

# Graduation Project

# AI Smart Cities

---

30th January, 2023



# **Traffic Accident: Detection**

Under supervision of

Dr. Amani Hassan

Eng. Dina Amr

Presented by

Eslam Mohamed 20190098

Maher Esmat 20190410

Mohanad Talat 20190561

Sayed Shaaban 20190254

Khalid Hassan 20190187

Ahmed Mohamed 20190071

# Table of Contents

List of Figures.....	5
List of Tables.....	5
List of Abbreviations.....	6
1. INTRODUCTION .....	7
1.1 Problem statement.....	7
1.2 Background.....	7
1.3 System purpose.....	8
1.4 System scope.....	8
1.5 System objectives and acceptance criteria.....	8
1.5.1 Objectives.....	8
1.5.2 Success criteria necessary for the project.....	8
1.6 Life cycle model.....	8
1.7 System methodology.....	10
1.8 Solution statement.....	11
1.8.1 System overview.....	11
1.8.2 Limitations.....	11
1.9 Expected results.....	12
1.10 Related Work .....	13
2. PLANNING.....	15
2.1 Project plan.....	15
2.2 Constraints.....	16
3. DATA.....	17

3.1 Version 1.....	18
4. MODEL.....	19
4.1 YOLO_V5.....	19
4.1.1. Architecture.....	19
4.1.2. Backbone Network.....	20
4.1.3. Detection Head.....	22
4.1.4. Object Scale.....	24
4.1.5. Anchor Boxes.....	24
4.1.6. Training.....	24
4.1.7. Inference.....	26
4.1.8. Model Sizes.....	26
4.1.9. Pre-trained Models and Transfer Learning.....	26
4.1.10. Open-Source and Framework Support.....	26
4.1.11. Pros in Yolo_v5 that are not found in other models.....	27
4.2. Faster R-CNN with ResNet-50.....	28
4.2.1. . Backbone Network.....	28
4.2.2. Feature Extraction.....	28
4.2.3. Region Proposal Network (RPN).....	29
4.2.4. Anchor Boxes.....	31
4.2.5. RoI (Region of Interest) Pooling.....	31
4.2.6. Object Classification and Localization.....	33
4.2.7. Training.....	33
4.2.8. Inference.....	35

4.2.9. Pros in Faster R-CNN with ResNet-50 that are not found in other models.....	35
4.3 SSD_mobilenet_v2_320*320.....	36
4.3.1. MobileNetV2 Backbone.....	37
4.3.2. Feature Pyramid Network (FPN).....	37
4.3.3. Detection Head.....	42
4.3.4. MultiBox Prior Boxes.....	42
4.3.5. Output Prediction.....	42
4.3.6. Input Resolution.....	42
4.3.7. Speed and Efficiency.....	42
4.3.8. Object Detection Performance.....	43
4.4. SSD_mobilenet_v2_fbnlite_64*640.....	43
4.5. SSD_mobilenet_v2_fbnlite_320*320.....	43
4.6. What is difference among SSD_mobilenet_v2_320*320 , SSD_mobilenet_v2_fbnlite_64*640 and ,SSD_mobilenet_v2_fbnlite_320*320.....	43
5. Conclusion .....	46
5.1. Final Report.....	46
5.1.1.Key findings.....	46
6. References.....	48

## List of Figures:

Figure 1 ➔ Project time plan.....	16
Figure 2 ➔ Exploring&analysing CADP.....	17
Figure 3 ➔ Sample about Expected Result.....	18
Figure 4 ➔ Sample about Expected Result.....	18
Figure 5 ➔ Sample about Expected Result.....	18
Figure 6 ➔ YOLO_V5 Architecture.....	20
Figure 7 ➔ YOLO_V5 Detection Head.....	22
Figure 8 ➔ Region Proposal Network (RPN) Architecture.....	30
Figure 9 ➔ Faster R-CNN with ResNet-50.....	31
Figure 10 ➔ SSD_mobilenet Architecture.....	39

## List of Tables:

Table 1 ➔ The most important models that have a close relationship with our project....	14
Table 2 ➔ Schedule of our project.....	15
Table 3 ➔ A report on the accuracy of each model used in this project.....	46

## List of Abbreviations:

- 1- FPS: Frames per second, which is the number of images that can be processed by the model per second
- 2- SSD: Single Shot Multi-Box Detector, which is a type of object detection model that uses a single deep neural network to predict object classes and bounding boxes
- 3- VGG: Visual Geometry Group, which is a research group at the University of Oxford that has developed several popular deep neural network architectures, including VGG-16 and VGG-19
- 4- ResNet: Residual Network, which is a type of deep neural network architecture that uses residual blocks to enable deeper networks
- 5- MobileNetV2: Mobile Network version 2, which is a type of lightweight deep neural network architecture that is designed for mobile and embedded devices
- 6- F-RCNN: Faster R-CNN, which is a type of object detection model that uses a region proposal network (RPN) to generate candidate object regions and a detection head to classify objects and refine their bounding boxes
- 7- RPN: Region Proposal Network, which is a neural network that generates candidate object regions in an image
- 8- Precision: The fraction of retrieved instances that are relevant, which is a measure of the accuracy of an object detection [ $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$  {all detected objects}.]
- 9- Recall: The fraction of relevant instances that are retrieved, which is a measure of the completeness of an object detection model [ $\text{Recall} = \text{TP} / (\text{TP} + \text{FP})$  all ground truth objects]

# 1. INTRODUCTION

## 1.1 Problem statement:

We have a real problem called traffic accidents, and those accidents are caused by Reasons, which are divided into two categories:

- 1) Natural disasters, such as earthquakes and hurricanes, which have nothing to do with humans.
- 2) Human-caused accidents are classified into two categories:
  - a) Accidents caused by the driver, such as not following traffic rules
  - b) Road accidents, such as when the road is not paved for traffic

We are looking forward to solving all of the causes of accidents, even if those causes are beyond the control of humans, and despite the presence of surveillance cameras on the road, as well as many traffic security units and a lot of first aid, there are still more than a million traffic accidents that occur annually. We believe that in order to solve these problems, the following must be done:

1. Using technology in general, and artificial intelligence in particular, to prevent the causes of accidents in a more practical way.
2. In the event of an accident, first aid and emergency services must arrive quickly, which will not happen unless someone immediately calls the police and an ambulance. Rather, the most amazing thing is that we can prevent an accident from occurring by predicting the occurrence of an accident as a result of the occurrence of certain factors, and this is what we want to access.

## 1.2 Background:

Any change that can cause an accident must be explored by the camera, because the causes of accidents are different

- Technical defects from the car itself
- The shape of the car suggests that it is not able to walk in complete balance
- No seat belt
- The driver is drunk
- The driver is asleep

And other causes of accidents, and what the camera will do specifically is to try to identify the car that has a problem or is subject to an accident and try to contact the police to stop it before the accident or help it if the accident actually occurred. This will be done



by training the camera on many data from images and videos that help to know the percentage of an accident in a car, and this will be done by using a models called R-CNN (Regional-based Convolutional Neural Network), YOLO (You-Only-Look-Once).

### 1.3 System Purpose:

- Minimizing accident rates
- The speed with which the accident is noticed, and thus the speed with which the police, first aid, and support are contacted
- Predicting the occurrence of an accident before it occurs by analyzing the causes and attempting to avoid it.

### 1.4 System Scope:

#### In Scope:

- The camera will predict whether there has been an accident or not

#### Out Scope:

- If there is indeed an accident, the police must come in a timely manner, in addition to the presence of the ambulance patrol also in a timely manner

### 1.5 System objectives and acceptance criteria:

#### 1.5.1 Objectives:

- Detect the accidents.
- Select the reasons for the occurrence of the accident.
- Predict the accident before it occurs.
- Our work is devoted toward the objective of making roads safer.


#### 1.5.2 Success criteria necessary for the project:

Our success criteria for this project require that we conduct an accurate, organized, and detailed analysis of accidents, and that the competent authorities are notified as soon as an accident occurs, as well as the causes of the accident.

### 1.6 Life cycle model:

#### ❖ We will use Agile as a life cycle type:

- **Data collection:** The major challenge in collecting data for traffic accidents is two-fold: (i) Abnormality: because the accidents are rare, although there are live-streams from traffic cameras mounted on the corner of road intersections, this is



infeasible to wait for an accident to happen; and (ii) Access: access to traffic camera data is often difficult. Due to this challenge, the data of traffic accidents is often not available for public uses. To this end, in this work, we use the traffic accidents captured from traffic camera views available on video sharing websites such as YouTube. The whole pipeline for data collection and annotation. Keyword search to collect the data for traffic accidents, we exploited the search engine and resources available in YouTube. We used keywords like "car accidents traffic camera" to search for relevant videos. This step returned 1014 YouTube videos. However, the collected videos from these queries contain many irrelevant items. To collect only relevant items, we employ three annotators to manually watch and report items as follows. All annotators are instructed to know that our objective is to collect only videos which contain at least one accident scene which is captured from a traffic CCTV footage.

