SOCIAL DISTANCE MONITORING USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

SANTHANAKUMAR V 822119104034

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING PATTUKKOTTAI

(A Constituent College of Anna University, Chennai)

ANNA UNIVERSITY: CHENNAI – 600 025

June 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report titled **"SOCIAL DISTANCE MONITORING USING MACHINE LEARNING"** is the bonafide work of **"SANTHANAKUMAR V (822119104034)"** of Computer Science & Engineering whose carried out this project work under my supervision.

SIGNATURE

Dr. S. SENTHILKUMAR, M.E., Ph.D.,
HEAD OF THE DEPARTMENT,
Computer Science & Engineering,
University College of Engineering,
Pattukkottai,
Thanjavur – 614 701

Submitted for the Project Viva Voice held on: _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I would first like to thank our college Dean **Dr. M. KARTHIKEYAN, M.Tech., Ph.D.,** for all the encouragement and support extended to us during the tenure of this project and who endorsed us throughout this project.

My heartfelt thanks to **Dr. S. SENTHILKUMAR, M.E., Ph.D.,** Department of Computer Science and Engineering, University College of Engineering, Pattukkottai, for the prompt and limitless help in providing the excellent infrastructure to do the project and to prepare the thesis.

I express my sincere thanks to the project committee members, Department of Computer Science and Engineering, Pattukkottai, for their invaluable guidance and technical support.

I would also like to thank all the teaching and non-teaching staffs of Computer Science and Engineering department, for their constant support and the encouragement given to us while we went about to achieving my project goals.

SANTHANAKUMAR V
822119104034

ABSTRACT

In this futuristic world of self-driving cars and voice assistants, computer science and machine learning play an important role in making our dream of automated living come true. This project is based on Object detection which is a direct application of object identification in the domain of Machine Learning and monitoring the social distance of people by finding the distance between people. Distance Measurement in detected objects is used in self-driving cars, or parking systems and as far now can be used for social distancing monitoring in this covid19 pandemic situation. This report deals with the process of making an Object detection model and monitoring the distance between the detected people from scratch. Using darknet, an open-source neural network, YOLOv4 is constructed and it can be an effective tool for object detection. Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization refers to identifying the location of one or more objects in an image and drawing a bounding box around their extent. Object detection does the work of combining these two tasks and localizes and classifies one or more objects in an image. We've trained the model with a custom database and the trained model is used to test new images and videos. mAP scores have been obtained for the trained weight files. The detected people's distance is measured by finding the Euclidean distance between the bounding boxes and a red bounding box is drawn, if the people are found to be close to each other. The final result is an ML model which can detect and monitor the social distance between people.

TABLE OF CONTENTS

LIST OF FIGURES		
S.NO	**FIGURE NAME**	**PAGE NO.**
1.1	AI vs ML vs DL ILLUSTRATION	2
1.2	Types of Machine Learning	3
1.3	Supervised Learning Process Flow	6
1.4	Unsupervised Learning Process Flow	7
1.5	Reinforcement Learning Process Flow	8
1.6	Object Detected Two Persons	10
1.7	Regional Proposal Network	12
1.8	Division of Image into Grids	18
1.9	Bounding Box Regression	19
1.10	Illustration of IOU	20
1.11	Combined Working of Three Techniques	21
3.1	Dataset Partition	34
3.2	Data is labeled as Person	35
3.3	Calculating Mean Absolute Error	
3.4	Graphical Representation of MSE	
3.5	Calculating Mean Squared Error	
3.6	Weights Representation	
3.7	Hidden Layers Representation	
3.8	Precision	
3.9	Representation of IOU	
3.10	Recall	
3.11	mAP Curve	

LIST OF ABBREVATIONS

S.NO	ABBREVATIONS	DESCRIPTION
1	ML	Machine Learning
2	AI	Artificial Intelligence
3	YOLO	You Only Look Once
4	mAP	mean Accurate Precision
5	IOU	Intersection over Union
6	RCNN	Region based Convolutional neural Network
7	MSE	Mean Squared Error
8	MAE	Mean Absolute Error
9	SSD	Single Shot Detectors

CHAPTER 1

INTRODUCTION

1.1 MACHINE LEARNING

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values. Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and predictive maintenance.

In this new era of technology, companies and developers around the world are talking about embracing artificial intelligence (AI), machine learning (ML), and deep learning (DL). AI is an umbrella discipline that covers everything related to making machines smarter. Machine Learning (ML) is commonly used along with AI but it is a subset of AI. ML refers to an AI system that can self-learn based on the algorithm. Systems that get smarter and smarter over time without human intervention is ML. Deep Learning (DL) is a machine learning (ML) applied to large data sets. Most AI work involves ML because intelligent behaviour requires considerable knowledge.

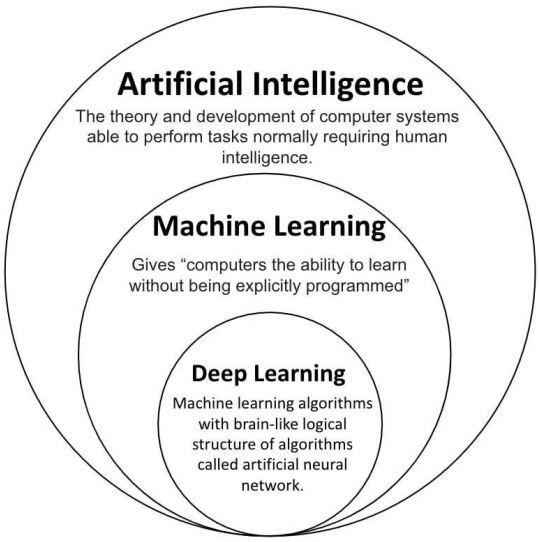


Figure 1.1 AI vs ML vs DL Illustration

Figure 1.1 shows that AI is the broader family consisting of ML and DL as it's components. ML is the subset of AI and DL is the subset of ML.

1.2 HISTORY OF MACHINE LEARNING

We might think that machine learning is a relatively new topic, but the concept of machine learning came into the picture in 1950, when Alan Turing published a paper answering the question "Can machines think?".

In 1957, Frank Rosenblatt designed the first neural network for computers, which is now commonly called the Perceptron Model. In 1959 Bernard Widrow and Marcian Hoff created two neural network models called Adeline, that could detect binary patterns and Madeline, that could eliminate echo on phone lines.

In 1967, the Nearest Neighbor Algorithm was written that allowed computers to use very basic pattern recognition.

Gerald DeJonge, in 1981 introduced the concept of explanation-based learning, in which a computer analyses data and creates a general rule to discard

unimportant information. During the 1990s work on machine learning shifted from a knowledge-driven approach to a more data-driven approach. During this period, scientists began creating programs for computers to analyze large amounts of data and draw conclusions or “learn” from the results. Which finally worked overtime after several developments formulated into the modern age of machine learning.

1.3 IMPORTANCE OF MACHINE LEARNING

Data is the lifeblood of all business. Data-driven decisions increasingly make the difference between keeping up with competition or falling further behind. Machine learning can be the key to unlocking the value of corporate and customer data and enacting decisions that keep a company ahead of the competition. It is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. It is quite hard to think of any industrial activity which can be done without the use of Machine learning or Artificial Intelligence. Machine learning is important because of its wide range of applications and its incredible ability to adapt and provide solutions to complex problems efficiently, effectively and quickly. To better understand the importance of machine learning these are certain instances where Machine learning is applied: online recommendation engines from Facebook, Netflix, Amazon, Apple’s Siri responding to your queries, facial recognition. It is quite hard for you to think of performing the above-mentioned tasks without the use of machine learning.

1.4 WORKING OF MACHINE LEARNING ALGORITHMS

Machine Learning algorithms utilize a variety of techniques to handle large amounts of complex data to make decisions. These algorithms complete the task of learning from data with specific inputs given to the machine. It's important to understand how these algorithms and a machine learning system as a whole work, so that we can get to know how these can be used in the future. It all starts with training the machine learning algorithm by using a training data set to create a model. When new input data is introduced to the ML algorithm, it makes a prediction. The predictions and results are evaluated for accuracy. If the prediction is not as expected, the algorithm is re-trained again and again until the desired output is obtained. This enables the ML algorithm to learn on its own and produce an optimal answer that will gradually increase in accuracy over time. After a desired level of accuracy is obtained, the machine learning algorithm is deployed.

A Decision Process: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabeled, your algorithm will produce an estimate about a pattern in the data.

An Error Function: An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.

An Model Optimization Process: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

1.5 TYPES OF MACHINE LEARNING METHODS

Machine Learning is broadly divided into three main areas, supervised learning, unsupervised and reinforcement learning. Each one of these has a specific action and purpose, yielding particular results by using various types of data. Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are three basic approaches: supervised learning, unsupervised learning, and reinforcement learning. The type of algorithm data scientists choose to use depends on what type of data they want to predict.

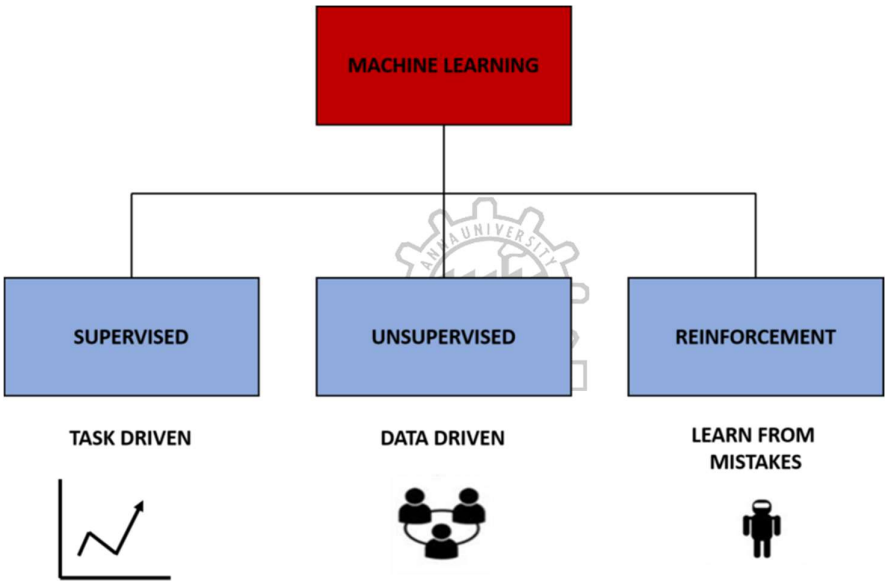


Figure 1.2 Types of Machine Learning

1.5.1 SUPERVISED MACHINE LEARNING

Supervised learning, also known as supervised machine learning, is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately. This occurs as part of the

cross validation process to ensure that the model avoids overfitting or underfitting. Supervised learning helps organizations solve a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Some methods used in supervised learning include neural networks, naïve bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and more.

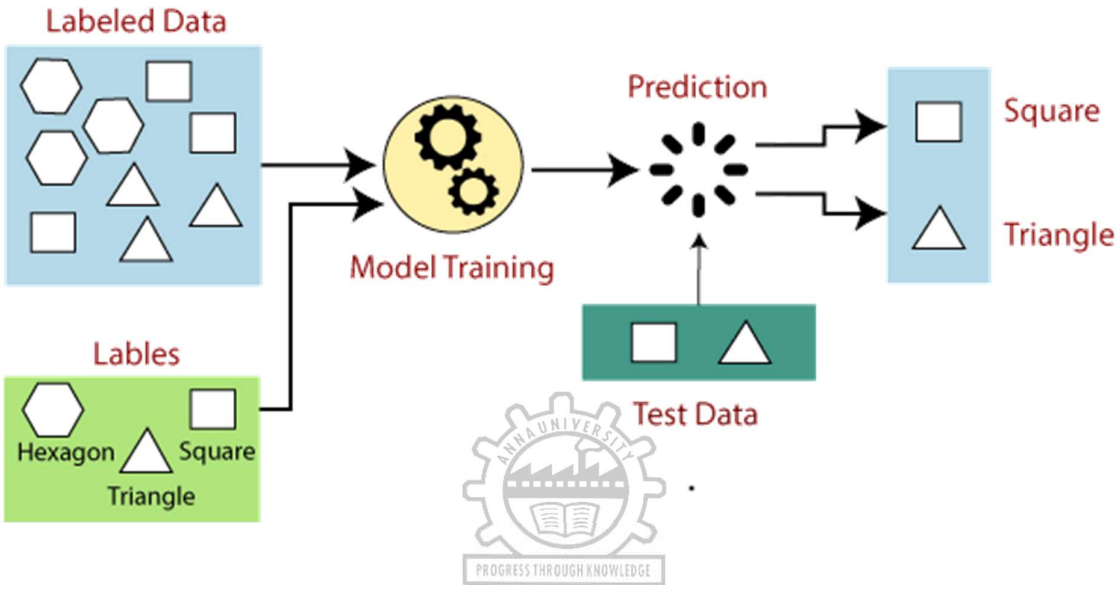


Figure 1.3 Supervised Learning process flow

Supervised learning algorithms are good for the following tasks:

- **Binary classification:** Dividing data into two categories.
- **Multi-class classification:** Choosing between more than two types of answers.
- **Regression modeling:** Predicting continuous values.
- **Ensembling:** Combining the predictions of multiple machine learning models to produce an accurate prediction.

