though some of the methods are good at locating anomalies in sparse condition, dense scene-based methods can generate many false negatives.

(e) **Curtailed tracks**

Since many anomaly detections are based on vehicle trajectories, underlying tracking algorithms are supposed to perform accurately. Even though tracking accuracies have increased in the last decade, many of the existing tracking algorithms do not work under different scenarios . Tracking under occlusion is also another challenge though humans can easily track them visually.

(f) **Lack of real-life datasets**

There is a need for real-life datasets to see the effectiveness of anomaly detection techniques. There are ample scopes and requirements for anomaly detection research. With the advancements in machine learning techniques , computer vision-based behavior analysis, anomaly detection and anomaly prediction can leapfrog in the coming years. Deep learning-based hybrid frameworks can handle diverse traffic scenarios. This can also help to build fully automatic traffic analysis frameworks capable of reporting events of interest to the stakeholders.

1.11 Conclusion

In this chapter, we have seen The definition of anomaly detection, the state of the art on anomaly detection. Then, we seen the techniques used to detect anomalies and its different research areas .

Chapter 2

STATE OF THE ART

"For generations, scientists and philosophers have tried to explain ordinary reasoning in terms of logical principles with virtually no success. I suspect this enterprise failed because it was looking in the wrong direction: common sense works so well not because it is an approximation of logic; logic is only a small part of our great accumulation of different, useful ways to chain things together."

prof. Marvin Lee Minsky, The Society of Mind (1987)

In this chapter a brief background about machine learning, deep learning, and computer vision. then describes in details the Convolutional Neural Network (CNN) approach to pattern recognition starting from the basic concepts of convolution and artificial neural network. Later, we will describe a specific CNN based object detection model, YOLO [88], which is the most accurate real-time object detector.

2.1 Machine Learning (ML)

Learning is a very interesting and articulated phenomenon. Learning processes include the "to gain knowledge, or understanding of, or skill in, by study, instruction, or experience," and "modification of a behavioral tendency by experience.". Since the birth of computing, researchers have been striving to implant such capabilities in computers. Solving this problem has been, and still remains,one of the most challenging and fascinating long-term goals in artificial intelligence.The study of computer modeling of learning processes in their multiple manifestations constitutes the subject matter of machine learning.

In 1959, Arthur Samuel defined machine learning as a *"Field of study that gives computers the ability to learn without being explicitly programmed ".*[20]

Tom M. Mitchell provided a widely quoted, more formal definition: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"* .[72]

2.1.1 Types of Learning

There are three main classes of machine learning techniques, as discussed below :

1. **Supervised learning**

   The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

   The important algorithms are:

   (A) Categorical: regression (Logistics, Neural Networks,..), classification (KNN, Trees, Naive Bayes, SVM,..).

   (B) Continuous: regression (Leaner, Polynomial..), classification (Decision tree, Random forest..).

2. **Unsupervised learning**

   No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end.

   The important algorithms are:

   (A) Categorical: association (Apriori, FP-Growth,...), Hidden Markov Model,. . . etc.

   (B) Continuous: clustering (SVD, PCA, K-means,. . .).

3. **Reinforcement learning**

   A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without a teacher explicitly telling it whether it has come close to its goal or not.

Between supervised and unsupervised learning another category of learning methods can be found. It is called ***Semi supervised learning*** it is a method used to enable machines to classify both tangible and intangible objects.

2.1.2 Categories Of Machine Learning

Another categorization of machine learning tasks arises considering the desired output of a machine-learned : [76]

1. **Classification**

   inputs are divided into two or more classes, . This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

2. **Regression**

   which is also a supervised problem, the outputs are continuous rather than discrete.

3. **Clustering**

   a set of input patters have to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

2.2 K-Means Clustering Algorithm

K-Means algorithm based on dividing [48] is a kind of cluster algorithm, and it is proposed by J.B.MacQueen. This algorithm which is unsupervised is usually used in data mining and pattern recognition.Aiming at minimizing cluster performance index, square-error and error criterion are foundations of this algorithm [36].

K-Means algorithm based on dividing is a kind of cluster algorithm, and has advantages of briefness, efficiency and celerity.

2.3 Deep Learning

A field of machine learning called DL is based on the idea of capturing hierarchical structure of data with composite models[65]: *"Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. "*

2.4 Artificial neural networks

Neural networks are a popular type of machine learning model. are inspired by biological neural networks (common in the brains of many mammals)[66]. A special case of a neural network called the convolutional neural network (CNN) is the primary focus of this thesis. Before discussing CNNs, we will discuss how regular neural networks work. A neuron, also known as "neuron" or "nerve cell", is an electrically excitable cell that processes and transmits information through electrical and chemical signals [39].

2.4.1 Basic Concepts

ANNs originated as systems inspired by biological neural networks. They were intended to simulate animals brain. So, by analogy, they consist of neurons and connections between them.

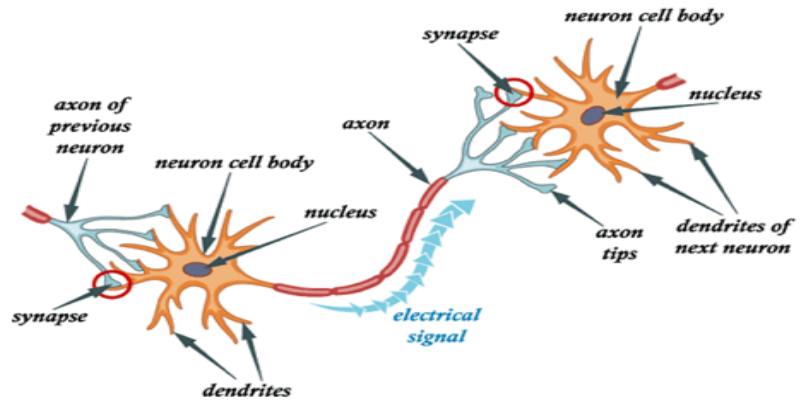


Figure 2.1: Structure of a biological neuron.

Mathematically, we model an artificial neuron as a function which calculates a weighted sum of the input vector x with a weight vector w, adds a bias term b and transforms the sum with a usually non-linear function σ called the activation function.

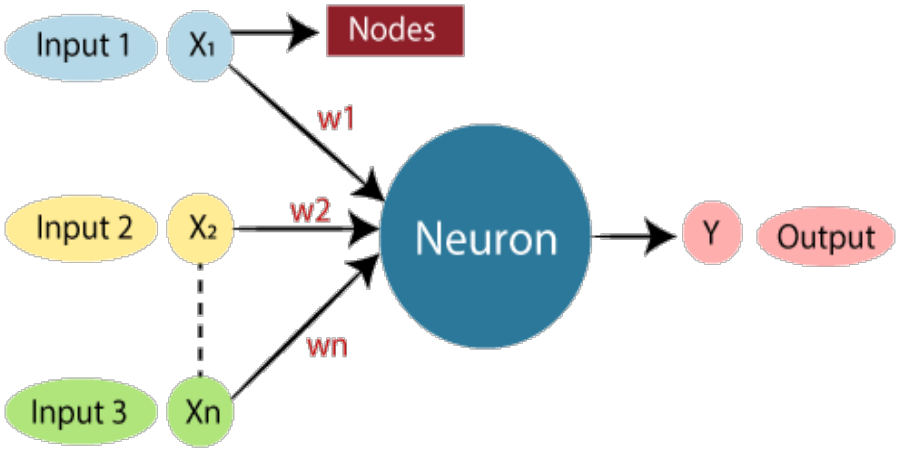


Figure 2.2: architecture of an artificial neuron

2.4.2 Multi-layer networks

Artificial neural networks consist of three layers The first layer is called input layer, the last layer is called output layer, and all the middle layers are hidden layers. each layer is made up of adaptive processing units, called neurons . Neurons between adjacent layers are connected for passing information as shown in Figure 2.3.

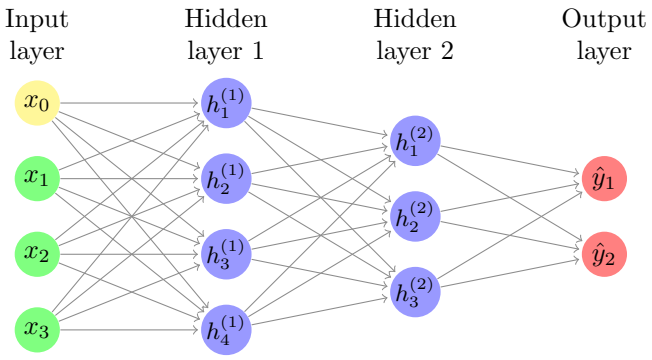


Figure 2.3: A fully-connected multi-layer neural network

2.4.3 Gradient Descent Methods

Gradient Descent (GD) method is one of the most common optimisation algorithms in machine learning field. and It is an optimization technique that is used to improve deep learning and neural network-based models by minimizing the cost function.

2.4.4 Back-propagation

"back-propagation" is an abbreviation, for "backward propagation of errors". It is a supervised learning technique for neural networks that calculates the gradient of descent for weighting different variables. Back-propagation is especially useful for deep neural networks working

on error prone projects, such as image or speech recognition.

2.4.5 Cost Function

A cost function is a function that measures the performance of a Machine Learning model for given data. or it is a measure of "how good" a neural network did with respect to it's given training sample and the expected output. This is typically expressed as a difference or distance between the predicted value and the actual value. and presents it in the form of a single real number.[56]

we used this functions :

- **binary cross entropy loss function :** Also called Sigmoid Cross-Entropy loss. It is a Sigmoid activation plus a Cross-Entropy loss. Unlike Softmax loss it is independent for each vector component (class), meaning that the loss computed for every CNN output vector component is not affected by other component values.

- **categorical cross entropy loss function :** Also called Softmax Loss. It is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the

2.4.6 Activation function

It's just a thing function that you use to get the output of node. It is also known as Transfer Function. Activation function transforms the output of a neuron. It is applied element wise on the output vector [25].

$$Y = \sigma(W^t x + b) \tag{2.1}$$

where x = input, w = weights, b = biases.

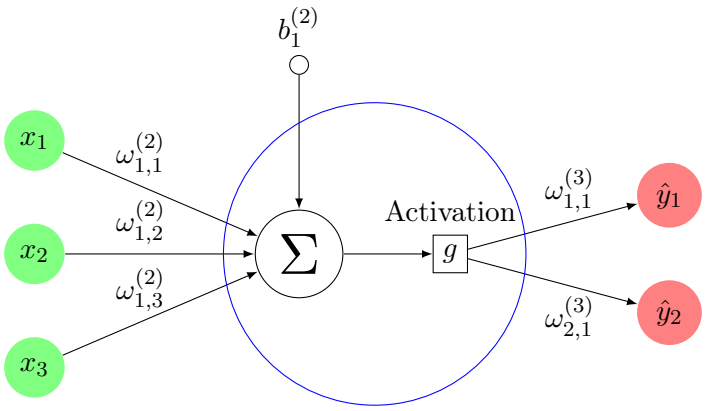


Figure 2.4: Neural network pipeline

1. **Sigmoid Function**

   A sigmoid function is a type of activation function, it is a non linear AF which are used in DL applications. Squashing functions limit the output to a range between 0 and 1, making these functions useful in the prediction of probabilities.

   The Sigmoid used mostly in feed forward neural networks. The Sigmoid function is given by the relationship

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

2. **Hyperbolic Tangent Function (Tanh)**

   is Tanh function also knows as Tangent Hyperbolic function. It's actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other and this function is given by [R2]

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

3. **Softmax Function**

   The Softmax function is another types of activation function but is handy when we are trying to handle classification problems and it used in neural computing. It is used to compute probability distribution from a vector of real numbers. The Softmax function produces an output which is a range of values between 0 and 1.[14] [R3]

$$f(xi) = \frac{\exp(xi)}{\sum \exp(xj)} \tag{2.4}$$

   The main difference between the Sigmoid and Softmax AF is that the Sigmoid is used in binary classification while the Softmax is used for multivariate classification tasks.[88]

4. **Softsign**

   softsign is an activation function for neural networks. the Softsign is another non-linear AF used in DL applications. The Softsign function is a quadratic polynomial.

5. **Rectified Linear Unit (ReLU) Function**

   most recent deep learning networks use rectified linear units (ReLUs) for the hidden layers. A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input.

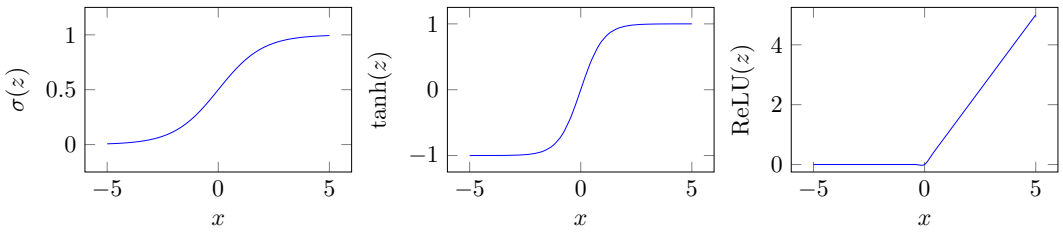


Figure 2.5: Commonly used activation functions

2.4.7 Regularization

Overfitting is an unneglectable problem in deep CNNs, which can be effectively reduced by regularization. In the following subsection, we introduce an effective regularization technique: Dropout [31]

2.4.8 Dropout

Dropout is one of the most effective and most commonly used regularization techniques for neural networks, It prevents over-fitting and provides a way of approximately combining exponentially many different neural network architectures efficiently.Dropout may be implemented on any or all hidden layers in the network as well as the visible or input layer. It is not used on the output layer.[57]

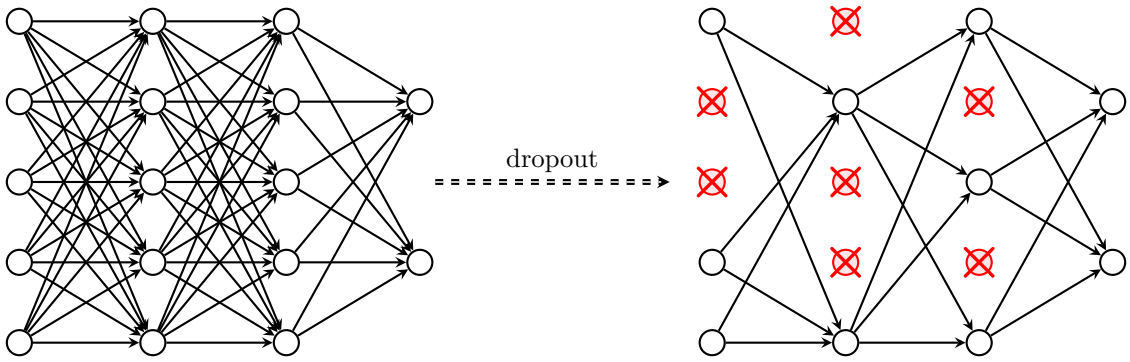


Figure 2.6: Dropout Neural Net Model

2.4.9 Overfitting

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, The overfitting is a common problem in the training process of neural network models and it greatly reduces the generalization ability of neural network models[22]

2.4.10 Steps for reducing over-fitting

we can try to do something about the overfitting. There are different options to do that. Reduce the network's capacity by removing layers or reducing the number of elements in the hidden layers. Apply regularization, which comes down to adding a cost to the loss function for large weights . Use Dropout layers, which will randomly remove certain features by setting them to zero or Add more data and Use data augmentation .

2.5 The Convolutional Neural Network (CNN)

"If we were magically shrunk and put into someone's brain while she was thinking, we would see all the pumps, pistons, gears and levers working away, and we would be able to describe their workings completely, in mechanical terms, thereby completely describing the thought processes of the brain. But that description would nowhere contain any mention of thought! It would contain nothing but descriptions of pumps, pistons, levers!"

G. W. Leibniz (1646–1716)

Convolutional networks (LeCun, 1989), also known as convolutional neural networks or CNNs is a special type of Neural Networks for processing data that has a known, which has shown exemplary performance on several competitions related to Computer Vision and Image Processing. Some of the exciting application areas of CNN include Image Classification and Segmentation, Object Detection [88], Video Processing [14], Natural Language Processing[50], and Speech Recognition[21].

The current part describes in details the Convolutional Neural Network (CNN) approach to pattern recognition starting from the basic concepts of convolution and artificial neural network. Later, we will describe a specific CNN based object detection model, YOLO [88], which is the most accurate real-time object detector.

2.5.1 What is Convolutional neural networks ?

Convolutional neural networks (CNN, ConvNet) is a class of deep, feedforward (not recurrent) artificial neural networks that are applied to analyzing visual imagery and that self optimise through learning.[59]

2.5.2 CNN architectures

CNNs are feedforward networks in that information flow takes place in one direction only, from their inputs to their outputs. The neurons within a CNN are split into a three dimensional structure, with each set of neurons analyzing a small region or feature of the image.

For example an image with H rows, W columns, and 3 channels (R, G, B color channels). Higher order tensor inputs, however, can be handled by CNN in a similar fashion. The input then sequentially goes through a number of processes. One processing step is usually called a layer, which could be a convolution layer, a pooling layer, a normalization layer, a fully connected layer ...ect .