- **Data preparation:** After collecting data, we must create a structure with which to feed the model. We cleaned the data to remove any noise or misleading data before collecting or removing it from the training set. In addition, we will normalize data by scaling or cropping images, converting them to a relevant format, and creating a folder structure for training. Preprocessing data is the most difficult challenge in our project and will take a long time to complete properly.
- **Model evaluation and training:** When we evaluate our model, we look closely at different models and their architectures to determine which architecture will be the best with our data, and we are going to do cross validation, also we split with an 80:20 ratio (train and test set).
- **Model validation:** After completing the training described above, we evaluate the model's quality. We work with the model to understand its behavior and which aspects are already well solved. We interpret necessary changes to the training set by inspecting the visual data in order to optimize result quality.
- **Comparison and Feedback:** At this point, we should share our progress with our customer. We present our findings on the model's quality and condition, as well as what worked and what did not work. A good working relationship with our customer is essential here. We discuss potential model improvements as a group.

- **Deployment:** Deploying our current model version serves as the quality baseline for the subsequent training iteration. If the model already adds value to the customer's prototype or even production, it can be used.

## 1.7 System methodology:

The methodology of a traffic accident system is determined by whether its features are adjusted and know what to do, or if some of them can be learned or modified during the system process.

- **Pre-processing:** Before applying a computer vision method to image data to extract some specific piece of information, it is usually necessary to process the data to ensure that it meets certain assumptions implied by the method. Here are some examples:
  - Re-sampling is used to ensure that the image coordinate system is accurate.
  - Noise reduction is used to ensure that sensor noise does not expose misleading data.
  - Enhancement of contrast to ensure that relevant information can be detected.
  - Scale space representation is used to improve image structures at local scales.
- **Feature extraction :** extracting various features from image data such as :
  - Localized points of the vehicles, or distances between each vehicle.
  - Shape or motion of each vehicle on the same road line can be associated with more complex features.
- **Detection:** A decision is made at some point in the processing as to which image points or regions of the image are relevant for further processing. Such as :
  - The selection of a specific set of points of interest.
  - Image segmentation of one or more image regions containing a specific object of interest.
- **High-level processing:** the input is a small set of data, such as a set of points or an image region containing specific objects. Then try to validate the data to ensure that model-based and system-specific assumptions are met.
- **Decision making:** Making the final application decision; at this point, we can ensure that everything is working properly or make any necessary changes.

## 1.8 Solution statement:


### 1.8.1 System overview:

Conflicts and accidents at intersections are one of the most serious issues in urban traffic management. Drivers caught in a bind may decide to accelerate at the time of the phase change from green to yellow, resulting in rear-end and angle crashes. Furthermore, despite all efforts to prevent dangerous driving behaviors, running a red light is still common. Due to the nature of traffic control systems or intersection geometry, other dangerous behaviors, such as sudden lane changes and unpredictable pedestrian/cyclist movements at the intersection, may also occur. The early detection of such trajectory conflicts is required for the development of countermeasures to mitigate their potential harms. Most traffic management systems currently monitor traffic surveillance cameras through manual perception of captured footage.

Because most intersections have surveillance cameras, automatic detection of traffic accidents based on computer vision technologies will be extremely beneficial to traffic monitoring systems. Numerous studies have used computer vision techniques in traffic surveillance systems for a variety of tasks. Automatic detection of traffic incidents not only saves a lot of unnecessary manual labor, but it also helps paramedics and emergency ambulances dispatch in a timely manner. In order to defuse severe traffic crashes, an automatic accident detection framework provides useful information for adjusting intersection signal operation and modifying intersection geometry.

### 1.8.2 Limitations

- Dealing with a massive amount of data. When it comes to images/videos, all computers see are numbers... Lots of numbers! That is what an image is to a computer (images are made up of pixels, which are represented as numbers). If you have a greyscale (black and white) image with a resolution of 1920 x 1080, this means your image is described by 2 million numbers ( $1920 * 1080 = 2,073,600$  pixels). I'm abstracting here about compression (see point 4 for more on this). When you switch to a color image, you need three times as many numbers because when you represent a colored pixel, you typically specify how much red, blue, and green it is made up of. Furthermore, If you trying to analyze images coming in from a video/camera stream at, say, 30 frames/sec (a common frame rate nowadays), you'll find yourself dealing with 180 million



numbers per second ( $3 * 2,073,600 * 30 = 180$  million pixels/sec). That is a lot of information to process! Machines struggle to do anything significant with 180 million numbers coming in per second, even with today's powerful processors and relatively large memory sizes. You must still be frugal and creative in your endeavors. Furthermore, the fact that the "default" resolution of images is increasing year after year does not help the situation.

- Another major factor contributing to the difficulty of computer vision is information loss during the digitizing process (going from real life to an image on a machine). Image processing is designed to take information from a 3D world (or 4D if we're talking about time in a video stream) and project it onto a 2D plane (i.e. a flat image). This means you're also losing a lot of information in the process, despite the fact that we still have a lot of data to deal with as is, as discussed in the previous point. Of course, you can try to simulate how we see with two eyes by taking two photos at the same time and extracting 3D information from them.
- Noise is frequently present during the digitizing process. For example, no camera will ever provide a perfect representation of reality, especially when it comes to the cameras on our phones (even though phone cameras are getting phenomenally good with each new release). Intensity levels, color saturation, and so on will always be merely attempts to capture our beautiful world. Another type of noise is a phenomenon known as an artifact. These are image distortions that can be caused by a variety of factors. Algorithms have been developed to attempt to remove lens flare from images, but this is still a work in progress.

### **1.9 Expected results:**

- Determine whether or not an accident is about to occur.
- Take accident-prevention measures.
- Notify the appropriate authorities so that they can take appropriate action such as sending ambulances and alert nearby hospitals.
- Learn from the accident in order to avoid similar incidents in the future.

### 1.10 Related work:

- **Object detection:** a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results.
- **Pedestrian Detection:** predicts information about the pedestrian's position, provides a comprehensive overview and arguments for replacing based on the detection in the current frame.

Continuous detection via pedestrian tracking achieves real-time performance for pedestrian detection. Shows that adding extra features, flow information, and context information are complementary additions that result in significant gains over other strong detectors. Body-part semantics and contextual information are employed. Proposes a Haar-like cascade classifier design for fast pedestrian detection. Proposes an extension to Faster R-CNN that uses contextual information with multi-level features to detect pedestrians in cluttered backgrounds by embedding pooling information from a larger area around the original area of interest.

- **Semantic segmentation:** the process of dividing a digital image into multiple image segments, also known as image regions or image objects, is known as image segmentation (sets of pixels). The goal of image segmentation is to simplify and/or change an image's representation into something more meaningful and easier to analyze. Image segmentation is commonly used to find objects and boundaries (lines, curves, and so on) in images. Image segmentation is the process of labeling every pixel in an image so that pixels with the same label share certain characteristics. Image segmentation produces a set of segments that cover the entire image, or a set of contours extracted from the image (see edge detection). Each pixel in a region is comparable in terms of some characteristic or computed property, such as color, intensity, or texture. Colors in adjacent regions differ significantly when compared to the same characteristic (s). When applied to a stack of images, as is common in medical imaging, the contours generated after image segmentation can be used to create 3D reconstructions using interpolation algorithms such as marching cubes.
- Most approaches treat car accidents as a classification problem with two labels (accident and no accident), and inputs such as weather, time features, speed limit, road width, and a plethora of others.
- This is what distinguishes our approach; we want to understand the causes of accidents so that we can predict when and where an accident will occur and take countermeasures, as well as notify concerned authorities such as ambulances if we were unable to prevent an accident.
- There are many models and algorithms that can address the problem of car accidents detection and prediction. Like YOLO, and R-CNN family Our objective is to compare between different models,

Algorithm	Features	Limitations
<b>CNN</b>	Divides the image into multiple regions and then classifies each region into various classes.	Needs a lot of regions to predict accurately and hence high computation time.
<b>R-CNN</b>	Uses selective search to generate regions.	High computation time as each region is passed to the CNN separately. Also, it uses three different models for making predictions.
<b>Fast R-CNN</b>	Maps are extracted. Selective search is used on these maps to generate predictions. Combines all the three models used in R-CNN together.	Selective search is slow and hence computation time is still high.
<b>Faster R-CNN</b>	Replaces the selective search method with region proposal network (RPN) which makes the algorithm much faster.	Object proposal takes time and as there are different systems working one after the other, the performance of systems depends on how the previous system has performed.
<b>Mask R-CNN</b>	Extending Faster R-CNN for Pixel Level Segmentation by adding a branch to Faster R-CNN that outputs a binary mask that says whether or not a given pixel is part of an object	R-CNN usually fails to detect object suffer from blur at low resolution
<b>YOLO</b>	We take an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding boxes, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.	It struggles with small objects within the image This is due to the spatial constraints of the algorithm.