1.5.2 UNSUPERVISED MACHINE LEARNING

Unsupervised machine learning in simple language means the ML model is self-sufficient in learning on its own. In unsupervised machine learning, there is no such provision of labeled data. The training data is unknown or unlabeled. This unknown data is fed to the machine learning model and is used to train the model. The model tries to find patterns and relationships in the dataset by creating clusters in it. The thing to be noted here is that unsupervised learning is not able to add labels to the clusters. For example, it cannot say this is a group of oranges or mangoes, but it will separate all the oranges frommangoes.

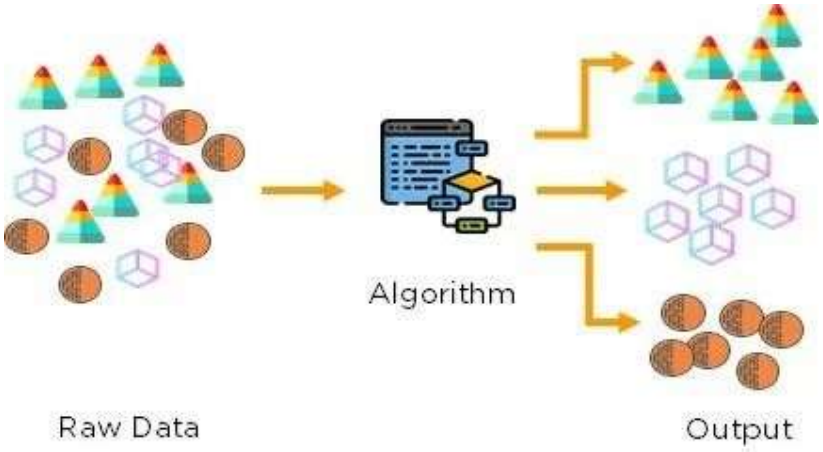


Figure 1.4 Unsupervised Learning process flow

Unsupervised learning algorithms are good for the following tasks:

- **Clustering:** Splitting the dataset into groups based on similarity.
- **Anomaly detection:** Identifying unusual data points in a data set.
- **Association mining:** Identifying sets of items in a data set that frequently occur together.
- **Dimensionality reduction:** Reducing the number of variables in a data set.

1.5.3 REINFORCEMENT MACHINE LEARNING

Reinforcement machine learning is a behavioral machine learning model that is similar to supervised learning, but the algorithm isn't trained using sample data. This model learns as it goes by using trial and error. A sequence of successful outcomes will be reinforced to develop the best recommendation or policy for a given problem. Reinforcement learning works by programming an algorithm with a distinct goal and a prescribed set of rules for accomplishing that goal. Data scientists also program the algorithm to seek positive rewards which it receives when it performs an action that is beneficial toward the ultimate goal and avoid punishments which it receives when it performs an action that gets it farther away from its ultimate goal. Reinforcement learning is often used in areas such as:

- **Robotics:** Robots can learn to perform tasks in the physical world using this technique.
- **Video gameplay:** Reinforcement learning has been used to teach bots to play a number of video games.
- **Resource management:** Given finite resources and a defined goal, reinforcement learning can help enterprises plan out how to allocate resources.

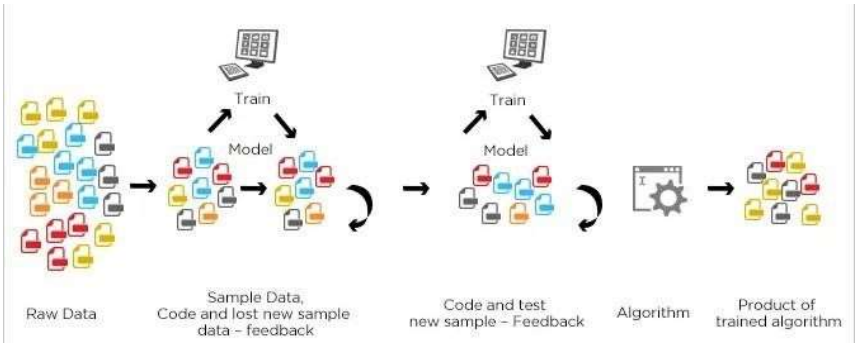


Figure 1.5 Reinforcement Learning process flow

1.6 MACHINE LEARNING USE-CASES

Speech recognition: It is also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, and it is a capability which uses natural language processing (NLP) to process human speech into a written format. Many mobile devices incorporate speech recognition into their systems to conduct voice search e.g. Siri or provide more accessibility around texting.

Customer service: Online chatbots are replacing human agents along the customer journey. They answer frequently asked questions (FAQs) around topics, like shipping, or provide personalized advice, cross-selling products or suggesting sizes for users, changing the way we think about customer engagement across websites and social media platforms. Examples include messaging bots on e-commerce sites with virtual agents, messaging apps, such as Slack and Facebook Messenger, and tasks usually done by virtual assistants and voice assistants.

Computer vision: This AI technology enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action. This ability to provide recommendations distinguishes it from image recognition tasks. Powered by convolutional neural networks, computer vision has applications within photo tagging in social media, radiology imaging in healthcare, and self-driving cars within the automotive industry.

Recommendation engines: Using past consumption behavior data, AI algorithms can help to discover data trends that can be used to develop more effective cross-selling strategies. This is used to make relevant add-on recommendations to customers during the checkout process for online retailers.

Automated stock trading: Designed to optimize stock portfolios, AI-driven high-frequency trading platforms make thousands or even millions of trades per day without human intervention.

1.7 OBJECT DETECTION

Object detection is a computer vision technique that works to identify and locate objects within an image or video. Specifically, object detection draws bounding boxes around these detected objects, which allow us to locate where said objects are in (or how they move through) a given scene. Object detection is commonly confused with image recognition, so before we proceed, it's important that we clarify the distinctions between them.

Image recognition assigns a label to an image. A picture of a person receives the label "person". A picture of two persons still receives the label "person". Object detection, on the other hand, draws a box around each person and labels the box "person". The model predicts where each object is and what label should be applied. In that way, object detection provides more information about an image than recognition.

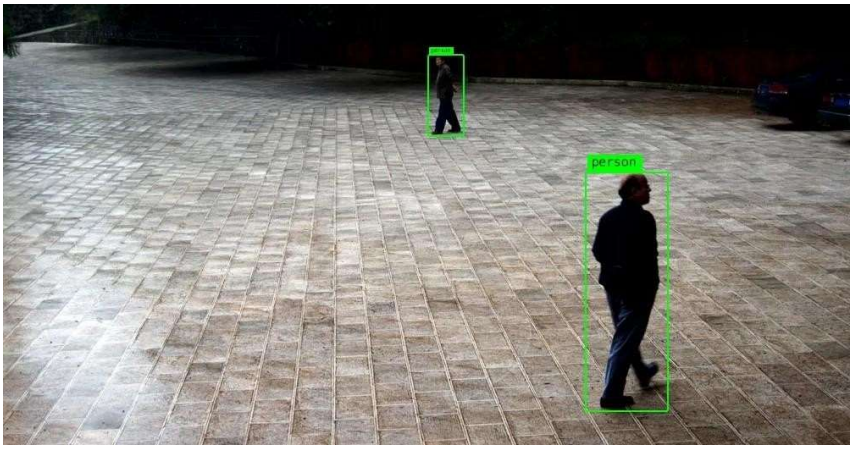


Figure 1.6 Object detected two persons

1.8 MODES OF OBJECT DETECTION

Object Detection can be broken down into machine learning-based approaches and deep learning-based approaches. In more traditional ML-based approaches, computer vision techniques are used to look at various features of an image, such as the color histogram or edges, to identify groups of pixels that may belong to an object. These features are then fed into a regression model that predicts the location of the object along with its label.

On the other hand, deep learning-based approaches employ convolutional neural networks (CNNs) to perform end-to-end, unsupervised object detection, in which features don't need to be defined and extracted separately.

1.9 IMPORTANCE OF OBJECT DETECTION

Object detection is inextricably linked to other similar computer vision techniques like image recognition and image segmentation, in that it helps us understand and analyze scenes in images or video. But there are important differences. Image recognition only outputs a class label for an identified object, and image segmentation creates a pixel-level understanding of a scene's elements. What separates object detection from these other tasks is its unique ability to locate objects within an image or video. This then allows us to count and then track those objects. Given these key distinctions and object detection's unique capabilities, we can see how it can be applied in a number of ways:

- Crowd counting
- Self-driving cars
- Video surveillance
- Face detection
- Anomaly detection

1.10 WORKING OF OBJECT DETECTION

Machine learning-based object detection models typically have two parts. An encoder takes an image as input and runs it through a series of blocks and layers that learn to extract statistical features used to locate and label objects. Outputs from the encoder are then passed to a decoder, which predicts bounding boxes and labels for each object. The simplest decoder is a pure regressor. The regressor is connected to the output of the encoder and predicts the location and size of each bounding box directly. The output of the model is the X, Y coordinate pair for the object and its extent in the image. Though simple, this type of model is limited. You need to specify the number of boxes ahead of time. If your image has two dogs, but your model was only designed to detect a single object, one will go unlabeled. However, if you know the number of objects you need to predict in each image ahead of time, pure regression-based models may be a good option.

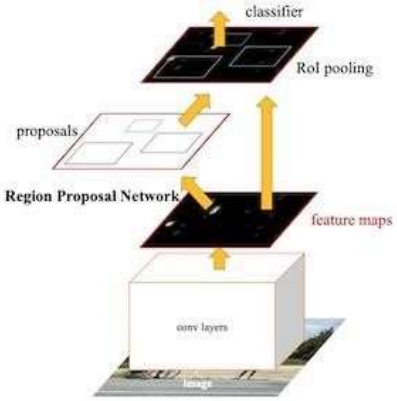


Figure 1.7 Region Proposal Network

An extension of the regressor approach is a region proposal network. In this decoder, the model proposes regions of an image where it believes an object might reside. The pixels belonging to these regions are then fed into a classification subnetwork to determine a label (or reject the proposal). It then

runs the pixels containing those regions through a classification network. The benefit of this method is a more accurate, flexible model that can propose arbitrary numbers of regions that may contain a bounding box. The added accuracy, though, comes at the cost of computational efficiency.

1.11 SINGLE SHOT DETECTORS

Single shot detectors (SSDs) seek a middle ground. Rather than using a subnetwork to propose regions, SSDs rely on a set of predetermined regions. A grid of anchor points is laid over the input image, and at each anchor point, boxes of multiple shapes and sizes serve as regions. For each box at each anchor point, the model outputs a prediction of whether or not an object exists within the region and modifications to the box's location and size to make it fit the object more closely. Because there are multiple boxes at each anchor point and anchor points may be close together, SSDs produce many potential detections that overlap. Post-processing must be applied to SSD outputs in order to prune away most of these predictions and pick the best one. The most popular post-processing technique is known as non-maximum suppression.

Finally, a note on accuracy. Object detectors output the location and label for each object. For an object's location, the most commonly-used metric is intersection-over-union (IOU). Given two bounding boxes, we compute the area of the intersection and divide by the area of the union. This value ranges from 0 (no interaction) to 1 (perfectly overlapping). For labels, a simple "percent correct" can be used. If your use case requires that object detection work in real-time, without internet connectivity, or on private data, you might be considering running your object detection model directly on an edge device like a mobile phone or IoT board.

1.12 OVERVIEW OF MODELARCHITECTURE

1.12.1 R-CNN, Faster R-CNN, Mask R-CNN

A number of popular object detection models belong to the R-CNN family. Short for region convolutional neural networks, these architectures are based on the region proposal structure discussed above. Over the years, they've become both more accurate and more computationally efficient. Mask R-CNN is the latest iteration, developed by researchers at Facebook, and it makes a good starting point for server-side object detection models.

1.12.2 YOLO, Mobile Net + SSD, Squeeze Det

There are also a number of models that belong to the single shot detector family. The main difference between these variants are their encoders and the specific configuration of predetermined anchors. Mobile Net + SSD models feature a Mobile Net-based encoder, Squeeze Det borrows the Squeeze Net encoder, and the YOLO model features its own convolutional architecture. SSDs make great choices for models destined for mobile or embedded devices.

1.12.3 Center Net

More recently, researchers have developed object detection models that do away with the need for region proposals entirely. CenterNet treats objects as single points, predicting the X, Y coordinates of an object's center and its extent (height and width). This technique has proven both more efficient and accurate than SSD or R-CNN approaches.