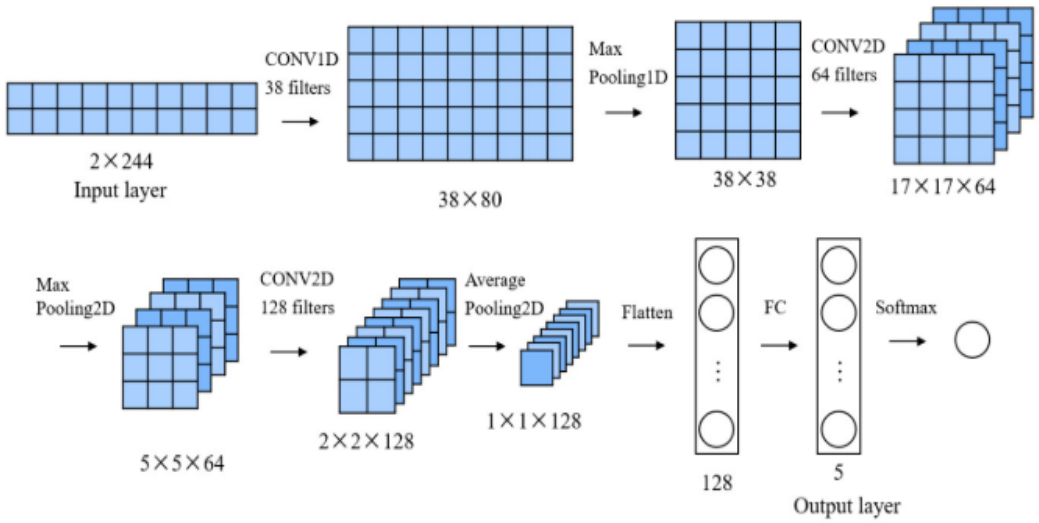


Figure 2.7: Schematic diagram of CNN structure.Figure from: [18]

1. **AlexNet** is considered as the first deep CNN architecture, it is refers to the deep convolutional neural network trained on the ILSVRC challenge data, AlexNet is composed of 5 convolution layers, 3 max-pooling 2×2 layers and fully connected layers .AlexNet was proposed by Krizhevesky et al.[41]

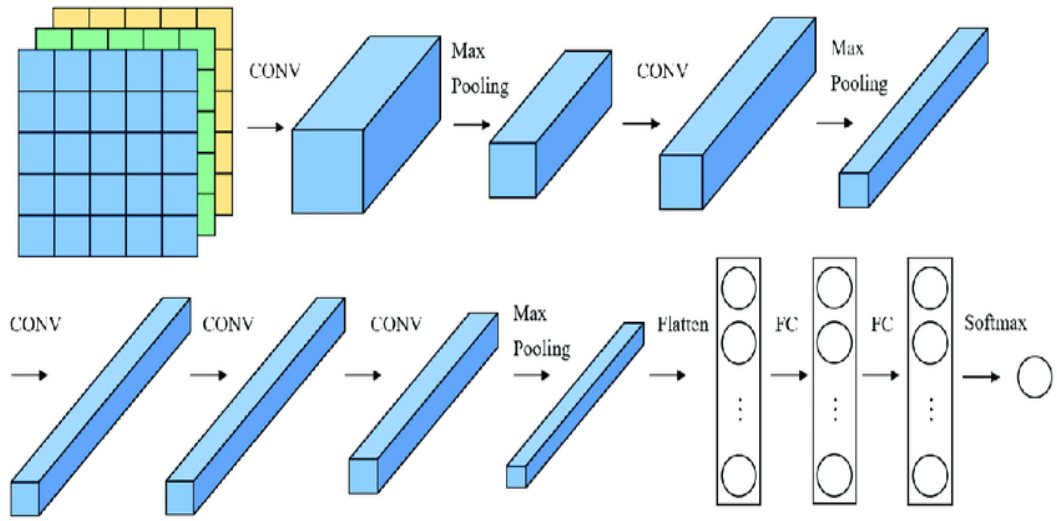


Figure 2.8: Schematic diagram of the AlexNet structure.Figure from: [18]

2. **LeNet**

   The first successful applications of CNNs. It was suggested by LeCuN in 1998 . It includes 5 convolutional layers and one fully connected layer and It is famous due to its historical importance as it was the first CNN.LeNet architecture was used to read zip codes, digits, etc.[6]

3. **GoogLeNet**

   GoogLeNet and is also known as Inception-V1 Built with a CNN, consisting of 22 layers. it inspired by LetNet, The main objective of the GoogleNet architecture was to achieve high accuracy with a reduced computational cost [6].

4. **VGGNet**

   VGGNet is designed to significantly increase the depthof the existing CNN architectures with 16 or 19 layers. We use it as an example to study the detailed structure of CNN networks released by the Oxford VGG group [6]

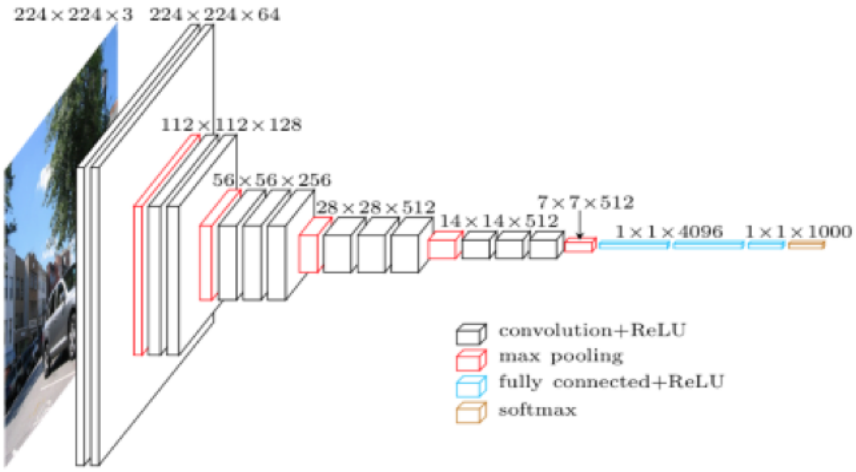


Figure 2.9: VGG-16 neural network architecture Figure From :[51].

5. **ResNet**

   is a very deep neural network learning technique proposed by He et al in 2015. it consisting 150 layer .The actual idea is that it reroutes the input and adds the concept learned in the previous network. The idea is that the next layer would learn from both the inputs of the previous layer and the previous layer itself [6]

   From these architectures, it is proved that CNN methods are highly efficient on image classification problems. As a result, these classical classification-oriented structures are largely used as backbones for other computer vision problems, such as detection, segmentation and image colouring.

   Furthermore, although the depth and width of the network are both increased, the computational complexity remains constant. However, as stated in [32], the degradation of training accuracy becomes a severe problem as CNNs go deeper, which makes deep neural networks hard to train. In order to solve this problem, ResNet is proposed in the paper. In this method, instead of optimizing the network layer by layer directly, it optimizes a residual mapping of a block of the layers so that the optimization process is

much easier and a network can easily reach a very deep level.Again it proves that deeper structures still have potential ability to improve the performance of the classification accuracy, as shown in Figure 2.10, which give a comparison of different network depths with the correlated Top-5 error(%) on some of the state-of the-art methods.

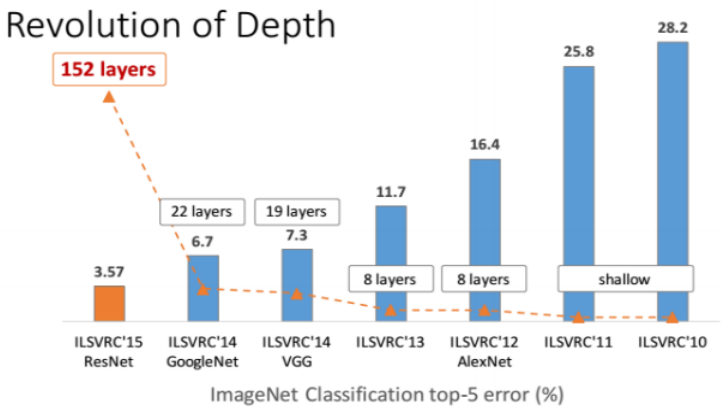


Figure 2.10: Comparison of network depth and accuracy on state-of-the-art methods [35].

From these architectures, it is proved that CNN methods are highly efficient on image classification problems. As a result, these classical classification-oriented structures are largely used as backbones for other computer vision problems, such as detection, segmentation and image colouring.

2.5.3 Layer of CNNs

A Convolutional Neural Network is composed by several kinds of layers, that are described in this section : convolutional layers, pooling layers and fully connected layers.

1. **Convolutional layer** Convolutional layer is essentially the primary layer in CNN architecture. The main task is to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map.which can be the original image or an another feature map.Its description includes four parts: number of channels; kernel spatial extent (kernel size); padding ; and stride size.[60]

2. **Max pooling layer**

   Pooling is an important concept of CNNit ,it is sometimes considered to be a separate layer .It lowers the computational burden by reducing the number of connections between convolutional layers. In general, they are used after multiple stages of other layers in order to reduce the computational well as minimizing the likelihood of overfitting.[60]

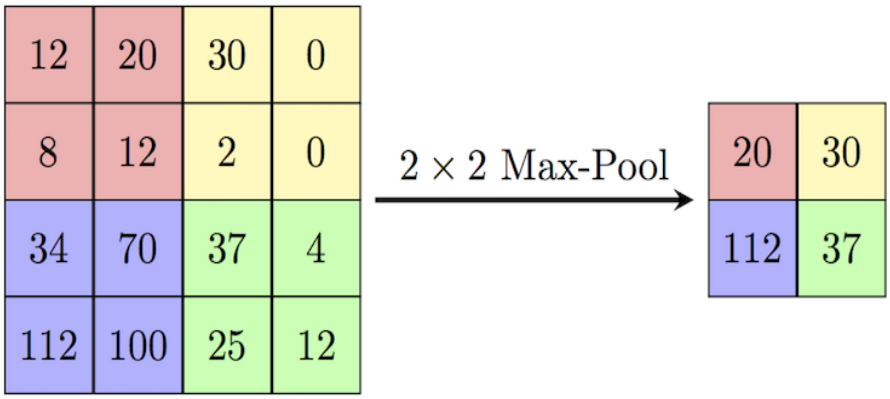


Figure 2.11: Schematic diagram of pooling layer calculation process (max-pooling layer).

3. **Full connection layer** Fully connected A fully connected layer is abbreviated as "FC." Fully connected layer is mostly used at the end of the network for classification. Unlike pooling and convolution. Therefore, each node in a fully-connected layer is directly connected to every node in both the previous and in the next layer [60]

2.5.4 Filters (Convolution Kernels)

A convolution is an operation that changes a function into something else. We do convolutions so that we can transform the original function into a form to get more information.

Convolutions have been used for a long time in image processing to blur and sharpen images, and perform other operations, such as, enhance edges and emboss.

A kernel is placed in the top-left corner of the image. The pixel values covered by the kernel are multiplied with the corresponing kernel values and the products are summated. The result is placed in the new image at the point corresponding to the centre of the kernel. An example for this first step is shown in the diagram below 2.12.

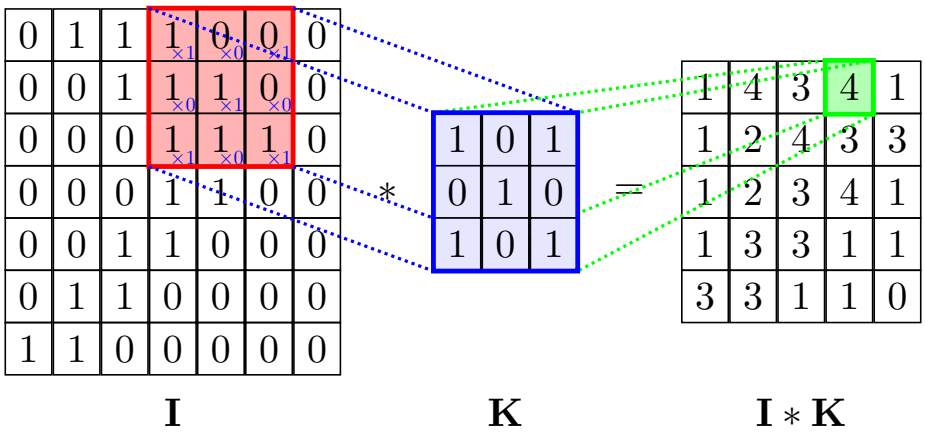


Figure 2.12: The kernel is moved over the image performing the convolution as it goes.

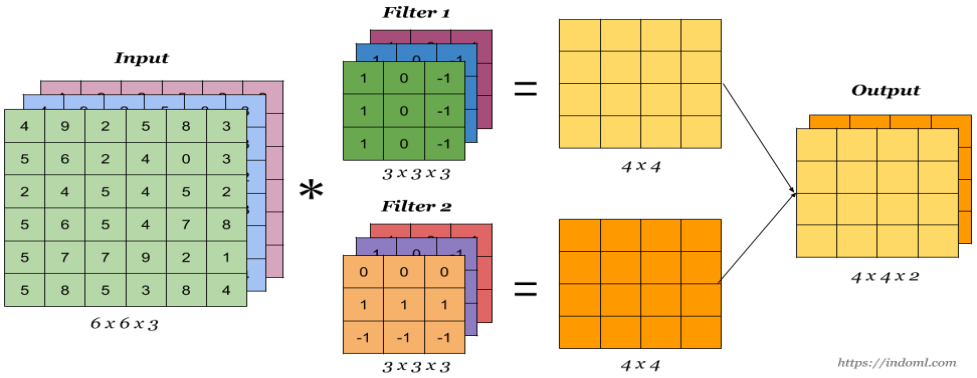


Figure 2.13: A step in the Convolution Process.

2.5.5 Applications of CNNs

Simple applications of CNNs which we can see in everyday life are obvious choices, like facial recognition software, image classification, speech recognition programs, etc. Some of the key applications of CNN are listed here -

(a) **Object Detection**

Object detection is a computer vision technique that allows us to identify and locate objects in an image or video. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations.[31]

(b) **Image Classification**

Compared with other methods CNNs achieve better classification accuracy on large scale datasets due to their capability of joint feature and classifier learning.Image classification refers to a process in computer vision that can classify an image according to its visual content. For example, an image classification algorithm may be designed to tell if an image contains a human figure or not. While detecting an object is trivial for humans, robust image classification is still a challenge in computer vision applications.[64]

(c) **Object Tracking**

Object tracking is to monitor an object's spatial and temporal changes during a video sequence, including its presence, position, size, shape, etc. This is done by solving the temporal correspondence problem, the problem of matching the target region in successive frames of a sequence of images taken at closely-spaced time intervals.

(d) **Text Detection and Recognition**

Text detection and recognition has emerged as an important problem in the past few years.Text detection aims to identify text regions of given images or videos which is also

an important prerequisite for many computer vision tasks, such as classification, video analysis.[43]

(e) **Analyzing Documents**

Convolutional neural networks can also be used for document analysis. This is not just useful for handwriting analysis, but also has a major stake in recognizers. For a machine to be able to scan an individual's writing, and then compare that to the wide database it has, it must execute almost a million commands a minute.

(f) **Decoding Facial Recognition**

Face recognition is the process of recognizing the face of a relevant person by a vision system. It has been a crucial human-computer interaction tool due to its usage in security systems, access-control, video surveillance and even it is used in social networks like Facebook as well. face recognition has once again attracted attention due to its non-intrusive nature and since it is main method of person identification for human when it is compared with other types of biometric techniques. Face recognition can also be easily checked without the subject person's knowledge in an uncontrolled environment.

(g) **Action Recognition in Video Sequences**

Action recognition in video sequences is a challenging problem of computer vision due to the similarity of visual contents, changes in the viewpoint for the same actions, and different illumination conditions . Generally, humanaction is a motion of body parts by interacting with objectsin the environment. In the context of videos, an action isrepresented using a sequence of frames, which humans caneasily understand by analyzing contents of multiple frames insequence.[77]

2.6 Recurrent neural networks

Recurrent neural networks (RNN's) are particular types of neural networks introduced in [[33], Hopfield 1982] as a mathematical model of particular biophysics processes. The first application of RNN's to solve large-scale tasks was done by [[68], Schuster et al. 1997] in the context of signal processing. More generally, these kinds of neural networks are particularly adapated to deal with sequential data.

A traditional fully connected feedforward network would have separate parameters for each input feature when modeling a time series. By comparison, a recurrent neural network shares the same weights across several time steps. The previous input of the system has an influence on the current and future outputs. Each output at a time step is produced using the same update rule applied to the previous outputs, Figure 2.14.

The initial hidden state can be initialized as h0 (commonly all zero), then the forward propagation update rule of RNN at each time step from t = 1 to t = τ

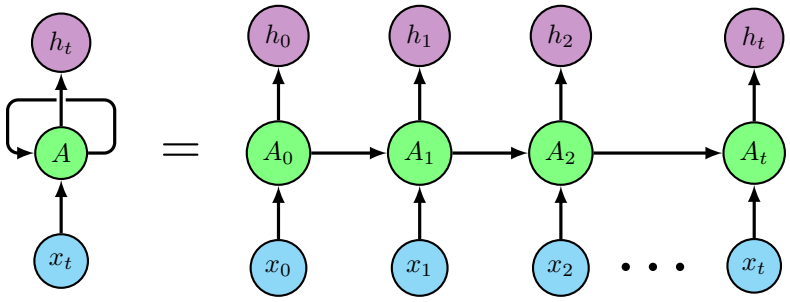


Figure 2.14: A standard RNN. The left hand side of the figure is a standard RNN.The state vector in the hidden units is denoted by A. On the right hand side is the same network unfolded in time to depict how the state is built over time. Image adapted from [70].

2.6.1 Long Short Term Memory(LSTM)

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation. An LSTM has four "gates": forget, remember, learn and use(or output) .It also has three inputs: long-term memory, short-term memory, and E. [70]

1. **Input gate** discover which value from input should be used to modify the memory. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1.

2. **Forget gate** is the gate you use to dump out all the unnecessary long term information. Kind of like when you study for a big exam, and the next day you forget everything. That's the power of the forget gate.

3. **Remember Gate** This gate takes the information from the forget gate and adds it to the information from the learn gate, to compute the new long term memory.

> Remember gate = Learn gate output + Forget gate output

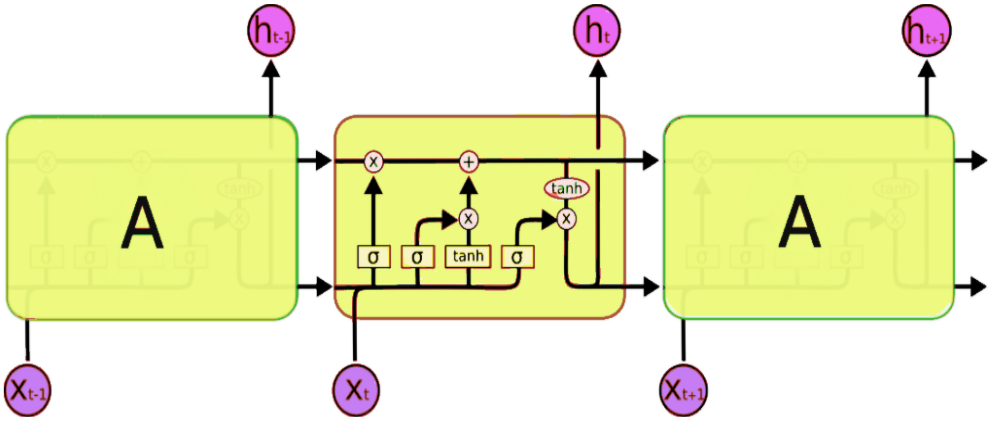


Figure 2.15: The structure of the Long Short-Term Memory (LSTM)

2.6.2 Anomaly Detection for Temporal Data using LSTM

The paper proposes the use of a Long Short-Term Memory recurrent neural network that is trained on normal data to predict normal time series patterns and predict future values where the prediction errors are used to detect anomalies. [70]

2.7 Computer Vision

"Just like to HEAR is not the same as to LISTEN, to TAKE PICTURES is not the same as to SEE" – *Fei-Fei Li*

Computer Vision is defined as a field of study that seeks to develop techniques to help computers "see" and processing, analyzing,and understanding the content of digital images such as photographs and videos. Applications of computer vision include image classification, visual detection, 3D scene reconstruction from 2D images, image retrieval, augmented reality, machine vision and traffic automation [67].