**Table 1**

## 2. PLANNING:

### 2.1 Project plan:

- Before we can work with any dataset, we need to do some preprocessing on it to filter the data, make labels, normalize it or any other operation. To find good features from the data to learn from.
- We can apply one of CNNs models (like mask RCNN (Region-Based Convolutional Neural Network) OR other Algorithms such as YOLO (you only look once)) on a video dataset.
- **Schedule**

Task	Start Date	Tenure
Searching for dataset	11/1/2022	40
Take DELL dataset	11/1/2022	15
Reading papers	11/15/2022	25
Searching for dataset from papers	12/1/2022	10
Searching for models	11/15/2022	35
Reading papers	11/15/2022	30
Learning about models	11/30/2022	20
Data preprocessing	12/20/2022	85
Download dataset	12/20/2022	15
Make data annotations	2/1/2023	20
Select good samples	2/25/2023	15
Applying the models on data	3/10/2023	60
Mask RCNN	3/10/2023	30
You look only once (YOLO)	4/10/2023	30

Table 2



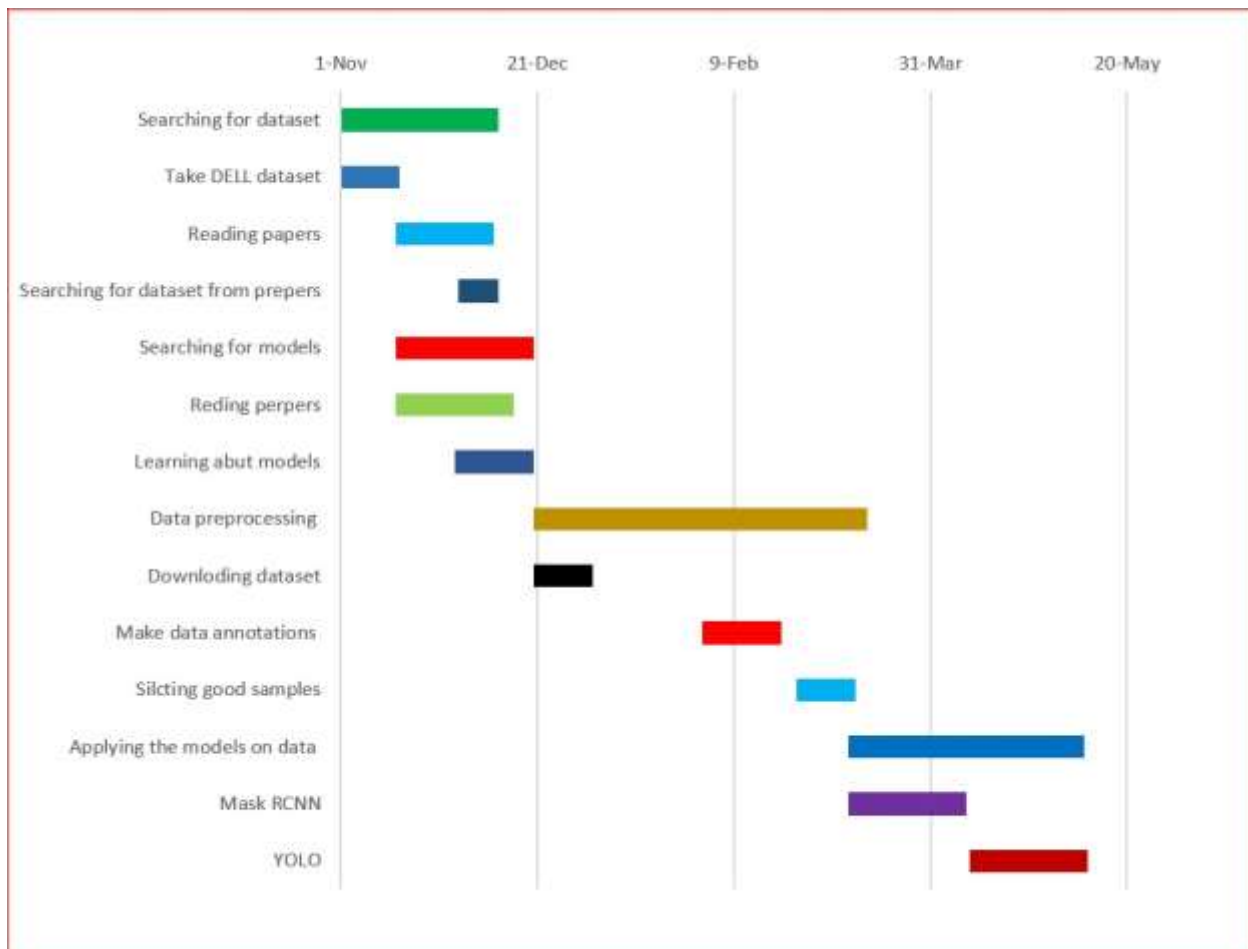


Figure 1

## 2.2 Constraints:

- In case we work with video models
  - We should put the camera in a good location so that it can see the room well and make sure that it wakes all time and is near to the processor so there is no delay in design.
  - Making sure all sensors and cameras have power and are connected to the system all time.

### 3. DATA:

- The data we using called CADP a novel dataset for CCTV (Closed Circuit Television) traffic camera based accident analysis.
- CADP dataset provides samples for accident detection and forecasting type analysis.
- Number of positive videos (1416 videos) in our dataset for only traffic accidents is much larger than that in UCF-Crimes (151 videos of road accidents) and DAD (about 600 videos). Note that, in CADP, there are videos with more than one accident.
- CADP contains videos collected from YouTube which are captured under various camera types and qualities, weather conditions and edited videos.
- The data contains different resolution (High and low). So we resized it to 500\*500.
- The videos are divided into frames which means the data contain images of frames of the video not the video itself.
- We used tools like make <https://www.makesense.ai> and labelme to annotate the data.
- Since the data contains different resolution and divided into frames, the low resolution has duplicate frames (redundant images), so it is our job to find it and remove it from the data.
- CADP dataset provides different scenarios, different weather, different time, and different cities.
- CADP dataset also contains some accidents that are not very clear to observer, so it is our job to remove the non-important frames of the data.

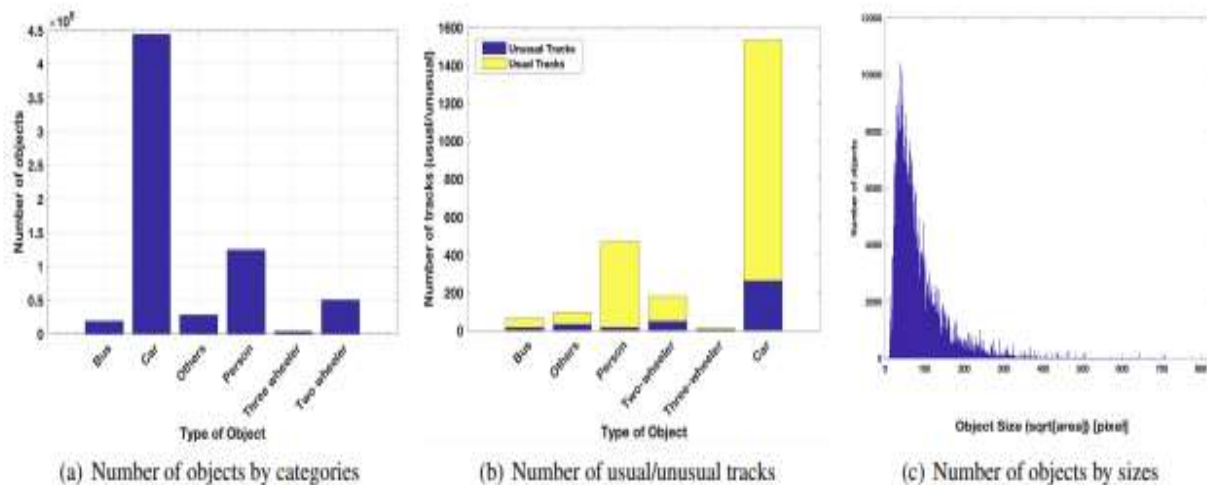


Figure 2

### 3.1 Version 1:

The data is annotated on only one object which is accident in general to make the model only learn the accidents and be able to detect them.

We will call it CAD\_V1



Figure 3



Figure 4



Figure 5

## 4. MODEL:

### 4.1. YOLO\_v5:

YOLO (You Only Look Once) is a popular object detection algorithm that has undergone several versions of development. YOLO v5, which was released in May 2020, is the latest version of the YOLO series. It is an improvement over its predecessors in terms of speed, accuracy, and ease of use. YOLO v5 is known for its real-time object detection capabilities and has gained significant popularity in the computer vision community.

Here's an overview of YOLO v5 and its key features:

#### 4.1.1 Architecture:

The first point regarding the architecture of YOLO v5 can be described in more detail. YOLO v5 follows a one-stage object detection approach, which means that it performs object detection in a single pass through the neural network.

In YOLO v5, the input image is divided into a grid of cells. Each cell is responsible for predicting objects that fall within its boundaries. The size of the predicted bounding boxes is relative to the cell size, allowing YOLO v5 to handle objects of different scales. This approach eliminates the need for pre-defined anchor boxes or region proposals used in some other object detection algorithms.

The backbone network in YOLO v5 is based on the CSPDarknet53 architecture. The backbone network extracts features from the input image. These features are then fed into the detection head of the network for making predictions.

The detection head takes the extracted features from the backbone network and performs predictions. It predicts the bounding box coordinates (x, y, width, height) for each detected object. The coordinates are relative to the grid cell in which the object falls.