1.13 OBJECT DETECTION APPLICATIONS

- **Video surveillance**

  Because state-of-the-art object detection techniques can accurately identify and track multiple instances of a given object in a scene, these techniques naturally lend themselves to automating video surveillance systems. For instance, object detection models are capable of tracking multiple people at once, in real-time, as they move through a given scene or across video frames. From retail stores to industrial factory floors, this kind of granular tracking could provide invaluable insights into security, worker performance and safety, retail foot traffic, and more.

- **Crowd counting**

  Crowd counting is another valuable application of object detection. For densely populated areas like theme parks, malls, and city squares, object detection can help businesses and municipalities more effectively measure different kinds of traffic whether on foot, in vehicles, or otherwise. This ability to localize and track people as they maneuver through various spaces could help businesses optimize anything from logistics pipelines and inventory management, to store hours, to shift scheduling, and more. Similarly, object detection could help cities plan events, dedicate municipal resources, etc.

- **Anomaly detection**

  Anomaly detection is a use case of object detection that's best explained through specific industry examples. In agriculture, for instance, a custom object detection model could accurately identify and locate potential instances of plant disease, allowing farmers to detect threats to their crop yields that would otherwise not be discernible to the naked human eye. And in health care, object detection could be used to help treat conditions that have specific and unique symptomatic lesions. One such example of this comes in the form of skin care and the treatment of acne—an object detection model could locate and identify instances of acne in seconds. What's particularly important and compelling about these potential use cases is how they leverage and provide knowledge and information that's generally only available to agricultural experts or doctors, respectively.

- **Self-driving cars**

  Real-time car detection models are key to the success of autonomous vehicle systems. These systems need to be able to identify, locate, and track objects around them in order to move through the world safely and efficiently. And while tasks like image segmentation can be (and often are) applied to autonomous vehicles, object detection remains a foundational task that underpins current work on making self-driving cars a reality.

1.14 YOLO - YOU ONLY LOOK ONCE

As a real-time object detection system, YOLO object detection utilizes a single neural network. The latest release of Image AI v2.1.0 now supports training a custom YOLO model to detect any kind and number of objects. Convolutional neural networks are instances of classifier-based systems where the system repurposes classifiers or localizers to perform detection and applies the detection model to an image at multiple locations and scales. Using this process, “high scoring” regions of the image are considered detections. Simply put, the regions which look most like the training images given are identified positively.

As a single-stage detector, YOLO performs classification and bounding box regression in one step, making it much faster than most convolutional neural networks. For example, YOLO object detection is more than 1000x faster than R-CNN and 100x faster than Fast R-CNN.

1.15 CHOOSING YOLO ALGORITHM

YOLO algorithm is important because of the following reasons:

- **Speed:** This algorithm improves the speed of detection because it can predict objects in real-time.
- **High accuracy:** YOLO is a predictive technique that provides accurate results with minimal background errors.
- **Learning capabilities:** The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

1.16 WORKING OF YOLOALGORITHM

YOLO algorithm works using the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

1.16.1 RESIDUAL BLOCKS

First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.

Figure 1.8 Division of image into grids

In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

1.16.2 BOUNDING BOX REGRESSION

A bounding box is an outline that highlights an object in an image. Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
- Bounding box center (bx, by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.

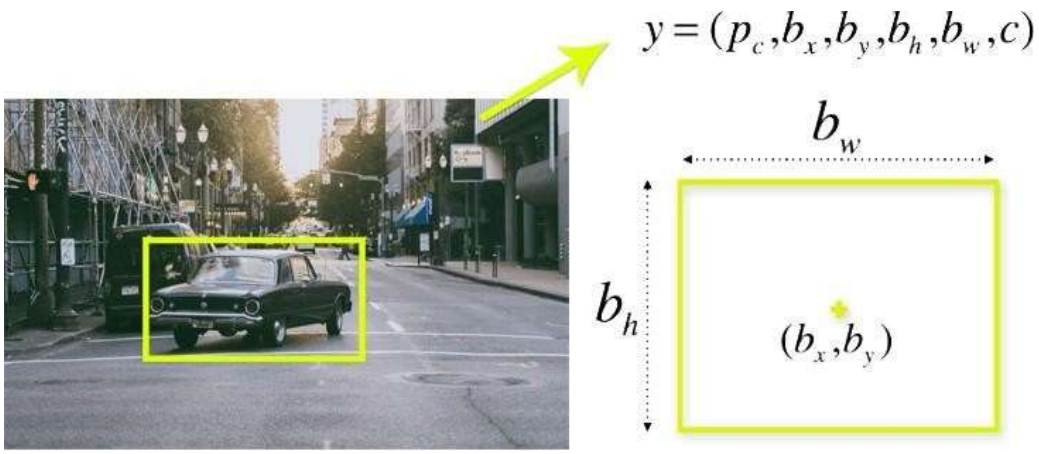


Figure 1.9 Bounding Box Regression

YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

1.16.3 INTERSECTION OVER UNION

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly. Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box. The following image provides a simple example of how IOU works.

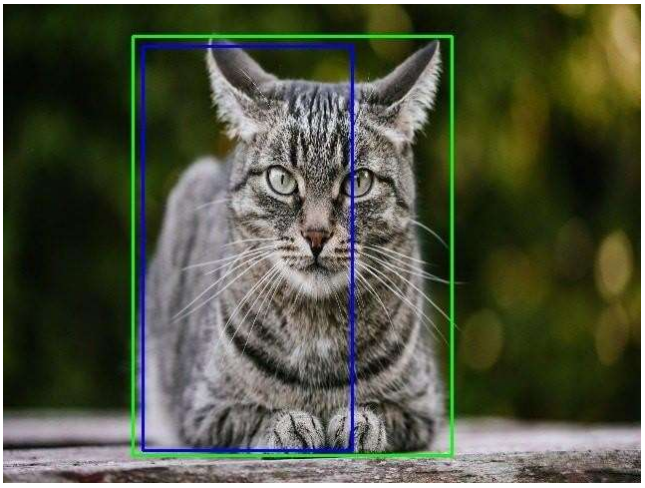


Figure 1.10 Illustration of IOU

In the image above, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.

1.17 COMBINATION OF THE THREE TECHNIQUES

First, the image is divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object.

For example, we can notice at least three classes of objects: a car, a dog, and a bicycle. All the predictions are made simultaneously using a single convolutional neural network.

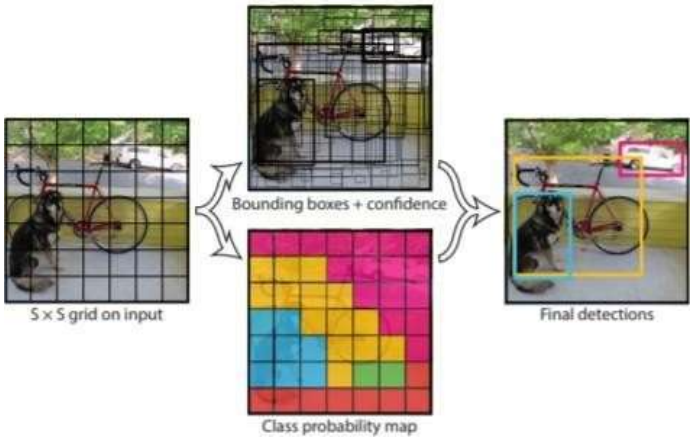


Figure 1.11 Combined working of three techniques

Intersection over union ensures that the predicted bounding boxes are equal to the real boxes of the objects. This phenomenon eliminates unnecessary bounding boxes that do not meet the characteristics of the objects (like height and width). The final detection will consist of unique bounding boxes that fit the objects perfectly.

For example, the car is surrounded by the pink bounding box while the bicycle is surrounded by the yellow bounding box. The dog has been highlighted using the blue bounding box.

1.18 PROBLEM STATEMENT

COVID-19 belongs to the family of coronavirus-caused diseases, firstly reported in Wuhan, China at the end of December 2020. China has announced its first death from the virus on January 11, a 61 years old man. On March 11,

World Health Organization (WHO) declared it a pandemic due to its spread over 114 countries. The basic objective is to reduce the physical contact between the infected and the healthy people. As prescribed by WHO, people should maintain at least 1 meter (m) distance from each other to control the spread of this disease. A monitoring system for overseeing the people identifying people at risk of getting affected by the covid.

1.19 OBJECTIVE

A Machine learning-based solution is proposed for the automatic detection of people and monitoring social distance in any respective environments. The first contribution of this project is the performance evaluation of YOLO v4 on low light conditions without applying any image cleansing approaches. So, the real-time application should have to give a timely response with high accuracy. Secondly, a social distance monitoring solution is proposed by considering precise speed-accuracy tradeoff and is evaluated on our custom dataset. From experimental results, it is aimed that the model will exhibit good performance with a balanced mAP score.

1.20 CONTRIBUTION

The contribution to detect the objects includes the proposed Alexey Bochkovskiy et al. ‘s working on the optimal speed and accuracy of the YOLO helped in choosing the right hyperparameters to change in order to achieve performance efficiency and to increase the accuracy of the model on detection. There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required.

CHAPTER 2

LITERATURE SURVEY

Jiwoong Choi et al.,(2019) proposed that the use of object detection algorithms is becoming increasingly important in autonomous vehicles, and object detection at high accuracy and a fast inference speed is essential for safe autonomous driving. A false positive (FP) from a false localization during autonomous driving can lead to fatal accidents and hinder safe and efficient driving. Therefore, a detection algorithm that can cope with mislocalizations is required in autonomous driving applications. This paper proposes a method for improving the detection accuracy while supporting a real-time operation by modeling the bounding box (b-box) of YOLOv3, which is the most representative of one-stage detectors, with a Gaussian parameter and redesigning the loss function. In addition, this paper proposes a method for predicting the localization uncertainty that indicates the reliability of the bounding box.

Aleksey Bochkovsky et al., (2020) proposed that there are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT,

Mish activation, Mosaic data augmentation, CmBN, Drop Block regularization, and CIoU loss, and combine some of them to achieve state-of-the-art results: 43.5% AP (65.7% AP50) for the MS COCO dataset at a real time speed of ~65 FPS on Tesla V100.

Zhaohui Zheng et al.,(2020) proposed that the bounding box regression is the crucial step in object detection. In existing methods, while ℓn-norm loss is widely adopted for bounding box regression, it is not tailored to the evaluation metric, i.e., Intersection over Union (IoU). Recently, IoU loss and generalized IoU (GIoU) loss have been proposed to benefit the IoU metric, but still suffer from the problems of slow convergence and inaccurate regression. In this paper, we propose a Distance-IoU (DIoU) loss by incorporating the normalized distance between the predicted box and the target box, which converges much faster in training than IoU and GIoU losses. Furthermore, this paper summarizes three geometric factors in bounding box regression, i.e., overlap area, central point distance and aspect ratio, based on which a Complete IoU (CIoU) loss is proposed, thereby leading to faster convergence and better performance. By incorporating DIoU and CIoU losses into state-of-the-art object detection algorithms, e.g., YOLO, SSD and Faster R-CNN, we achieve notable performance gains in terms of not only IoU metric but also GIoU metric. Moreover, DIoU can be easily adopted into non-maximum suppression (NMS) to act as the criterion, further boosting performance improvement.

Van Etten et al., (2019) proposed that detecting small objects over large areas remains a significant challenge in satellite imagery analytics. Among the challenges is the sheer number of pixels and geographical extent per image: a single DigitalGlobe satellite image encompasses over 64 km 2 and over 250 million pixels. To address these issues, we propose a pipeline (SIMRDWN) that evaluates satellite images of arbitrarily large size at native resolution at a rate of

≥0.2 km 2 /s. Building upon the TensorFlow Object Detection API paper, this pipeline offers a unified approach to multiple object detection frameworks that can run inference on images of arbitrary size. The SIMRDWN pipeline includes a modified version of YOLO (known as YOLT), along with the models of the TensorFlow Object Detection API: SSD, Faster R-CNN, and R-FCN. We evaluate large test images at native resolution and find mAP scores of 0.2 to 0.8 for vehicle localization, with the YOLT architecture achieving both the highest mAP and fastest inference speed.

Olga Russakovsky et al.,(2015) proposed that the ImageNet Large Scale Visual Recognition Challenge is a benchmark in object category classification and detection on hundreds of object categories and millions of images. The challenge has been run annually from 2010 to present, attracting participation from more than fifty institutions. This paper describes the creation of this benchmark dataset and the advances in object recognition that have been possible as a result. We discuss the challenges of collecting large-scale ground truth annotation, highlight key breakthroughs in categorical object recognition, provide a detailed analysis of the current state of the field of large-scale image classification and object detection, and compare the state-of-the-art computer vision accuracy with human accuracy. We conclude with lessons learned in the five years of the challenge, and propose future directions and improvements.