2.7.1 Object detection

Humans can detect and identify objects present in an image. The human visual system is fast and accurate and can also perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought.[45]

Object Detection: This is done through, Locate the presence of objects with a bounding box and types or classes of the located objects in an image.[45]

The development process of CNN is to build more stable and more efficient system for object detection. As along with the definition of proposal, Region with CNN, Fast Region with CNN (Fast R-CNN), and Faster Region with CNN (Faster R-CNN) have been proposed.

1. **Regions-based Convolutional Neural Network (R-CNN)**

   R-CNN has been widely used for object detection. Where R-CNN mainly plays as a classifier, the RCNN algorithm proposes a bunch of boxes in the image and checks if any of these boxes contain any object. Its accuracy depends on the performance of the region proposal module.[23]

2. **Fast R-CNN**

   The approach is similar to the R-CNN algorithm.solved some of the drawbacks of R-CNN to build a faster object detection algorithm and get all the regions of interest (regions containing some object).In practice, the results prove that these innovations are efficient in accelerating training time and testing time while also improving detection accuracy.[23]

3. **Faster R-CNN**

   Faster R-CNN is the modified version of Fast R-CNN. It uses search selective to find out the regions of interests and passes them to a ConvNet.The major difference between them is that Fast RCNN uses selective search for generating Regions of Interest, while Faster RCNN uses "Region Proposal Network", aka RPN.[10]

4. **Single Shot MultiBox Detector**

   Single Shot MultiBox Detector (SSD) is the cutting-edge real-time object detection deep neural network. SSD is a one-stage object detection network which, provided an input image, uses a feed-forward CNN approach to produce a predefined number of bounding boxes and confidence scores for objects in the given image. The early layers in SSD are based on a pretrained classification network (VGG 16 in the paper, but any high-quality image classifier could be used), called the base network (see figure 2.16), truncated prior to the layers used for classification.

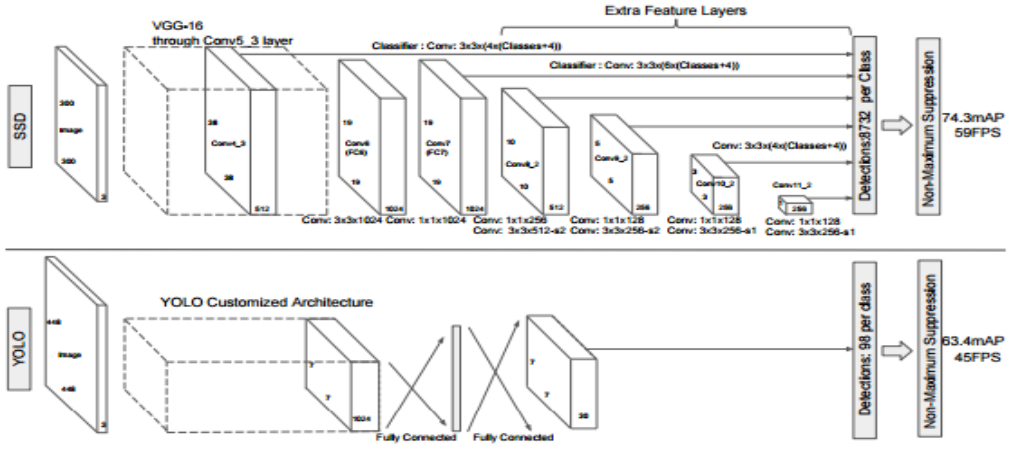


Figure 2.16: A comparison between the two main one-stage object detection networks: SSD and YOLO. Figure from:[49]

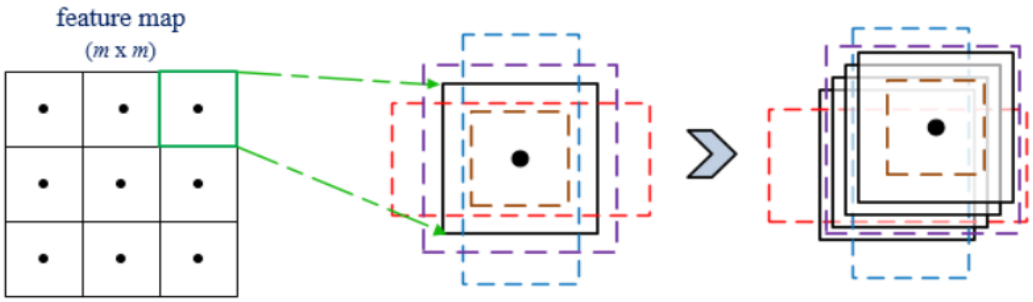


Figure 2.17: An example of SSD default boxes.

2.7.2 Neural Networks framework and Detection algorithms

An important choice regards the framework used to deploy the neural network and it is linked not only to platform compatibility but also to the detection algorithm and to the model chosen for the task, since different frameworks provide different implementations and some algorithms may not be found for certain frameworks. A key objective for analyzing live data is achieving good detection performance by keeping the computational cost as low as possible, and to achieve such a result single-pass detection algorithms such as YOLO [35] and SSD [49] were favoured to more complex ones such as Faster R-CNN [64] (for an overview of the key features of the state of the art detection algorithms the reader can refer to section 2.7.1). The first testing embryo of the detector was developed with Darknet [61] for the first platform, the Nvidia Jetson TK1, but the official framework was working only on the development laptop while it was outputting empty detection vectors on the board. The only way it could work on the board was by using a fork which only supported up to YOLOv2, but this solution was later dropped in favour of a different framework, Tensorflow, offering more flexibility and a better support.

2.7.3 Object Detection-based Deep Learning

Many detection systems repurpose classifiers or localizers to perform detection. They apply the classification to an image at multiple locations and scales. CNN classifier, like RCNN, can be used for this application. This approach gives good results but require many number of iterations to process a single image [75][42]. Many detection algorithms using selective search [75] with region proposals have been proposed to avoid exhaustive sliding window. With deep learning, the detection problem can be tackled in new ways, with algorithms like YOLO and SSD. We have implemented those two algorithms for object detection in real time , and obtained very good results (qualitative evaluation). Fig. 3.34 shows an example of our deep learning model applied in the localization and identification of vehicles within MIO-TCD dataset at the city Canada and the United States.

Figure 2.18: vehicles detection-based deep learning (Yolo and SSD).

2.7.4 Real Time Object Detection

Real time object detection is a sub category of computer vision where the aim is to detect and classify objects in real time. This means that the inference time of the deep learning models has to be fast enough to deal with a input under real time constraints, such as video, whilst also maintaining a good accuracy

Real time object detection is a field that has quite a short history as the hardware which was bleeding edge a few years ago did not have the computational power to perform inference in real time, 33 ms. The paper Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [64] showed that there was a possibility to do inference fast enough to reach 15 fps. A few months later the first YOLO paper [35] was released. YOLO was extremely fast and could achieve inference speeds to allow for 45 fps, whilst achieving a mAP score1 of 63.4. Real time object detectors were incrementally improved by SSD [49], YOLO900 [63] and YOLOv3 [62].

2.8 Related work

In this section, we briefly review the previous works about Object detection, Object tracking and anomaly detection .

2.8.1 Object detection

Object detection is a core problem in computer vision. Detecting vehicles in a video stream is an object detection problem. Fast, robust object detection systems are fundamental to the success video processing systems. Such systems are capable of searching for a specific class of objects, such as faces, people,bairplanes, or vehicles [84].

Currently, with the advent of CNN networks, have made significant advancements in image classification [42] and object detection [64]. Where RCNN based methods have become the mainstream of object detection. They can be categorized into two different classes:

single-stage (YOLO [88], SSD [49]) and two-stage networks (RCNN [23], Fast RCNN [23], Faster RCNN [10]) . two-stage networks are a methods very accurate, but come at a high computational cost (low frame-rate); in other words, they are not well suited for real-time vehicle detection. In contrast, Single-stage detectors are fast but less accurate. YOLO has scaling issues and cannot detect small objects accurately.

An alternative way of doing object detection is by combining these two tasks into one network. This is possible because instead of having a network produce region proposals.You Only Look Once (YOLO) is a state-of-the-art algorithm among single shot detector approaches for object detection [64]. YOLO trains on full images and is the fastest approach while maintaining high accuracy. This makes it a good fit for real-time vehicle detection. in [19], Histogram of Oriented Gradient (HOG) method is used for pedestrian detection; in [81], a multi-directional CNN method is used for vehicle license plate detection;in [30] a proposed accident detection system can be trained by using regression based algorithm called Optimized-YOLO algorithm which can be applied on CPU based devices. However, very few papers apply the above mentioned methods to detection of abnormal behavior of vehicles on roads.

2.8.2 Object tracking

Visual object tracking is one of the fundamental tasks in computer vision, and the cornerstone of many visual analytics applications in video surveillance, smart homes/cities, including surveillance [8], vehicle tracking [46], robotics [65], and medical imaging [71]. The goal of object tracking is to accurately locate objects of interest based on a series of consecutive video frames.

The tracking of vehicles is a key component in the accident detection , and accurate tracking in real world video remains a challenge because the scenario typically contains abrupt changes in the appearance and motion of the target.in this paper [44] ,Kwon and Lee proposed a robust tracking method by alleviating the motion smoothness constraint in abrupt motion .

The task of estimating the motion path of an object in successive video frames consists of two steps: object detection and object tracking. As shown in Section 2.8.1, the state-of-the-art approaches in object detection generate bounding boxes for detected objects in addition to the probability of detecting a given class of objects, such as cars, trucks, or persons.

2.8.3 Traffic anomaly detection

The abnormal events are instances which are different from the modelled regularities. where Anomaly detection in traffic videos has been recently gaining attention due to its importance in intelligent transportation systems.One of the most important, is the detection of abnormal events such as traffic accidents, violations and crimes.

one of the main outcomes of a video anomaly detection system is the real-time decision

making capability. Events such as traffic accidents, robbery, and fire require immediate counter actions to be taken in a timely manner. Despite its importance, a very limited body of research has focused on video anomaly detection techniques.

in this paper [28] studied detection of abnormal vehicle trajectories such as illegal U-turn, In [74], the author proposes a method based on multi-lane trajectories by comparing dense and free-flowing traffic conditions using Kalman filters. [37] utilizes Markov random fields (MRF) and hidden Markov model (HMM) in recognizing abnormal vehicle behavior atintersections.In [47], the temporal outliers on roads are detected based on historical similarity trends. [54] extracts human skeleton trajectory patterns, and hence is limited to only the detection of abnormalities in human behavior

2.9 Conclusion

In this chapter, we have showed the basic concepts of general neural networks.and we provided the background information of the fully connected neural network, then we have showed a subclass of neural network, Convolution Neural Network (CNN), which is widely used in image recognition. Later, we will describe a specific CNN based object detection model, YOLO [88], which is the most accurate real-time object detector.

Chapter 3

METHODOLOGY AND FRAMEWORK

In this Chapter 3 describes all the research methods that are adopted and used in this research. After that, we continues with the description of the network used.

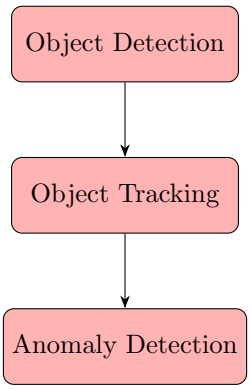


Figure 3.1: Block diagram of the our Methodology

3.1 Overview of traffic accident detection system

Traffic anomaly detection of videos has attracted great interests in computer vision field. Due to the scarcity of data and scene diversities. As shown in the flowchart below, image sequences (video Frames) are extracted from the video surveillance. In this Chapter, we suggested a vision based traffic accident detection system. This model first extracts the vehicles from the video image of CCTV camera, the second stage is a tracker in order to focus each car. The model then makes decision Anomaly / No.

3.2 Object Detection

3.2.1 Comparisons with State-of-the-art Detection Methods

Table 3.2 shows the performance comparison between DP-SSD, Faster R-CNN [51], YOLO [88], YOLO2 [13], SSD [49], DSSD [79], R-SSD [55], RefineDet [85] , SIN [82]and YOLO3 [62]. In addition to YOLO3, our DP-SSD512 outperforms all the other methods significantly.

Method	Input	mAP(%)	FPS
Faster R-CNN(VGG16)	-	72.70	11.23
YOLO	448 × 448	62.52	42.34
YOLOv2	416 × 416	73.82	64.65
YOLOv2 544 × 544	544 × 544	75.96	39.14
SSD300	300 × 300	74.18	58.78
SSD512	512 × 512	76.83	27.75
DSSD (ResNet-101)	321 × 321	76.03	8.36
R-SSD300	300 × 300	75.02	43.78
R-SSD512	512 × 512	77.73	24.19
RefineDet320	320 × 320	76.97	46.83
RefineDet512	512 × 512	77.68	29.45
SIN	-	77.26	10.79
YOLOv3	416 × 416	88.09	51.26
DP-SSD300	300 × 300	75.43	50.47
DP-SSD512	512 × 512	77.94	25.12

Figure 3.2: Detection results between different methods on UA-DETRAC dataset. In addition to YOLO3, DP-SSD512 outperforms all other methods on accuracy in real-time, and YOLO2 achieves the highest speed of 64.65 FPS. [86]

Both the YOLO2 models (YOLO2 416×416 and YOLO2 544×544) obviously improve the performance compared to YOLO by 11.3% and 13.44% of mAP respectively due to the elimination of fully connected layers and introduction of anchor boxes, but they are 4.12% and 1.98% lower than DP-SSD512, respectively. In addition, DP-SSD512 obtains an increased mAP of 1.11% compared to state-of-the-art SSD512 detection model, and the lower resolution network DP-SSD300 achieves 75.43%, which is worse than SSD512 but is 1.25% higher than SSD300. Although residual network is utilized by DSSD, there is still a 1.91% mAP degraded in accuracy with 76.03% mAP compared to DP-SSD512. Surprisingly, YOLOv3 achieves 88.09% in mAP, which beats all models and is better than DP-SSD512 by 10.15%, the reason is the good residual structure and multi-scale prediction method used in YOLO3. However, DP-SSD has fewer network parameters than YOLO3 (138M vs 214M), thus it is easy to train. As for speed, all models except Faster R-CNN , DSSD (ResNet-101) , and SIN in Table 4.1 can detect vehicles in real time, and YOLO2 outperforms all other methods with a processing speed of 64.65 FPS.

3.2.2 You only look once (YOLO)

Deep neural networks (DNNs) have shown prominent performance in the field of object detection. However, Object detection is an important task in many popular fields such as medical diagnosis. YOLO is one of the fastest object detection methods with good real time performance and high accuracy, created by Redmon, J., et al. [88]. and it has been improved since it was proposed, including YOLO-V1, YOLO-V2, YOLO-V3. we frame object detection as a regression problem to spatially separated bounding boxes and associated

class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation.[34]

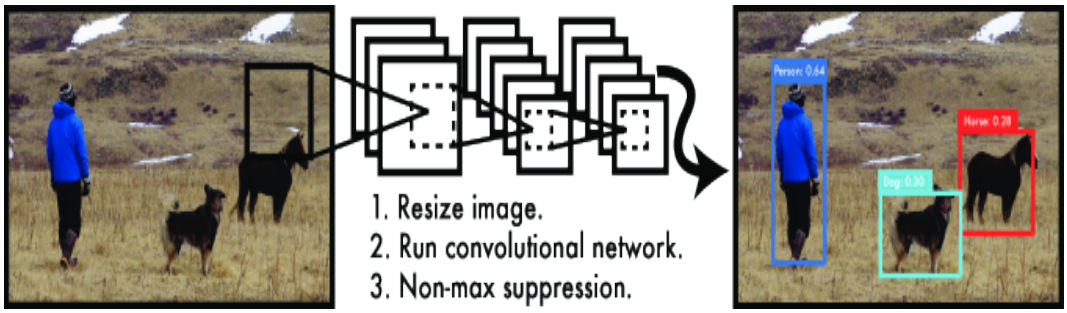


Figure 3.3: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence. Figure from:[35]

3.2.3 YOLO-V2

One enhancement of YOLO is the YOLOv2, which is a state-of the-art real-time detection system and faster than other detection systems across a variety of detection datasets. Yolov2 uses a custom deep architecture darknet-19, an originally 19-layer network supplemented with 11 more layers for object detection. Yolov2 still suffers from inability of detecting small objects due to the loss of fine-grained features. Instead of predicting bounding box offsets as SSD, YOLOv2 predicts location coordinates relative to the location of the grid cell normalized to (0,1]. [49] [13]

3.2.4 YOLO-V3

YOLOv3 is the latest contribution to the YOLO family model and is inspired by recent advances in object detection [51]. It is the latest variant of a popular object detection algorithm YOLO .YOLO V3 consists of the backbone network called Darknet-53, the upsampling network, and the detection layers called YOLO layers [88]. The backbone network of YOLO V3 is based on Darknet-53 to extract feature images from the input image. YOLO V3 uses a three-scale YOLO layer that is responsible for detecting objects of different scales. In the first YOLO layer, the grid resolution is 1/32 of the input image detecting large objects. The resolution of the last YOLO layer is 1/8, which is capable of detecting small objects.Its network structure is illustrated in Figure 3.4.

