OVERCROWDED WARNING MECHANISM FOR UN-REGULATED PUBLIC SPACES

DISSERTATION

Submitted in partial fulfilment of the requirements of the MTech Data Science and Engineering Degree program

By

KSHITIZ KUMAR

2020SC04230

Under the supervision of

Parshwanath Desai

(Program Manager)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE Pilani (Rajasthan) INDIA

(March 2023)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Dissertation entitled "**Overcrowded Warning Mechanism for Un-regulated Public Spaces**" submitted by Mr **Kshitiz Kumar** ID No. **2020SC04230** in partial fulfilment of the requirements of DSECLZG628T Dissertation, embodies the work done by him/her under my supervision.

[illegible]

(Signature of the Supervisor)

Place: Bengaluru

Date: 14/03/2023

Parshwanath Desai

Program Manager, QuEST Global

ABSTRACT

Public open spaces which are meant for mass gathering or usage sometimes lack a regulatory mechanism to control the number of people utilizing them simultaneously. Unfortunate circumstances can cause stampedes in public spaces or breakage of manmade structures due to overload. These kinds of events could have been managed to some extent if there was any kind of auto-control mechanism to restrict the number of people entering when it is more than what the place or structure can sustain.

The technology of Computer Vision could be useful to find a potential solution to this problem. Nowadays most public spaces are surveillance under CCTV cameras for security purposes and they are often connected to a computer for storing and monitoring. We can utilize the concepts of Deep Learning to develop a model which can run on these computers and track the people’s count using the live feed from the CCTV camera. This tool then can alert the authorities if the people count is more than the safety standards of that place. Also, we can display a warning message at the entrance to make newcomers aware of the overcrowded condition.

As this model doesn’t require much infrastructure, it can be easily installed and can be widely used. Also, updates and maintenance can be performed remotely. This kind of regulatory mechanism can help government authorities to manage the usage of public spaces and ensure public safety and longevity of the construction.

LIST OF SYMBOLS & ABBREVIATIONS USED

- ANN - Artificial Neural Network
- SORT - Simple Online Real-time Tracking
- YOLO – You Look Only Once
- COCO – Common Objects in Context
- MLP – Multi-Layered Perceptron
- AI - Artificial Intelligence
- NN - Neural networks
- DNN - Deep Neural Networks
- CCTV - closed-circuit television
- PC - Personal Computer
- NMS - Non-Maximum Suppression
- CSP - Cross Stage Partial
- SPP - Spatial Pyramid Pooling
- PANet - Path Aggregation Network
- SPP - Spatial Pyramid Pooling
- SiLU - Sigmoid Linear Unit
- CIoU - Complete Intersection over Union
- IOU - intersection-over-union
- FPS – Frame per Second

LIST OF FIGURES

Table of Contents

Chapter 01

INTRODUCTION

Places like temples, music concerts, tourist destinations, etc. sometimes face unfortunate events like stampedes or structural damage due to a large number of people visiting them simultaneously. Due to the absence of a regulation mechanism in open spaces, it becomes difficult to manage the number of people. There is a need for a warning mechanism that can alert the authorities in case of count more than what it can sustain.

Computer Vision is a field of Artificial Intelligence. It enables computers to detect and track various objects from photos and video. It works on the principles of Deep Learning. Deep learning

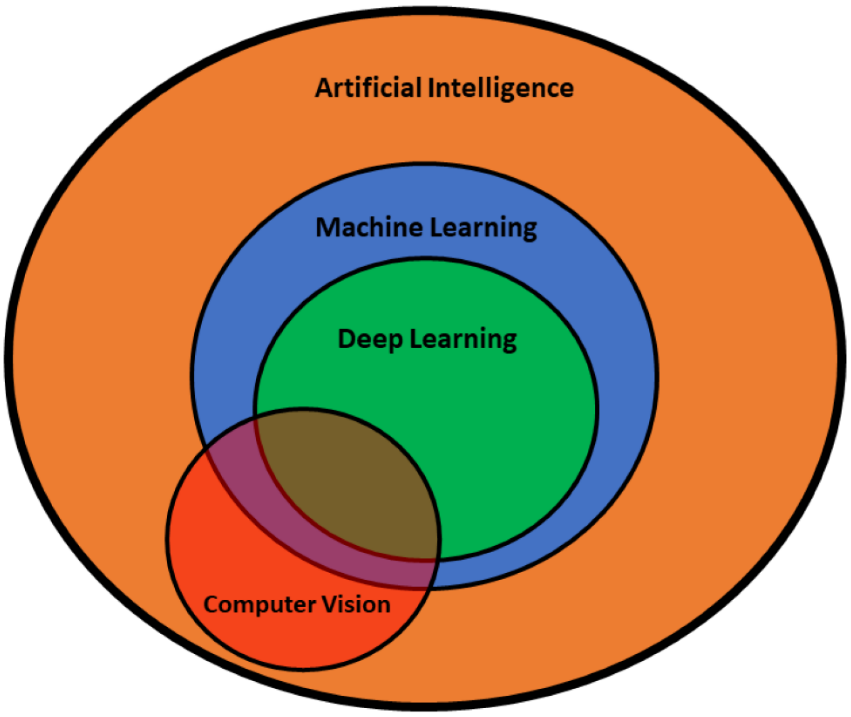


Figure 1. Venn diagram of Artificial Intelligence, Machine Learning, Deep Learning, and Computer Vision

1.1 Deep Learning

Deep learning generates insights using *Neural Networks*. A neural network consists of a dense connection of artificial neurons called *Perceptron.* A deeply connected network of perceptron, also called MLP (Multi-layers perceptron) forms a neural network.

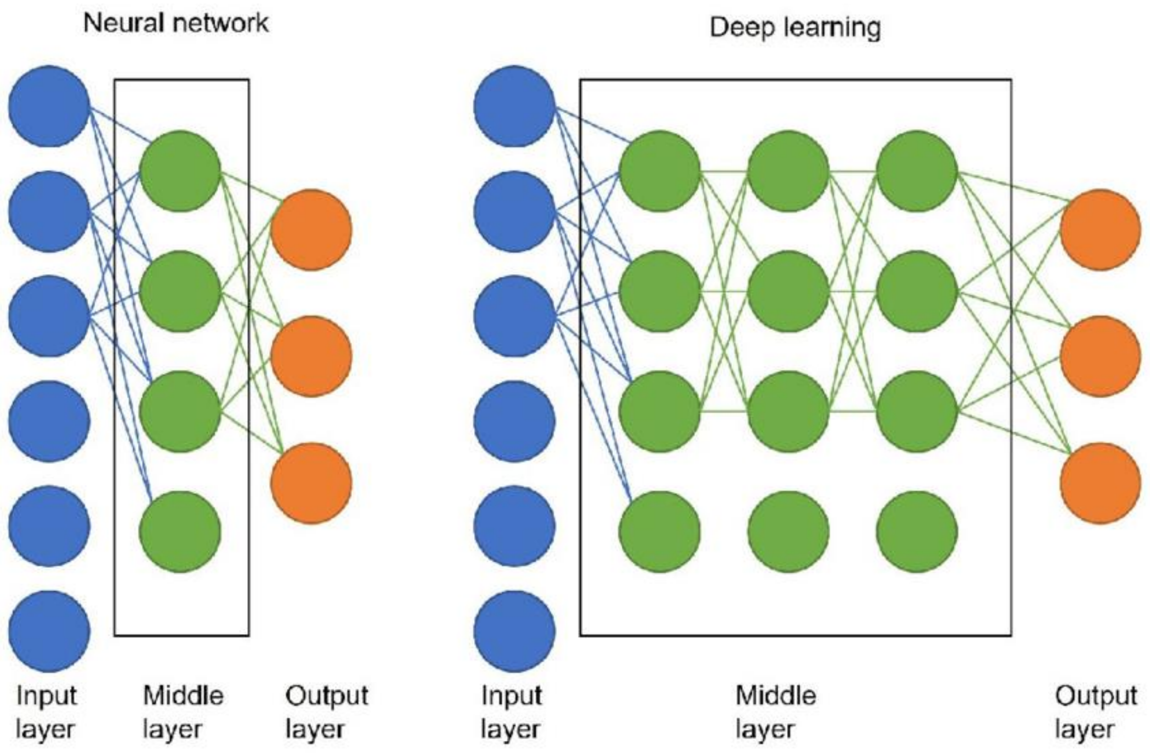


Figure 2: Deep learning and a neural network

1.1.1 Neural Network

Neural networks (NN) also known as Artificial Neural Networks (ANN) are inspired by neurons in the Human brain. An ANN usually involves a large number of processors operating in parallel and arranged in tiers. The first tier receives the raw input information -- analogous to optic nerves in human visual processing. Each successive tier receives the output from the tier preceding it, rather than the raw input -- in the same way neurons further from the optic nerve receive signals from those closest to it. The last tier produces the output of the system.

Most of today's neural nets are organized into layers of nodes, and they're "feed-forward," meaning that data moves through them in only one direction. An individual node might be connected to several nodes in the layer beneath it, from which it receives data, and several nodes in the layer above it, to which it sends data. However, you can also train your model through backpropagation; that is, move in opposite direction from output to input. Backpropagation allows us to calculate and attribute the error associated with each neuron, allowing us to adjust and fit the algorithm appropriately.

To each of its incoming connections, a node will assign a number known as a “weight.” When a neural net is being trained, all of its weights and thresholds are initially set to random values. Training data is fed to the bottom layer — the input layer — and it passes through the succeeding layers, getting multiplied and added together in complex ways, until it finally arrives, radically transformed, at the output layer. During training, the weights and thresholds are continually adjusted until training data with the same labels consistently yield similar outputs.