Joseph Redmon et al.,(2016) proposed a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, 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. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. Our

unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives in the background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

Sara Beery et al.,(2020) proposed that in static monitoring cameras, useful contextual information can stretch far beyond the few seconds typical video understanding models might see: subjects may exhibit similar behavior over multiple days, and background objects remain static. Due to power and storage constraints, sampling frequencies are low, often no faster than one frame per second, and sometimes are irregular due to the use of a motion trigger. In order to perform well in this setting, models must be robust to irregular sampling rates. In this paper we propose a method that leverages temporal context from the unlabeled frames of a novel camera to improve performance at that camera. Specifically, we propose an attention-based approach that allows our model, Context R-CNN, to index into a long term memory bank constructed on a per-camera basis and aggregate contextual features from other frames to boost object detection performance on the current frame.

Menglong Zhu et al.,(2019) proposed that with a single eye fixation lasting a fraction of a second, the human visual system is capable of forming a rich representation of a complex environment, reaching a holistic understanding which facilitates object recognition and detection. This phenomenon is known as recognizing the ”gist” of the scene and is accomplished by relying on

relevant prior knowledge. This paper addresses the analogous question of whether using memory in computer vision systems can not only improve the accuracy of object detection in video streams, but also reduce the computation time. By interleaving conventional feature extractors with extremely lightweight ones which only need to recognize the gist of the scene, they show that minimal computation is required to produce accurate detections when temporal memory is present. In addition, they show that the memory contains enough information for deploying reinforcement learning algorithms to learn an adaptive inference policy.

Vivek Rathod et al.,(2018) presented a simple tweak of the Single Shot Multibox Detector (SSD) family of detectors, which is effective in reducing model size while maintaining the same quality. They share box predictors across all scales, and replace convolution between scales with max pooling. This has two advantages over vanilla SSD: (1) it avoids score miscalibration across scales; (2) the shared predictor sees the training data over all scales. Since we reduce the number of predictors to one, and trim all convolutions between them, model size is significantly smaller. We empirically show that these changes do not hurt model quality compared to vanilla SSD.

Joseph Redmon et al.,(2018) presented some updates to YOLO. They made a bunch of little design changes to make it better. They also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP50 in 51 ms on a Titan X, compared to 57.5 AP50 in 198 ms by RetinaNet, similar performance but 3.8× faster.

Mingxing Tan et al.,(2020) proposed that model efficiency has become increasingly important in computer vision. In this paper, we systematically study neural network architecture design choices for object detection and propose several key optimizations to improve efficiency. First, we propose a weighted bi-directional feature pyramid network (BiFPN), which allows easy and fast multiscale feature fusion; Second, we propose a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time. Based on these optimizations and better backbones, we have developed a new family of object detectors, called Efficient, which consistently achieve much better efficiency than prior art across a wide spectrum of resource constraints. In particular, with single model and single-scale, our EfficientDet-D7 achieves state of-the-art 55.1 AP on COCO test-dev with 77M parameters and 410B FLOPs1, being 4x – 9x smaller and using 13x – 42x fewer FLOPs than previous detectors.

Xingyi Zhou et al.,(2019) proposed that the detection identifies objects as axis-aligned boxes in an image. Most successful object detectors enumerate a nearly exhaustive list of potential object locations and classify each. This is wasteful, inefficient, and requires additional post-processing. In this paper, we take a different approach. We model an object as a single point the center point of its bounding box. Our detector uses keypoint estimation to find center points and regresses to all other object properties, such as size, 3D location, orientation, and even pose. Our center point based approach, CenterNet, is end-to-end differentiable, simpler, faster, and more accurate than corresponding bounding box based detectors. CenterNet achieves the best speed-accuracy trade-off on the MS COCO dataset, with 28.1% AP at 142 FPS, 37.4% AP at 52 FPS, and 45.1% AP with multi-scale testing at 1.4 FPS. We use the same

approach to estimate the 3D bounding box in the KITTI benchmark and human pose on the COCO keypoint dataset. Our method performs competitively with sophisticated multi-stage methods and runs in real-time.

Xiangyu Zhang et al.,(2015) proposed that deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers 8× deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset.

Jonathan Huang et al.,(2017) proposed to serve as a guide for selecting a detection architecture that achieves the right speed/memory/accuracy balance for a given application and platform. To this end, we investigate various ways to trade accuracy for speed and memory usage in modern convolutional object detection systems. A number of successful systems have been proposed in recent years, but apples-to-apples comparisons are difficult due to different base feature extractors (e.g., VGG, Residual Networks), different default image resolutions, as well as different hardware and software platforms. We present a unified implementation of the Faster R-CNN, R-FCN and SSD systems, which we view as “meta-architectures” and trace out the speed/accuracy trade-off curve created by using alternative feature extractors and varying other critical

parameters such as image size within each of these meta-architectures. On one extreme end of this spectrum where speed and memory are critical, we present a detector that achieves real time speeds and can be deployed on a mobile device. On the opposite end in which accuracy is critical, we present a detector that achieves state-of-the-art performance measured on the COCO detection task.

Chien-Yao Wang et al.,(2021) presented that the YOLOv4 object detection neural network based on the CSP approach, scales both up and down and is applicable to small and large networks while maintaining optimal speed and accuracy. We propose a network scaling approach that modifies not only the depth, width, resolution, but also structure of the network. YOLOv4- large model achieves state-of-the-art results: 55.5% AP (73.4% AP50) for the MS COCO dataset at a speed of ~16 FPS on Tesla V100, while with the test time augmentation, YOLOv4-large achieves 56.0% AP (73.3 AP50). To the best of our knowledge, this is currently the highest accuracy on the COCO dataset among any published work. The YOLOv4-tiny model achieves 22.0% AP (42.0% AP50) at a speed of ~443 FPS on RTX 2080 Ti, while by using TensorRT, batch size = 4 and FP16-precision the YOLOv4-tiny achieves 1774 FPS.

Jianxin Wu et al.,(2017) stated that's how a Convolutional Neural Network (CNN) operates from a mathematical perspective. This note is self-contained, and the focus is to make it comprehensible to beginners in the CNN field. The Convolutional Neural Network (CNN) has shown excellent performance in many computer vision and machine learning problems. Many solid papers have been published on this topic, and quite some high quality open source software packages have been made available. There are also well-written CNN tutorials and software manuals. However, I believe that an introductory CNN material specifically prepared for beginners is still needed. Research

papers are usually very terse and lack details. It might be difficult for beginners to read such papers. A tutorial targeting experienced researchers may not cover all the necessary details to understand how a CNN runs.

Saad Albawi et al.,(2017) proposed that the term Deep Learning or Deep Neural Network refers to Artificial Neural Networks (ANN) with multilayers. Over the last few decades, it has been considered to be one of the most powerful tools and has become very popular in the literature as it can handle a huge amount of data. The interest in having deeper hidden layers has recently begun to surpass classical methods performance in different fields; especially in pattern recognition. One of the most popular deep neural networks is the Convolutional Neural Network (CNN).

Cong Tang et al.,(2017) proposed that object detection based on deep learning is an important application in deep learning technology, which is characterized by its strong capability of feature learning and feature representation compared with the traditional object detection methods. The paper first makes an introduction of the classical methods in object detection, and expounds the relation and difference between the classical methods and the deep learning methods in object detection. Then it introduces the emergence of object detection methods based on deep learning and elaborates the most typical methods nowadays in object detection via deep learning. In the statement of the methods, the paper focuses on the framework design and the working principle of the models and analyzes the model performance in the real-time and the accuracy of detection. Eventually, it discusses the challenges in object detection based on deep learning and offers some solutions for reference.

Ross Girshick et al.,(2014) proposed that Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last

few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012 achieving a mAP of 53.3%. Since we combine region proposals with CNNs, we call our method R-CNN: Regions with CNN features.

Xiaoou Tang et al.,(2012) presented in this paper, a new Mixture model of Dynamic pedestrian-Agents (MDA) is proposed to learn the collective behavior patterns of pedestrians in crowded scenes. Collective behaviors characterize the intrinsic dynamics of the crowd. From the agent-based modeling, each pedestrian in the crowd is driven by a dynamic pedestrian-agent, which is a linear dynamic system with its initial and termination states reflecting a pedestrian's belief of the starting point and the destination. Then the whole crowd is modeled as a mixture of dynamic pedestrian-agents.

CHAPTER 3

SOCIAL DISTANCE MONITORING USING MACHINE LEARNING

3.1 PROPOSED METHODOLOGY

In this project, darknet's YOLO v4 object detection algorithm is used. Custom dataset is collected with multiple classes with their respective labels and class names, which will be trained and tested for accuracy scores and better evaluation. The following steps are followed on successful completion of this project.

- Dataset Collection
- Annotation
- Model Training
- Model Testing
- Performance Evaluation
- Measuring the Euclidean Distance

3.2 DATASET COLLECTION

A COCO(Common Objects in Context) dataset consisting of multiple classes is collected. Here we take a single class 'Person' for detection. Classes are sometimes called targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).In layman's terms, classes are a number of different/unique traffic signs present in our custom dataset. The label is the final choice, such as dog, fish, iguana, rock, etc. Once you've trained your model, you will give it sets of new input containing those features; it will return the predicted "label" (Person).

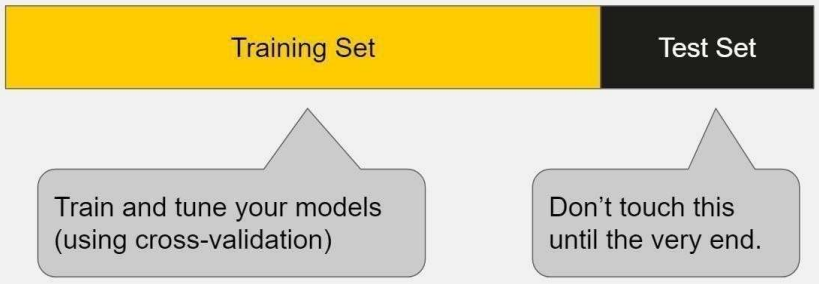


Figure 3.1 Dataset Partition

Training sets are used to fit and tune your models. Test sets are put aside as "unseen" data to evaluate your models. Importance of data is of much importance in our project as data is either a very scarce quantity or in very high amounts. Think of your data as a limited resource. You can spend some of it to train your model. You can spend some of it to evaluate (test) your model. But you can't reuse the same data for both. If you evaluate your model on the same data you used to train it, your model could be very overfit and you wouldn't even know. A model should be judged on its ability to predict new, unseen data. Therefore, you should have separate training and test subsets of your dataset.

3.3 ANNOTATION

A software named Label-Img is used for this process. It creates an .xml file with the given label and the (x1,y1)(x2,y2) coordinates. For YOLO v4, we need the label and the coordinates in a .txt file. The format of an labelled txt file is: <class no> <x1> <y1> <x2> <y2>

Example input:

0 0.518657 0.484375 0.753731 0.859375

Explained:

0 - class no(in this case speed_limit_5) 0.518657 - x1 coordinate and

respectively.

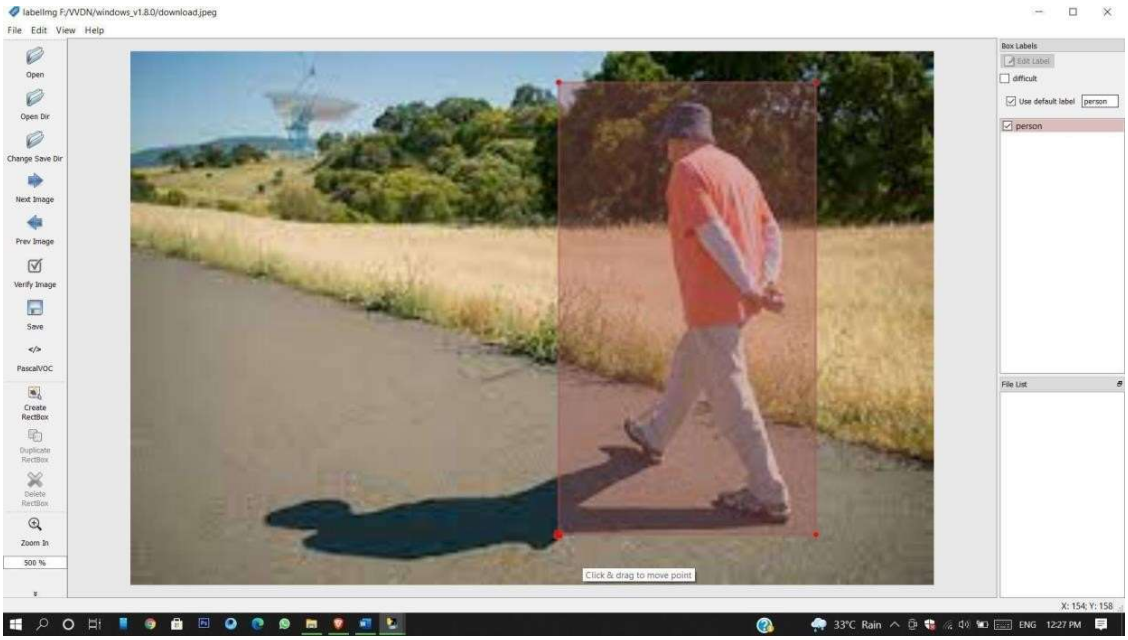


Figure 3.2 Data is labeled as 'person'

The coordinate value produced by Label-Img software is in pixel count, but the requirement of the YOLOv3 is in the range 0-1,thus an python script was created to change the values to be in specific range of 0-1 before training proceeds. The name of the .txt file and the .jpg/png/jpeg /webp file should be the same for the model to process it.