In addition to the bounding box coordinates, the detection head predicts objectness scores, which indicate the presence of an object within a bounding box. This score represents the likelihood of an object being present in a particular box.

Furthermore, the detection head predicts class probabilities for multiple classes. This means that YOLO v5 is capable of detecting and classifying objects from a predefined set

of classes. Each bounding box prediction is associated with class probabilities for different object categories.

During training, YOLO v5 optimizes a loss function that combines objectness loss, localization loss, and classification loss. The objectness loss penalizes incorrect objectness predictions, the localization loss penalizes inaccurate bounding box predictions, and the classification loss penalizes incorrect class predictions. By minimizing this loss function, YOLO v5 learns to improve the accuracy of the predicted bounding boxes and class probabilities.

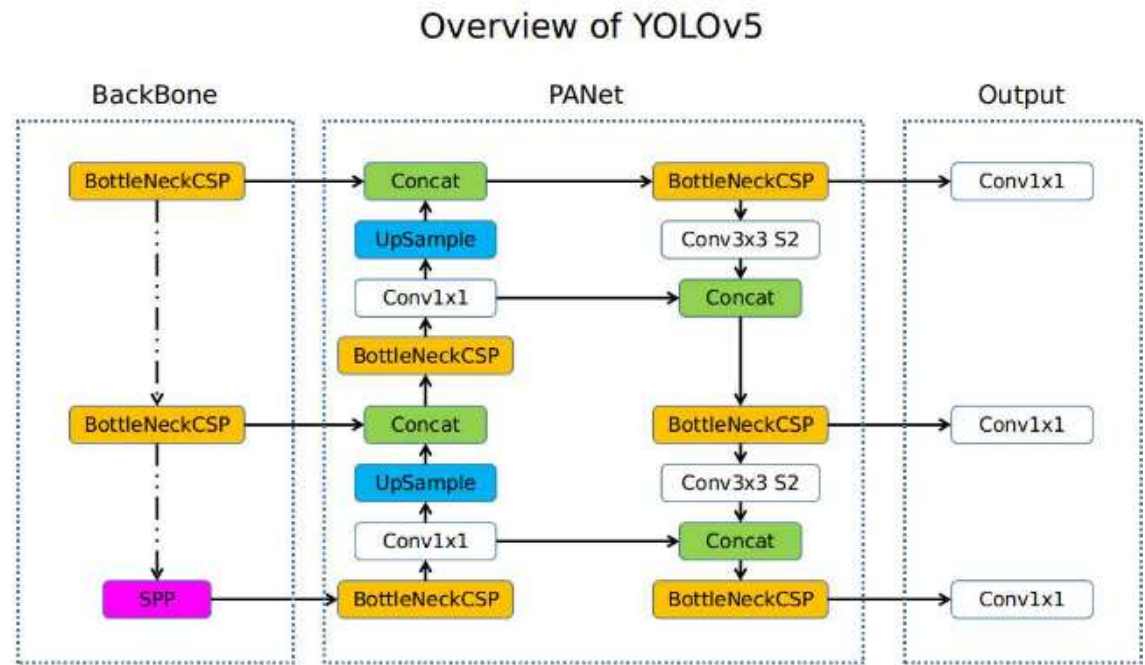



Figure 6

In summary, YOLO v5's architecture follows a one-stage object detection approach, where the input image is divided into a grid of cells. The backbone network extracts features from the image, which are then used by the detection head to make predictions. These predictions include bounding box coordinates, objectness scores, and class probabilities. The model is trained by optimizing a loss function that considers objectness, localization, and classification. This architecture enables YOLO v5 to perform efficient and accurate object detection in real-time scenarios.

#### 4.1.2. Backbone Network:

The backbone network in YOLO v5 refers to the underlying architecture that is responsible for extracting high-level features from the input image. In YOLO v5, the



backbone network is based on the CSPDarknet53 architecture, which is an enhanced version of the Darknet architecture used in previous versions of YOLO.

**Here are some key details about the backbone network in YOLO v5:**

1. **CSPDarknet53 Architecture:** The backbone network in YOLO v5 is based on the CSPDarknet53 architecture. CSP stands for Cross-Stage Partial Network, which introduces a split-transform-merge strategy to enhance feature representation and computational efficiency.
2. **Feature Extraction:** The main purpose of the backbone network is to extract features from the input image that are relevant for object detection. These features capture different levels of abstraction, allowing the subsequent layers to make accurate predictions.
3. **Layer Organization:** The CSPDarknet53 architecture consists of a series of convolutional layers, pooling layers, and shortcut connections. These layers are organized in a deep neural network structure to capture both low-level and high-level image features.
4. **Convolutional Layers:** Convolutional layers are the core building blocks of the backbone network. They perform convolutions on the input image to extract spatial features, such as edges, corners, and textures. The number of convolutional layers and their configurations may vary depending on the specific variant of YOLO v5 being used (e.g., YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x).
5. **Pooling Layers:** Pooling layers are used to downsample the feature maps, reducing their spatial dimensions while retaining important features. This downsampling helps in capturing more abstract and higher-level representations of the input image.
6. **Shortcut Connections:** YOLO v5 incorporates skip or shortcut connections within the backbone network. These connections allow information to flow directly from earlier layers to later layers, facilitating the propagation of low-level details and aiding in gradient flow during training. Shortcut connections help in preserving fine-grained spatial information and can improve the model's ability to detect small objects.

7. **Pretrained Weights:** The backbone network in YOLO v5 is often initialized with pretrained weights. These weights are usually obtained by training the backbone network on large-scale image classification datasets, such as ImageNet. By starting with pretrained weights, the backbone network is already capable of extracting meaningful features from images, which helps in the subsequent object detection task

In summary, the backbone network in YOLO v5, based on the CSPDarknet53 architecture, plays a crucial role in extracting high-level features from the input image. It consists of convolutional layers, pooling layers, and shortcut connections. The backbone network is designed to capture both low-level and high-level features, aiding in accurate object detection. Pretrained weights are often used to initialize the backbone network, leveraging knowledge learned from large-scale image classification tasks.

#### 4.1.3. Detection Head:

The detection head in YOLO v5 is responsible for taking the extracted features from the backbone network and making predictions for object detection. It receives the feature maps as input and produces bounding box coordinates, objectness scores, and class probabilities for the detected objects.

Here are some key details about the detection head in YOLO v5:

1. **Prediction Grid:** The detection head operates on a grid that divides the input image into cells. Each cell is responsible for predicting objects that fall within its boundaries. The size of the grid and the number of cells may vary depending on the specific variant of YOLO v5 being used.

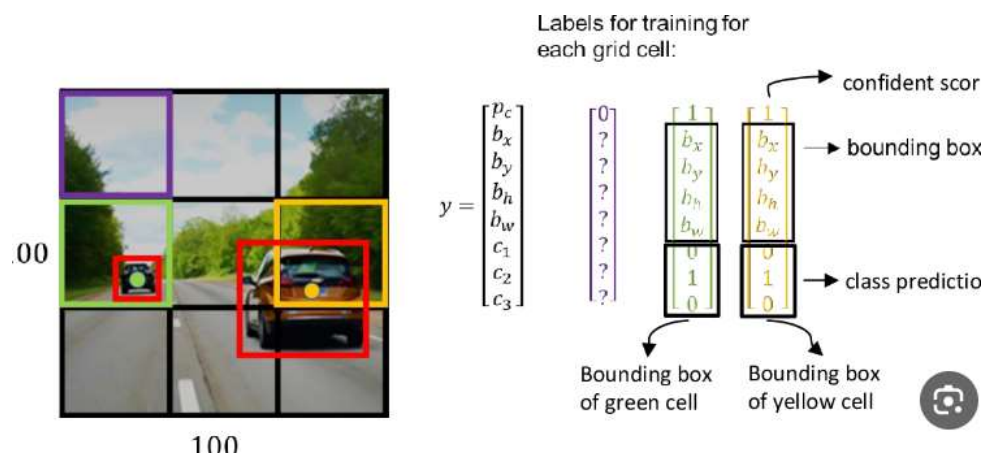



Figure 7



2. **Bounding Box Predictions:** For each grid cell, the detection head predicts the bounding box coordinates of the detected objects. These coordinates include the x and y coordinates of the box's center, as well as the width and height of the box. The coordinates are relative to the cell and can be transformed to the original image coordinates if needed.

3. **Objectness Scores:** The detection head also predicts objectness scores for each bounding box. The objectness score represents the likelihood of an object being present in a particular box. It indicates the confidence that the predicted box contains an object of interest.

4. **Class Probabilities:** In addition to bounding box predictions and objectness scores, the detection head predicts class probabilities for multiple predefined classes. Each bounding box is associated with class probabilities for different object categories. These probabilities represent the confidence that the detected object belongs to a specific class.


5. **Anchors or Scales:** YOLO v5 does not explicitly use anchor boxes like some other object detection models. Instead, it learns to assign bounding boxes to different objects based on their sizes and positions within the grid cells. The model determines the scales of the predicted bounding boxes based on the characteristics of the training data.

6. **Non-Maximum Suppression (NMS):** After generating bounding box predictions, objectness scores, and class probabilities, the detection head applies a post-processing step called non-maximum suppression (NMS). NMS removes duplicate or overlapping detections to produce a final set of non-overlapping and highly confident object detections.