All layers between Input and Output are called Hidden layers.

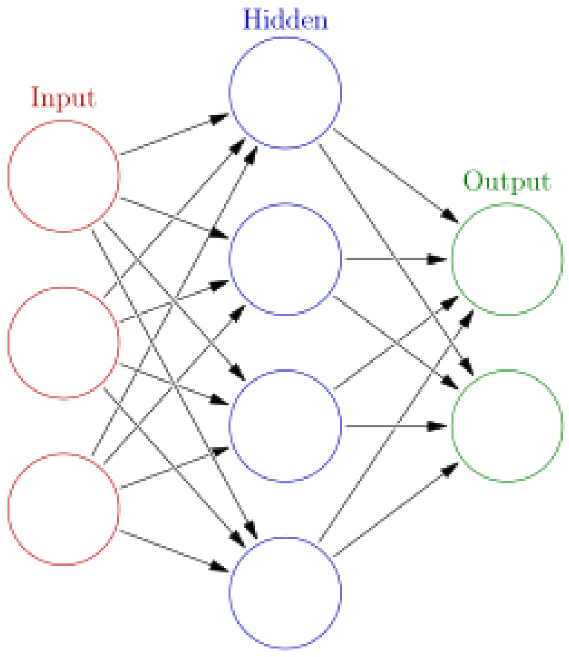


Figure 3: Neural Network Architecture

1.1.2 Deep Learning

In general Deep Neural Network is a complex form of Neural Network with many Hidden layers. A neural network that consists of more than three layers—which would be inclusive of the inputs and the output—can be considered a deep learning algorithm. Adding multiple hidden layers makes it computationally very expensive.

Deep Neural Networks (DNNs) are such types of networks where each layer can perform complex operations such as representation and abstraction that make sense of images, sound, and text. Deep learning has aided image classification, language translation, and speech recognition. It can be used to solve any pattern recognition problem and without human intervention.

Deep learning systems require large amounts of data to return accurate results; accordingly, information is fed as huge data sets. When processing the data, artificial neural networks are able to classify data with the answers received from a series of binary true or false questions involving highly complex mathematical calculations.

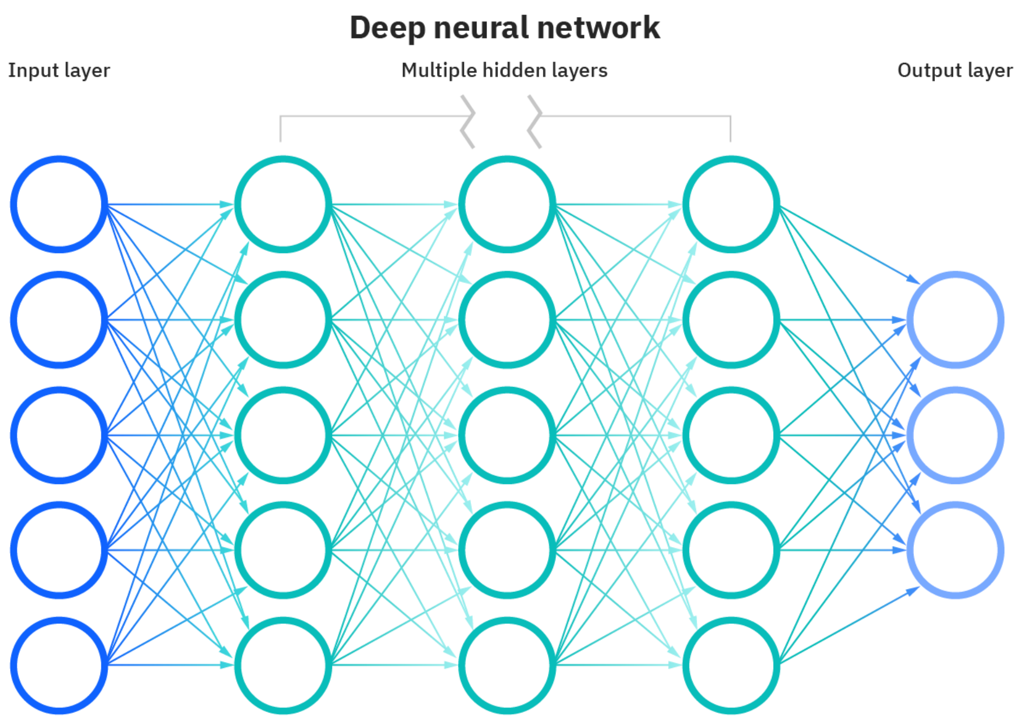


Figure 4: Deep Neural Network Architecture

1.2 Computer Vision

Using reference images the Computer Vision model uses deep networks to train itself and generate optimal values of weights. The trained model can then be used to detect objects in new photos or videos. Latest advancements enabled it to produce a high-quality result in a live video feed.

Below are a few key aspects that Computer Vision seeks to recognize in the photographs:

- Object Recognition: The objects in the image, and their positions.
- Object Classification: The broad category that the object lies in.
- Object Detection: The location of the object.
- Object Segmentation: The pixels belonging to that object

1.2.1 Object Recognition

Object recognition is the technique of identifying the object present in images and videos. It is one of the most important applications of machine learning and deep learning. The goal of this field is to teach machines to understand (recognize) the content of an image just like humans do.

1.2.2 Object Classification

In Image classification, it takes an image as an input and outputs the classification label of that image with some metric (probability, loss, accuracy, etc.). For example, an image of a car must be labelled as CAR and an image of a truck must be labelled as TRUCK etc.

1.2.3 Object Detection

Object Detection algorithms act as a combination of image classification and object localization. It takes an image as input and produces one or more bounding boxes with the class label attached to each bounding box. These algorithms are capable enough to deal with multi-class classification and localization as well as to deal with the objects with multiple occurrences.

1.2.4 Object Segmentation

Object segmentation is a further extension of object detection in which we mark the presence of an object through pixel-wise masks generated for each object in the image. This technique is more granular than bounding box generation because this helps us in determining the shape of each object present in the image. This granularity helps us in various fields such as medical image processing, satellite imaging, etc.

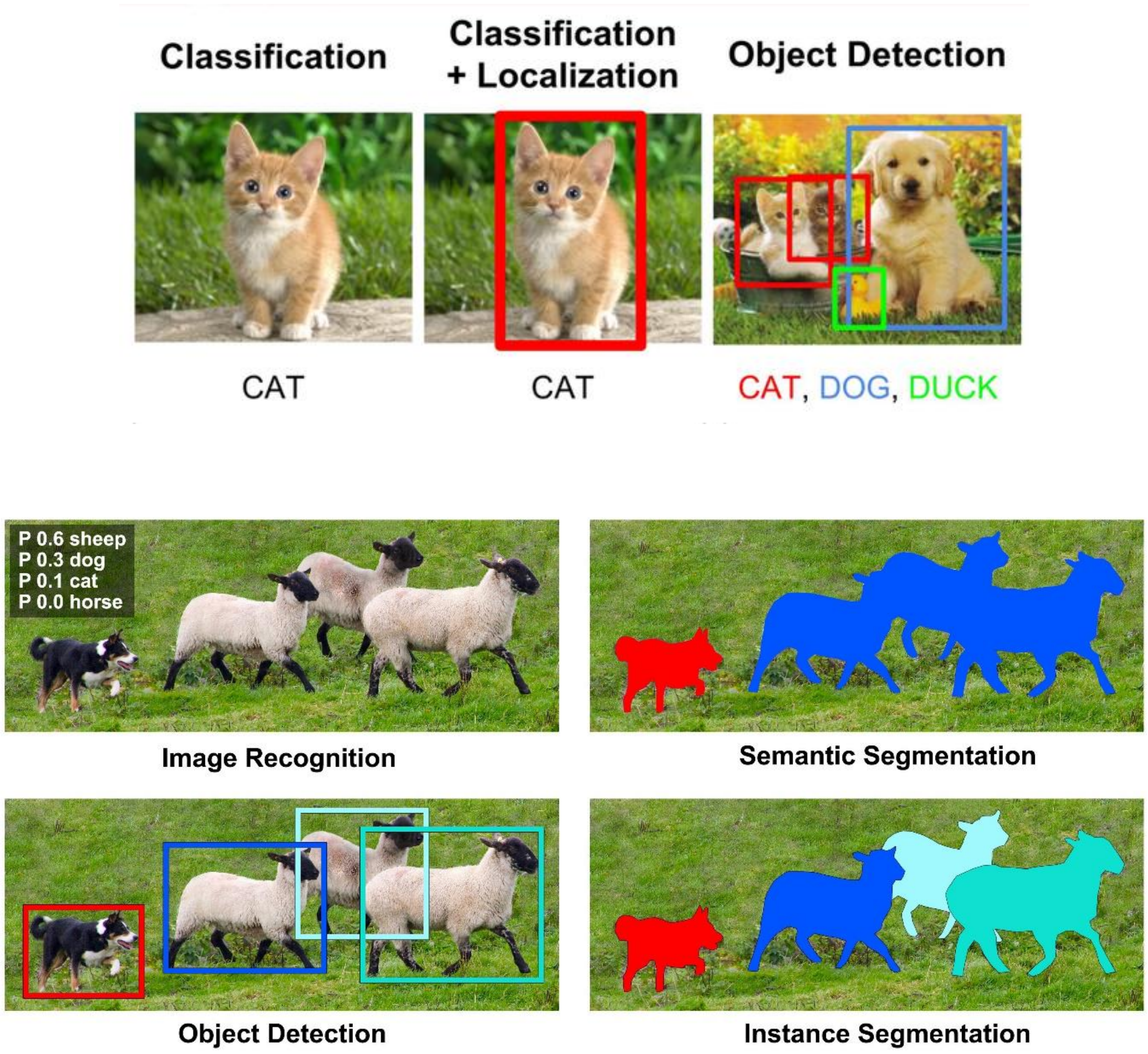


Figure 5: Computer Vision Aspects

1.3 Problem Statement

Places like temples, music concerts, tourist destinations, etc. sometimes face unfortunate events like stampedes or structural damage due to a large number of people visiting them simultaneously. Due to the absence of a regulation mechanism in open spaces, it becomes difficult to manage the number of people. There is a need for a warning mechanism that can alert the authorities in case of count more than what it can sustain.