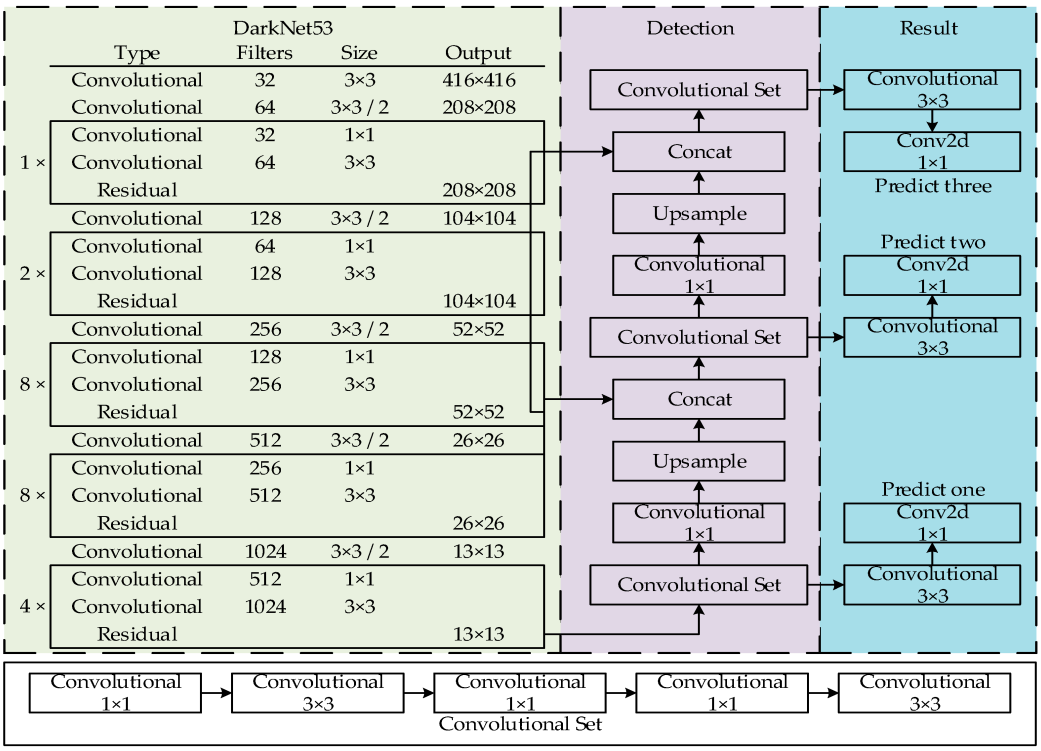


Figure 3.4: YOLOv3 network structure. Figure from [87]

3.2.5 Architecture Of YOLO

YOLO architecture is made up of 27 CNN layers, with 24 convolutional layers, followed by 2 Fully Connected layers and a final detection layer. The input image is first divided into a S×S grid. Next, B bounding boxes are defined in every grid cell, For each grid cell, there will only be one set of class scores C for all bounding boxes in that region. Each of the bounding boxes is characterized by a quintuple (x, y, w, h, c). The (x, y) coordinates are the center offset of the bounding box compared to the bounds of the grid cell. Moreover, x, y, w and h are normalized by the image width and height, and thus all between [0, 1].The value of Pr(Object) is 1 when a grid cell contains a center of a ground truth box, else its value is 0,where 0 means no overlap and 1 means that the predicted box is same as the ground truth.

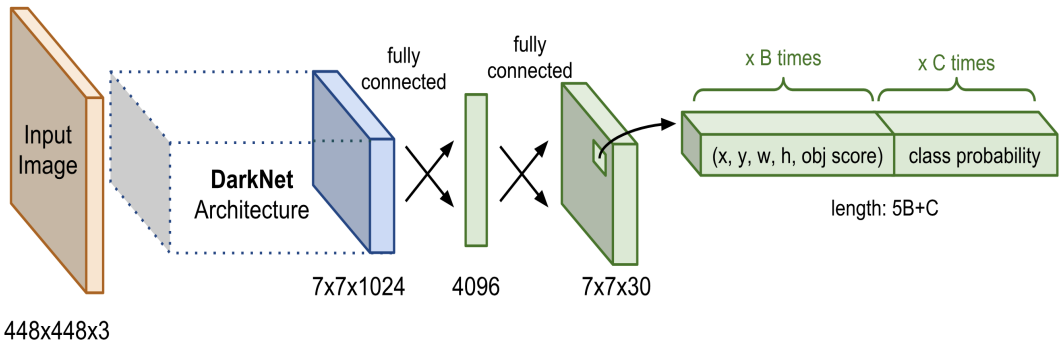


Figure 3.5: The Network Architecture Of YOLO.

YOLO divides an image in S×S grids. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores tell that how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.

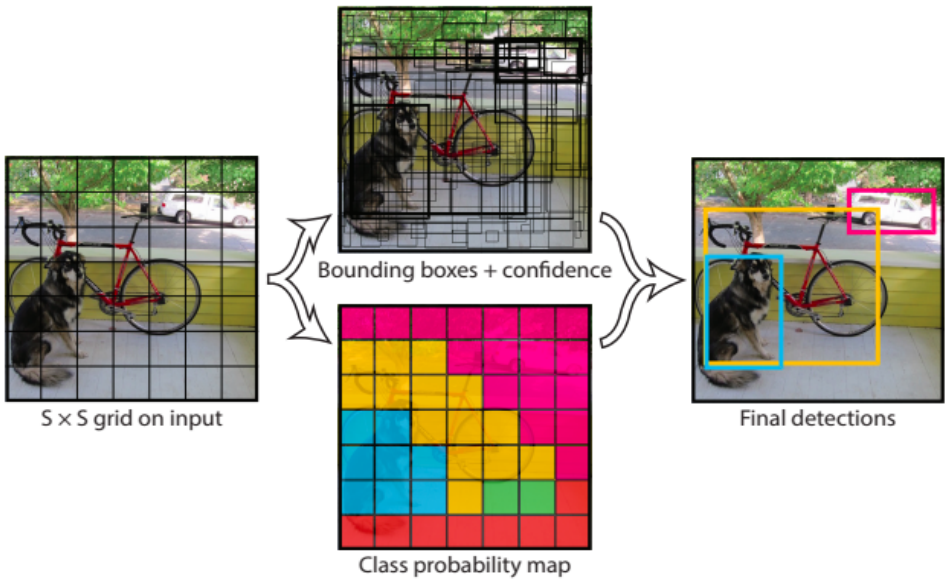


Figure 3.6: The YOLO detection model divides the input image into an S×S grid, and for each grid cell it predicts B bounding boxes,confidence for those boxes, and C class probabilities. Figure from: [26]

3.2.6 Grid cell

we are splitting our image into cells, we using a 15×15 grid. Each cell is responsible for predicting 5 bounding boxes, the 5 boxes called anchors

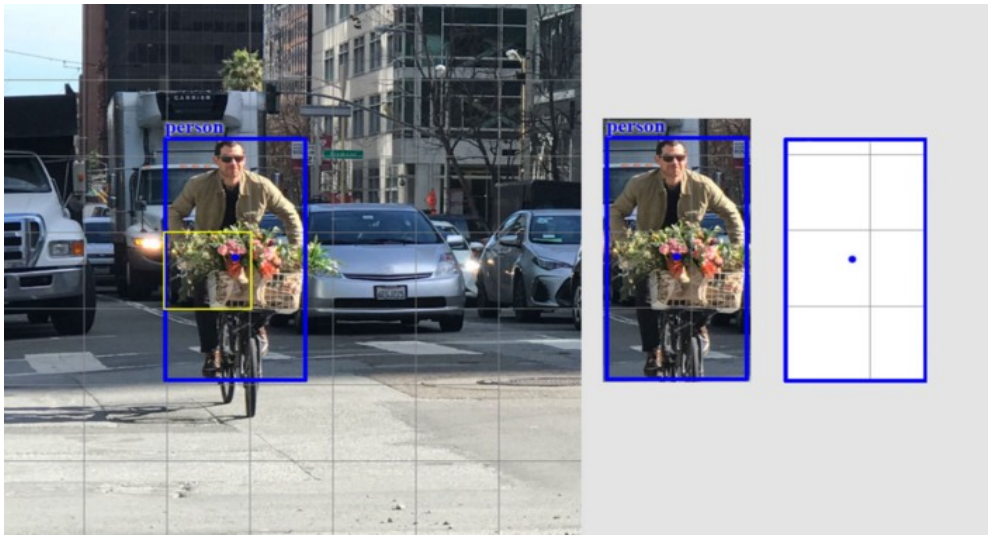


Figure 3.7: Yellow box is grid cell, blue box is the prediction box

3.2.7 Anchor Box

Anchor boxes and also called priors have a defined aspect ratio, and they tried to detect objects that nicely fit into a box with that ratio. Instead of choosing priors by hand, we run k-means clustering on the training set bounding boxes to automatically find good priors, which is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid). we want priors that lead to good IOU scores, which is independent of the size of the box. Thus for our distance metric we use:

$$d(box, centroid) = 1 - IOU(box, centroid) \tag{3.1}$$

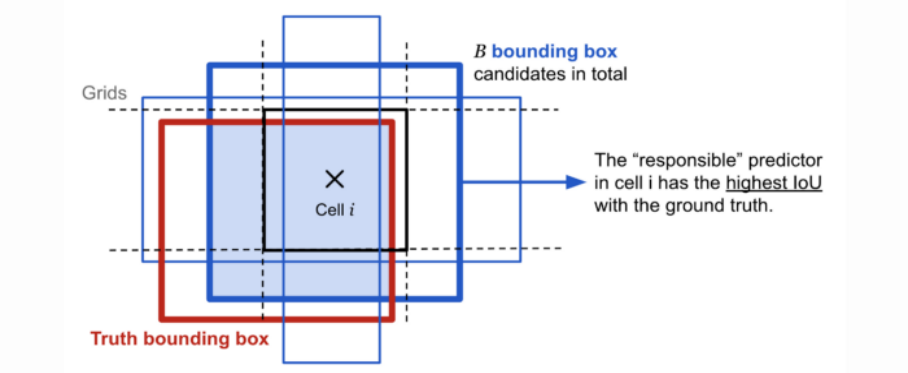


Figure 3.8: Anchor boxes.

We run k-means for various values of k and plot the average IOU with closest centroid, We choose k = 5

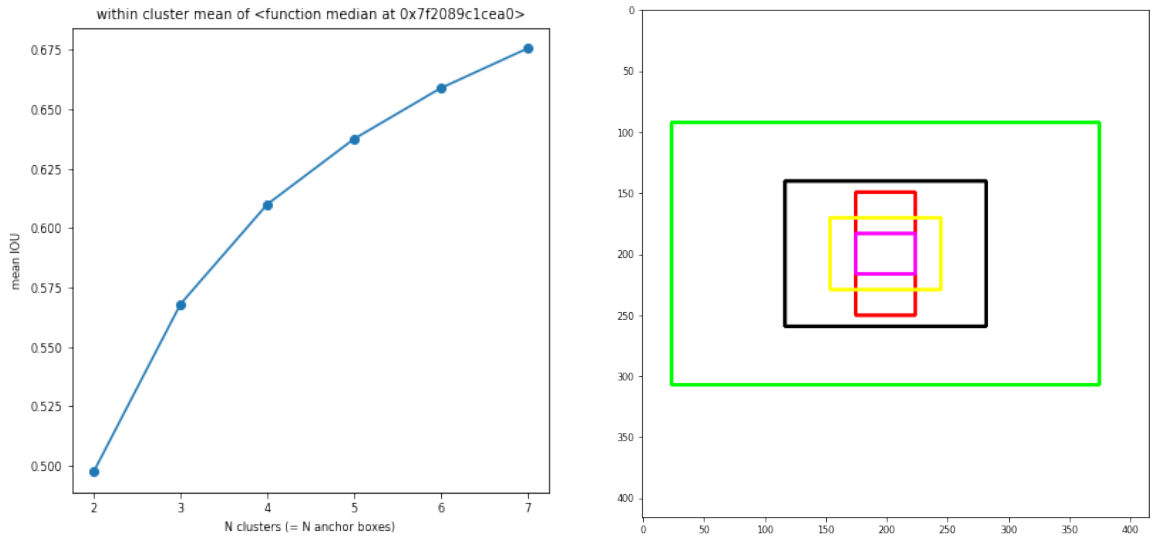


Figure 3.9: iou by the number of anchors And Anchors.

3.2.8 Intersection over Union (IoU)

Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. Intersection is the overlapping area between the predicted bounding box and ground truth, and union is the total area between both predicted and ground truth, represents a fraction between 0 and 1.

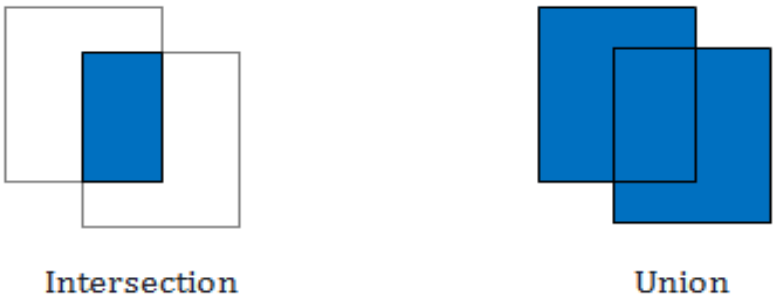


Figure 3.10: Illustration depicting the definitions of intersection and union.

This competition is evaluated on the average scores of each test images given intersection over union (IoU). The IoU of predicted area and ground truth area As we mentioned before in this part is calculated as:

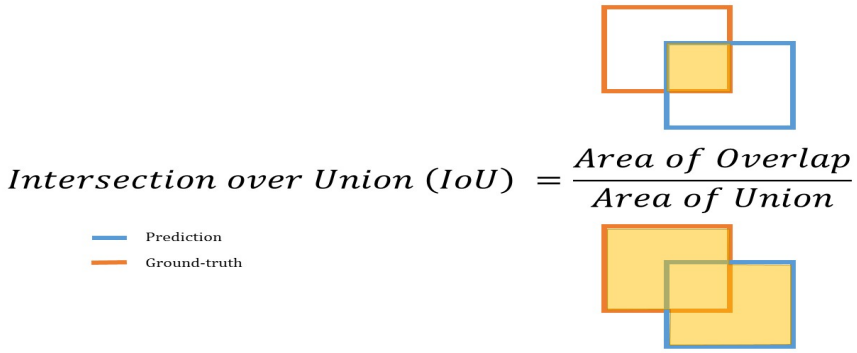


Figure 3.11: Intersection over Union (IOU) calculation diagram.

3.2.9 Network

YOLOv2 used darknet-19 as backbone

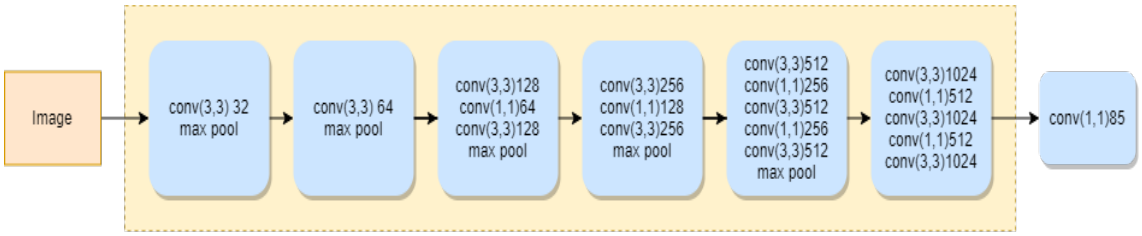
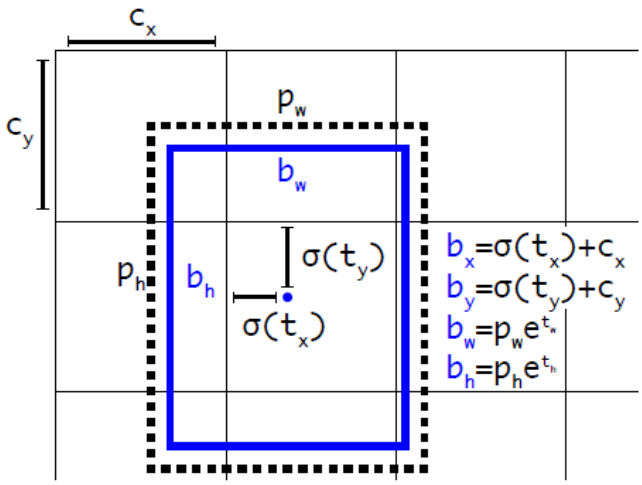


Figure 3.12: Flowchart of YOLO network

The network output GridxGridx85:
85 = Number of Anchors * (Number of classes + 5)
Number of classes = 12
5 = confidence(to), tx, ty, tw, th
If the cell is offset from the top left corner of the image by (cx, cy) and the bounding box prior has width and height pw, ph, then the predictions correspond to:



$$
\begin{aligned}
b_x &= \sigma(t_x) + C_x \\
b_y &= \sigma(t_y) + C_y \\
b_h &= p_h e^{t_h} \\
b_w &= p_w e^{t_w} \\
P_r(object) &= \sigma(t_o) \\
\sigma &: sigmoid
\end{aligned}
\tag{3.2}
$$

Figure 3.13: Bounding boxes with dimension priors and location prediction.[62]

3.2.10 Loss Function

We used yolov3 loss because gives us better results.The loss function composes of:

1. The Classification loss.
2. The Localization loss (errors between the predicted boundary box and the ground truth).
3. The confidence loss (the objectness of the box).

1. **Classification loss** If an object is detected, the classification loss at each cell is the Categorical Cross Entropy (CCE) of the class conditional probabilities instead of Binary Cross Entropy in yolov3 loss :

$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} CCE(p(c_{ij}), p(\hat{c}_{ij})) \tag{3.3}$$

Where:

Π_{ij}^{obj}= 1 if an object appears in cell $i = 0$, otherwise 0 .

$p(\hat{c}_{ij})$: conditional class probabilities for class c in cell i .

2. **Localization loss** The localization loss measures the errors in the predicted boundary box locations and sizes. We only count the box responsible for detecting the object.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} [(t_{xi} - \hat{t}_{xi})^2 + (t_{yi} - \hat{t}_{yi})^2] + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} [(t_{wi} - \hat{t}_{wi})^2 + (t_{hi} - \hat{t}_{hi})^2] \tag{3.4}$$

Where:

$\Pi_{ij}^{obj}=$ 1 if the j the boundary box in cell i is responsible for detecting the object, otherwise 0

λ_{coord}: make more weight in localization loss (default=5) .