7. **Loss Function Optimization:** During training, the detection head optimizes a loss function that combines objectness loss, localization loss, and classification loss. The objectness loss penalizes incorrect objectness predictions, the localization loss penalizes inaccurate bounding box predictions, and the classification loss penalizes incorrect class predictions. By minimizing this loss function, the detection head learns to improve the accuracy of the predicted bounding boxes and class probabilities.

In summary, the detection head in YOLO v5 processes the extracted features from the backbone network to make predictions for object detection. It predicts bounding box





coordinates, objectness scores, and class probabilities. The detection head operates on a grid and assigns bounding boxes to objects based on their sizes and positions within the grid cells. Non-maximum suppression is applied to remove duplicate detections. During training, the detection head optimizes a loss function to improve the accuracy of the predictions.

#### **4.1.4. Object Scale:**

YOLO v5 divides the input image into a grid of cells. Each cell is responsible for predicting objects that fall within its boundaries. The size of the predicted bounding boxes is relative to the cell size, allowing YOLO v5 to handle objects of different scales.


#### **4.1.5. Anchor Boxes:**

In YOLO v5, anchor boxes are not explicitly used. Instead, YOLO v5 learns to assign bounding boxes to different objects based on their sizes and positions within the grid cells.

#### **4.1.6. Training:**

Training YOLO v5 involves the process of optimizing the model's parameters to accurately detect objects in images. Here's an overview of the training process:

- 1. Dataset Preparation:** First, you need to gather a labeled dataset for training YOLO v5. The dataset should consist of images along with corresponding bounding box annotations and class labels for the objects present in the images. The annotations typically include the coordinates of the bounding boxes and the class labels associated with each object.
- 2. Data Augmentation:** Data augmentation techniques are commonly employed to increase the diversity and robustness of the training data. Augmentation techniques include random resizing, flipping, rotation, translation, and color distortion. These techniques help the model generalize better and handle variations in object appearance, scale, and orientation.
- 3. Model Initialization:** YOLO v5 can be initialized with pretrained weights from models trained on large-scale image classification datasets like ImageNet. The pretrained weights provide a good starting point, as the backbone network is already capable of extracting meaningful features. The model can then be fine-tuned for the object detection task.



4. **Loss Function:** YOLO v5 utilizes a combination of loss functions during training to optimize the model. The loss function consists of three main components:

- **Objectness Loss:** This component penalizes incorrect objectness predictions. It encourages the model to predict high objectness scores for grid cells that contain objects and low objectness scores for cells without objects

- **Localization Loss:** The localization loss penalizes inaccurate bounding box predictions. It encourages the model to accurately predict the coordinates (x, y, width, height) of the bounding boxes for the objects.


- **Classification Loss:** The classification loss penalizes incorrect class predictions. It encourages the model to assign high probabilities to the correct classes associated with the objects

5. **Optimization:** The model is optimized by minimizing the combined loss function. This is typically achieved using gradient descent optimization algorithms such as Stochastic Gradient Descent (SGD) or Adam. The gradients are computed with respect to the model parameters, and the optimizer updates the parameters iteratively to minimize the loss.

6. **Training Process:** During training, batches of images are fed into the model. The model performs forward propagation to generate predictions, and the loss function is calculated based on the predicted outputs and the ground truth annotations. The gradients are then computed through backpropagation, and the optimizer updates the model's parameters accordingly. This process is repeated for multiple iterations or epochs until the model converges and achieves satisfactory performance.

7. **Evaluation:** To monitor the model's performance during training, evaluation metrics such as mean average precision (mAP) are commonly used. These metrics assess the accuracy and quality of the model's object detections. The model's performance can be evaluated on a separate validation or test dataset to assess its generalization capabilities.

8. **Hyperparameter Tuning:** YOLO v5 has several hyperparameters that need to be set before training, such as learning rate, batch size, and the number of training iterations. These hyperparameters can be tuned through experimentation to find the best configuration that balances training speed and accuracy.



**9. Transfer Learning:** YOLO v5 supports transfer learning, which allows you to fine-tune the model on a custom dataset with relatively fewer labeled examples. By initializing the model with pretrained weights and adjusting the final layers to match the desired number of classes, the model can quickly adapt to the specific object detection task.

By following these steps, you can train YOLO v5 on your labeled dataset and optimize it to accurately detect objects in images.

#### **4.1.7. Inference:**

During inference, YOLO v5 takes an input image and passes it through the network to generate bounding box predictions and class probabilities. Non-maximum suppression (NMS) is then applied to remove duplicate or overlapping detections, resulting in a final set of object detections.

#### **4.1.8. Model Sizes:**

YOLO v5 comes in different sizes or variants, such as YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. These variants differ in terms of the number of layers, model size, and computational requirements. Smaller variants are faster but less accurate, while larger variants are more accurate but slower.

#### **4.9. Pre-trained Models and Transfer Learning:**

YOLO v5 provides pre-trained models trained on large-scale datasets, such as COCO (Common Objects in Context) or Open Images. These pre-trained models can be used as a starting point for transfer learning on custom datasets, reducing the training time and resource requirements.

#### **4.10. Open-Source and Framework Support:**

YOLO v5 is an open-source project and is implemented using the PyTorch deep learning framework. It has an active community of contributors and is widely used in various applications, including autonomous driving, surveillance, robotics, and more.

Overall, YOLO v5 offers an efficient and accurate solution for real-time object detection tasks, providing a balance between speed and accuracy. Its ease of use, availability of pre-trained models, and compatibility with popular deep learning frameworks make it a popular choice among researchers and developers in the computer vision field.

#### 4.1.11. Pros in Yolo\_v5 that are not found in other models:

YOLO v5 offers several advantages and unique features compared to other object detection models. Here are some pros of YOLO v5 that set it apart:

1. **Speed and Efficiency:** YOLO v5 is known for its impressive speed and efficiency in object detection. It achieves real-time or near real-time performance, making it suitable for applications that require fast inference speeds. The one-stage detection approach, absence of region proposals, and optimized architecture contribute to its efficiency.
2. **Accuracy and Flexibility:** YOLO v5 achieves competitive accuracy while maintaining its speed. It introduces improvements over previous YOLO versions, such as the CSPDarknet53 backbone network, which enhances feature representation. YOLO v5 also offers different variants (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x) that provide a trade-off between speed and accuracy, allowing users to choose the most suitable variant for their specific requirements.
3. **Easy Implementation and Deployment:** YOLO v5 has been designed to be user-friendly and easy to implement. It provides a well-documented and open-source codebase that facilitates implementation and customization. The model is implemented using popular deep learning frameworks such as PyTorch, which simplifies integration into existing workflows and deployment on various platforms.
4. **Transfer Learning Capability:** YOLO v5 supports transfer learning, enabling the model to be fine-tuned on new datasets with relatively fewer labeled examples. By initializing the model with pretrained weights, YOLO v5 can leverage knowledge learned from large-scale classification datasets, speeding up training and adapting the model to specific object detection tasks.
5. **On-Device Inference:** YOLO v5's efficient architecture makes it suitable for on-device inference, such as deploying the model on edge devices or embedded systems. Its low computational requirements and smaller model size enable real-time object detection on resource-constrained devices.
6. **Strong Community Support:** YOLO v5 has gained significant popularity and has a strong community of developers and researchers contributing to its development and improvement. This active community provides resources,

tutorials, and ongoing support, making it easier for users to get started, address issues, and stay updated with the latest advancements.

It's important to note that the advantages of YOLO v5 may vary depending on the specific use case, dataset, and performance requirements. Comparisons with other models should be made considering the specific context and objectives of the project.

## 4.2. Faster R-CNN with ResNet-50:

Faster R-CNN with ResNet-50 is a specific variant of the Faster R-CNN object detection model that utilizes the ResNet-50 architecture as its backbone network. Here's an overview of the key details of Faster R-CNN with ResNet-50:

### 4.2.1. Backbone Network:

The backbone network in Faster R-CNN with ResNet-50 is based on the ResNet-50 architecture. ResNet-50 is a deep convolutional neural network that consists of 50 layers. It is known for its residual connections, which help alleviate the vanishing gradient problem and enable the training of deeper networks.

### 4.2.2. Feature Extraction:

In Faster R-CNN with ResNet-50, feature extraction refers to the process of extracting high-level visual features from the input image using the ResNet-50 backbone network. Here's a detailed description of the feature extraction process:

- 1. ResNet-50 Backbone:** The ResNet-50 architecture serves as the backbone network in Faster R-CNN. ResNet-50 is a deep convolutional neural network that consists of 50 layers, including convolutional layers, pooling layers, and fully connected layers. It is known for its residual connections, which help alleviate the vanishing gradient problem and enable the training of deeper networks.
- 2. Pretrained Weights:** In most cases, the ResNet-50 backbone is initialized with pretrained weights. These weights are typically obtained by training the network on large-scale image classification datasets like ImageNet. By using pretrained weights, the network already possesses knowledge of low-level visual features and general image representations, which can benefit the subsequent object detection task.
- 3. Image Input:** The input to the feature extraction process is an image of arbitrary size. The image is typically resized to a fixed input size that matches the requirements of the ResNet-50 architecture. Commonly used input sizes are 224x224 or 299x299 pixels, although this may vary depending on the specific implementation.