This kind of regulatory mechanism can help government authorities to manage the usage of public spaces and ensure public safety and longevity of the construction.

1.4 Project Overview

The objective is to develop a tool that can count the number of people present at the venue. And, alert if the people count is more than the safety threshold. The safety threshold is the maximum number of people the venue can sustain without damaging the structure. It can also determine based upon the available exits facility which can allow them to exit the venue safely in case of any miss-happening.

The tool works with,

- Limited data input:
    - It can work with the live video feed from the CCTV cameras, generally installed at the entrance and exit.
- Limited infrastructure:
    - The tool can be installed on the PC connected to these cameras, with one PC consolidating the total count

And is,

- Cost-effective:
    - It can work with the existing infrastructure with a slight upgrade
- Easy to implement

It is developed in Python language using the deep learning algorithms of Computer Vision. It takes a video file as input. Humans are detected and their movement is tracked at the exit or entry gate. A number of people entering and exiting are monitored to obtain the current number of people at the premises.

Chapter 02

LITERATURE REVIEW

2.1 Computer Vision Algorithms

Since the inception of Computer Vision, many algorithms have been developed to enhance computer vision abilities. These algorithms can be broadly divided into two categories.

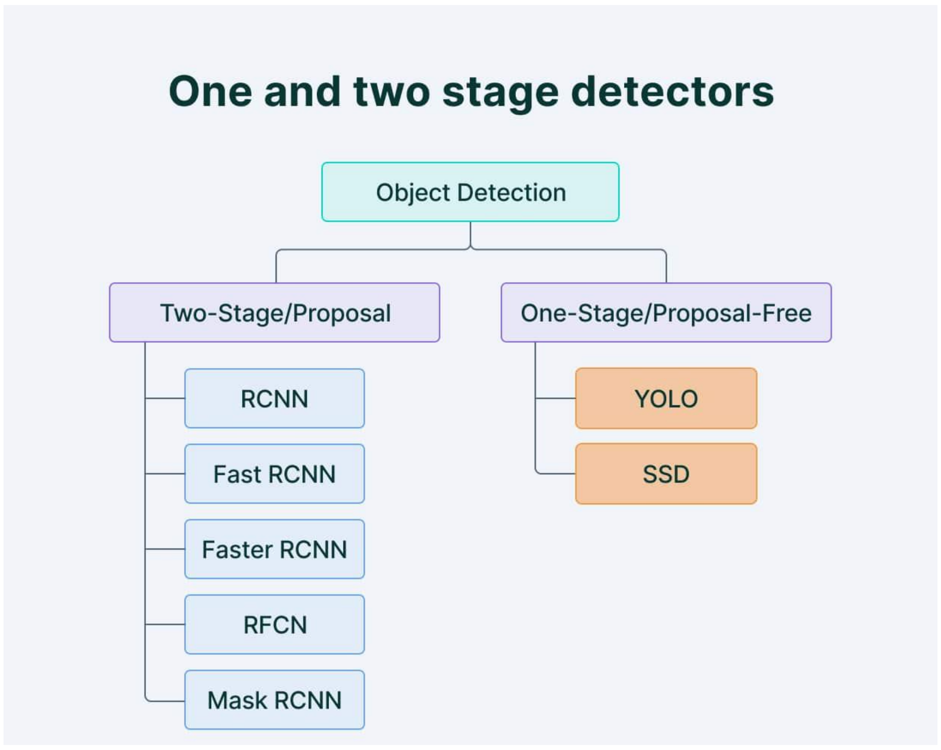


Figure 6: Object Detection Algorithms

Two-shot object detection

Two-shot object detection uses two passes of the input image to make predictions about the presence and location of objects. The first pass is used to generate a set of proposals or potential object locations, and the second pass is used to refine these proposals and make final predictions. This approach is more accurate than single-shot object detection but is also more computationally expensive.

Single-shot object detection

Single-shot object detection uses a single pass of the input image to make predictions about the presence and location of objects in the image. It processes an entire image in a single pass, making them computationally efficient.

However, single-shot object detection is generally less accurate than other methods, and it's less effective in detecting small objects. Such algorithms can detect objects in resource-constrained environments in real-time.

Overall, the choice between single-shot and two-shot object detection depends on the specific requirements and constraints of the application.

Generally, single-shot object detection is better suited for real-time applications, while two-shot object detection is better for applications where accuracy is more important.

Fast R-CNN

Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9x faster than R-CNN, is 213x faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3x faster, tests 10x faster, and is more accurate.

Reference - https://arxiv.org/abs/1504.08083

Faster R-CNN

State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet and Fast R-CNN have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. It uses a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection.

Reference - https://arxiv.org/abs/1506.01497

YOLO

YOLO, 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.

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 far less likely to predict false detections where nothing exists. Finally, YOLO learns very general representations of objects.

Reference - https://arxiv.org/abs/1506.02640

Chapter 03

DEEP LEARNING MODEL – YOLOv5

To count the persons we require such a model which can generate real-time results. As it is critical to raise the alarm when the count is more than the threshold. In this project, we will be using the YOLO algorithm to develop the computer vision model. The main reason to use this model is because of its ability to produce almost instant results. Also, YOLO is currently the latest model developed in this area.

YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image.

YOLO divides an input image into an S × S grid. If the centre 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 reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is non-maximum suppression (NMS). NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

Some of the reasons why YOLO is leading the competition include its:

- Speed
- Detection accuracy
- Good generalization
- Open-source

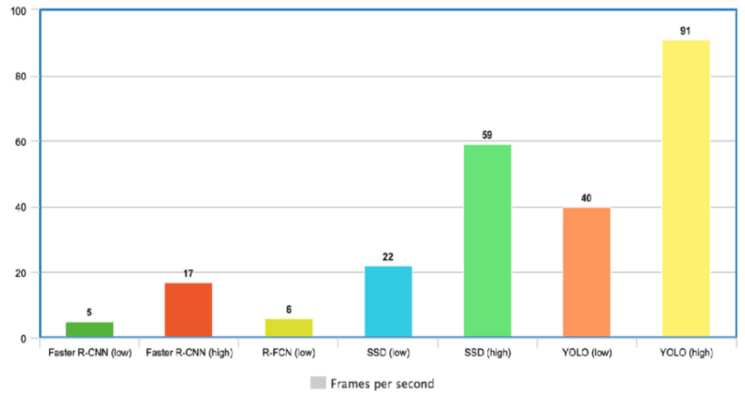


Figure 7: YOLO speed comparison

3.1 Introduction

YOLOv5 was released in five different sizes:

- **n** for extra small (Nano) size model.
- **s** for the small-size model.
- **m** for medium size model.
- **l** for large size model
- **x** for extra-large size model

These model only differ in terms of the number of layers and parameters. Their working principle remains the same. But due to the difference in the number of layers and parameters, their Speed and Accuracy change significantly. Heavier the model, the lower will be its speed and the higher will be its accuracy.