3. **Confidence loss**

   If an object is detected in the box, the confidence loss (measuring the objectness of the box) is:

$$\sum_{i=0}^{s^2}\sum_{j=0}^{B}\Pi_{ij}^{obj}BCE(C_i,\hat{C}_i) \tag{3.5}$$

Where:

$\Pi_{ij}^{obj}=$ 1 if the j th boundary box in cell i is responsible for detecting the object, otherwise 0

$\hat{C}_i$: is the box confidence score of the box j in cell i.

BCE: is Binary Cross Entropy .

If an object is not detected in the box, the confidence loss is:

$$\lambda_{noobj}\sum_{i=0}^{s^2}\sum_{j=0}^{B}\Pi_{ij}^{noobj}\Pi_{ij}^{ignore}BCE(C_i,\hat{C}_i) \tag{3.6}$$

Where:

$\Pi_{ij}^{obj}=$ is the complement of Π_{ij}^{obj} .

$\hat{C}_i$: is the box confidence score of the box j in cell i.

BCE: is Binary Cross Entropy .

λ_{noobj}: weights down the loss when detecting background in the cell because there is more background cells compare to objects

Π_{ij}^{ignore}: If the bounding box prior is not the best but does overlap a ground truth object by more than some threshold (0.5) we ignore the prediction. So, equal 0 when IOU between ground truth object and prediction greater than 0.5 else equal 1

The total loss in the sum of classification, localization and confidence losses

$$
\begin{aligned}
F = & \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} [(t_{xi} - \hat{t}_{xi})^2 + (t_{yi} - \hat{t}_{yi})^2] \\
& + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} [(t_{wi} - \hat{t}_{wi})^2 + (t_{hi} - \hat{t}_{hi})^2] \\
& + \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} BCE(C_i, \hat{C}_i) \\
& + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{noobj} \Pi_{ij}^{ignore} BCE(C_i, \hat{C}_i) \\
& + \sum_{i=0}^{s^2} \sum_{j=0}^{B} \Pi_{ij}^{obj} CCE(p(c_{ij}), p(\hat{c}_{ij}))
\end{aligned}
\tag{3.7}
$$

3.2.11 Non-maximal suppression

YOLO can make duplicate detections for the same object. To fix this, YOLO applies non-maximal suppression to remove duplications with lower confidence. Non-maximal suppression In three steps:

1. Sort the predictions by the confidence scores.
2. Start from the top scores, ignore any current prediction if we find any previous predictions that have the same class and IoU> threshold (0.5) with the current prediction.
3. Repeat step 2 until all predictions are checked.

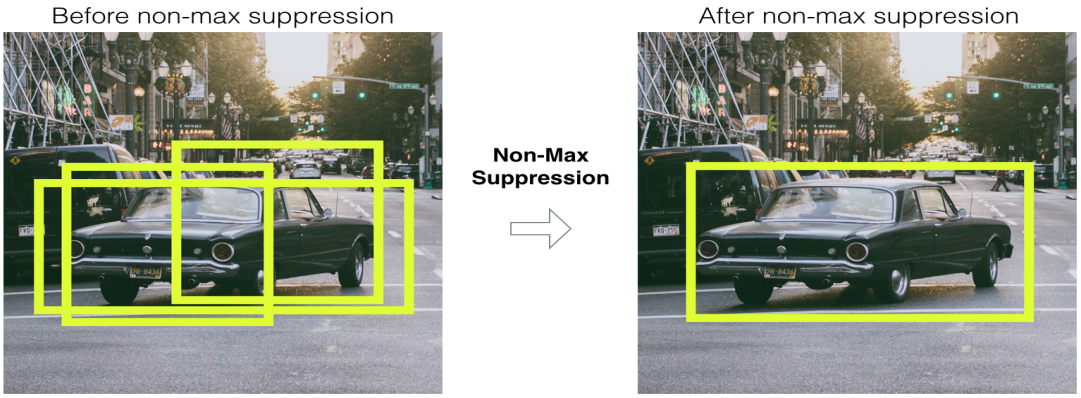


Figure 3.14: Non-Max Suppression From [12]

3.2.12 mAP(mean Average Precision)

To evaluate the performance of an object detector. Normally, we focus on the accuracy (mean average precision, mAP). In terms of accuracy, there are many different approaches used to evaluate the accuracy of a model or an algorithm for object detection, but mAP is the primary one. To understand mAP, we would need to first review precision and recall.

Precision measures how accurate is your predictions. i.e. the percentage of your predictions are correct. or it is the number of true positive divided by the sum of true positive and false positive,calculated by:

$$Precision = \frac{TP}{TP + FP} \tag{3.8}$$

Recall measures is defined by the ratio of the positive instances that are correctly detected by the detector, calculated by:

$$Recall = \frac{TP}{TP + FN} \tag{3.9}$$

Where :

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

Average precision (AP): The definition of AP is to computer the area under the precision-recall curve above. The average precision's value always falls between 0 and 1 based on the PR curve since Precision and recall are always between 0 and 1, too.

3.2.13 Datasets

We use combination of two datasets:

3.2.13.1 MIO-TCD Dataset

We performed our experiments on the MIOvision Traffic Camera Dataset (MIO-TCD) dataset [52] where the images are obtained from real-world traffic surveillance cameras. The dataset consists of total 786.702 images with 648.959 in the classification dataset and 137.743 in the localization dataset. taken from 8,000 different traffic surveillance cameras deployed all over the USA and Canada. The images cover a wide range of urban traffic scenarios and typically cover one or two traffic lanes captured from the side of the road with a wide-angle lens ,is a large dataset used for a 2017 CVPR challenge and for which CNN methods got accuracies of

up to 98% [78]. These images are taken at different times of the day and different times of the year. Additionally, the images are taken from a different angle, scale and resolution.This dataset aims to provide a rigorous benchmarking facility for training and testing existing and new algorithms for the classification and localization of moving vehicles in traffic scenes.

The dataset is divided in two parts : the "classification challenge dataset" and the "localization challenge dataset". The total number of training images in the dataset for the classification task is 648,959 with 11 different classes namely: articulated truck (1.98%), background (30.68%), bus (1.98%), bicycle (0.44%), car (49.96%), motorcycle (0.38%), nonmotorized vehicle (0.34%), pedestrian (1.2%), pickup truck (9.76%), single unit truck (0.98%) and work van (1.86%). Figure 3.15 shows a few examples of images taken from the dataset.

Figure 3.15: Sample images from the MIO-TCD dataset.[1]

3.2.13.2 Urban dataset

The dataset Contains8794containing one (or more) foreground object(s) with one of the following 10 labels : Pedestrian, Particular, Cab, Motorcycle, Bus, Truck, Minivan, Bicycle, scooter,Articulated truck. Split it into (7680 training, 1114 validation) [2]

3.2.13.3 Data preprocessing

We resize the input images to 480 with keeping the aspect ratio of the image(letterbox), And we use different types of data augmentation

1. **Data augmentation**

   Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, a data-space solution to the problem of

limited data, such as image classification, to virtually enlarge the training dataset size and avoid overfitting.[80]

2. **Horizontal flip**

   In our proposed method we need to train our model but the problem is that we are having only small size of training which are not sufficient for training a good-fit model that is generalized. To solve this problem we can apply various kind of preprocessing methods. horizontal flipping is a technique of Data augmentation and it is commonly used to train large neural networks.

   we can flip images horizontally and vertically. Some frameworks do not provide function for vertical flips. But, a vertical flip is equivalent to rotating an image by 180 degrees and then performing a horizontal flip. Below are examples for images that are flipped.

Figure 3.16: flipping image

3. **Add noise**

   Adding noise means that the network is less able to memorize training samples because they are changing all of the time, resulting in smaller network weights and a more robust network that has lower generalization error. The noise means that it is as though new samples are being drawn from the domain in the vicinity of known samples, smoothing the structure of the input space. This smoothing may mean that the mapping function is easier for the network to learn, resulting in better and faster learning. [11]

Figure 3.17: Adding noise

4. **Random Shear**: shifts one part of the image like a parallelogram

Figure 3.18: Sheared image

5. **Random Translate**: In translation, the image is moved either along the x-axis or y-axis.

Figure 3.19: translated image

6. **Random scale**: In scaling or resizing, the image is resized to the given size e.g. the width of the image can be doubled.

Figure 3.20: scaling image

7. **Hue, Saturation and Value (HSV)**: HSV augmentation is a method of decomposing an image in to colour attributes and it is randomly adjust the brightness and contrast of gray scale images by using basic math operations.

Figure 3.21: RGB image transformed to HSV color space

3.2.14 Training

We train the network with starting learning rate = 0.001 and divided by 10 when the loss stop decreasing, with batch size (16)

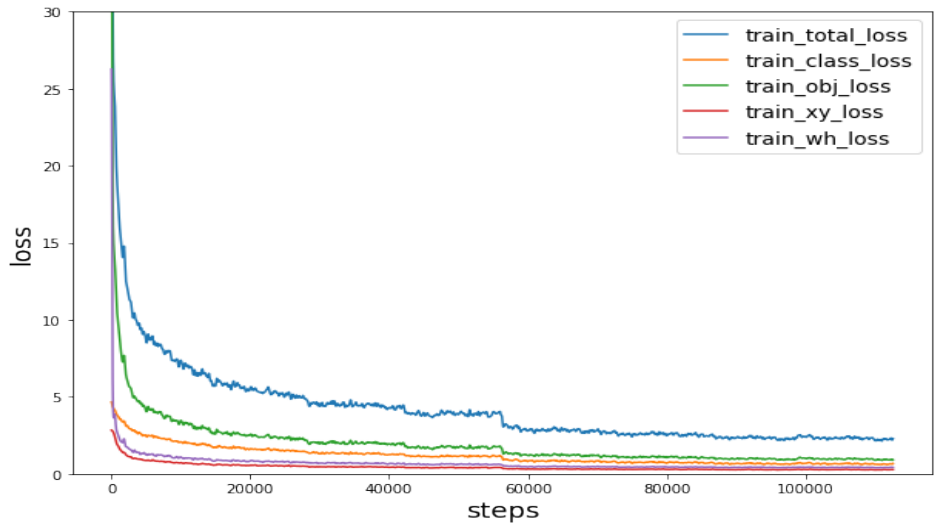


Figure 3.22: training losses

3.2.15 Evaluation Metrics

Average Precision of each category :

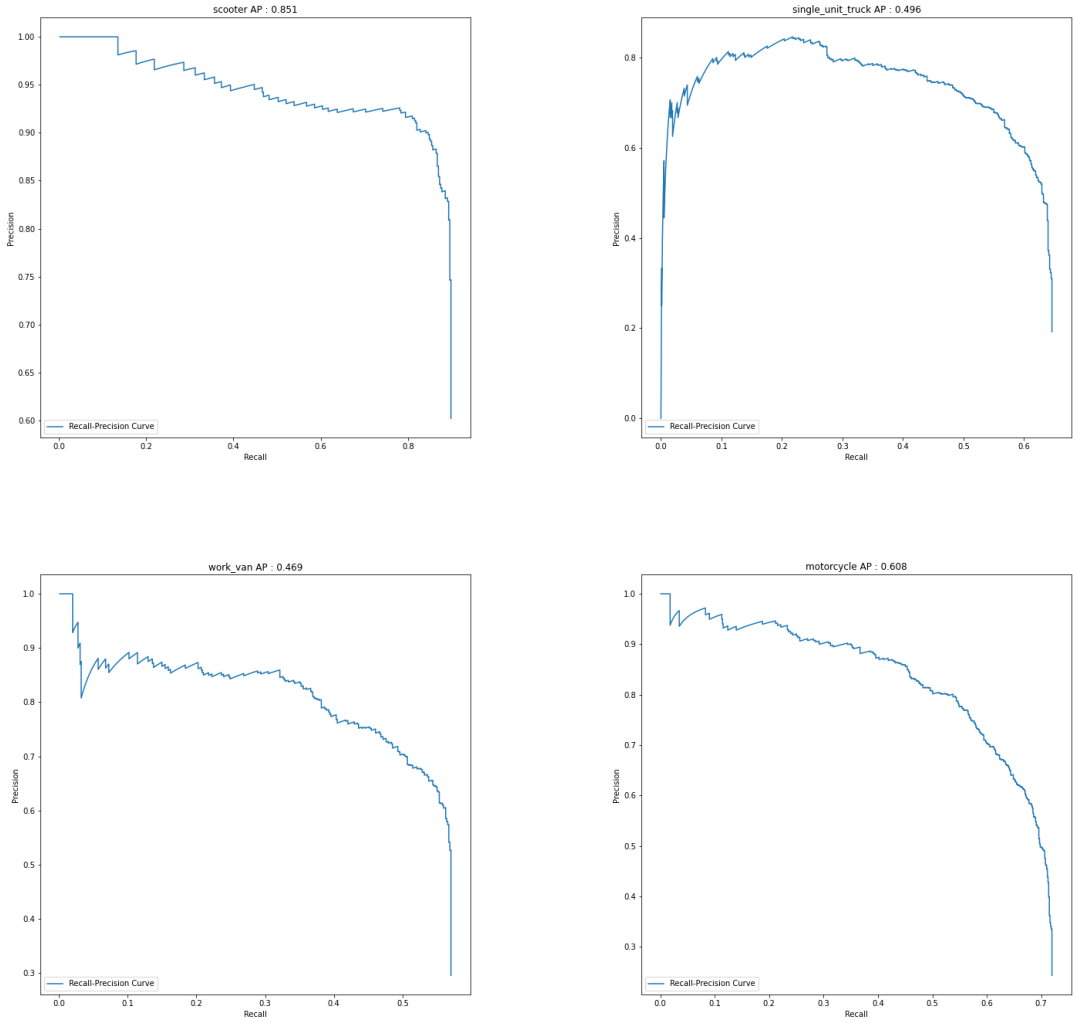
- **we evaluate the model based on IoU threshold** = 0.5.

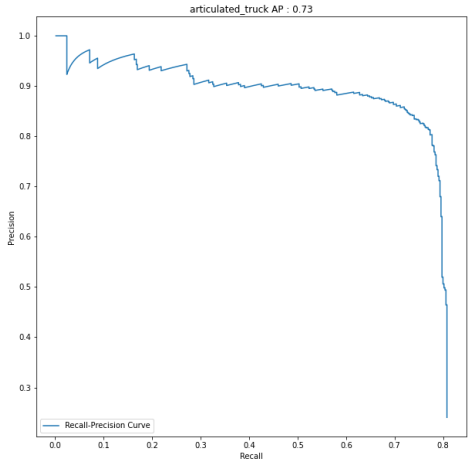
- **Mean Average Precision (MAP)@0.5:** 0.605

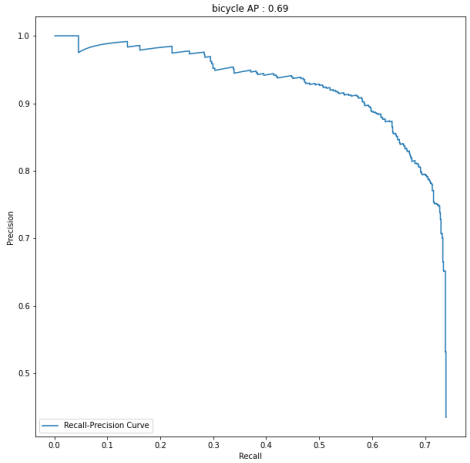
Articulated Truck	Bicycle	Bus	Car	Motorcycle	Motorized Vehicle	Non-motorized Vehicle	Pedestrian	Pickup Truck	Single Unit Truck	Work Van	Scooter
0.73	0.68	0.77	0.80	0.60	0.28	0.35	0.37	0.81	0.49	0.46	0.85

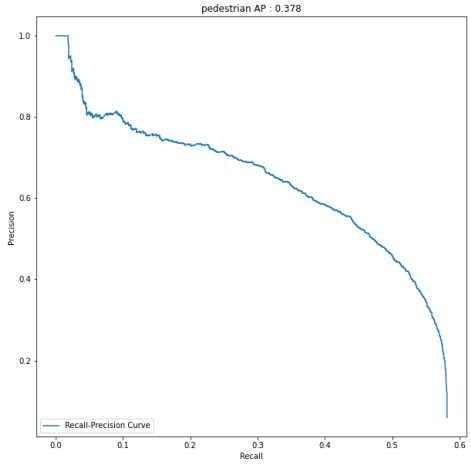
Figure 3.23: Evaluation Metrics

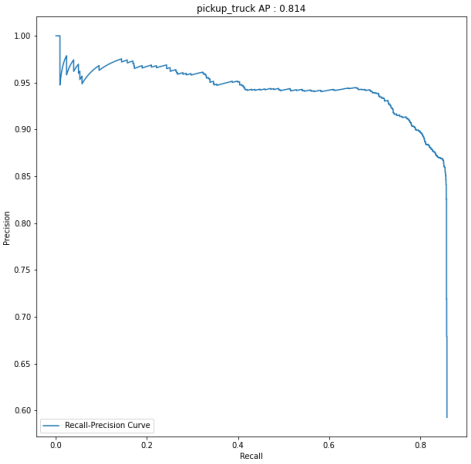
where Shown in Fig.bellow are the results for this experiment.