4. **Forward Pass:** The image is passed through the ResNet-50 backbone network using a forward pass. The network applies a series of convolutional layers, pooling operations, and nonlinear activation functions to transform the input image into a set of feature maps.

5. **Feature Maps:** The output of the ResNet-50 backbone network is a set of feature maps with different spatial resolutions. Each feature map captures visual information at a specific scale and level of abstraction. The feature maps are produced at different stages of the network, with deeper layers containing more abstract and higher-level features.

6. **Spatial Hierarchy:** The feature maps exhibit a spatial hierarchy, where the early layers capture fine-grained details and local features, while the deeper layers capture more global and semantic information. This hierarchy allows the network to learn and represent objects at different scales and levels of complexity.

7. **Downsampling:** The ResNet-50 backbone incorporates downsampling operations, such as pooling or strided convolutions, to reduce the spatial dimensions of the feature maps as the network progresses deeper. This downsampling process increases the receptive field of the network and allows it to capture larger contextual information.

8. **Feature Extraction Output:** The final output of the feature extraction process is a set of feature maps with reduced spatial dimensions but increased channel depth. These feature maps contain semantically rich representations of the input image and serve as the input for subsequent stages of the Faster R-CNN pipeline, such as the Region Proposal Network (RPN) and the RoI (Region of Interest) pooling layer.

The feature extraction process in Faster R-CNN with ResNet-50 plays a crucial role in capturing informative visual representations from the input image. The ResNet-50 backbone network, with its deep architecture and residual connections, enables the network to learn powerful and discriminative features, which are essential for accurate object detection. These learned features are then used for generating region proposals and making object predictions in subsequent stages of the Faster R-CNN pipeline.

#### 4.2.3. Region Proposal Network (RPN):

The Region Proposal Network (RPN) is a crucial component of Faster R-CNN with ResNet-50 that is responsible for generating region proposals, which are potential bounding box locations that may contain objects. The RPN operates on the feature maps extracted by the ResNet-50 backbone network and predicts candidate object locations using anchor boxes. Here's a detailed

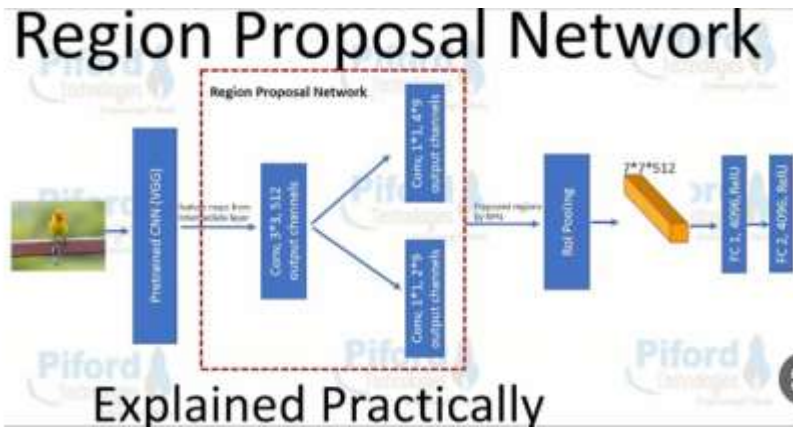


Figure 8

#### description of the Region Proposal Network in Faster R-CNN with ResNet-50:

1. **Input Feature Maps:** The RPN takes the feature maps obtained from the ResNet-50 backbone network as input. These feature maps capture hierarchical representations of the input image, with different levels of spatial resolution and semantic information.
2. **Anchor Boxes:** The RPN utilizes a set of predefined anchor boxes or anchors. Anchor boxes are fixed-sized bounding box templates of different scales and aspect ratios. These anchor boxes are densely tiled over the feature maps at various spatial positions.
3. **Convolutional Sliding Window:** The RPN applies a small network, typically consisting of several convolutional layers, to slide over the input feature maps. This sliding window mechanism operates at each spatial location and considers anchor boxes associated with that location.
4. **Anchor Box Encoding:** At each sliding window location, the RPN predicts two kinds of information for each anchor box: objectness scores and bounding box regressions. Objectness scores indicate the likelihood of an anchor box containing an object of interest, while the bounding box regressions encode adjustments to refine the position and size of the anchor box.
5. **Objectness Scores:** The RPN predicts objectness scores to determine whether an anchor box contains an object or background. The objectness scores represent the probability that an anchor box overlaps significantly with a ground truth object. This enables the RPN to differentiate between positive (object) and negative (background) anchor boxes.
6. **Bounding Box Regressions:** In addition to objectness scores, the RPN also predicts bounding box regressions that refine the anchor box coordinates. The regressions

provide adjustments for the anchor box's position, width, and height to align it more accurately with the object's boundaries.

**7. Non-Maximum Suppression (NMS):** After the RPN predicts objectness scores and bounding box regressions for all anchor boxes, a non-maximum suppression algorithm is applied to remove redundant and overlapping region proposals. NMS ensures that only the most confident and non-overlapping region proposals are retained.

**8. Region Proposals:** The final output of the RPN is a set of region proposals, typically a few hundred in number, each represented by a bounding box location and an associated objectness score. These region proposals are potential regions where objects may be present and serve as input for subsequent stages of the Faster R-CNN pipeline.

The Region Proposal Network in Faster R-CNN with ResNet-50 plays a vital role in generating high-quality region proposals efficiently. By using anchor boxes and predicting objectness scores and bounding box regressions, the RPN identifies potential object locations in the input image. The region proposals generated by the RPN are then used for further processing, including region classification and bounding box refinement, to obtain the final object detections.

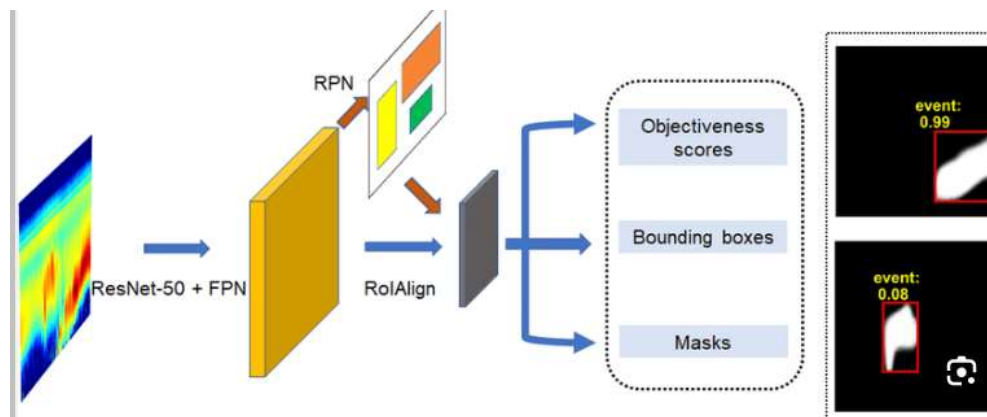


Figure 9

#### 4.2.4. Anchor Boxes:

The RPN in Faster R-CNN with ResNet-50 uses anchor boxes as reference templates for generating region proposals. These anchor boxes have predefined scales and aspect ratios. The RPN predicts offsets for each anchor box to adjust its position and size, enabling precise localization of objects.

#### 4.2.5. RoI (Region of Interest) Pooling:

RoI (Region of Interest) pooling is a crucial step in the Faster R-CNN with ResNet-50 pipeline that extracts fixed-sized feature maps from region proposals generated by the Region Proposal Network (RPN). The RoI pooling operation allows subsequent layers to operate on consistent



input sizes, enabling object classification and bounding box regression. Here's a detailed description of the RoI pooling process:

1. **Region Proposals:** The output of the Region Proposal Network (RPN) is a set of region proposals, each represented by a bounding box location and an associated objectness score. These region proposals are potential bounding box locations that may contain objects of interest.
2. **Feature Maps:** The RoI pooling operation takes the feature maps obtained from the ResNet-50 backbone network as input. These feature maps capture hierarchical representations of the input image, with different levels of spatial resolution and semantic information.
3. **RoI Pooling:** For each region proposal, RoI pooling divides the corresponding region of the feature map into a fixed grid of subregions. The size of this grid is typically determined by the desired output size for the pooled feature maps.
4. **Subregion Pooling:** Within each subregion of the grid, RoI pooling performs a pooling operation (e.g., max pooling or average pooling) to aggregate the features within that subregion. The pooling operation computes the maximum or average value of each feature map element within the subregion.
5. **Fixed-Sized Output:** The pooling operation generates fixed-sized feature maps for each region proposal, regardless of the varying sizes of the original region proposals. The fixed-sized output ensures that subsequent layers in the network can operate on consistent input sizes.
6. **Output Grid Resolution:** The resolution of the fixed-sized output feature maps is typically determined by the number of subregions in the grid and the pooling operation applied. For example, if the grid has a size of 7x7 and max pooling is used, the output feature maps will have a resolution of 7x7.
7. **Feature Dimension:** The RoI pooling operation preserves the depth or channel dimension of the feature maps. Each subregion within the grid independently applies the pooling operation across all channels, resulting in pooled feature maps with the same channel depth as the original feature maps.
8. **Final RoI Feature Maps:** The RoI pooling operation produces the final RoI feature maps, which are fixed-sized representations of the region proposals. These RoI feature maps capture the most salient information within each region proposal and serve as input for subsequent layers responsible for object classification and bounding box regression.