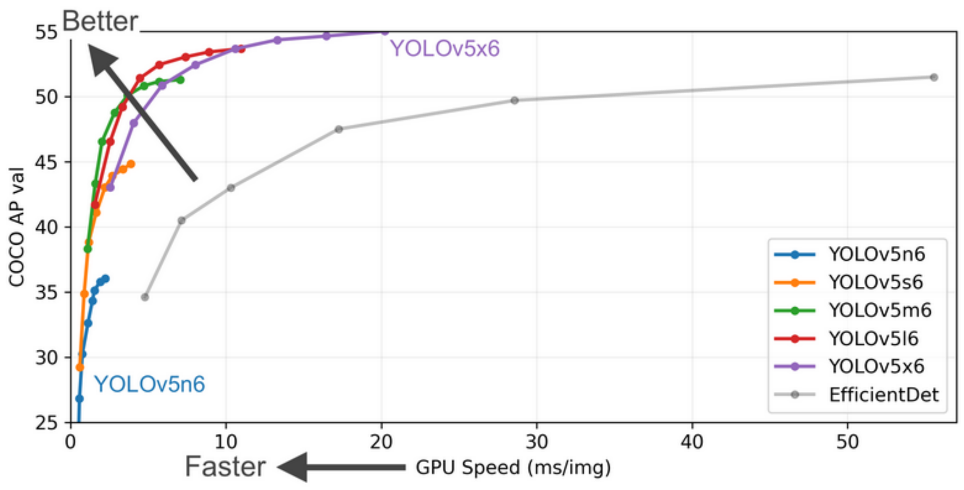
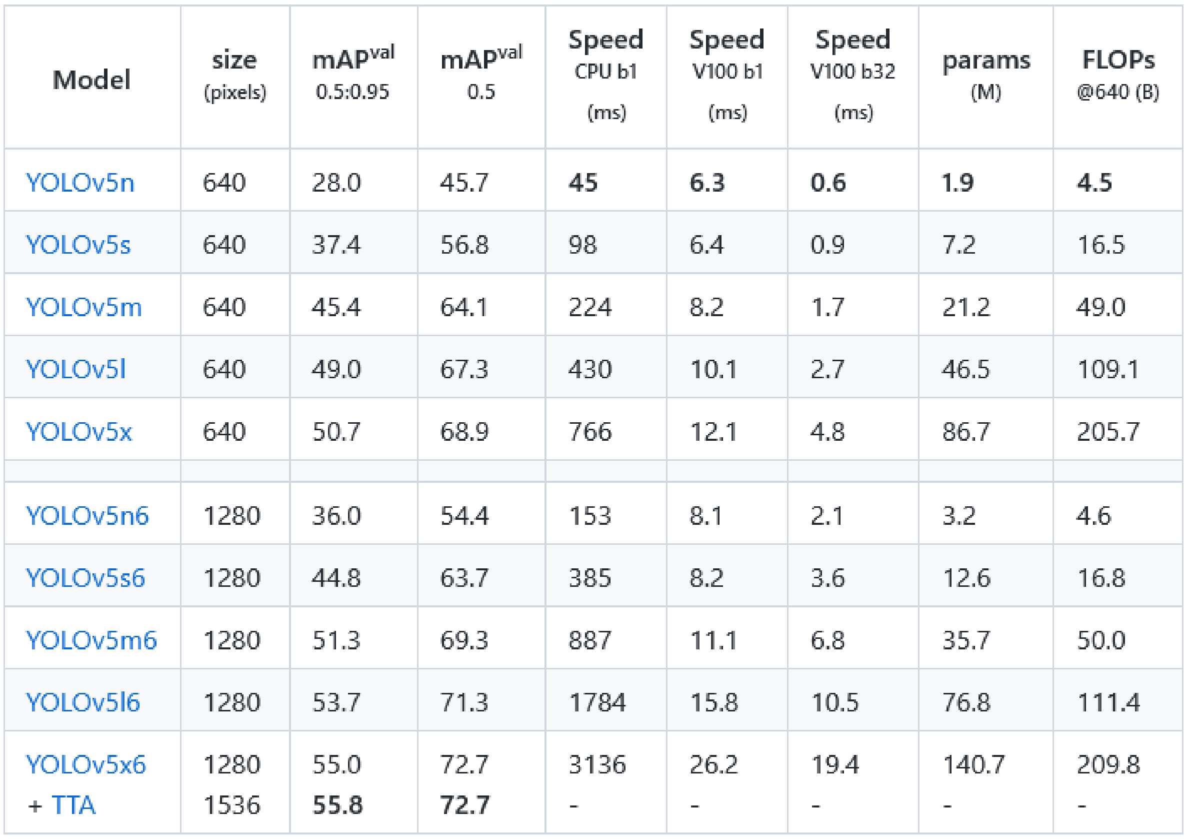


Figure 8: YOLOv5 models speed comparison



Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

3.2 YOLOv5 Architecture

All the YOLOv5 models are composed of the same 3 components: **CSP-Darknet53** as a backbone, **SPP** and **PANet** in the model neck and the **head** used in YOLOv4.

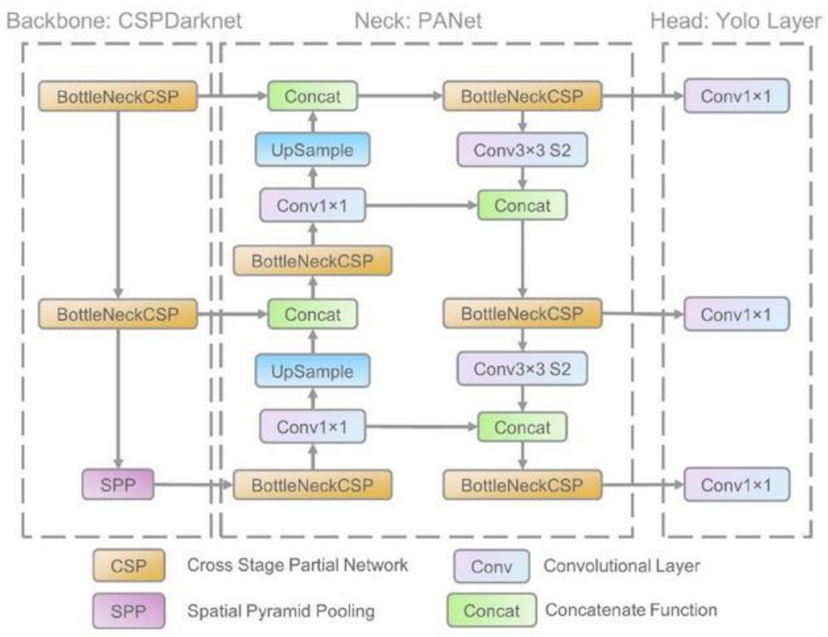


Figure 9: YOLOv5 Architecture

3.2.1 Backbone of YOLOv5

CSP-Darknet53

YOLOv5 uses CSP-Darknet53 as its backbone. CSP-Darknet53 is just the convolutional network **Darknet53** used as the backbone for YOLOv3 to which the authors applied the **Cross Stage Partial** (CSP) network strategy.

Cross-Stage Partial Network

YOLO is a deep network, it uses residual and dense blocks in order to enable the flow of information to the deepest layers and to overcome the **vanishing gradient problem**. However one of the perks of using dense and residual blocks is the problem of **redundant gradients**. CSPNet helps tackle this problem by truncating the gradient flow.

3.2.2 Neck of YOLOv5

YOLOv5 brought two major changes to the model neck. First, a variant of Spatial Pyramid Pooling (SPP) has been used, and the Path Aggregation Network (PANet) has been modified by incorporating the BottleNeckCSP in its architecture.

Path Aggregation Network (PANet)

PANet is a feature pyramid network, it has been used in the previous version of YOLO (YOLOv4) to improve information flow and to help in the proper localization of pixels in the task of mask prediction. In YOLOv5 this network has been modified by applying the CSPNet strategy to it as shown in the network's architecture figure.

Spatial Pyramid Pooling (SPP)

SPP block [4] performs an aggregation of the information that receives from the inputs and returns a fixed-length output. Thus it has the advantage of significantly increasing the receptive field and segregating the most relevant context features without lowering the speed of the network. This block has been used in previous versions of YOLO (yolov3 and yolov4) to separate the most important features from the backbone, however, in YOLOv5 (6.0/6.1) SPPF has been used, which is just another variant of the SPP block, to improve the speed of the network

3.2.3 Head of YOLOv5

YOLOv5 uses the same head as YOLOv3 and YOLOv4. It is composed of three convolution layers that predict the location of the bounding boxes, the scores and the objects classes.

3.3 YOLOv5 Activation Function

It uses SiLU (Sigmoid Linear Unit) and the Sigmoid activation function. SiLU has been used in the convolution operations in the hidden layers and the Sigmoid activation function has been used with the convolution operations in the output layer.

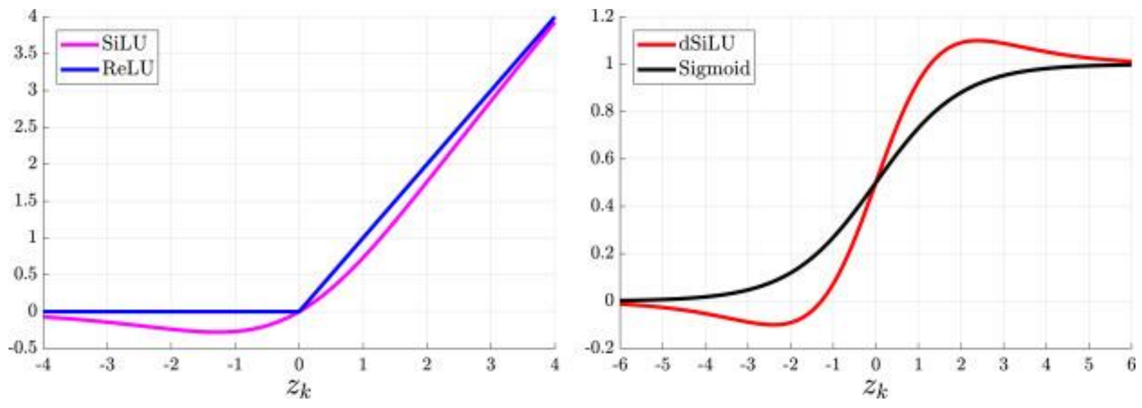


Figure 10: YOLOv5 Activation Functions

3.4 YOLOv5 Loss Function

YOLOv5 returns three outputs: the classes of the detected objects, their bounding boxes and the confidence scores**.** Thus, it uses **BCE** (Binary Cross Entropy) to compute the class's loss and the objectness loss. While **CIoU** (Complete Intersection over Union) loss to compute the location loss. The formula for the final loss is given by the following equation,

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$$

Chapter 04

MODEL TRAINING DATASET – COCO

The YOLO model has been pre-trained over COCO (Common Objects in Context) image dataset. COCO dataset is a large-scale object detection, image segmentation, and captioning dataset published by Microsoft. Machine Learning and Computer Vision engineers popularly use the COCO dataset for various computer vision projects.