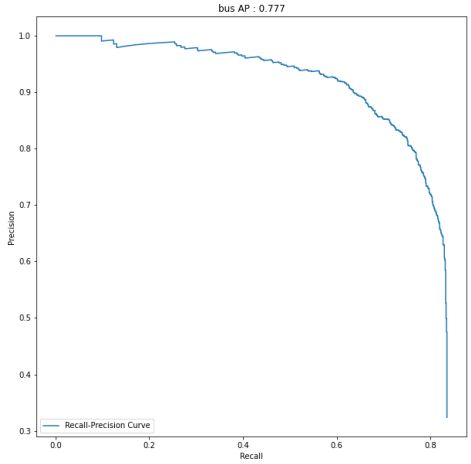


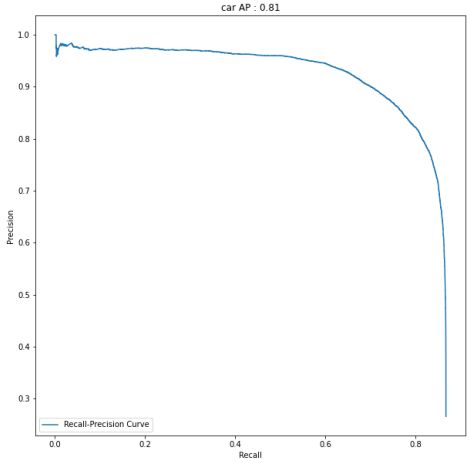

articulated_truck AP : 0.73
Precision
Recall
Recall-Precision Curve


bicycle AP : 0.69
Precision
Recall
Recall-Precision Curve


pedestrian AP : 0.378
Precision
Recall
Recall-Precision Curve


pickup_truck AP : 0.814
Precision
Recall
Recall-Precision Curve


bus AP : 0.777
Precision
Recall
Recall-Precision Curve


car AP : 0.81
Precision
Recall
Recall-Precision Curve

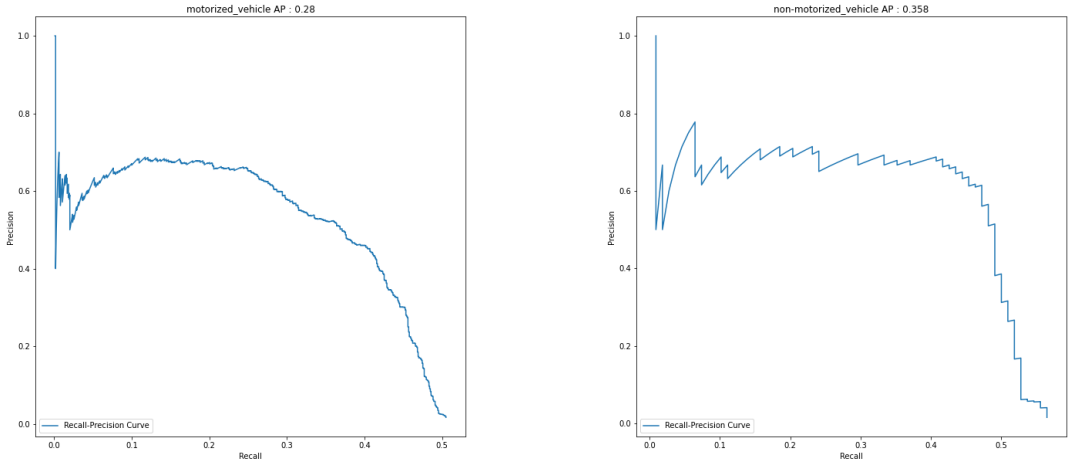


Figure 3.24: Precision-Recall curve of each category

3.3 Object Tracking

The aim of an object tracker is to generate the trajectory of an object over time by locating its position in every frame of the video [83]. But tracking has two definition one is in literally it is locating a moving object or multiple object over a period of time using a camera.

Due to the lack of tracking datasets, we wanted to use a method that does not require a training stage, so we used Optical Flow

3.3.1 Optical flow

Optical flow is the motion of objects between consecutive frames of sequence, caused by the relative movement between the object and camera.By assuming that pixel intensities of an object are constant between consecutive frames [24]. [24]

Pros: This method can get the complete movement information and detect the moving object from the background relatively well.

Cons: large quantity of calculation, sensitivity to noise, and poor anti-noise performance, make it not suitable for real-time implementation [5].

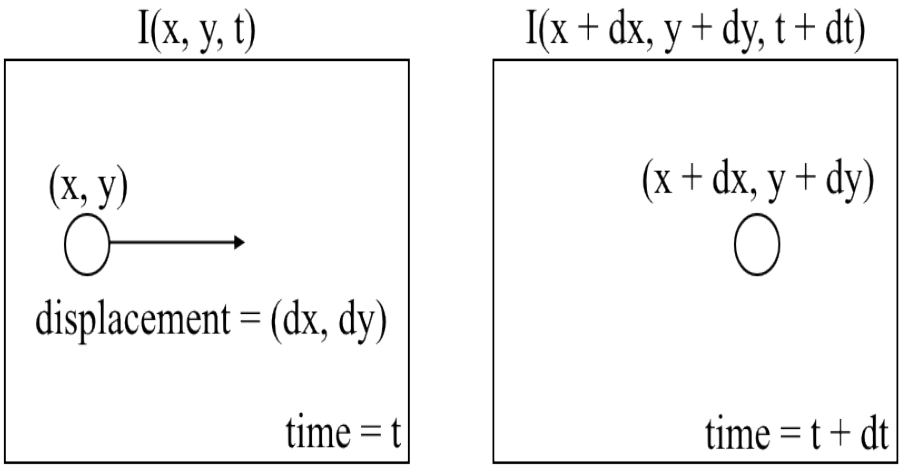


Figure 3.25: Optical flow problem

Where between consecutive frames we can express the image intensity(I)as a function of space*(x,y)* and time *(t)*, if we take the first image *I(x,y,t)* and move its pixels by *(dx,dy)* over t time, we obtain the new image *I(x+dx,y+dy,t+dt)* [24].

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{3.10}$$

we take the Taylor Series Approximation of the right side of the equation and remove common terms.

$$\begin{aligned} I(x + dx, y + dy, t + dt) &= I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \dots \\ \Rightarrow \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt &= 0 \end{aligned} \tag{3.11}$$

we divide by dt to derive the optical flow equation:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \tag{3.12}$$

where *u=dx/dt and v=dy/dt.*

dI/dx,dI/dy, and *dI/dt* are the image gradients (directional change in the intensity or color in an image) along the horizontal axis, the vertical axis, and time. Hence, solving *u(dx/dt)* and *v(dy/dt)* to determine movement over time. You maynotice that we cannot directly solve the optical flow equation for u and v since there is only one equation for two unknown variables.

We will implement some methods such as the Lucas-Kanade method to address this issue.

1. **Dense optical flow**

   Dense optical flow,gives the flow vectors of the entire frame (all pixels) - up to one flow vector per pixel. Dense optical flow has higher accuracy at the cost of being slow/computationally expensive.

2. **Sparse optical flow**

   Sparse optical flow selects a sparse feature set of pixels to track its velocity vectors (motion). The extracted features are passed in the optical flow function from frame to frame to ensure that the same points are being tracked.

Since we will track specific objects, we use sparse optical flow.Before using the tracking, we need to get good features to track

3.3.2 Shi-Tomasi method

Shi-Tomasi Corner Detection was published by J.Shi and C.Tomasi in their paper 'Good Features to Track'. Here the basic intuition is that corners can be detected by looking for significant change in all direction.

We consider a small window on the image then scan the whole image, looking for corners.

A Shi-Tomasi detector is a fully based on the detector of Harris . only, there is one difference in a " criteria of selection " recognize this detector on the detector of Harris . It works very good where even the Harris detector fails. hence a little change that this detector did to the Harris detector.[69]

Figure 3.26: good Features To Track

3.3.3 Lucas-Kanade (LK):Sparse Optical Flow

Lucas and Kanade proposed an effective technique to estimate the motion of interesting features by comparing two consecutive frames [38].The Lucas-Kanade method works under the following assumptions:

- Two consecutive frames are separated by a small-time increment (dt) such that objects are not displaced significantly (the method work best with slow-moving objects).
- A frame portrays a "natural" scene with textured objects exhibiting shades of gray that change smoothly.

First, under these assumptions, we can take a small 3x3 window (neighborhood) around the features detected by Shi-Tomasi and assume that all nine points have the same motion.

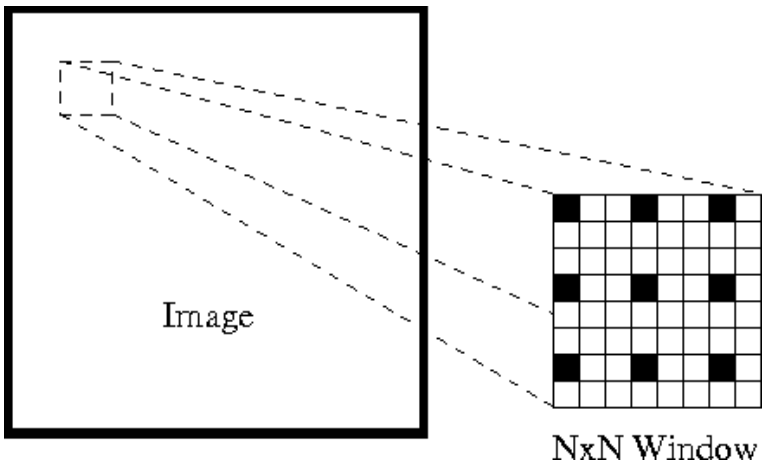


Figure 3.27: Optical flow is estimated for the black pixels

represented as:

$$I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \tag{3.13}$$
$$I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \tag{3.14}$$
$$\vdots$$
$$I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \tag{3.15}$$

where q_1,q_2,...,q_n denote the pixels inside the window (e.g.n = 9 for a 3x3 window) and I_x (q_i),I_y (q_i), and I_t (q_i) denote the partial derivatives of image I with respect to position (x,y) and time t, for pixel q_i at the current time.

This is just the Optical Flow Equation (that we described earlier) for each of the n pixels, since there is tow variables and 9 equation which can be solve.

3.3.4 Method

We run object detector every 20 second to detect objects and then we detect features to track using Shi-Tomasi method and we run sparse Lucas-Kanade method to track the features and update bounding boxes and repeat the operation.

In Our work, we get at an extremely high frame rate (55 FPS).

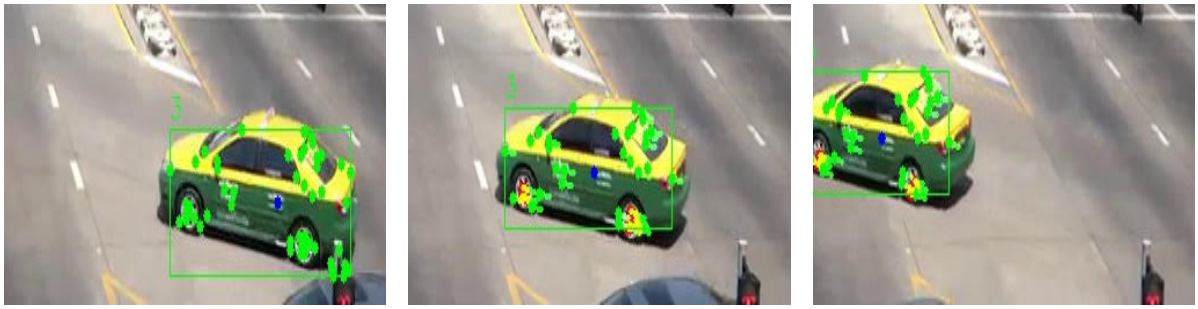


Figure 3.28: Object tracking method

3.4 Anomaly Detection(Accident Detection)

3.4.1 Method

We need to create a classifier that takes video as input and output Accident/No accident.

1. **Extract videos:** We can pass a video every 3-5 seconds to the accident classification model to classifier the accidents but this approach takes more power of the system if there are no objects (car, motor, truck...) in the video. Instead we use the object detector and object tracking to pass the videos that can contain accidents, and instead of pass the full resolution of the video we use object detector to pass a mini videos that contain the objects [7]

2. **Network:** Because the input is a sequence of frames (video), we use at first VGG16 to extract the features from the frame then we pass the output of convolutions to LSTM layers and to Dense layers to output the classification accident/no accident.

   We use 5 frames of each video, we use Time Distributed layer by keras to pass the frames into the same convolution block to make sure we extract the same features of each frame

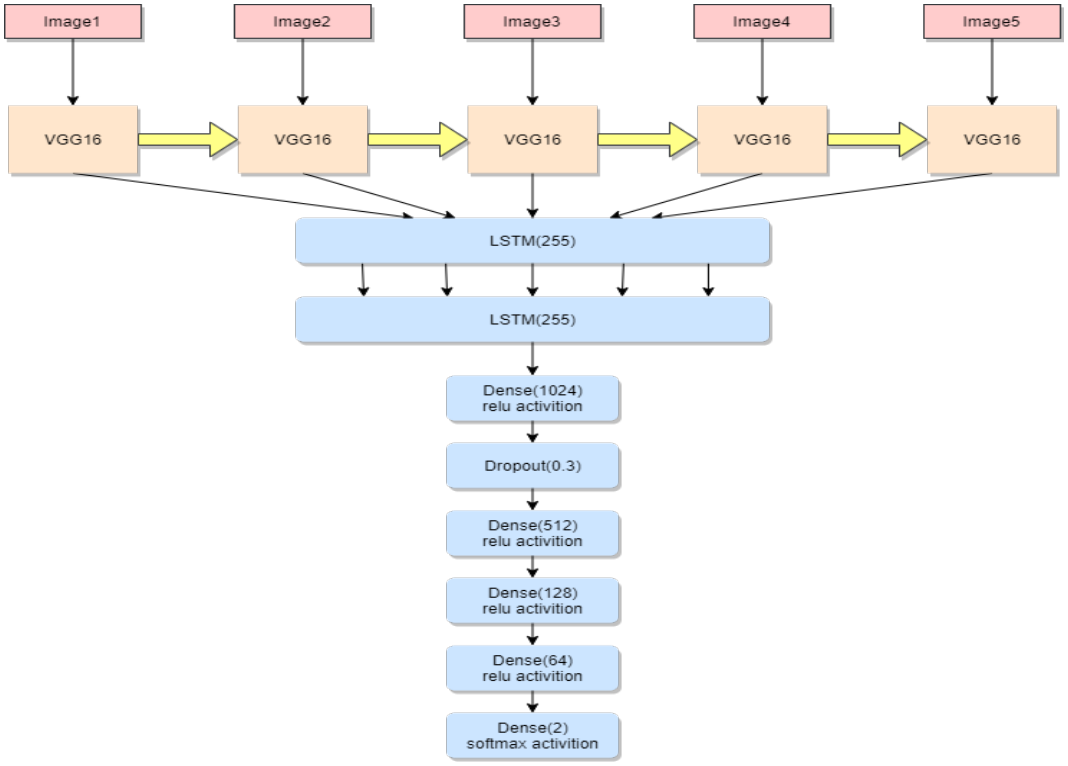


Figure 3.29: Network

Note: The Convolution Block is the same block distribute for each input image

VGG16(without Fully Connected layers)

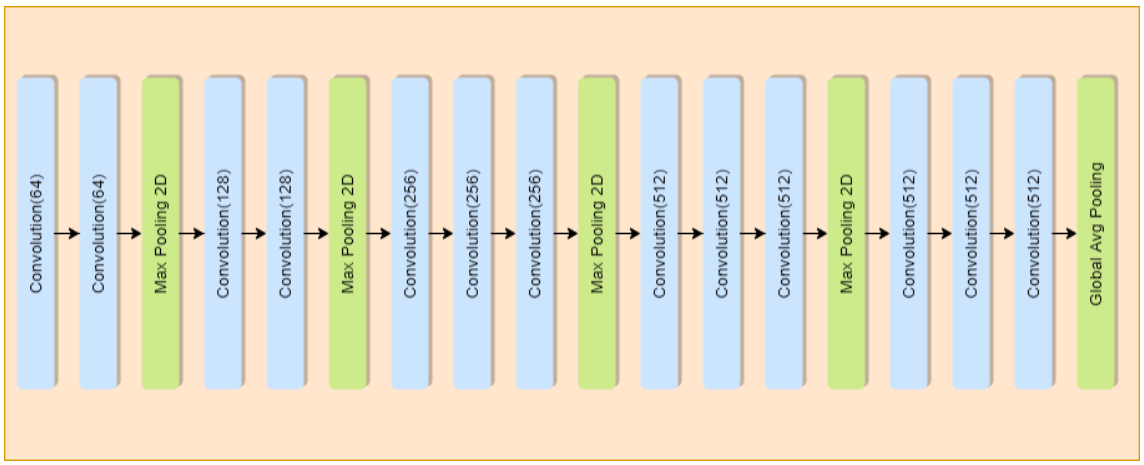


Figure 3.30: VGG16(without Fully Connected layers)

3. **Loss Function**

   We use Categorical Cross Entropy (CCE) for the loss.

$$Loss = CCE(y, y') \tag{3.16}$$

   Where:

   y: is true value 1 is accident and 0 no accident

   y': is prediction value.

3.4.2 Dataset

1. **CADP dataset**

   consists of 1,416 video segments collected from YouTube, the videos are different sizes.

   But the dataset contains video clips from the camera in the car that had been in an accident, and we don't need it and some videos the bounding box of the accident don't contains the accident. So, we make our dataset by the same videos from YouTube and add some others.

2. **Labels The Dataset**

   We develop a program that can help us label the videos by the start and end of the accident and the bounding box of the accident.