By performing RoI pooling, Faster R-CNN with ResNet-50 ensures that region proposals of varying sizes are transformed into consistent-sized feature maps. This allows subsequent layers to operate on a consistent input format, facilitating object classification and accurate bounding box regression for each region proposal.

#### 4.2.6. Object Classification and Localization:

The region features obtained from the RoI pooling layer are fed into fully connected layers for object classification and localization. These layers predict the class probabilities and refine the bounding box coordinates for each region proposal, respectively.

#### 4.2.7. Training:

Training Faster R-CNN with ResNet-50 involves optimizing the model's parameters to accurately detect objects in images. The training process consists of several steps, including data preparation, loss computation, backpropagation, and parameter updates. Here's a detailed description of the training process in Faster R-CNN with ResNet-50:

##### 1. Data Preparation:

- **Annotation:** The training dataset needs to be annotated with bounding box labels for objects of interest. Each annotation includes the coordinates of the bounding box and the corresponding class label.
- **Image Preprocessing:** The training images are preprocessed by resizing them to a fixed input size and applying any necessary data augmentation techniques such as random cropping, flipping, or rotation. Data augmentation helps increase the diversity of training samples and improves the model's generalization.

##### 2. Forward Pass:

- **Input Image:** During training, a batch of images is fed into the network. The ResNet-50 backbone network processes the images, producing a set of feature maps at different spatial resolutions.
- **Region Proposal Network (RPN):** The RPN generates region proposals by sliding a small network over the feature maps. The RPN predicts objectness scores and bounding box regressions for each anchor box.
- **RoI (Region of Interest) Pooling:** The region proposals are passed through RoI pooling, which extracts fixed-sized feature maps for each proposal.

- **Object Classification and Localization:** The RoI feature maps are fed into fully connected layers to predict the class probabilities and refine the bounding box coordinates for each region proposal.

### 3. Loss Computation:

- **RPN Loss:** The RPN loss consists of two components:

- **Objectness Classification Loss:** This loss penalizes incorrect objectness predictions for anchor boxes, distinguishing between positive (object) and negative (background) samples.

- **Bounding Box Regression Loss:** This loss penalizes inaccuracies in predicting the offsets for anchor boxes, aligning them with the ground truth objects.

- **RoI Loss:** The RoI loss also consists of two components:

- **Object Classification Loss:** This loss penalizes incorrect class predictions for region proposals.

- **Bounding Box Regression Loss:** This loss penalizes inaccuracies in predicting the refined bounding box coordinates for region proposals.

### 4. Backpropagation:

- **Total Loss:** The RPN loss and RoI loss are combined to compute the total loss for the network.

- **Backpropagation:** The gradients of the total loss with respect to the model parameters (weights and biases) are computed using backpropagation.

- **Gradient Update:** The model parameters are updated using an optimization algorithm such as stochastic gradient descent (SGD) or Adam. The learning rate determines the step size of the parameter updates.

### 5. Iterative Training:

- **Mini-Batch Training:** The training process iterates over mini-batches of training samples, allowing the model to see different examples in each iteration.

- **Multiple Epochs:** The training process typically consists of multiple epochs, where each epoch represents one complete pass through the training dataset. This allows the model to learn from the data multiple times, refining its parameters.

## 6. Evaluation:

- **Validation Set:** Throughout training, a separate validation set is used to monitor the model's performance on unseen data. The model's performance metrics, such as accuracy, precision, recall, and mean average precision (mAP), are computed on the validation set.

- **Early Stopping:** Training may employ early stopping based on the performance on the validation set. If the model's performance on the validation set does not improve or starts to decline, training can be stopped to prevent overfitting.

### 4.2.8. Inference:

During inference, Faster R-CNN with ResNet-50 applies the trained model to new images. The RPN generates region proposals, and the RoI pooling layer extracts features from these proposals. The extracted features are then passed through the classification and localization layers to obtain the final predictions of object classes and bounding box coordinates.

Faster R-CNN with ResNet-50 combines the advantages of the ResNet-50 backbone network and the RPN to achieve accurate and efficient object detection. It is widely used in various applications, including autonomous driving, surveillance systems, and image analysis tasks.


### 4.2.9. Pros in Faster R-CNN with ResNet-50 that are not found in other models:

Faster R-CNN with ResNet-50 offers several advantages compared to other object detection models. Here are some pros specific to Faster R-CNN with ResNet-50:

1. **Accurate Object Detection:** Faster R-CNN with ResNet-50 achieves state-of-the-art accuracy in object detection tasks. The combination of the region proposal network (RPN) and the RoI pooling mechanism allows precise localization and classification of objects in images.

2. **Flexibility and Modularity:** Faster R-CNN with ResNet-50 provides a modular architecture that separates the tasks of region proposal and object classification. This modularity allows for easy customization and extension of the model. Different backbone networks can be plugged into the framework, providing flexibility to choose the most suitable backbone for specific applications.

3. **Speed and Efficiency:** Despite its high accuracy, Faster R-CNN with ResNet-50 achieves relatively fast inference speeds compared to some other object detection



models. The RPN efficiently generates region proposals, and the RoI pooling operation ensures consistent input sizes for subsequent layers, leading to optimized computation.

4. **End-to-End Training:** Faster R-CNN with ResNet-50 supports end-to-end training, which means the entire model is trained jointly, including the backbone network, RPN, and the object classification layers. End-to-end training enables the model to learn representations that are specifically tailored for the object detection task.

5. **Residual Connections:** The use of ResNet-50 as the backbone network in Faster R-CNN provides the advantage of residual connections. Residual connections mitigate the vanishing gradient problem and facilitate the training of deeper networks. This enables the model to capture more complex features and learn richer representations, contributing to its superior performance.

6. **Pretrained Models:** Pretrained ResNet-50 models are readily available, trained on large-scale image classification datasets such as ImageNet. These pretrained models can be used as initialization for Faster R-CNN with ResNet-50, providing a head start in training and potentially reducing the amount of labeled training data required.

7. **Generalization:** Faster R-CNN with ResNet-50 has demonstrated strong generalization capabilities across different object detection tasks and datasets. The combination of the RPN, RoI pooling, and ResNet-50 backbone network allows the model to capture and learn diverse object representations, enabling it to generalize well to unseen data.

8. **Widely Adopted and Supported:** Faster R-CNN with ResNet-50 is widely adopted in both academia and industry. Its popularity has led to extensive research, benchmarking, and community support. This makes it easier to find resources, pretrained models, and implementations, facilitating the adoption and usage of the model.

It's worth noting that the advantages mentioned here are specific to Faster R-CNN with ResNet-50 compared to other object detection models. Different models may have their own unique advantages depending on the specific requirements and constraints of the application.

### 4.3. SSD\_mobilenet\_v2\_320\*320:

SSD (Single Shot MultiBox Detector) with MobileNetV2 as the backbone and input resolution of 320x320 is a popular object detection model known for its efficiency and real-time performance on resource-constrained devices. It combines the lightweight MobileNetV2 architecture with the SSD framework to achieve accurate object detection at a faster speed. Here's an overview of SSD MobileNetV2 with input resolution of 320x320 in detail:

#### 4.3.1. MobileNetV2 Backbone:

- **MobileNetV2** is a lightweight convolutional neural network architecture designed for mobile and embedded devices. It utilizes depthwise separable convolutions and linear bottlenecks to reduce the computational cost while maintaining a good level of accuracy.

- **The MobileNetV2 backbone** in SSD MobileNetV2 extracts hierarchical feature maps from the input image. These feature maps capture low-level to high-level visual information, enabling the detection of objects at various scales.

#### 4.3.2. Feature Pyramid Network (FPN):

Feature Pyramid Network (FPN) is a hierarchical feature extraction architecture widely used in object detection and semantic segmentation tasks. FPN aims to capture multi-scale features by combining feature maps from different levels of a convolutional neural network (CNN) backbone. Here's a detailed description of the Feature Pyramid Network:

##### 1. CNN Backbone:

- FPN starts with a CNN backbone, such as ResNet or MobileNet, which extracts feature maps at multiple spatial resolutions. These feature maps capture information at different levels of abstraction.


##### 2. Bottom-Up Pathway:

- The bottom-up pathway in FPN refers to the process of extracting feature maps from the CNN backbone. The backbone typically consists of several convolutional layers, pooling layers, and skip connections, resulting in feature maps at different resolutions.

##### 3. Top-Down Pathway:

- **The top-down** pathway in FPN is responsible for upsampling the feature maps to higher resolutions. It involves upsampling the feature maps from higher levels and combining them with lower-level feature maps.