3.4 MODEL TRAINING

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model. This iterative process is called "model fitting".

The accuracy of the training dataset or the validation dataset is critical for the precision of the model. Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all

attributes involved. YOLO v4 is an Supervised learning algorithm and here it is possible when the training data contains both the input and output values. Each set of data that has the inputs and the expected output is called a supervisory signal. The training is done based on the deviation of the processed result from the documented result when the inputs are fed into the model. .cfg file located in the root folder is responsible for all the operations happening inside the model and modifying that .cfg file helps us fine tune the model.

The default values are Batch=64, Subdivision = 16, Max_batches = 5000 (classes*2000, but not less than number of training images and not less than 6000), Steps: change line steps to 80% and 90% of max_batches, Classes = 58(here the total number of unique traffic signs are 58) Filters = (classes + coords + 1)*<no of mask> In YOLOv4, filers = (58+4+1)*3 = 169

For training, the batch and subdivisions should be greater than 1. Change the values of classes in lines 610,696 and 783. Change the filters to 169 in 603,689 and 776. And we have to explicitly tell the model where the files and folders are located before training. And it is done by specifying the values in the obj.data file.

classes = 84

train = data/train.txt

valid = data/test.txt

names = data/obj.names

backup = backup/

The backup folder is where the weight files are stored after each 1000 iterations while the model is getting trained. For training purposes, we use the default

given weight file as the initial weight file yolov4.conv.137. There are various processes involved under model training.

3.4.1 BACKPROPAGATION

The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function are then considered to be a solution to the learning problem.

3.4.2 GRADIENT DESCENT

In model training , sets of images are considered and each set is worked to find a value called Gradient descent. Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point, because this is the direction of steepest descent. Conversely, stepping in the direction of the gradient will lead to a local maximum of that function; the procedure is then known as gradient ascent.

3.4.3 LEARNING RATE

The size of these steps is called the learning rate. With a high learning rate we can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. With a very low learning rate, we can confidently move in the direction of the negative gradient since we are recalculating it so frequently. A low learning rate is more precise, but calculating the gradient is time-consuming.

3.4.4 COST FUNCTION

Cost function is a function that measures the performance of a Machine Learning model for given data. Cost Function quantifies the error between predicted values and expected values and presents it in the form of a single real number. Depending on the problem Cost Function can be formed in many different ways. The purpose of Cost Function is to be either:

- **Minimized** - then returned value is usually called cost, loss or error. The goal is to find the values of model parameters for which Cost Function returns as small a number as possible.
- **Maximized** - then the value it yields is named a reward. The goal is to find values of model parameters for which the returned number is as large as possible.

3.4.5 MEAN ABSOLUTE ERROR

Regression metric which measures the average magnitude of errors in a group of predictions, without considering their directions. In other words, it's a mean of absolute differences among predictions and expected results where all individual deviations have even importance.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}^{(i)} - y^{(i)}|$$

Figure 3.3 Calculating Mean Absolute Error

where: i - index of sample, ŷ - predicted value, y - expected value, m - number of samples in dataset.

3.4.6 MEAN SQUARED ERROR

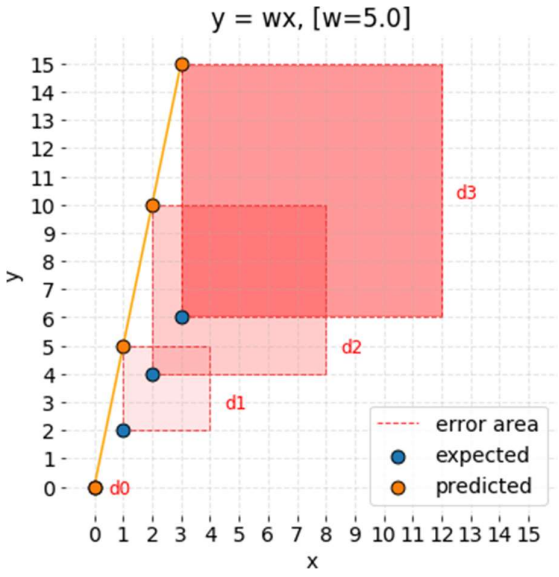


Figure 3.4 Graphical representation of MSE

One of the most commonly used and firstly explained regression metrics. Average squared difference between the predictions and expected results. In other words, an alteration of MAE where instead of taking the absolute value of differences, they are squared. In MAE, the partial error values were equal to the distances between points in the coordinate system. Regarding MSE, each partial error is equivalent to the area of the square created out of the geometrical distance between the measured points. All regional areas are summed up and averaged.

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

Figure 3.5 Calculating Mean Squared Error

where: i - index of sample, ŷ - predicted value, y - expected value, m - number of samples in dataset.

3.4.7 WEIGHTS AND BIASES

Weights and biases (commonly referred to as w and b) are the learnable parameters of a machine learning model. Neurons are the basic units of a neural network. In an ANN, each neuron is in a layer and is connected to each neuron in the next layer. When the inputs are transmitted between neurons, the weights are applied to the inputs along with the bias.

Weights control the signal (or the strength of the connection) between two neurons. In other words, a weight decides how much influence the input will have on the output. Biases, which are constant, are an additional input into the next layer that will always have the value of 1. Bias units are not influenced by the previous layer (they do not have any incoming connections) but they do have outgoing connections with their own weights.

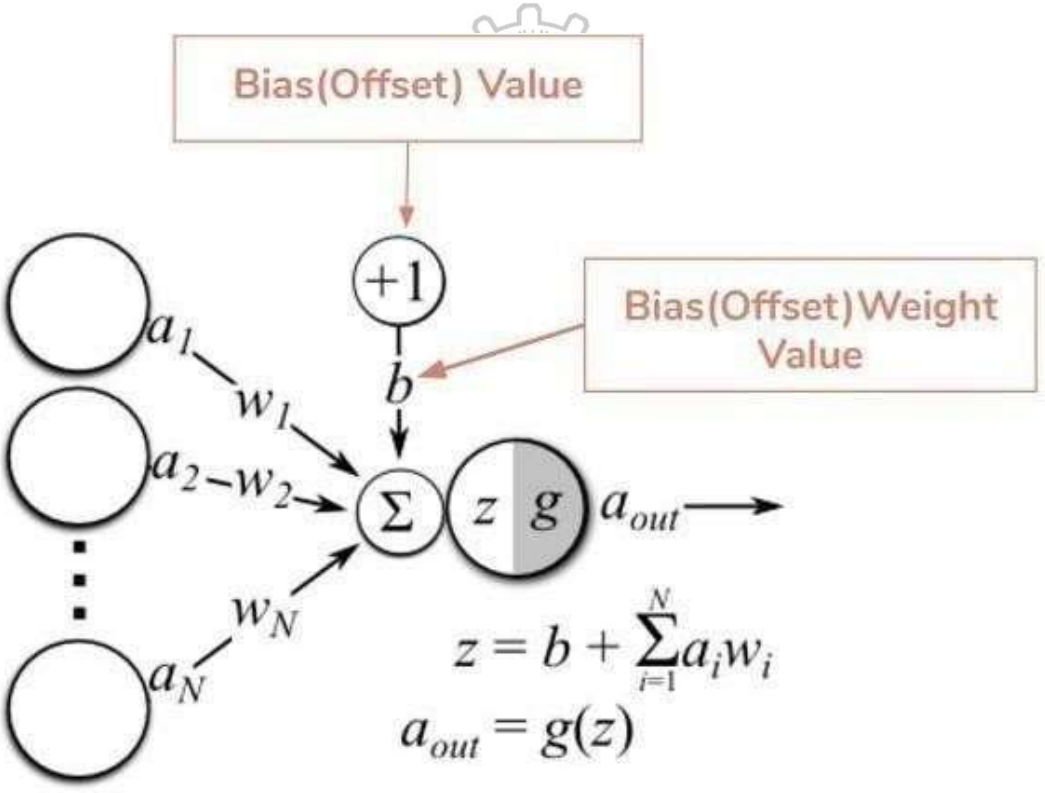


Figure 3.6 Weights Representation

What happens inside the model when it gets trained is that for each iteration some set of images is considered, and the model tests itself and checks with the known result (supervised learning) and corrects itself by adjusting the weight file. The amount of correction is done and depends on a function called Gradient descent. At each iteration the cost function tries to obtain a value as small as possible, and Gradient descent is based on the observation that if the multivariable function F(x) is defined and differentiable.

3.4.8 BATCHES

Batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration. The batch size can be one of three options: batch mode: where the batch size is equal to the total dataset thus making the iteration and epoch values equivalent.

3.4.9 SUB DIVISION

It's nothing but how many mini batches one split your batch in. Batch=64 -> loading 64 images for this "iteration". Subdivision=8 -> Split batch into 8 "mini-batches" so 64/8 = 8 images per "minibatch" and this gets sent to the gpu for processing. That will be repeated 8 times until the batch is completed and a new iteration will start with 64 new images. When batching one is averaging over more images the intent is not only to speed up the training process but also to generalize the training more.

3.4.10 DECAY

The learning rate is a parameter that determines how much an updating step influences the current value of the weights. Weight decay is an additional term in the weight update rule that causes the weights to exponentially decay to zero, if no other update is scheduled.

3.4.11 FILTERS

Filters typically are applied to data in the data processing stage or the preprocessing stage. Filters enhance the clarity of the signal that's used for machine learning. Detect trends or remove seasonal effects in noisy sales or economic data.

3.4.12 ACTIVATION FUNCTIONS

An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.

3.4.13 HIDDEN LAYERS

In neural networks, a hidden layer is located between the input and output of the algorithm, in which the function applies weights to the inputs and directs them through an activation function as the output. In short, the hidden layers perform nonlinear transformations of the inputs entered into the network.

3.4.14 WORKING OF HIDDEN LAYERS

Hidden layers, simply put, are layers of mathematical functions each designed to produce an output specific to an intended result. For example, some forms of hidden layers are known as squashing functions. These functions are particularly useful when the intended output of the algorithm is a probability because they take an input and produce an output value between 0 and 1, the range for defining probability.

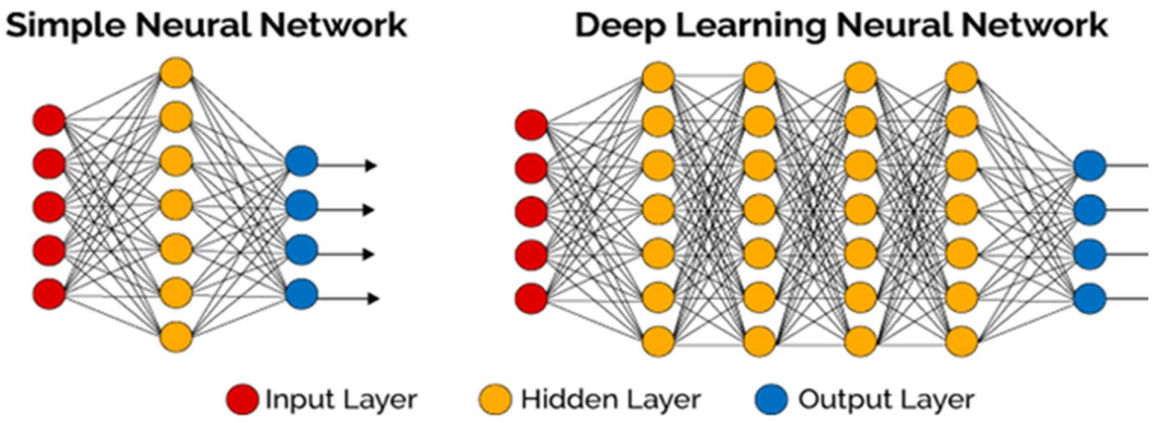


Figure 3.7 Hidden Layers Representation

Hidden layers allow for the function of a neural network to be broken down into specific transformations of the data. Each hidden layer function is specialized to produce a defined output. For example, hidden layer functions that are used to identify human eyes and ears may be used in conjunction by subsequent layers to identify faces in images. While the functions to identify eyes alone are not enough to independently recognize objects, they can function jointly within a neural network.

Training command is: *darknet.exe detector train data/obj.data yolo-obj.cfg darknet53.conv.74 > /content/drive/'My Drive'/train.log -dont_show*

After the model training gets completed, we end up with weight files in /root/backup.

3.5 MODEL TESTING

Before testing is conducted, we should consider the fact that we need the model to consider each image to be tested at a time, thus we should change batch and subdivisions to 1 in the .cfg file. The trained model is tested with any sample video or image to check how well the object is detected.