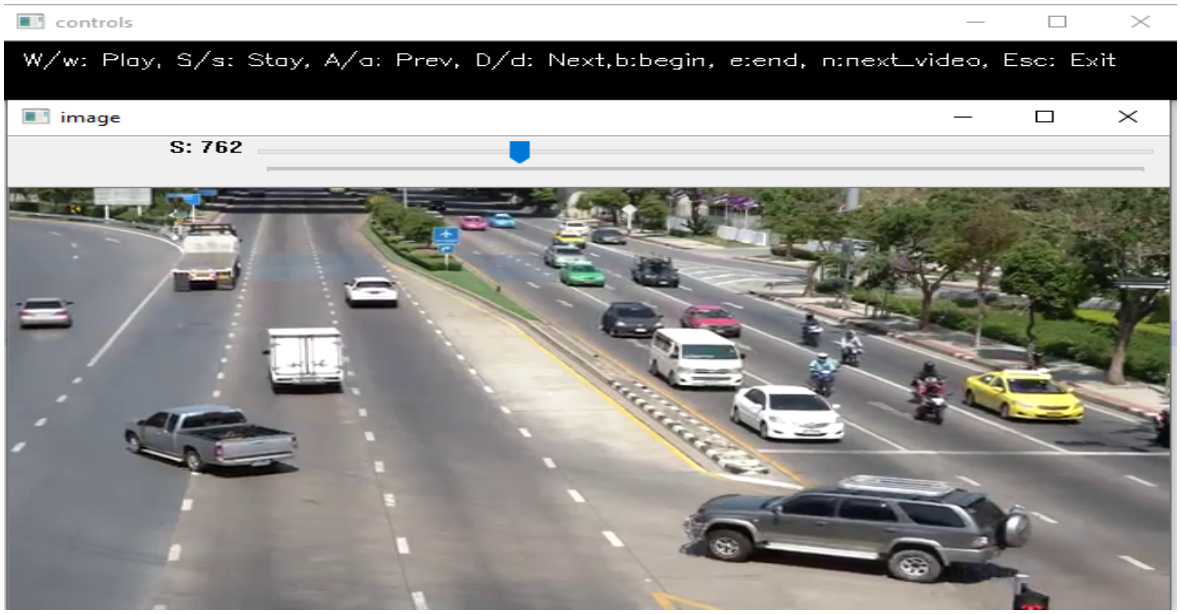


Figure 3.31: label program

(a) begin accident **(b)** accident **(c)** end accident

Figure 3.32: Car crash detection with the proposed method

We label 1230 accident videos. For no accident videos we selected 65 videos from YouTube and we make crops and every 10-30 seconds we record mini videos from the crops

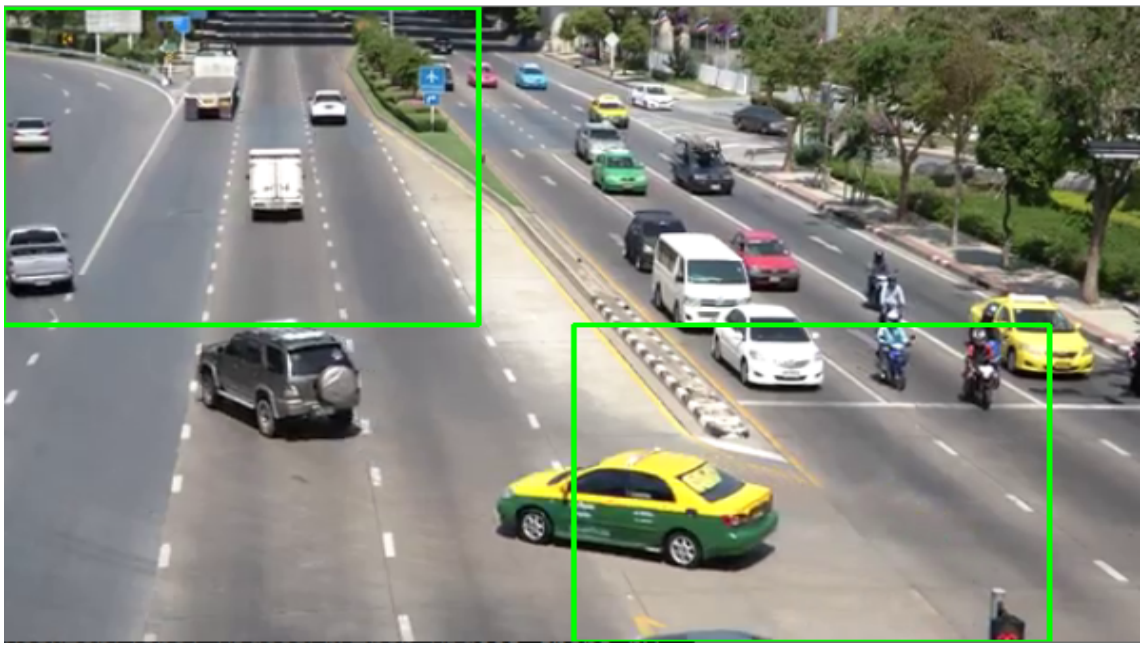


Figure 3.33: label program

Figure 3.34: Frame (On the left crop1 and on the right crop2).

We chose 96 videos from each class for validation.

Data preprocessing

We need from each video 5 frames to pass through the Network. We selected the distributed frames from the entire video, and we resize it to (224,224)

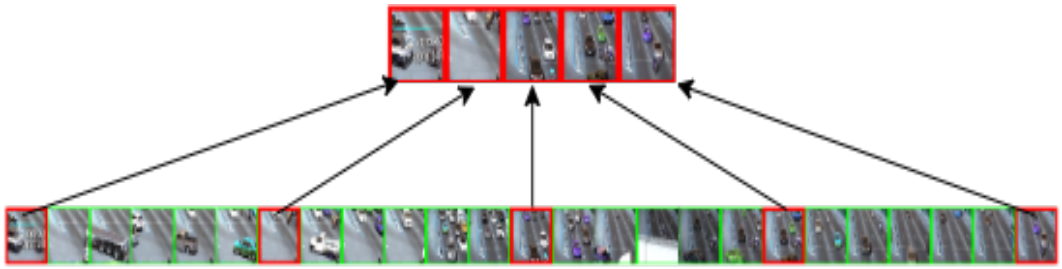


Figure 3.35: distributed frames

Data augmentation

We use the same data augmentation with same value in each 5 frames, hsv(hue, saturation, value, also known as or HSB [hue, saturation, brightness]), horizontal flip,shear, scale,add noise.

3.4.3 Training

We train the network for 30 epochs with batch size = 32 and we use Adam optimizer with learning rate = 0.0001,and we use transfer learning from vgg16 imagenet dataset.

3.4.4 Evaluation

We use accuracy as metric to evaluate the model, and archive 94.5% accuracy on the validation data

Anomaly Detection Speed 50 Frames per Second

3.5 Challenges

1. Our main challenge was to gather accident videos and manually categorize videos into accident and non-accident frames.
2. To design a deep convolutional neural networks model for this project.
3. Limited hardware resorces like GPU's.

3.6 Experimental Analysis

The implementation proposed in this thesis is executed in:

1. google Colaboratory
2. OS: Linux
3. CPU: 2x Intel(R) Xeon(R) CPU @ 2.20GHz
4. RAM: 12.7 GB
5. GPU: Nvidia K80/T4 16GB
6. Disk: 100 GB
7. maximum 12 hours in session

3.7 Conclusion

In this section, We proposed a model based on Anomaly detection in road traffic using regression based algorithm called YOLO(you only look once) algorithm on the sample vehicle datasets and the vehicle detection process has been successfully .

General Conclusion and Future Scope

In this thesis, we propose and develop a framework to determine whether an abnormal event has occurred in a traffic environment. One of the most applicable usages of this system is Traffic surveillance by CCTV cameras in an urban environment. This system is divided to three stages, in the first one, the framework uses a state-of-the-art method (YOLO) to detect objects (vehicles and pedestrians) in giving image, It takes a frame in the input, applies YOLO algorithm, and produces as output, a bounding box associated with its class for every object. The object tracking stage takes that output along with its frame and tracks the objects over the next frames. Finally, the anomaly detection stage utilizes these information to determine the possibility of anomaly occurrence in the scene using a well-trained classification model.

Through the aid of this monitoring system, we can lower down the loss of lives caused by accidents. Such kind of systems can also be adopted for exclusive category of anomalies, like over speed incidences and red-light violations. This feature opens the door for an optimal and efficient anomaly prediction to curb traffic problems in the near future.

Bibliography

[1] http://podoce.dinf.usherbrooke.ca/challenge/dataset/.

[2] F. A. and . C. F. (2019.April). *Vehicle and pedestrian video-tracking with classification based on deep convolutional neural networks. In 2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA) (pp. 1-5). IEEE.*

[3] M. Ahmed and M. R. I. Abdun Mahmood. A survey of anomaly detection techniques in financial domain.

[4] M. Ahmed, A. N. Mahmood, and J. Hu. A survey of network anomaly detection techniques.

[5] S. A.K. and G. Patil. Efficient background subtraction and shadow removal technique for multiple human object tracking. international journal. 2013.

[6] S. ALBAWI, T. A. Mohsmmed, and S. AL-Zawi. Understanding of a convolutional neural network.

[7] Arceda, V. E. M., Riveros, and E. L. (2018, october). fast car crash detection in video. in 2018 xliv latin american computer conference (clei) (pp. 632-637). ieee.

[8] A. Banharnsakun and S. Tanathong. A hierarchical clustering of features approach for vehicle tracking in traffic environments.

[9] beijing. A surveillance video based anomaly detection technology for intelligent transportation. April 01 2015.

[10] A. Bhandare, M. Bhide, P. Gokhale, and R. Chandavarkar. Applications of convolutional neural networks.

[11] by Jason Brownlee on December 12 2018 in Deep Learning Performance. Train neural networks with noise to reduce overfitting.

[12] P. by Jędrzej Świeżewski. https://appsilon.com/object-detection-yolo-algorithm/ .ph.d.22 may, 2020.

[13] A. R. Caballo and C. J. Aliac. Yolo-based tricycle detection from traffic video.

[14] A. Chadha, A. Abbas, Y. Andreopoulos, and S. M. IEEE. Video classification with cnns: Using the codec as a spatio-temporal activity sensor.

[15] P. K. Chan and M. V. Mahoney. Modeling multiple time series for anomaly detection.

[16] V. Chandola, A. Banerjee, and V. Kumar. University of minnesota, anomaly detection : A survey. 2009.

[17] S. Chawla. Deep learning for anomaly detection : A survey. January 24 2019.

[18] H. Chu, X. Liao, P. Dong, Z. Chen, X. Zhao, and J. Zou. An automatic classification method ofwell testing plot based on convolutional neural network (cnn).

[19] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection.

[20] A. Dey. Machine learning algorithms: A review.

[21] D.Nagajyothi and P.Siddaiah. Speech recognition using convolutional neural networks.

[22] Dr.M.R.NarasingaRao, V. V. Prasad, P. Teja, Md.Zindavali, and O. Reddy. Aa survey on prevention of overfitting in convolution neural networks using machine learning techniques.

[23] J. Du. Understanding of object detection based on cnn family and yolo.

[24] C. en Lin a 2019. Introduction to motion estimation with optical flow available online: https://nanonets.com/blog/optical-flow/ (accessed on 18/9/2020).

[25] A. G. Enyinna Nwankpa, Winifred Ijomah and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning.

[26] R. B. G. et al. Rich feature hierarchies for accurate object detection and semantic segmentation available online : https://arxiv.org/pdf/1311.2524.pdf.

[27] U. Fiore, F. Palmieri, A. Castiglione, and A. D. Santis. Network anomaly detection with the restricted boltzmann machine.

[28] Z. Fu, W. Hu, and T. Tan. Similarity based vehicle trajectory clustering and anomaly detection.

[29] P. Galeano, D. Peña, and R. S. Tsay. Outlier detection in multivariate time series via projection pursuit.

[30] D. Gour and A. Kanskar. Optimized-yolo: Algorithm for cpu to detect road traffic accident and alert system.

[31] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liub, X. Wangb, L. Wangb, G. Wangb, J. Caic, and T. Chenc. Recent advances in convolutional neural networks.

[32] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. in proceedings of the ieee conference on computer vision and pattern recognition. pages 770–778, 2016.

[33] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities.

[34] W. F. M. IEEE), L. WANG, and P. REN. Tinier-yolo: A real-time object detection method for constrained environments.

[35] R. J. and D. S. G. R. F. A. You only look once: Unified, real-time object detection. in proceedings of the ieee conference on computer vision and pattern recognition (2016).

[36] J. A. K and D. R. C. Algorithms for clustering data[m].new jersey:prentice-hall,1988.

[37] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi. Traffic monitoring and accident detection at intersections.

[38] L. B. D. Kanade.T. (1981). an iterative image registration technique with an application to stereo vision.

[39] E. R. Kandel, J. H. Schwartz, and T. M. J. et al. Principles of neural science, volume 4. mcgraw-hill new york.2000.

[40] H. Kaur, G. Singh, and J. Minhas. A review of machine learning based anomaly detection techniques.

[41] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. A survey of the recent architectures of deep convolutional neural networks.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks.

[43] C. R. Kulkarni and A. B. Barbadekar. Text detection and recognition: A review.

[44] J. Kwon and K. M. Lee. Wang-landau monte carlo-based tracking methods for abrupt motions.

[45] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning.

[46] C. Li and T. Hua. Human action recognition based on template matching.

[47] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal outlier detection in vehicle traffic data.

[48] Y. Li and H. Wu. A clustering method based on k-means algorithm.2012 international conference on solid state devices and materials science.

[49] W. Liu, D. Anguelov, C. S. Dumitru Erhan, S. Reed, C. Fu, and A. C. Berg. Ssd: Single shot multibox detector.

[50] M. M. Lopez and J. Kalita. Deep learning applied to nlp.

[51] M. Loukadakis, J. Cano, and M. O'Boyle. Accelerating deep neural networks on low power heterogeneous architectures.

[52] Z. Luo, F. B.Charron, C. Lemaire, J. Konrad, S. Li, A. Mishra, A. Achkar, J. Eichel, and P.-M. Jodoin. Mio-tcd: A new benchmark dataset for vehicle classification and localization. in press at ieee tip, 2018.

[53] M. Mdini. Anomaly detection and root cause diagnosis in cellular networks.

[54] R. Morais, V. Le, T. Tran, B. Saha, M. Mansour, and S. Venkatesh. Learning regularity in skeleton trajectories for anomaly detection in videos.

[55] J. J. P. H. K. N. Enhancement of ssd by concatenating feature maps for object detection. arxiv,2017; arxiv:1705.09587v1.

[56] A. Ng and M. K. Cowan. Machine learning, (www.coursera.org). *Stanford University.*

[57] N. S. nitish, G. H. hinton, A. K. kriz, I. S. ilya, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting.

[58] C. C. Noble and D. J. Cook. Graph-based anomaly detection.

[59] K. O'Shea and R. Nash. An introduction to convolutional neural networks.

[60] K. O'Shea and R. Nash. An introduction to convolutional neural networks.

[61] J. Redmon. Darknet: Open source neural networks in c, 2013. available online : http://pjreddie.com/darknet (accessed on 14 june 2020).

[62] J. Redmon and A. Farhadi. Yolov3: An incremental improvement available online : https://arxiv.org/pdf/1804.02767.pdf.

[63] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. in 2017 ieee conference on computer vision and pattern recognition (cvpr). July 2017.

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

[65] S. Russell and P. Norvig. and artificial intelligence. a modern approach. artificial intelligence. prentice-hall, egnlewood cliffs, 25, 1995.

[66] S. Russell and P. Norvig. and artificial intelligence. a modern approach. artificial intelligence. prentice-hall, egnlewood cliffs, 25, 1995.

[67] M. R. S. Sai and S. V. SindhushaRella. Object detection and identification a project report.

[68] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks.

[69] Shi and Tomasi. Good features to track," 9th ieee conference on computer vision and pattern recognition. june 1994.

[70] A. SINGH. Anomaly detection for temporal data using long short-term memory (lstm).

[71] I. Smal and al. Multiple object tracking in molecular bioimaging by raoblackwellized marginal particle filtering.

[72] A. Smola and S. Vishwanathan. *Introduction to Machine Learning The Wikipedia Guide.*

[73] G. Sreenu and M. A. S. Durai. Intelligent video surveillance: a review through deep learning techniques for crowd analysis.

[74] S.Sivaraman, B. Morris, and M. Trivedi. Learning multi-lane trajectories using vehicle-based vision.

[75] J.-C. Su. State of the art object detection algorithms. university of california, san diego, 9500 gilman dr. la jolla, ca. 2014.

[76] Szeliski and R. Computer vision: algorithms and applications. springer science business media. 2010.

[77] A. ULLAH, J. AHMAD, K. MUHAMMAD, M. SAJJAD, and S. W. BAIK. Action recognition in video sequences using deep bi-directional lstm with cnn features.

[78] http://podoce.dinf.usherbrooke.ca/challenge/tswc2017/.

[79] F. C. L. W., R. A., T. A., and B. A.C. Dssd: Deconvolutional single shot detector. arxiv, 2017;arxiv:1701.06659v1.

[80] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. M. M. 2020. Understanding data augmentation for classification: when to warp?

[81] L. Xie and al. A new cnn-based method for multi-directional car licens plate detection.

[82] L. Y, W. R, S. S, and C. x. Structure inference net: Object detection using scene-level contextand instance-level relationships.

[83] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. acm computing surveys (csur), 38(4):13, 2006.

[84] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. acm computing surveys (csur) (german).2006.

[85] Z. S. W. L. B. X. L. Z. and L. S.Z. Single-shot refinement neural network for object detection.arxiv. 2018. arxiv:1711.06897.

[86] F. Zhang, C. Li, and F. Yang. Vehicle detection in urban traffic surveillance images based on convolutional neural networks with feature concatenation.

[87] H. Zhao, Y. Zhou, L. Zhang, Y. Peng, X. H. H. Peng, and X. Cai. Mixed yolov3-lite: A lightweight real-time object detection method.

[88] Z.-Q. Zhao, S. tao Xu, and X. Wu. Object detection with deep learning: A review.