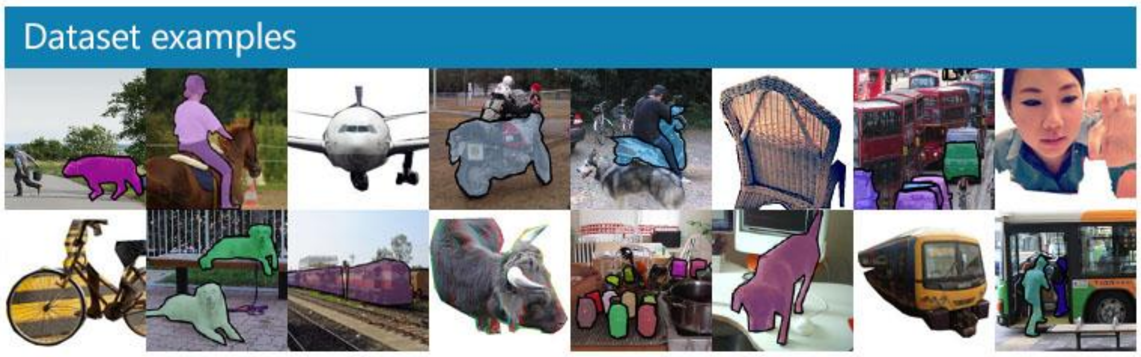


Figure 11: COCO dataset example

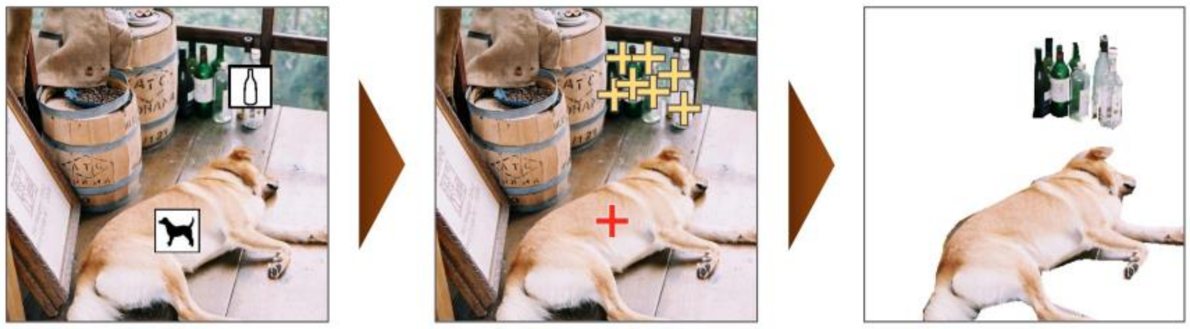
COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labelled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with key points

4.1 Annotation Pipeline

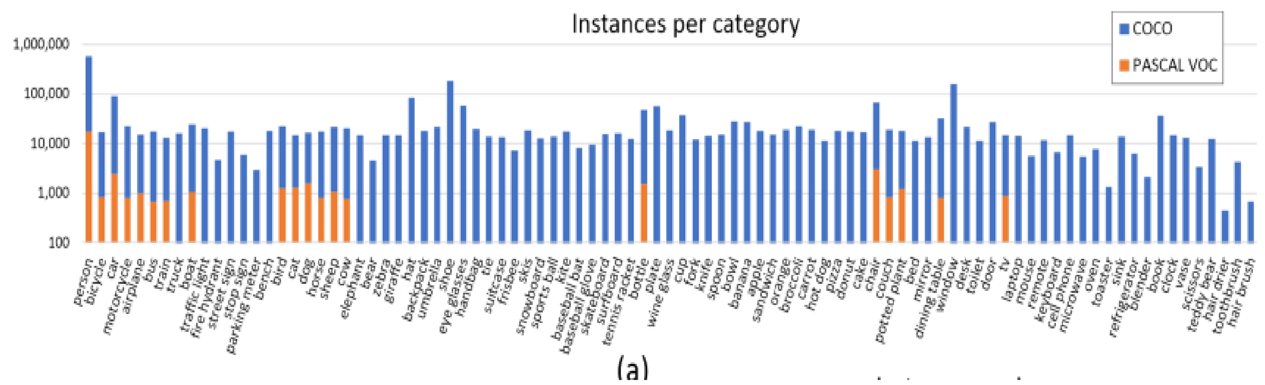
The annotation pipeline is split into 3 primary tasks:

a) labelling the categories present in the image
b) locating and marking all instances of the labelled categories
c) segmenting each object instance


(a) Category labeling (b) Instance spotting (c) Instance segmentation

4.2 COCO vs other Datasets

Other dataset include ImageNet [1], PASCAL VOC 2012 [2], and SUN [3]. Each of these datasets varies significantly in size, list of labelled categories and types of images. ImageNet was created to capture a large number of object categories, many of which are fine-grained. SUN focuses on labelling scene types and the objects that commonly occur in them. Finally, PASCAL VOC's primary application is object detection in natural images. MS COCO is designed for the detection and segmentation of objects occurring in their natural context.



(a)

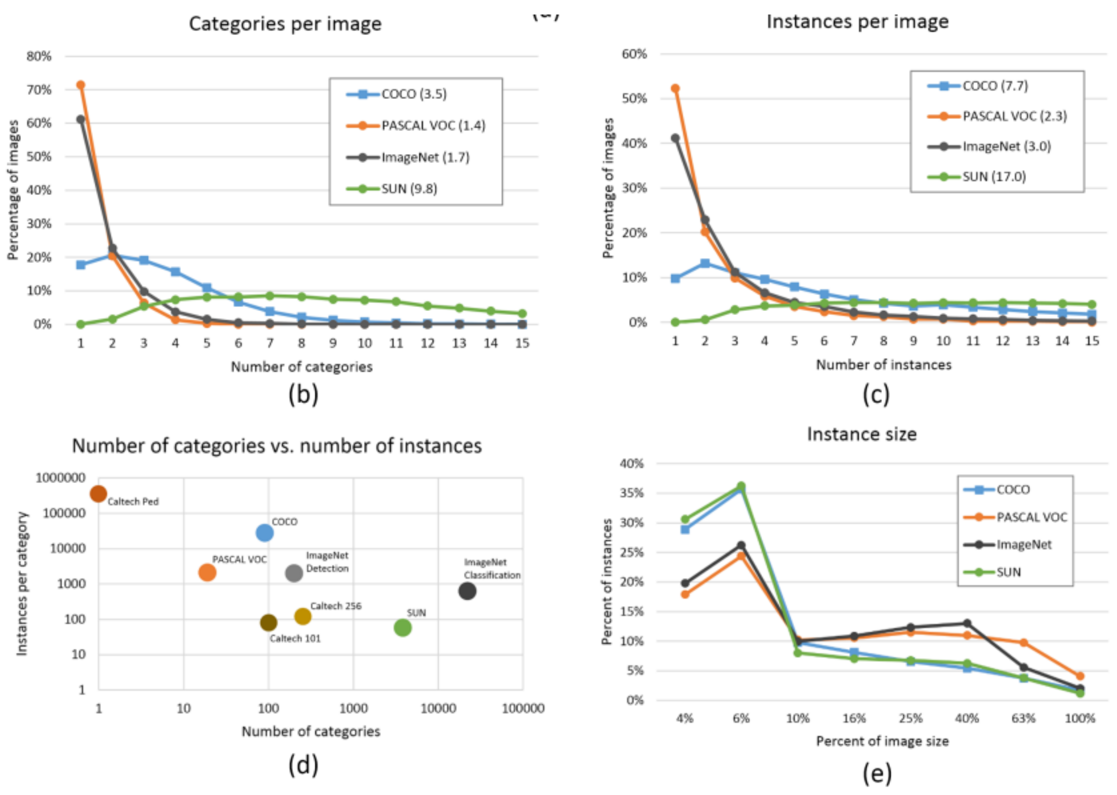


Figure 12: COCO vs other Dataset

Chapter 05

OBJECT TRACKING - DEEP SORT

Tracking in deep learning is the task of predicting the positions of objects throughout a video using their spatial as well as temporal features. More technically, Tracking is getting the initial set of detections, assigning unique ids, and tracking them throughout frames of the video feed while maintaining the assigned ids. Tracking is generally a two-step process:

1. **A detection module for target localization:** The module responsible for detecting and localization of the object in the frame using some object detector like YOLOv4, CenterNet, etc.
2. **A motion predictor**: This module is responsible for predicting the future motion of the object using its past information.

5.1 Introduction

DeepSORT is a computer vision tracking algorithm for tracking objects while assigning an ID to each object. DeepSORT is an extension of the SORT (Simple Online Real time Tracking) algorithm. DeepSORT introduces deep learning into the SORT algorithm by adding an appearance descriptor to reduce identity switches, Hence making tracking more efficient.

5.2 Simple Online Real time Tracking (SORT)

SORT is an approach to object tracking where rudimentary approaches like Kalman filters and Hungarian algorithms are used to track objects and claim to be better than many online trackers. SORT is made of 4 key components which are as follows:

- **Detection:** This is the first step in the tracking module. In this step, an object detector detects the objects in the frame that are to be tracked. These detections are then passed on to the next step. Detectors like FrRCNN, YOLO, and more are most frequently used.
- **Estimation:** In this step, it propagate the detections from the current frame to the next which is estimating the position of the target in the next frame using a constant velocity model. When detection is associated with a target, the detected bounding box is used to update the target state where the velocity components are optimally solved via the Kalman filter framework
- **Data association:** We now have the target bounding box and the detected bounding box. So, a cost matrix is computed as the intersection-over-union (IOU) distance between each detection and all predicted bounding boxes from the existing targets. The assignment is solved optimally using the Hungarian algorithm. If the IOU of detection and target is less than a certain threshold value called IOU_{min} then that

assignment is rejected. This technique solves the occlusion problem and helps maintain the IDs.