Testing command is:

For Image - *darknet.exe detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights*

For Video - *darknet.exe detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights test.mp4*

3.6 PERFORMANCE EVALUATION

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and overfitted models. I have tried different algorithm models and changed hyperparameters and evaluated their performances. Our model is evaluated by a metric, mAP score.

mAP stands for Mean Average Precision. Precision measures how accurate your predictions are. i.e. the percentage of your predictions are correct. It measures how many of the predictions that your model made were actually correct.

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

Figure 3.8 Precision

TP = True Positives (Predicted as positive as was correct)

FP = False Positives (Predicted as positive but was incorrect)

Object detection systems make predictions in terms of a bounding box and a class label. For each bounding box, we measure an overlap between the predicted bounding box and the ground truth bounding box. This is measured by IoU (intersection over union).

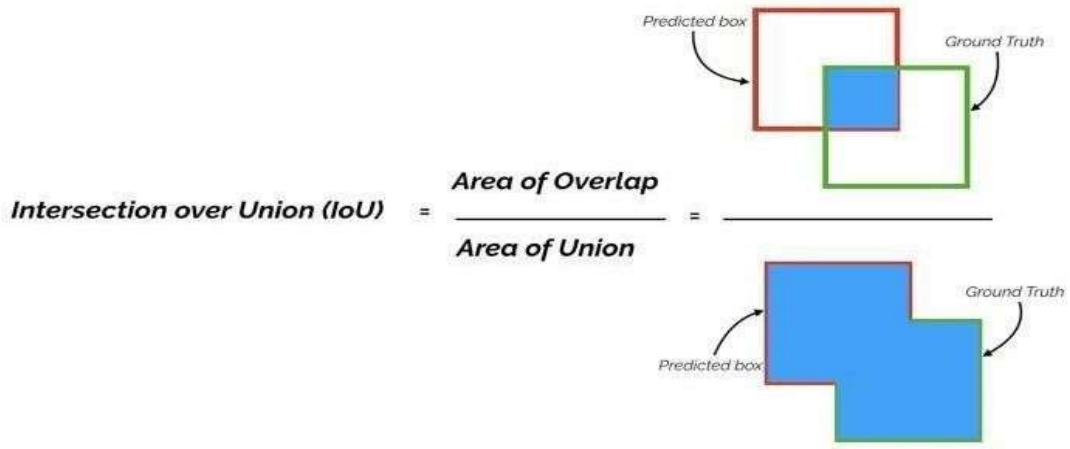


Figure 3.9 Representation of IOU

Recall measures how well you find all the positives. For example, we can find 80% of the possible positive cases in our top K predictions.

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

Figure 3.10 Recall

TP = True Positives (Predicted as positive as was correct)

FN = False Negatives (Failed to predict an object that was there)

The general definition for the Average Precision (AP) is finding the area under the precision- recall curve above. mAP (mean average precision) is the average of AP. We got a mAP score of 89.33% on detecting people.

Command to find Map score:

!'/darknet' detector map 'data/obj.data' 'backup/yolov4_custom_test.cfg' '/content/drive/My Drive/yolov4_custom_6000.weights'

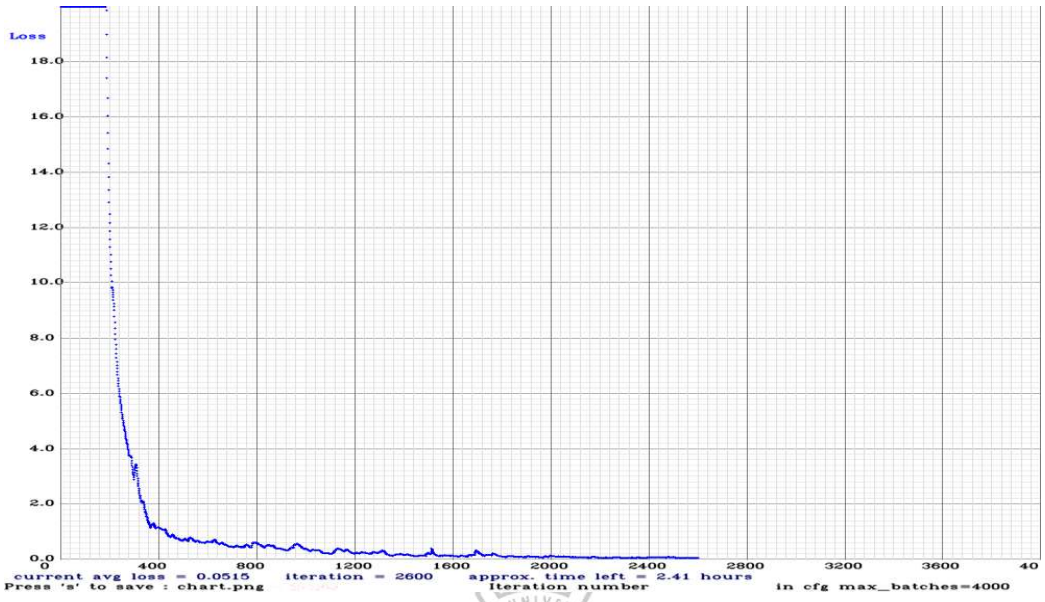


Figure 3.11 mAP Curve

3.6.1 HYPERPARAMETERS

A model parameter is a configuration variable that is internal to the model and whose value can be estimated from data. A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data. We cannot know the best value for a model hyperparameter on a given problem. We may use rules of thumb, copy values used on other problems, or search for the best value by trial and error. Thus we train the model numerous times to get the desired best results. Changing learning rate(increasing/decreasing), max batches, sub division might have an effect on the result of a model. Thus different results are stored and the best model is produced as a result. Changing learning rate(increasing/decreasing), max

batches, sub division might have an effect on the result of a model. Thus different results are stored and the best model is produced as a result.

3.7 MEASURING THE EUCLIDEAN DISTANCE

$$ED = \sqrt{(x_A - x_c)^2 + (y_A - y_c)^2}$$

Figure 3.12 Calculating Euclidean Distance

The Euclidean distance between two points in either the plane or 3-dimensional space measures the length of a segment connecting the two points. It is the most obvious way of representing distance between two points. shows the Euclidean distance of tracking algorithms and the proposed method. From above, it is clear that the proposed method has the least Euclidean distance between the centroid of the tracked bounding box and actual centroid in comparison to other methods. The x-axis shows the ground truth values of the centroid of the object in different frames.

Using python script and the trained model, the object person is detected from the frame. There is a criteria before the bounding box is drawn around the object using openCV. The Euclidean distance between any two bounding boxes must be less than 100 to ensure the safety distance advised by the World Health Organization.

3.8 SOFTWARE REQUIREMENTS

The model training requires intense computational power, thus to meet the required demands, we need CUDA, CUDA® is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs).

The easiest and most affordable way is to use Google's Collaboratory. Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. Each user gets a 12hr free continuous runtime which is enough for the model to train for around 4000 iterations. Google colab provides GPU runtime which offers more memory for higher computational power.

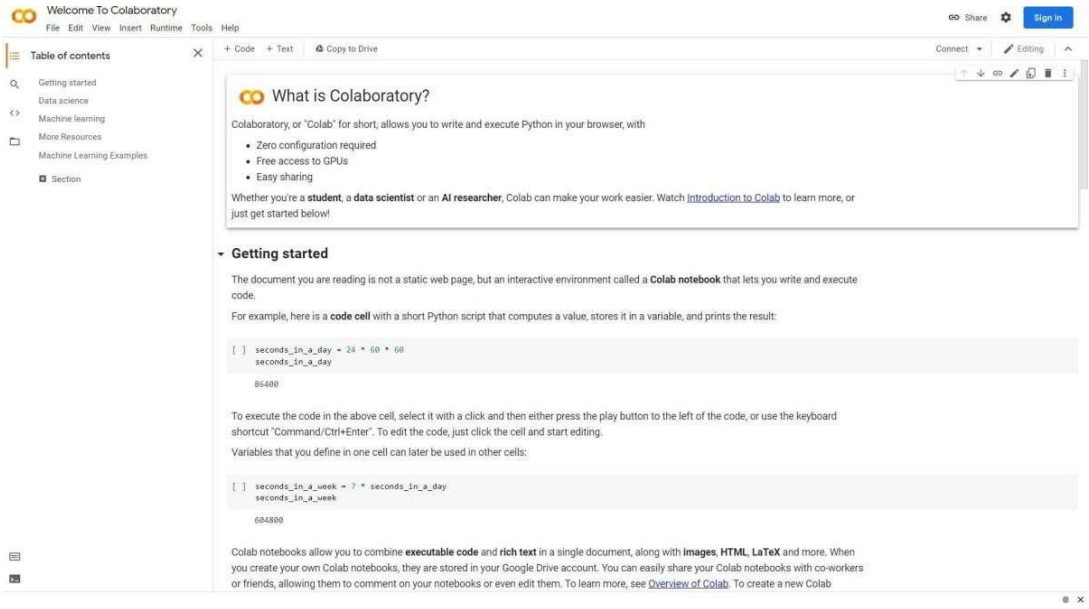


Figure 3.13 Colaboratory home screen

Furthermore we need CUDAnn, a neural network engine architecture that sets up the environment for the models to process in nVidia GPU. Then we need to have Visual Studio 2019 to compile the darknet architecture on a local machine. Other dependencies like python and opencv library are also needed for successful object detection.