Appendix A

Original Project Code

In this appendix all original code written for this project is presented. The complete code base for the project can be found at : https://github.com/OussamaBenAbdallah/MasterThesis

A.1 Object detection pseudo code

```
##############Model##############
class ConvLayer(Layer):
  def __init__(self, filters=32, size=(3,3), alpha=0.1, name='conv_layer', **kwargs):
    super(ConvLayer, self).__init__(name=name, **kwargs)
    self.conv = Conv2D(filters, size, padding='same')
    self.batch_normalization = BatchNormalization()
    self.activation = LeakyReLU(alpha=alpha)

  def call(self, inputs):
    x = self.conv(inputs)
    x = self.batch_normalization(x)
    x = self.activation(x)
    return x

class YoloModel(Model):
    def __init__(self, name=None):
        super(YoloModel, self).__init__(name=name)

        self.conv_layer1_1 = ConvLayer(32, (3,3), name='conv_layer1_1')
        self.maxpool1_1 = MaxPooling2D(pool_size=(2,2))

        self.conv_layer1_2 = ConvLayer(64, (3,3), name='conv_layer1_2')
        self.maxpool1_2 = MaxPooling2D(pool_size=(2,2))

        self.conv_layer1_3 = ConvLayer(128, (3,3), name='conv_layer1_3')
        self.conv_layer2_3 = ConvLayer(64, (1,1), name='conv_layer2_3')
        self.conv_layer3_3 = ConvLayer(128, (3,3), name='conv_layer3_3')
        self.maxpool1_3 = MaxPooling2D(pool_size=(2,2))

        self.conv_layer1_4 = ConvLayer(256, (3,3), name='conv_layer1_4')
        self.conv_layer2_4 = ConvLayer(128, (1,1), name='conv_layer2_4')
        self.conv_layer3_4 = ConvLayer(256, (3,3), name='conv_layer3_4')
        self.maxpool1_4 = MaxPooling2D(pool_size=(2,2))
```

```
        self.conv_layer1_5 = ConvLayer(512, (3,3), name='conv_layer1_5')
        self.conv_layer2_5 = ConvLayer(256, (1,1), name='conv_layer2_5')
        self.conv_layer3_5 = ConvLayer(512, (3,3), name='conv_layer3_5')
        self.conv_layer4_5 = ConvLayer(256, (1,1), name='conv_layer4_5')
        self.conv_layer5_5 = ConvLayer(512, (3,3), name='conv_layer5_5')

        self.maxpool1_5 = MaxPooling2D(pool_size=(2,2))

        self.conv_layer1_6 = ConvLayer(1024, (3,3), name='conv_layer1_6')
        self.conv_layer2_6 = ConvLayer(512, (1,1), name='conv_layer2_6')
        self.conv_layer3_6 = ConvLayer(1024, (3,3), name='conv_layer3_6')
        self.conv_layer4_6 = ConvLayer(512, (1,1), name='conv_layer4_6')
        self.conv_layer5_6 = ConvLayer(1024, (3,3), name='conv_layer5_6')

        self.conv_final = Conv2D(N_ANCHORS * (N_CLASSES + 5), (1,1), padding='same')

    def call(self, x, training=False):
        x = self.conv_layer1_1(x)
        x = self.maxpool1_1(x)

        x = self.conv_layer1_2(x)
        x = self.maxpool1_2(x)

        x = self.conv_layer1_3(x)
        x = self.conv_layer2_3(x)
        x = self.conv_layer3_3(x)
        x = self.maxpool1_3(x)

        x = self.conv_layer1_4(x)
        x = self.conv_layer2_4(x)
        x = self.conv_layer3_4(x)
        x = self.maxpool1_4(x)

        x = self.conv_layer1_5(x)
        x = self.conv_layer2_5(x)
        x = self.conv_layer3_5(x)
        x = self.conv_layer4_5(x)
        x = self.conv_layer5_5(x)
        x = self.maxpool1_5(x)

        x = self.conv_layer1_6(x)
        x = self.conv_layer2_6(x)
        x = self.conv_layer3_6(x)
        x = self.conv_layer4_6(x)
        x = self.conv_layer5_6(x)

        x = self.conv_final(x)
        return x
End Function
```

A.2 Loss function

```
class YoloLoss():
    def __init__(self):
        self.MSE_loss_object = keras.losses.MeanSquaredError()
        self.BCE_loss_object = keras.losses.BinaryCrossentropy()
        self.ignore_thresh = 0.5
        self.lambda_coord = 5.0
        self.lamda_noobj = 0.5
        self.lambda_class = 1.0
    def __call__(self, y_true, y_pred):

        true_box_rel, true_box_abs, true_obj, true_class = prepare_y_true(y_true)
        true_xy_rel = true_box_rel[..., 0:2]
        true_wh_rel = true_box_rel[..., 2:4]
        true_xy_abs = true_box_abs[..., 0:2]
        true_wh_abs = true_box_abs[..., 2:4]
        true_box_abs = xywh_to_x1x2y1y2(true_box_abs)

        pred_box_rel, pred_box_abs, pred_obj, pred_class = prepare_y_pred(y_pred)
        pred_xy_rel = pred_box_rel[..., 0:2]
        pred_wh_rel = pred_box_rel[..., 2:4]
        pred_xy_abs = pred_box_abs[..., 0:2]
        pred_wh_abs = pred_box_abs[..., 2:4]
        pred_box_abs = xywh_to_x1x2y1y2(pred_box_abs)

        xy_loss = self.calc_xy_loss(true_obj, true_xy_rel, pred_xy_rel, None)
        wh_loss = self.calc_wh_loss(true_obj, true_wh_rel, pred_wh_rel, None)
        class_loss = self.calc_class_loss(true_obj, true_class, pred_class)

        ignore_mask = self.calc_ignore_mask(true_obj, true_box_abs, pred_box_abs)

        obj_loss = self.calc_obj_loss(true_obj, pred_obj, ignore_mask)

        return xy_loss + wh_loss + class_loss + obj_loss, (xy_loss, wh_loss,
            ↪ class_loss, obj_loss)

    def calc_ignore_mask(self, true_obj, true_box, pred_box):
        true_obj = tf.squeeze(true_obj, axis=-1)
        true_box_filtered = tf.boolean_mask(true_box, tf.cast(true_obj, tf.bool))
        best_iou = tf.reduce_max(broadcast_iou(pred_box, true_box_filtered), axis=-1)
        ignore_mask = tf.cast(best_iou < self.ignore_thresh, tf.float32)
        ignore_mask = tf.expand_dims(ignore_mask, axis=-1)
        return ignore_mask

    def calc_xy_loss(self, true_obj, true_xy, pred_xy, weight = 5):
        xy_loss = tf.reduce_sum(tf.square(true_xy - pred_xy), axis=-1)
        true_obj = tf.squeeze(true_obj, axis=-1)
        xy_loss = true_obj * xy_loss #* weight
        xy_loss = tf.reduce_sum(xy_loss, axis=(1, 2, 3)) * self.lambda_coord
        return xy_loss

    def calc_wh_loss(self, true_obj, true_wh, pred_wh, weight=None):
        wh_loss = tf.reduce_sum(tf.square(true_wh - pred_wh), axis=-1)
        true_obj = tf.squeeze(true_obj, axis=-1)
        wh_loss = true_obj * wh_loss #* weight
```

```
        wh_loss = tf.reduce_sum(wh_loss, axis=(1, 2, 3)) * self.lambda_coord
        return wh_loss

    def calc_class_loss(self, true_obj, true_class, pred_class):
        class_loss = tf.square(true_class - pred_class)
        class_loss = tf.reduce_sum(class_loss, axis=(-1))
        true_obj = tf.squeeze(true_obj, axis=-1)
        class_loss = class_loss * true_obj
        class_loss = tf.reduce_sum(class_loss, axis=(1, 2, 3)) * self.lambda_class
        return class_loss

    def calc_obj_loss(self, true_obj, pred_obj, ignore_mask=None):
        obj_l = tf.square(true_obj - pred_obj)
        obj_loss = true_obj * obj_l
        noobj_loss = (1 - true_obj) * obj_l  * ignore_mask
        obj_loss = tf.reduce_sum(obj_loss, axis=(1, 2, 3, 4))
        noobj_loss = tf.reduce_sum(noobj_loss, axis=(1, 2, 3, 4)) * self.lamda_noobj
        return obj_loss + noobj_loss
```

A.3 Object Tracking pseudo code

```
out = cv2.VideoWriter('/content/video20.avi', cv2.VideoWriter_fourcc(*"MJPG"),
    ↪ 30,(480, 480))

url = "https://www.youtube.com/watch?v=MNn9qKG2UFI"
videoPafy = pafy.new(url)
best = videoPafy.allstreams[-2]
print(videoPafy.allstreams)
print(best)
color = (0, 255, 0)
# Parameters for Shi-Tomasi corner detection
feature_params = dict(maxCorners=50, qualityLevel=0.01, minDistance=1, gradientSize
    ↪ =3, blockSize=10)
# Parameters for Lucas-Kanade optical flow
lk_params = dict(winSize = (15,15), maxLevel = 2, criteria = (cv2.TERM_CRITERIA_EPS |
    ↪  cv2.TERM_CRITERIA_COUNT, 10, 0.03))

cap = cv2.VideoCapture(best.url)

color = (0, 255, 0)
error_color = (0, 0, 255)
center_color = (255, 0, 0)

matching_iou_threshold = 0.2
initial_sizes = []
bboxes = []
all_points = []
ids = []
tracking = False
def calc_iou(bbox1, bbox2):
    area1 = (bbox1[2] - bbox1[0]) * (bbox1[3] - bbox1[1])
    area2 = (bbox2[2] - bbox2[0]) * (bbox2[3] - bbox2[1])
    intersection_area = np.maximum(np.minimum(bbox1[2], bbox2[2]) - np.maximum(bbox1
        ↪ [0], bbox2[0]), 0) * \
                        np.maximum(np.minimum(bbox1[3], bbox2[3]) - np.maximum(bbox1
                            ↪ [1], bbox2[1]), 0)
    return intersection_area/ (area1 + area2 - intersection_area)

def remove_disapper(tracking_indices):
    global all_points, bboxes, initial_sizes, ids
    all_points = [all_points[i] for i in tracking_indices]
    initial_sizes = [initial_sizes[i] for i in tracking_indices]
    bboxes = [bboxes[i] for i in tracking_indices]
    ids = [ids[i] for i in tracking_indices]

def get_features(objects, frame, feature_params):
    global all_points, bboxes, initial_sizes, ids
    first_frame = frame
    old_gray = cv2.cvtColor(first_frame, cv2.COLOR_BGR2GRAY)
    old_gray = cv2.medianBlur(old_gray, 5)
    image_size = frame.shape[0]
    tracking_indices = []
    for obj in objects:
        new_bbox = obj[1] * image_size
        new_bbox = new_bbox.astype(np.int32)
```

```
        # print(new_bbox)
        roi = old_gray[new_bbox[1]:new_bbox[3], new_bbox[0]:new_bbox[2]]
        box_points = cv2.goodFeaturesToTrack(roi, mask=None, **feature_params)
        if box_points is None:
            continue
        box_points[:, :, 0] += new_bbox[0]
        box_points[:, :, 1] += new_bbox[1]
        max_iou = 0
        match_bbox_index = -1
        for i, bbox in enumerate(bboxes):
            iou = calc_iou(bbox, new_bbox)
            if (iou > max_iou) & (iou > matching_iou_threshold):
                max_iou = iou
                match_bbox_index = i
        if match_bbox_index == -1:
            if len(ids) not in tracking_indices:
                tracking_indices.append(len(ids))
            bboxes.append([new_bbox[0], new_bbox[1], new_bbox[2], new_bbox[3]])
            all_points.append(box_points)
            initial_sizes.append([new_bbox[2] - new_bbox[0], new_bbox[3] - new_bbox
                ↪ [1]])
            ids.append((max(ids) + 1) if ids else 0)
        else:
            bboxes[match_bbox_index] = [new_bbox[0], new_bbox[1], new_bbox[2],
                ↪ new_bbox[3]]
            all_points[match_bbox_index] = box_points
            initial_sizes[match_bbox_index] = [new_bbox[2] - new_bbox[0], new_bbox[3]
                ↪  - new_bbox[1]]
            if match_bbox_index not in tracking_indices:
                tracking_indices.append(match_bbox_index)
    print(tracking_indices)
    remove_disapper(tracking_indices)
    return old_gray

def remove_all():
    initial_sizes = []
    bboxes = []
    all_points = []
    ids = []
output = 0
num_frames = 0
while cap.isOpened():
    num_frames += 1
    if num_frames == 1500:
        break
    _, frame = cap.read()
    image_size = 480
    frame = letterbox_image(frame, image_size)
    # plt.imshow(frame)
    mask = np.zeros_like(frame)
    if num_frames % 20 == 1:
        print('num_frames: ', num_frames)
        tracking = True
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image = image.astype(np.float32)
        image = image / 255.0
        prediction = model.predict(np.reshape(image, (1,image_size, image_size, 3)))
```

```
        makePred = MakePrediction(0.5, 0.5, 20)
        objects = makePred(prediction)
        if len(objects) == 0:
            remove_all()
            tracking = False
            continue
        else:
            old_gray = get_features(objects, frame, feature_params)

    if tracking:
        new_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        new_gray = cv2.medianBlur(new_gray, 5)
        for index, old_points in enumerate(all_points):
            # print(len(old_points))
            new_points, status, error = cv2.calcOpticalFlowPyrLK(old_gray, new_gray,
                ↪ old_points, None, **lk_params)
            # good_old = old_points[(status == 1) & (error < 50)]
            # good_new = new_points[(status == 1) & (error < 50)]
            good_old = old_points[(status == 1)]
            good_new = new_points[(status == 1)]
            error = error[status == 1]
            if len(good_new) == 0:
                bboxes.pop(index)
                initial_sizes.pop(index)
                all_points.pop(index)
                ids.pop(index)
                continue
            xs, ys = 0, 0
            error_threshold = 0 if len(good_new[error > 2]) == 0 else 2
            xs = np.sum(good_new[error >= error_threshold,0])
            ys = np.sum(good_new[error >= error_threshold,1])
            center = (int(xs / len(good_new[error >= error_threshold])), int(ys / len
                ↪ (good_new[error >= error_threshold])))
            # print(good_new[...,0])
            if len(good_new[error >= error_threshold]) > 10:
                distances = np.sqrt(np.square(center[0]-good_new[...,0])+np.square(
                    ↪ center[1]-good_new[...,1]))
                good_old = good_old[distances < 2*np.max(distances[error >=
                    ↪ error_threshold])]
                good_new = good_new[distances < 2*np.max(distances[error >=
                    ↪ error_threshold])]

            if len(good_new) <= 3:
                bboxes.pop(index)
                initial_sizes.pop(index)
                all_points.pop(index)
                ids.pop(index)
                continue
            for i, (new, old) in enumerate(zip(good_new, good_old)):
                # Returns a contiguous flattened array as (x, y) coordinates for new
                    ↪ point
                a, b = new.ravel()
                # Returns a contiguous flattened array as (x, y) coordinates for old
                    ↪ point
                c, d = old.ravel()
                # mask = cv2.line(mask, (a, b), (c, d), color, 2)
                if error[i] < 2:
```

```
                    frame = cv2.circle(frame, (a, b), 2, color, -1)
                # else:
                #     frame = cv2.circle(frame, (a, b), 3, error_color, -1)
            upper_left = (int(np.min(good_new[...,0])), int(np.min(good_new[...,1])))
            bottom_right = (int(np.max(good_new[...,0])), int(np.max(good_new[...,1])
                ↪ ))
            frame = cv2.rectangle(frame, upper_left, bottom_right, color, 1)
            cv2.putText(frame, str(ids[index]), (upper_left[0], upper_left[1] - 5),
                ↪ cv2.FONT_HERSHEY_SIMPLEX, 0.5, (36, 255, 12), 1)
            frame = cv2.circle(frame, center, 3, center_color, -1)
            # Overlays the optical flow tracks on the original frame
            frame = cv2.add(frame, mask)
            # Updates oldious good feature points
            old_points = good_new.reshape(-1, 1, 2)
            all_points[index] = old_points
            bboxes[index] = [upper_left[0],upper_left[1],bottom_right[0],bottom_right
                ↪ [1]]
        # Updates oldious frame
        old_gray = new_gray.copy()
    out.write(frame)

    if (num_frames % 100 == 1):
        plt.figure(figsize=(10, 10))
        plt.imshow(frame)
    if cv2.waitKey(20) & 0xFF == ord('q'):
        break
cv2.destroyAllWindows()
cap.release()
```

A.4 Anomaly detection pseudo code

```
class MyModel(Model):
    def __init__(self):
        super(MyModel, self).__init__()
        momentum = 0.9
        self.vgg = TimeDistributed(vgg, input_shape=INSHAPE)
        self.g_avg_pool4 = TimeDistributed(GlobalAveragePooling2D())

        self.lstm1 = LSTM(256, return_sequences=True)
        self.lstm2 = LSTM(128)

        self.dense1 = Dense(1024, activation='relu')
        self.dropout = Dropout(0.3)
        self.dense2 = Dense(512, activation='relu')
        self.dense3 = Dense(128, activation='relu')
        self.dense4 = Dense(64, activation='relu')

        self.final_dense = Dense(len(CLASSES), activation='softmax')


    def call(self, x, training=False):
        x = self.vgg(x)
        x = self.g_avg_pool4(x)

        x = self.lstm1(x)
```

```
        x = self.lstm2(x)

        x = self.dense1(x)
        x = self.dropout(x)
        x = self.dense2(x)
        x = self.dense3(x)
        x = self.dense4(x)

        return self.final_dense(x)

# Create an instance of the model
model = MyModel()

class LossObject():
    def __init__(self, a):
        self.a = a
        self.b = max(a-1, 1)
    def __call__(self, y_true, y_pred):
        y_true = tf.cast(y_true, tf.float32)
        y_pred = tf.cast(y_pred, tf.float32)
        weight = self.a - self.b*y_true[:,1]
        # loss = keras.backend.categorical_crossentropy(y_true, y_pred)# * weight
        # print(loss.shape)
        loss = keras.backend.binary_crossentropy(y_true, y_pred)
        loss = tf.reduce_sum(loss, axis=-1)# * weight
        return loss

loss_object = LossObject(3)
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
train_loss = tf.keras.metrics.Mean(name='train_loss')
train_accuracy = tf.keras.metrics.CategoricalAccuracy(name='train_accuracy')

test_loss = tf.keras.metrics.Mean(name='test_loss')
test_accuracy = tf.keras.metrics.CategoricalAccuracy(name='test_accuracy')

@tf.function
def train_step(images, labels):
    with tf.GradientTape() as tape:
        # training=True is only needed if there are layers with different
        # behavior during training versus inference (e.g. Dropout).
        predictions = model(images, training=True)
        loss = loss_object(labels, predictions)
        loss = keras.backend.mean(loss)
    gradients = tape.gradient(loss, model.trainable_variables)
    optimizer.apply_gradients(zip(gradients, model.trainable_variables))

    train_loss(loss)
    train_accuracy(labels, predictions)

@tf.function
def test_step(images, labels):
    # training=False is only needed if there are layers with different
    # behavior during training versus inference (e.g. Dropout).
    predictions = model(images, training=False)
    t_loss = loss_object(labels, predictions)
    t_loss = keras.backend.mean(t_loss)
    # tf.print(t_loss)
```

```
    test_loss(t_loss)
    test_accuracy(labels, predictions)

EPOCHS = 30
# optimizer.learning_rate = 0.00001
for epoch in range(0, EPOCHS):
    # Reset the metrics at the start of the next epoch
    train_loss.reset_states()
    train_accuracy.reset_states()
    test_loss.reset_states()
    test_accuracy.reset_states()
    for i in range(len(data.trainset)):
        images, labels = data.trainset[i][0], data.trainset[i][1]
        train_step(images, labels)

    for i in range(len(data.valset)):
        test_images, test_labels = data.valset[i][0], data.valset[i][1]
        test_step(test_images, test_labels)

    template = 'Epoch {}, Loss: {}, Accuracy: {}, Test Loss: {}, Test Accuracy: {}'
    print(template.format(epoch + 1,
                          train_loss.result(),
                          train_accuracy.result() * 100,
                          test_loss.result(),
                          test_accuracy.result() * 100))
    try:
        save_path = f'/content/gdrive/My Drive/Riad/models/cadp_dataset/144/'
        if not os.path.exists(save_path):
            os.mkdir(save_path)
        model_name = f'model_E{epoch+1}_VL{round(float(test_loss.result()), 2)}_VA{
            ↪ round(float(test_accuracy.result()), 2)}'
        model.save(os.path.join(save_path, model_name))
    except Exception:
        pass
```