- **Creation and Deletion of Track Identities:** This module is responsible for the creation and deletion of IDs. Unique identities are created and destroyed according to the IOU_{min}. If the overlap of detection and target is less than IOU_{min} then it signifies the untracked object. Tracks are terminated if they are not detected for TLost frames, you can specify what the amount of frame should be for TLost. Should an object reappear, tracking will implicitly resume under a new identity.

5.3 Deep Sort

SORT performs very well in terms of tracking precision and accuracy. But SORT returns track with a high number of ID switches and fails in case of occlusion. This is because of the association matrix used. DeepSORT uses a better association metric that combines both motion and appearance descriptors. DeepSORT can be defined as the tracking algorithm which tracks objects not only based on the velocity and motion of the object but also the appearance of the object.

For the above purposes, a well-discriminating feature embedding is trained offline just before implementing tracking. The network is trained on a large-scale person re-identification dataset making it suitable for tracking context. To train the deep association metric model in the DeepSORT cosine metric learning approach is used.

According to DeepSORT's paper, "The cosine distance considers appearance information that is particularly useful to recover identities after long-term occlusions when motion is less discriminative." That means cosine distance is a metric that helps the model recover identities in case of long-term occlusion and motion estimation also fails. Using these simple things can make the tracker even more powerful and accurate.

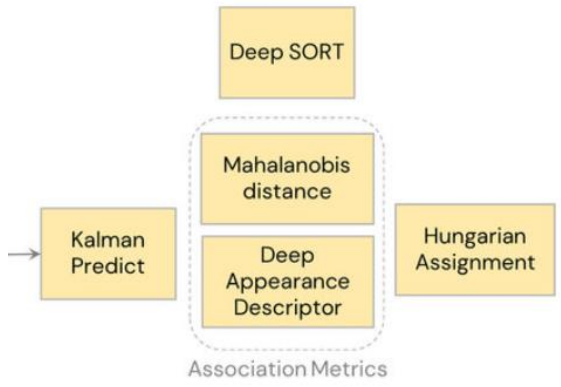


Figure 13: Deepsort Architecture

Chapter 06

APPLICATION OVERVIEW

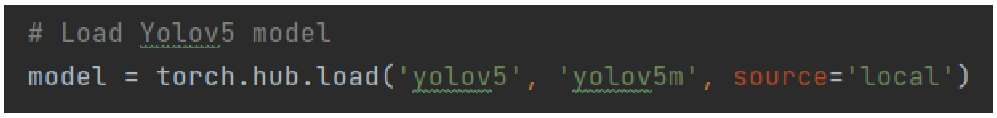
6.1 Program Input/output

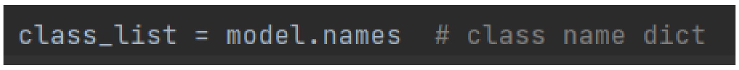
Tool Input – Sample video feed from CCTV

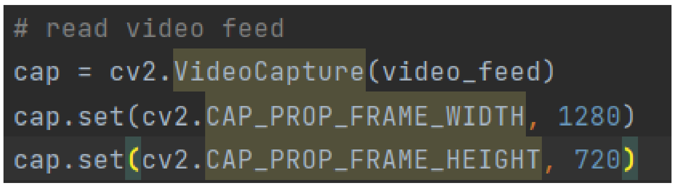
Tool Output – Count of number of people inside the premises at a given time

A video file consists of multiple frames. Each frame is like an image. Object detection is performed on each frame which is later combined to track the object's movement.

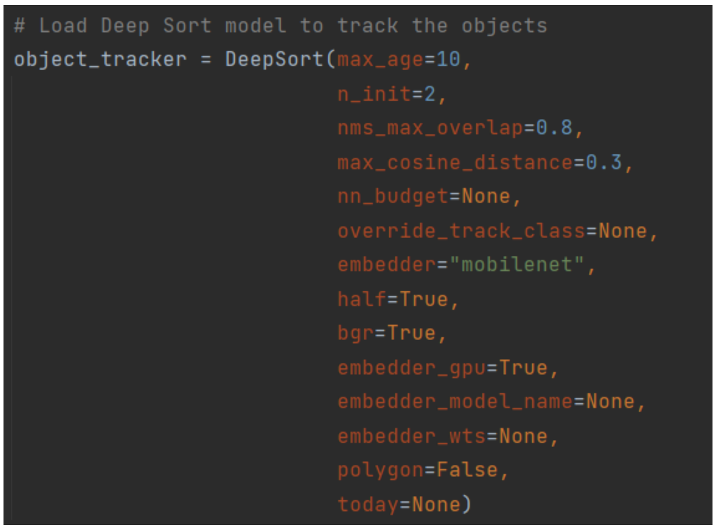
6.2 Code Overview

- The tool works with a pre-trained YOLOv5 model.
  yolov5m - This file is the medium-size yolov5 model developed on the PyTorch framework.
  
  ```
  # Load Yolov5 model
  model = torch.hub.load('yolov5', 'yolov5m', source='local')
  ```
  The model is stored locally instead of running directly from its Git repository. This saves time and increases its speed to some extent.

- **model.names** – This has a list of all class names which can be identified by the model
  
  ```
  class_list = model.names  # class name dict
  ```

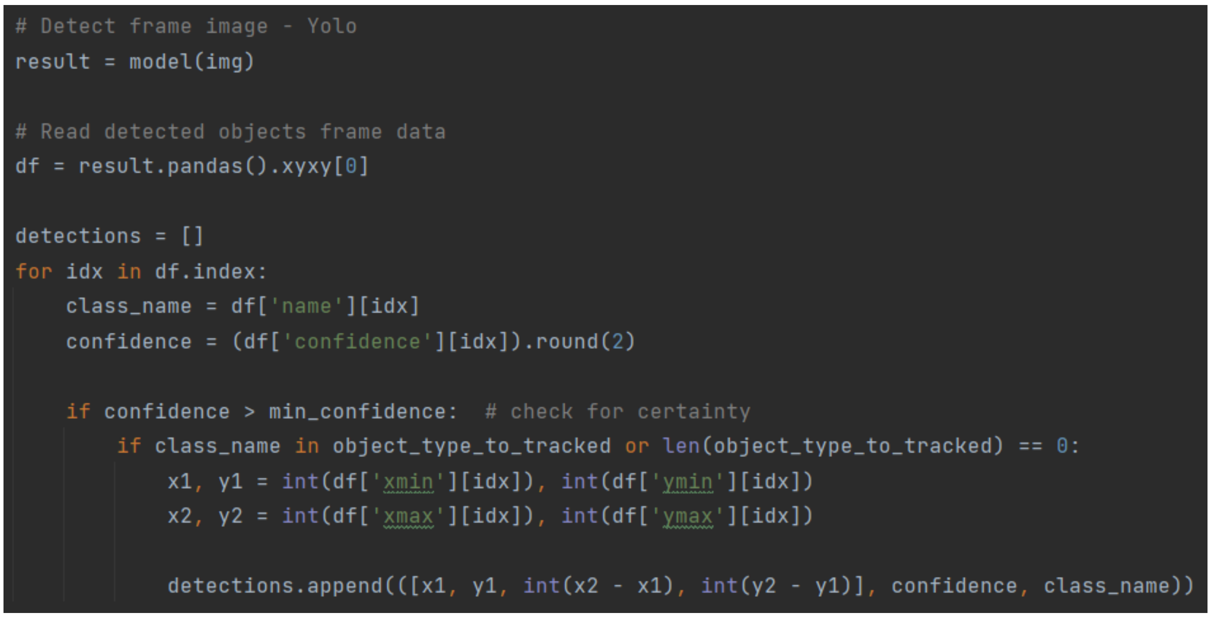
- **VideoCapture** – The video file is read by using OpenCV function
  
  ```
  # read video feed
  cap = cv2.VideoCapture(video_feed)
  cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
  cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
  ```

- **Deep Sort –** This model ensures tracking of the identified object in every frame of the video.



```python
# Load Deep Sort model to track the objects
object_tracker = DeepSort(max_age=10,
                          n_init=2,
                          nms_max_overlap=0.8,
                          max_cosine_distance=0.3,
                          nn_budget=None,
                          override_track_class=None,
                          embedder="mobilenet",
                          half=True,
                          bgr=True,
                          embedder_gpu=True,
                          embedder_model_name=None,
                          embedder_wts=None,
                          polygon=False,
                          today=None)
```