CHAPTER 4

RESULT ANALYSIS

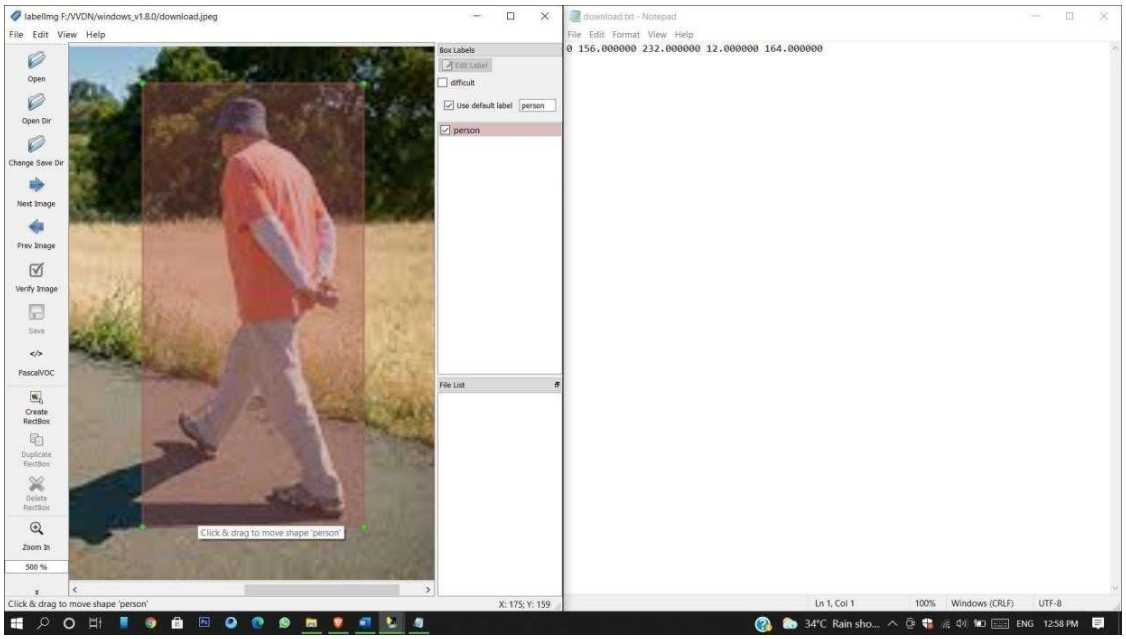


Figure 4.1 Labeled Image and its corresponding .txt file

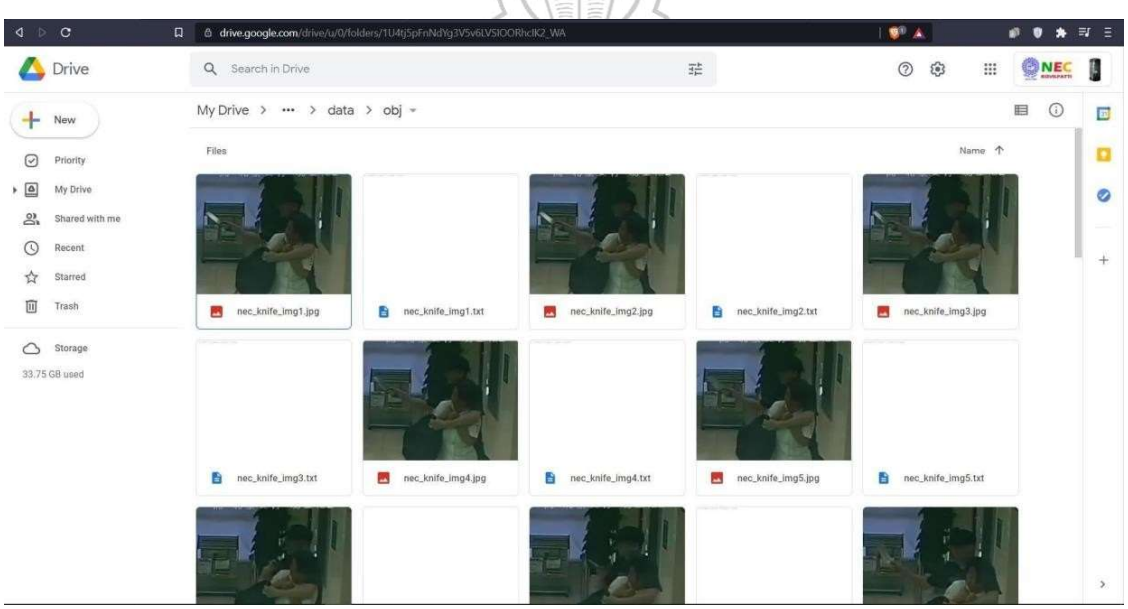


Figure 4.2 Dataset collected and uploaded to the drive

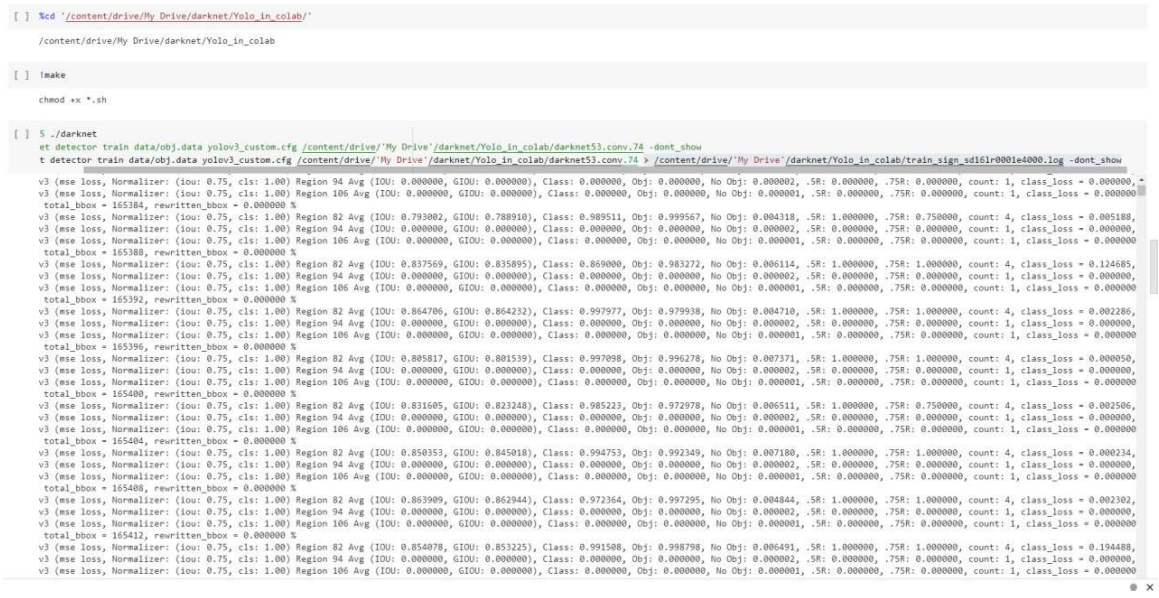


Figure 4.3 Model training

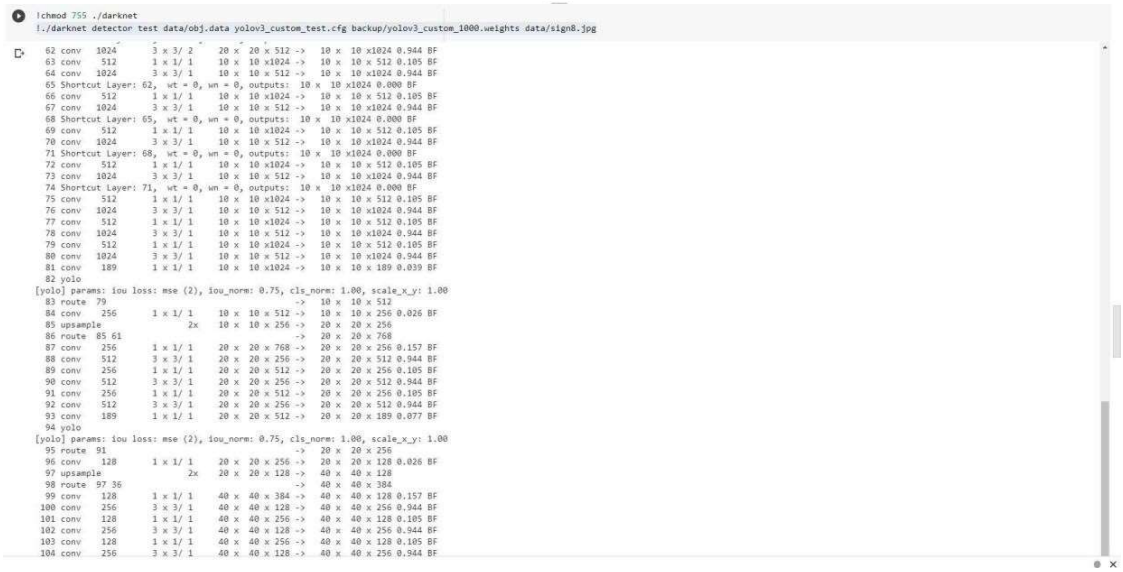


Figure 4.4 Model Testing

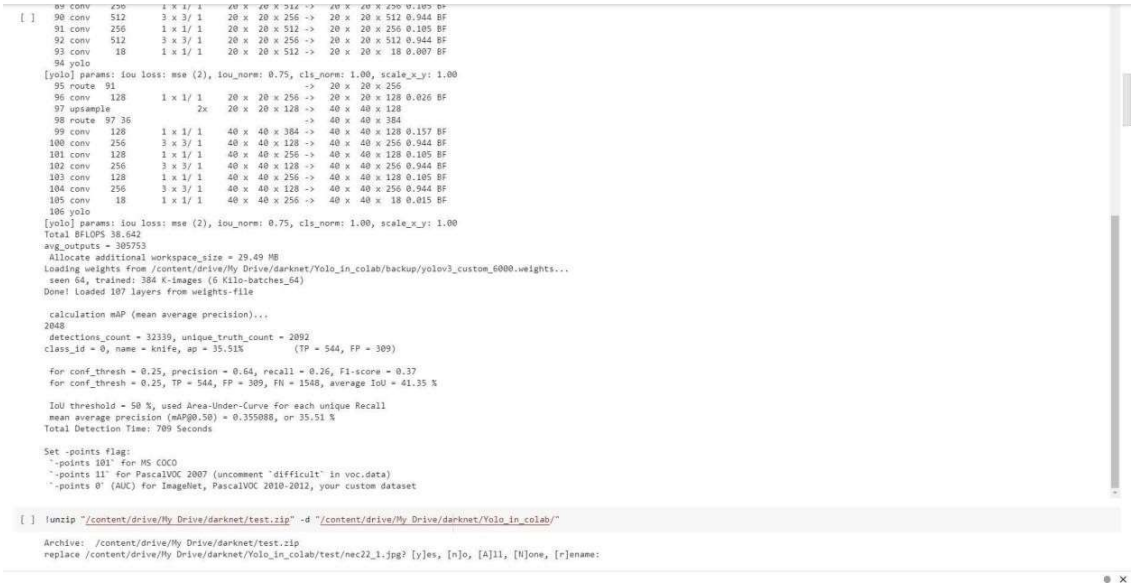


Figure 4.5 mAP Score Evaluation

The final samples from videos are attached as frames below:

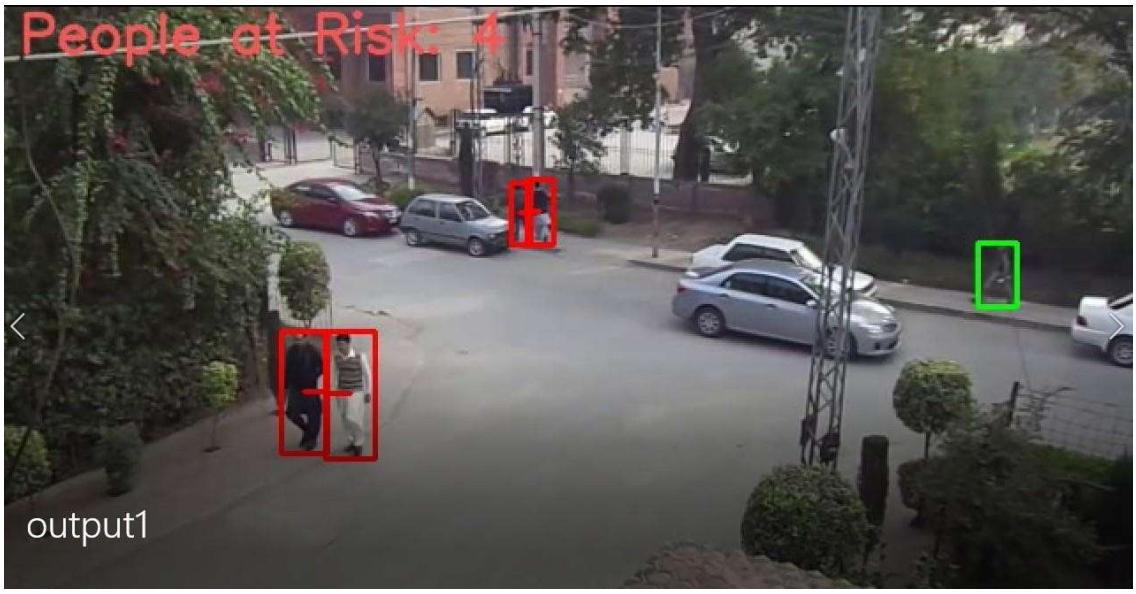


Figure 4.6 Output Sample 1

The above image is the final output. The model detects the object "Person" and here we have indicated the Red box to detect the persons who are nearby not having the distance criteria between the bounding boxes and the person having the distance criteria is indicated in a green bounding box such

that by finding the distance between the objects, we have implemented the model for developing an Social Distance Monitoring Application.

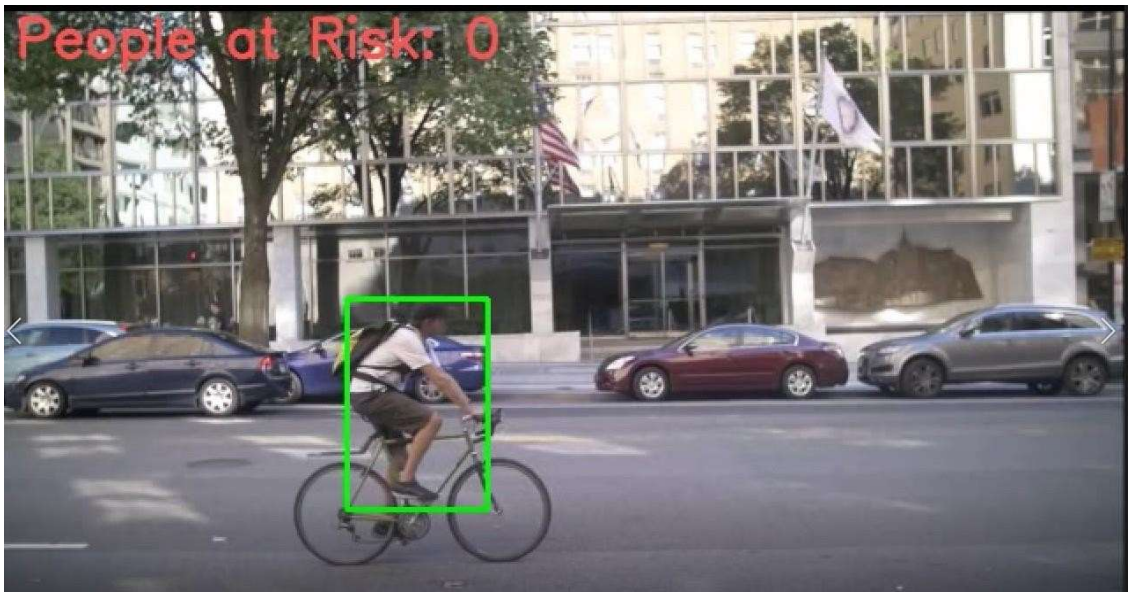


Figure 4.7 Output Sample 2

Figure 4.8 Output Sample 3

If the detected object has a distance of more than 100, then the bounding box is drawn as green by storing the object count in a green bounding box list. If the detected object is less, then a red bounding box is drawn to highlight and

monitor the people at risk of getting infected. The red bounded box counts are stored in a list and the number of people at risk is printed on top of the frame real-time to view the immediate number of people at risk of getting infected by the covid’19.