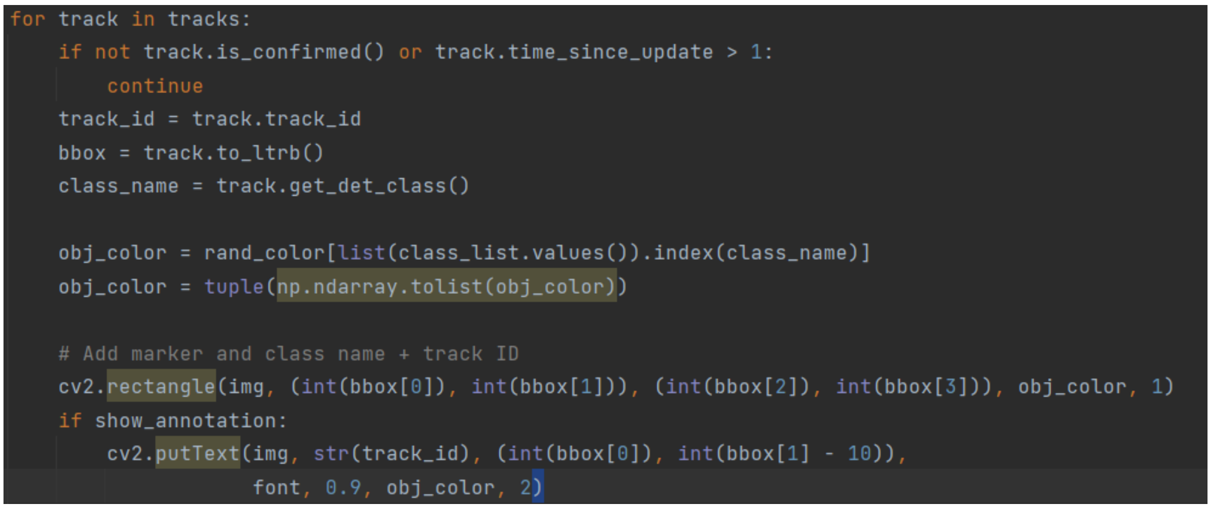
- **Object Detection** – Objects are detected in the given frame using a loaded Yolov5m model.



```python
# Detect frame image - Yolo
result = model(img)

# Read detected objects frame data
df = result.pandas().xyxy[0]

detections = []
for idx in df.index:
    class_name = df['name'][idx]
    confidence = (df['confidence'][idx]).round(2)

    if confidence > min_confidence:  # check for certainty
        if class_name in object_type_to_tracked or len(object_type_to_tracked) == 0:
            x1, y1 = int(df['xmin'][idx]), int(df['ymin'][idx])
            x2, y2 = int(df['xmax'][idx]), int(df['ymax'][idx])

            detections.append(([x1, y1, int(x2 - x1), int(y2 - y1)], confidence, class_name))
```

Only the Person Class is identified using the model. The identified object must have a confidence score of 0.5 or more.

- **Object Tracking** – Persons are tracked in each frame. Each person is marked with a unique track ID.



```
for track in tracks:
    if not track.is_confirmed() or track.time_since_update > 1:
        continue
    track_id = track.track_id
    bbox = track.to_ltrb()
    class_name = track.get_det_class()

    obj_color = rand_color[list(class_list.values()).index(class_name)]
    obj_color = tuple(np.ndarray.tolist(obj_color))

    # Add marker and class name + track ID
    cv2.rectangle(img, (int(bbox[0]), int(bbox[1])), (int(bbox[2]), int(bbox[3])), obj_color, 1)
    if show_annotation:
        cv2.putText(img, str(track_id), (int(bbox[0]), int(bbox[1] - 10)),
                    font, 0.9, obj_color, 2)
```

- **Object Counting** – After detecting the person, the next task is to monitor their movement at the entry or exit gate.

- **Warning** – If the person count is more than the safety threshold limit of the place, then an alert is shown to the public - be cautious while entering the premises or better wait for some time to avoid the rush.

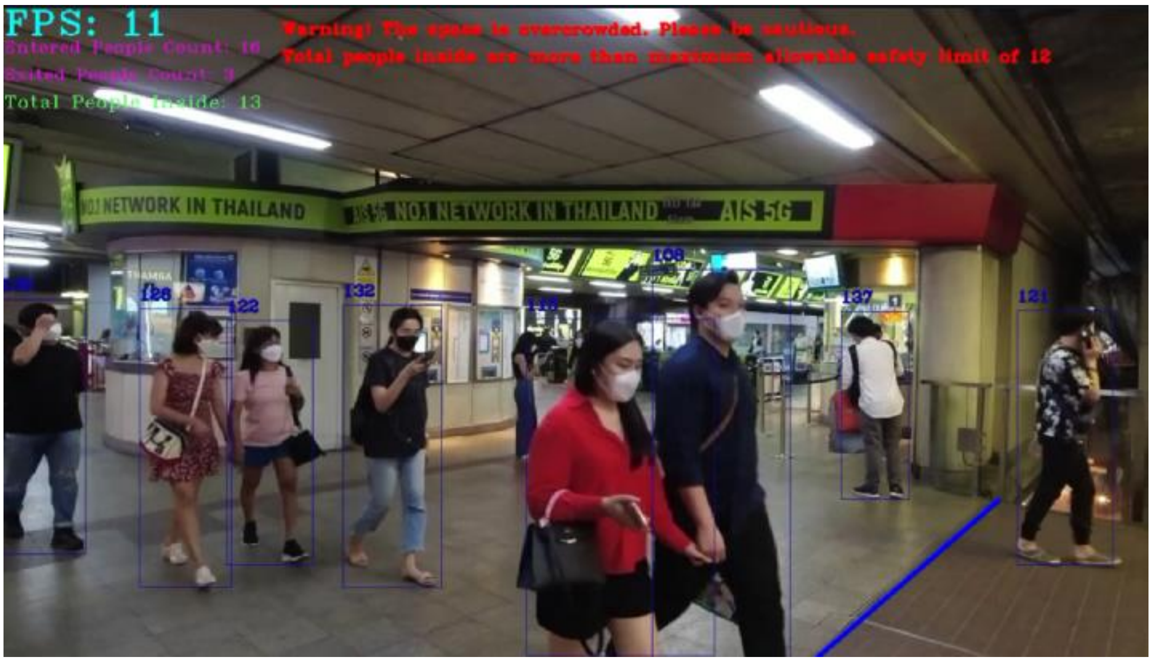


Figure 14: Application output on a sample video

6.3 Hardware Configuration

This application is running on a personal laptop with the following configuration:

- OS – Windows 11 pro
- Hard disk type – SSD
- RAM – 16 GB
- GPU - NVIDIA GeForce GTX 1650, 4 GB

6.4 Result

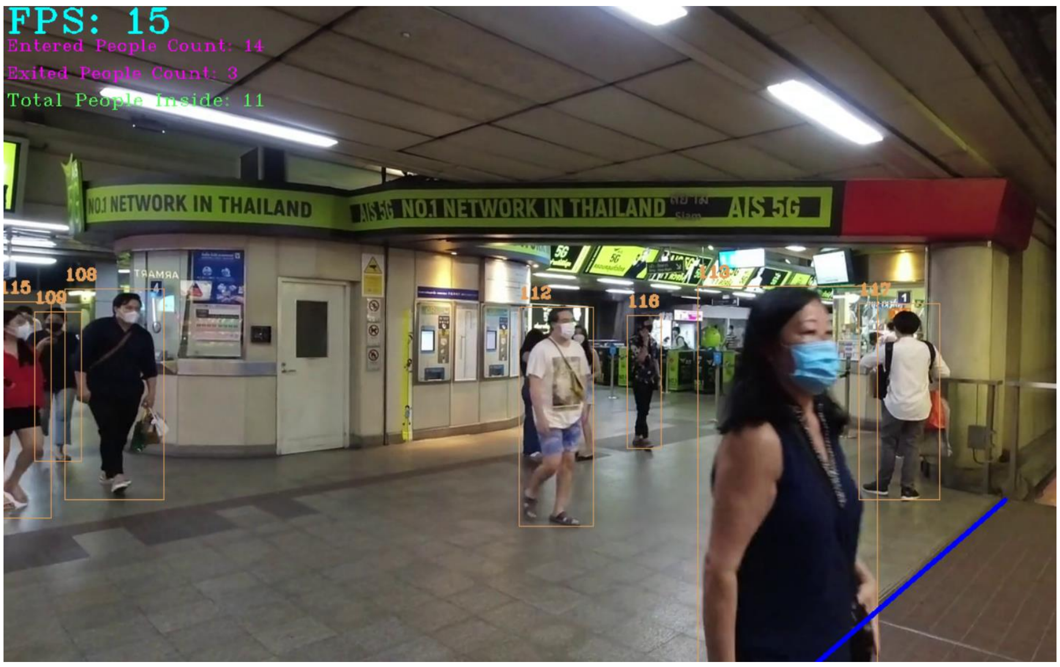
Results shown below are for the yolov5 – medium model

FPS – Frame per second

Accuracy – Proper identification and counting of people passing the entry line

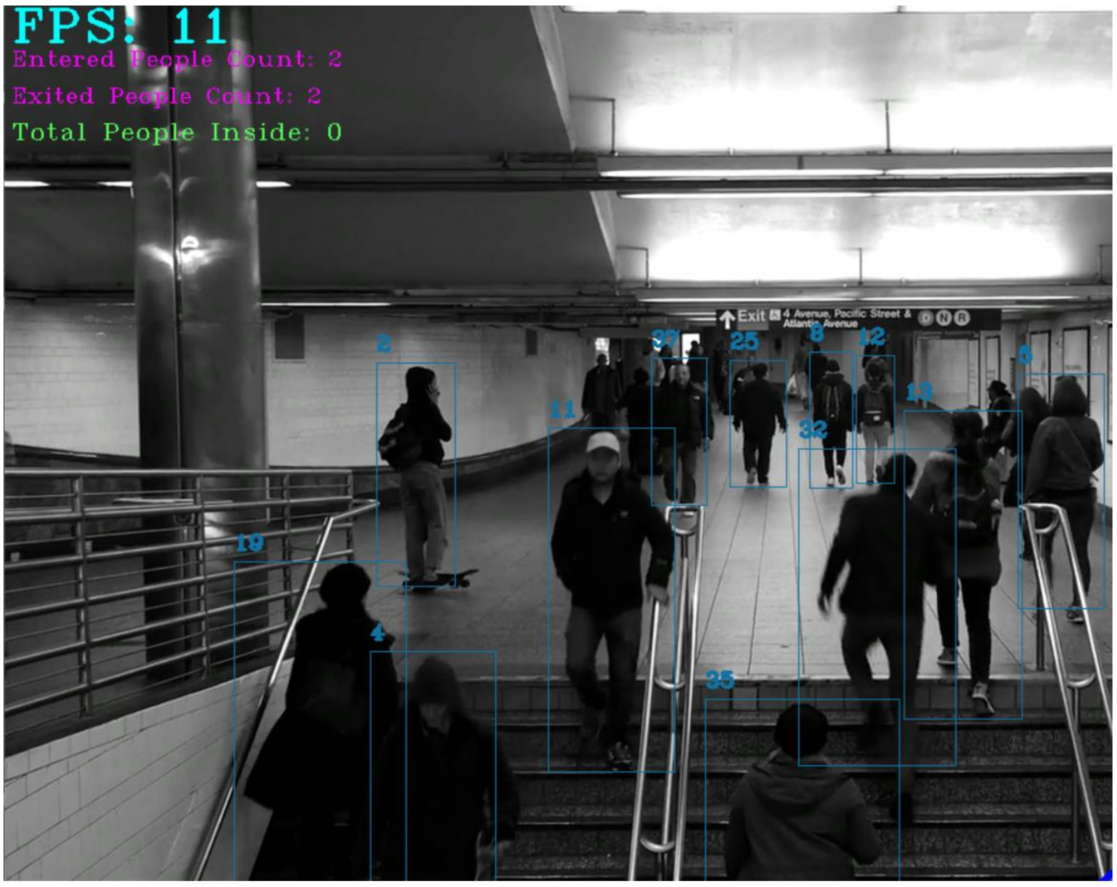
The tool is tested on different intensities of crowds,

- **Light Intensity:**
  Around 5 -10 people in each frame



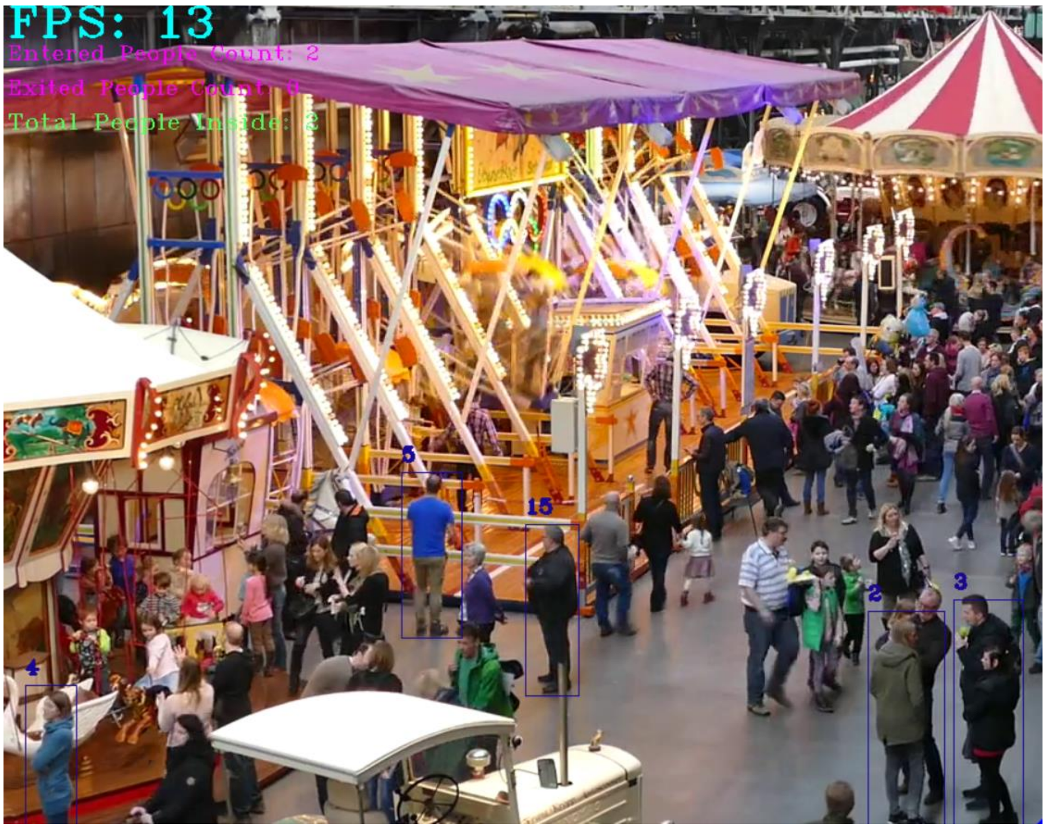
FPS (avg.) – 15
Accuracy – 99% (calculated manually)

- **Moderate Intensity:**
  Around 12 - 15 people per frame



FPS (avg.) – 12
Accuracy – 95% (calculated manually)

- **High Intensity**
  For 30+ people per frame

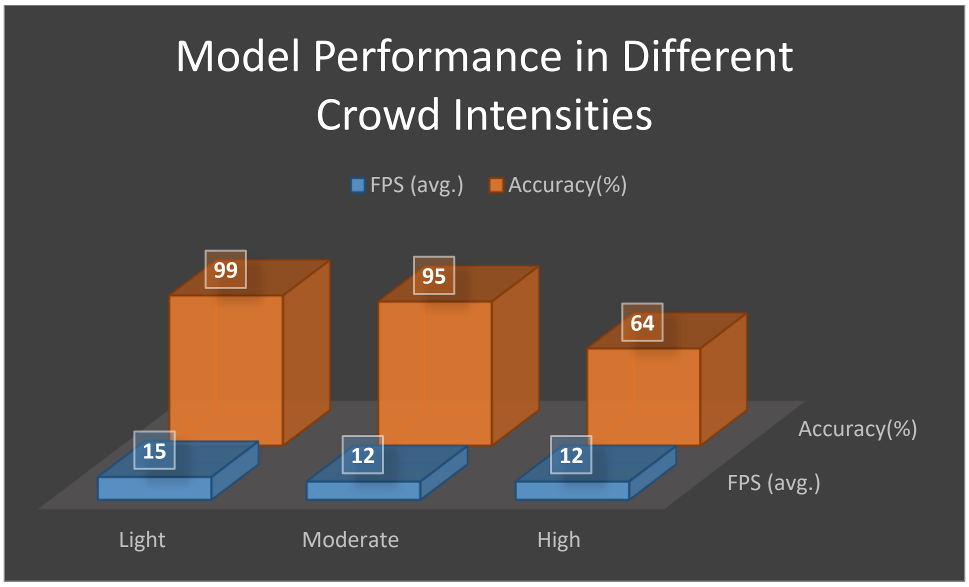

FPS: 13
Entered People Count: 2
Exited People Count: 0
Total People Inside: 2

FPS (avg.) -12
Accuracy – 64% (calculated manually)

6.5 Inference

The following observation are made for different test conditions,

- Tool can process almost the same number of frames per second in all load conditions
- Object (Person) identification decreases with an increase in the number of people per frame
- Object tracking/counting efficiency decreases with an increase in the number of people per frame



6.6 Recommendations

For High crowd intensity places, better to cover the region with multiple CCTV cameras (with no common area) and operate on them individually.

The process only the entry area instead of the whole frame. This will increase the accuracy.

CONCLUSION AND FUTURE SCOPE

1. Conclusion

The research for this dissertation is focused on the automatic detection and monitoring of people flow in a public space. Famous places like monuments, religious shrines, concerts etc. face high footfall on special occasions. It is not possible to predict the inflow every time. So, a regulatory mechanism is necessary to at least monitor the count. If the count is more than the expectation, authorities must be intimated to take necessary action.

To tackle this issue, an application has been developed that uses principles of Computer Vision and monitors the count on real time basis. Calculated count from the application can then be displayed on an electronic screen which is visible to all the on goers.

Experts can decide on a threshold safety limit. Safety threshold is the maximum number of people the venue can sustain without damaging the structure. It is also determined based upon the available exits facility which can allow them to exit the venue safely in case of any miss-happening.

When the count is more than the limit an alert is displayed on the electronic screen. This alert is not only useful for the authorities but also for the newcomers who are about to enter the facility. They can become cautious and wait till safety is restored.

This application alone does not guarantee safety, however, it can play a major role in ensuring it.

2. Future Scope

Throughout the work, it was possible to identify some work fronts that can be addressed in the future:

- More advanced CV models can be used to get real-time results.
- Some models require more hardware support for efficiently running. We can develop lighter models with high accuracy.
- Whole system can be centrally connected with a dedicated call centre team, which can monitor and alert when needed.

REFERENCES

- IJCRT - A Literature Survey on Computer Vision Towards Data Science
- V7 Labs
- YOLO - https://arxiv.org/abs/1506.02640
- Faster R-CNN - https://arxiv.org/abs/1506.01497
- Fast R-CNN - https://arxiv.org/abs/1504.08083
- YOLO object detection with OpenCV - https://pyimagesearch.com
- Wikipedia - https://en.wikipedia.org/wiki/Artificial_neural_network
- IBM Clouds - https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks
- Image Segmentation - https://manipulation.csail.mit.edu/segmentation.html
- Science Direct - https://www.sciencedirect.com/science/article/pii/S0893608017302976