CHAPTER 5

CONCLUSION & FUTURE WORK

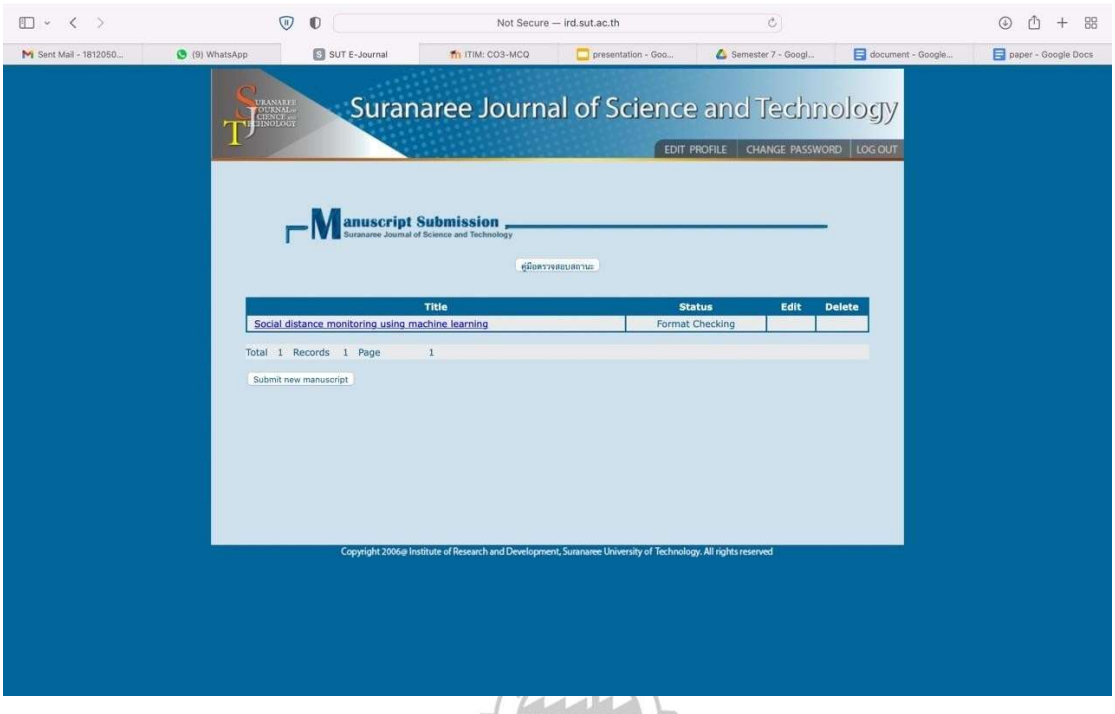
5.1 CONCLUSION

With the current technological advancements, it's no doubt that in the next few years there will be automated vehicle transportations and much more applications to be developed for monitoring such as the Social Distancing application. All the ML applications need to be maintained in an efficient manner, thus comes our trained model which helps machines to identify objects all around the world and finds the distance between them much more efficiently. With a much larger dataset and more iterated files can be more efficient than the current one, with much computational power we can power multiple camera feeds at the same time. Thus the outcome of this model can help people track, monitor and measure easier and save time for the advancement of our daily routine in the near future in almost all the fields.

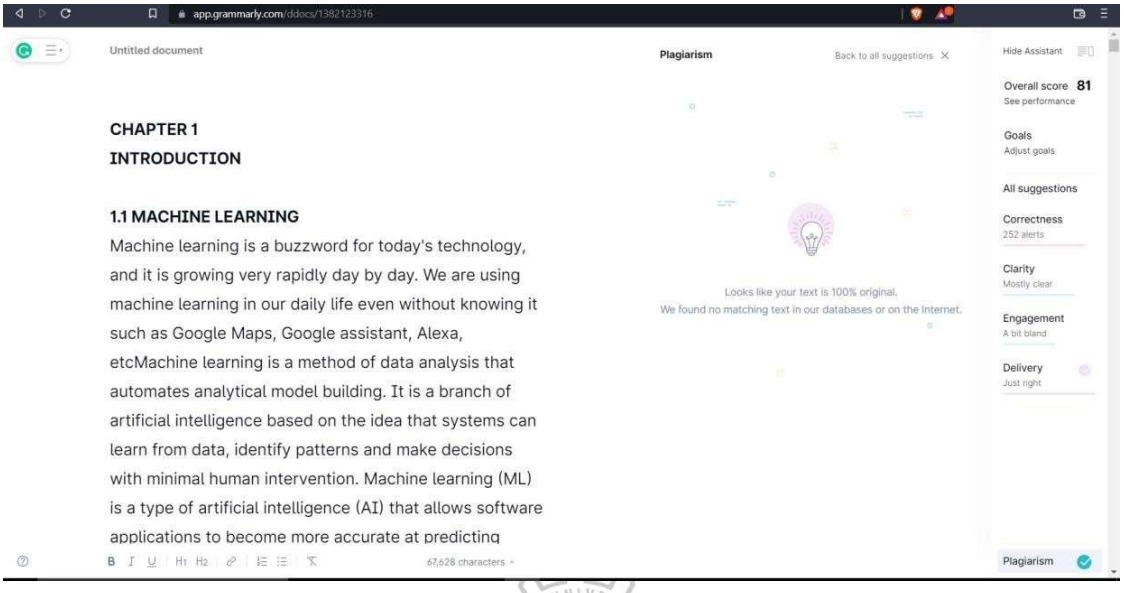
5.2 FUTURE WORKS

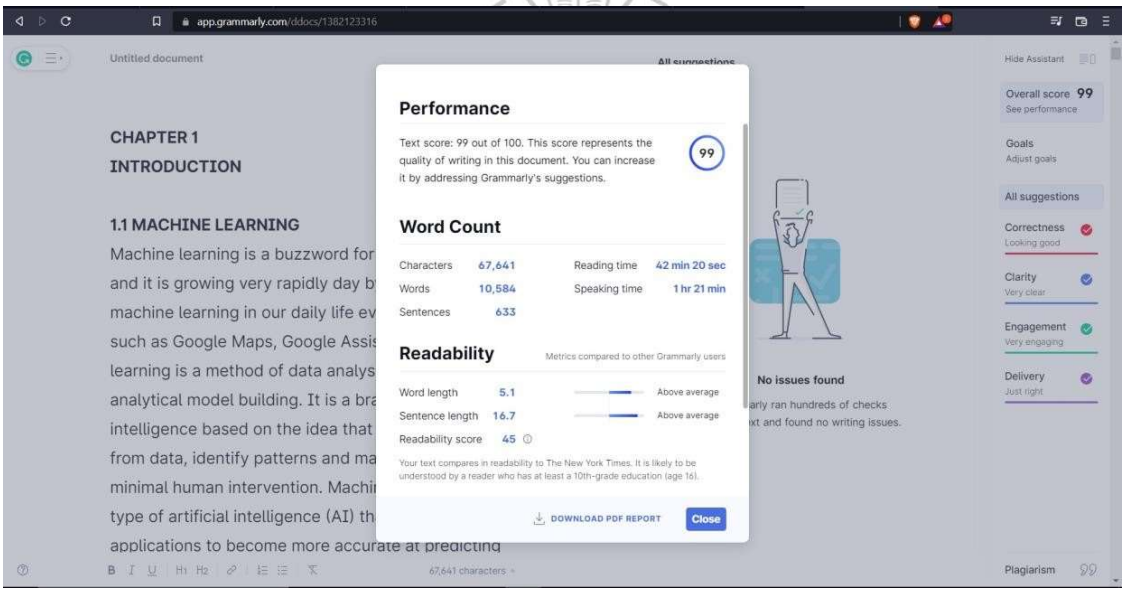
Further it was found that the finding the distance between the detected objects can be improved by calculating the depth of the image using computer vision. By applying differentiable warping to frames and comparing the result to adjacent ones, it provides several improvements. We can address occlusions geometrically and differentially, directly using the depth maps as predicted during training. We can introduce randomized layer normalization, a novel regularize, and we account for object motion relative to the scene.

OUTCOME PROOF



APPENDICES


app.grammarly.com/ddocs/1382123316
Untitled document
CHAPTER 1
INTRODUCTION
1.1 MACHINE LEARNING
Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etcMachine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting
67,628 characters
Plagiarism
Back to all suggestions
Looks like your text is 100% original.
We found no matching text in our databases or on the Internet.
Hide Assistant
Overall score 81
See performance
Goals
Adjust goals
All suggestions
Correctness
252 alerts
Clarity
Mostly clear
Engagement
A bit bland
Delivery
Just right
Plagiarism


app.grammarly.com/ddocs/1382123316
Untitled document
CHAPTER 1
INTRODUCTION
1.1 MACHINE LEARNING
Performance
Text score: 99 out of 100. This score represents the quality of writing in this document. You can increase it by addressing Grammarly's suggestions.
99
Word Count
Characters 67,641
Words 10,584
Sentences 633
Reading time 42 min 20 sec
Speaking time 1 hr 21 min
Readability
Metrics compared to other Grammarly users
Word length 5.1 Above average
Sentence length 16.7 Above average
Readability score 45
Your text compares in readability to The New York Times. It is likely to be understood by a reader who has at least a 10th-grade education (age 16).
DOWNLOAD PDF REPORT
Close
No issues found
Hide Assistant
Overall score 99
See performance
Goals
Adjust goals
All suggestions
Correctness
Looking good
Clarity
Very clear
Engagement
Very engaging
Delivery
Just right
Plagiarism

REFERENCES

1. Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee, 2019, 'Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving', Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 502–511.

2. Aleksey Bochkovsky, Chien-Yao Wang, Hong-Yuan Mark Liao, 2020, ' YOLOv4: Optimal Speed and Accuracy of Object Detection', Proceedings of the IEEE International Conference on Computer Vision (ICCV),250-258.

3. Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-IoU Loss, 2020, 'Faster and better learning for bounding box regression', In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).

4. Van Etten, A., 2019, 'Satellite imagery multiscale rapid detection with windowed networks', IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 735–743).

5. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei, 2015, 'ImageNet Large Scale Visual Recognition Challenge', Proceedings of the IEEE International Conference on Computer Vision (ICCV), 120-132.

6. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, 2016, 'You Only Look Once: Unified, Real-Time Object Detection', IEEE Winter Conference on Applications of Computer Vision (WACV) (pp.430–435).

7. Sara Beery, Guanhang Wu, Vivek Rathod, Ronny Votel, Jonathan Huang,2018, 'Context R-CNN: Long Term Temporal Context for Per-Camera Object Detection', California Institute of Technology presentation of Google(pp 340-351).

8. Mason Liu, 2019, 'Looking Fast and Slow: Memory-Guided Mobile Video Object Detection', IEEE International Conference on Computer Vision (ICCV), 230-236.

9. Pengchong Jin Vivek Rathod Xiangxin Zhu, 2019, 'Pooling Pyramid Network for Object Detection', Proceedings of the IEEE International Conference on Computer Object Detection (ICCOD), 375 - 382.

10. Joseph Redmon Ali Farhadi, 2018, 'YOLOv3: An Incremental Improvement' University of Washington, IEEE Transactions on Neural Networks and Learning Systems(ITNNL) (PP. 330-343).

11. Mingxing Tan Ruoming Pang Quoc V. Le, 2019, 'EfficientDet: Scalable and Efficient Object Detection', Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020).

12. Xingyi Zhou, UT Austin, Dequan Wang, 2019, 'Objects as Points', Journal of Machine Learning Research (pp. 210-220).

13. Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun, 2015, 'Deep Residual Learning for Image Recognition', Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

14. Jonathan Huang, Vivek Rathod, Chen Sun Menglong Zhu, Anoop Korattikara, 2016, 'Speed/accuracy trade-offs for modern convolutional object detectors', Journal of Machine Learning Research (pp. 109-121)

15. Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao, 2018, 'Scaled-YOLOv4: Scaling Cross Stage Partial Network', Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 170-182).

16. Jianxin Wu, 2017, 'A Filter Level Pruning Method for Deep Neural Network Compression', IEEE Transactions on Neural Networks and Learning Systems (pp. 540-553).

17. Saad Albawi, 2017, Conference: The International Conference on Engineering and Technology(pp. 734-748).

18. C. Tang, Y. Feng, X. Yang, C. Zheng and Y. Zhou, 2017, 'The Object Detection Based on Deep Learning', International Conference on Information Science and Control Engineering (ICISCE) (pp. 723-728).

19. R. Girshick, J. Donahue, T. Darrell and J. Malik, 2014, 'Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation', IEEE Conference on Computer Vision and Pattern Recognition (pp. 580-587).

20. B. Zhou, X. Wang, and X. Tang, 2012, 'Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents' , IEEE Conference on Computer Vision and Pattern Recognition. IEEE (pp. 2871-2878).