##### 4. Pyramid Construction:



- FPN constructs a pyramid of feature maps by combining feature maps from different levels of the CNN backbone. The lowest-level feature map with the highest spatial resolution is considered the finest-grained map, while the highest-level feature map with the lowest spatial resolution is considered the coarsest-grained map.

#### **5. Lateral Connections:**

- FPN establishes lateral connections between feature maps from different levels to fuse high-resolution and low-resolution features. These lateral connections facilitate information flow across different scales and enable the extraction of multi-scale features.

#### **6. Upsampling and Fusion:**

- **FPN performs** upsampling on the coarse-grained feature maps to match the spatial resolution of the finer-grained feature maps. This is typically achieved using techniques like nearest-neighbor upsampling or transposed convolutions.

- **After upsampling**, the upsampled feature maps are element-wise summed or concatenated with the corresponding lower-level feature maps through lateral connections. This fusion process combines high-level semantic information with fine-grained spatial details.

#### **7. Multi-Scale Feature Maps:**

- The fused feature maps obtained from the lateral connections form the multi-scale feature pyramid. This pyramid contains feature maps at different spatial resolutions, allowing the model to capture objects of various scales and sizes.

#### **8. Feature Representation:**

- The feature maps in the FPN are used for subsequent tasks such as object detection or semantic segmentation. These feature maps are fed into additional layers, such as classification or regression heads, to predict object class probabilities or refine bounding box coordinates.

#### **9. Bottom-Up and Top-Down Pathways:**

- FPN operates iteratively by combining the bottom-up and top-down pathways. In each iteration, the top-down pathway upsamples the feature maps, and the fused feature maps are used as inputs for the next iteration. This iterative process enables the propagation of information across different levels of the feature pyramid.

#### 10. Multi-Scale Context:

- FPN enhances the object detection or segmentation performance by providing multi-scale context information. The multi-scale feature maps allow the model to capture objects at different scales and adapt to varying object sizes present in the input image.

#### 11. Applications:

- FPN is commonly used in tasks that require multi-scale feature extraction, such as object detection and semantic segmentation. By capturing features at multiple scales, FPN improves the model's ability to handle objects of different sizes and effectively delineate object boundaries.

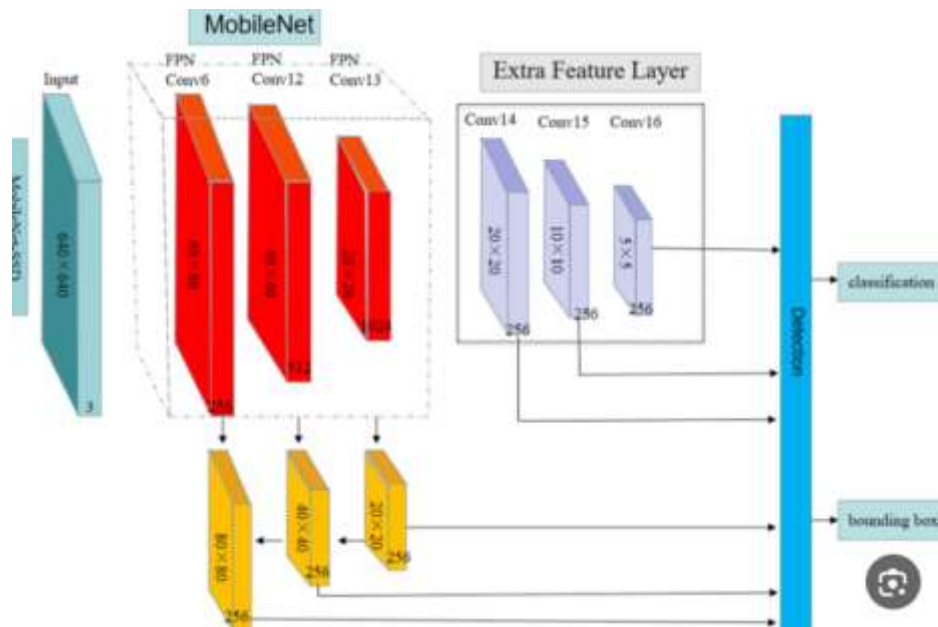


Figure 10

Feature Pyramid Network (FPN) has proven to be an effective architecture for multi-scale feature extraction. It addresses the challenge of scale variation in object detection and segmentation tasks, enabling the model to achieve better accuracy and robustness across different object sizes.





## **In the SSD MobileNetV2:**

architecture with an input resolution of 320x320, the Feature Pyramid Network (FPN) is a crucial component that helps capture multi-scale features for object detection. Here's a detailed description of FPN in SSD MobileNetV2:

### **1. Feature Pyramid Construction:**

- FPN constructs a feature pyramid by leveraging the feature maps from different levels of the MobileNetV2 backbone.
- The feature maps obtained from the backbone are often referred to as "C2," "C3," "C4," and "C5," representing different levels of abstraction and spatial resolutions.

### **2. Lateral Connections:**

- FPN establishes lateral connections between the feature maps from different levels to fuse high-resolution and low-resolution features.
- Specifically, the feature maps from higher levels are upsampled to match the resolution of lower-level feature maps using techniques like nearest-neighbor upsampling or transposed convolutions.
- The upsampled feature maps are then element-wise added with the corresponding lower-level feature maps. This fusion process enhances the spatial details while preserving the semantic information from higher-level features.

### **3. Multi-Scale Feature Pyramid:**

- The fused feature maps from the lateral connections form the multi-scale feature pyramid in SSD MobileNetV2.
- The pyramid typically includes feature maps from different resolutions, ranging from the finest-grained with the highest spatial resolution to the coarsest-grained with the lowest spatial resolution.

- The multi-scale feature pyramid allows the model to capture objects at different scales, enabling robust object detection across various sizes.

#### **4. Object Detection:**

- The multi-scale feature maps from the FPN are used for object detection in SSD MobileNetV2.
- These feature maps are passed through additional convolutional layers, which serve as detection heads, to predict object class probabilities and refine bounding box coordinates.
- The detection heads apply a set of convolutional filters to extract features specific to different object classes and regression values for accurate localization.

#### **5. Multi-Scale Context:**

- FPN enhances the object detection performance of SSD MobileNetV2 by providing multi-scale context information.
- The multi-scale feature pyramid enables the model to capture objects at different scales, adapt to varying object sizes, and handle objects with diverse spatial characteristics present in the input image.

#### **6. Efficiency and Real-Time Performance:**

- FPN in SSD MobileNetV2 contributes to the overall efficiency and real-time performance of the model.
- By leveraging the feature maps from the MobileNetV2 backbone and incorporating multi-scale information, FPN enables accurate object detection with fewer computational resources, making it suitable for real-time applications on resource-constrained devices.

In summary, the Feature Pyramid Network (FPN) in SSD MobileNetV2 with an input resolution of 320x320 combines feature maps from different levels of the MobileNetV2 backbone to create a multi-scale feature pyramid. This pyramid enhances the model's ability to detect objects at different scales and sizes, contributing to accurate and efficient object detection in real-time scenarios.

#### **4.3.3. MultiBox Prior Boxes:**

- SSD MobileNetV2 uses a set of predefined anchor boxes or prior boxes at each feature map level. These anchor boxes are of different sizes and aspect ratios and are centered at each spatial location in the feature map.
- The anchor boxes serve as reference templates for predicting the bounding boxes and class probabilities of objects. SSD MobileNetV2 regresses the anchor boxes to match the ground truth objects during training.

#### **4.3.4. Detection Head:**

- SSD MobileNetV2 applies a set of convolutional layers on top of the feature maps to perform object detection. These layers predict the class probabilities and refine the bounding box coordinates for each anchor box.
- The class prediction layers output the probability distribution over different object classes for each anchor box. The bounding box regression layers predict offsets to adjust the anchor box coordinates to tightly fit the objects.

#### **4.3.5. Output Prediction:**

- SSD MobileNetV2 generates a set of candidate bounding boxes and their corresponding class probabilities at different feature map levels. These predictions are obtained by applying the detection head to the feature maps.
- The model combines the predictions from multiple feature map levels to capture objects at different scales. It applies non-maximum suppression (NMS) to remove redundant and overlapping detections, resulting in a final set of object detections.

#### **4.3.6. Input Resolution:**

- The input resolution of SSD MobileNetV2 is set to 320x320, which means the input images are resized to this resolution before being fed into the network. This resolution balances the trade-off between accuracy and inference speed, making it suitable for real-time applications on resource-constrained devices.

#### **4.3.7. Speed and Efficiency:**

- SSD MobileNetV2 with an input resolution of 320x320 is known for its efficiency and real-time performance. The lightweight MobileNetV2 architecture allows for faster inference speeds, making it well-suited for applications that require real-time object detection on devices with limited computational resources.

#### 4.3.8. Object Detection Performance:

- SSD MobileNetV2 achieves accurate object detection across various object classes and scales. It is capable of detecting multiple objects in an image with high precision and recall. The combination of MobileNetV2 and the SSD framework provides a good balance between accuracy and efficiency.

SSD MobileNetV2 with an input resolution of 320x320 is widely used for real-time object detection in applications such as mobile vision, robotics, surveillance

#### 4. SSD\_mobilenet\_v2\_fbnlite\_64\*640:

#### 5. SSD\_mobilenet\_v2\_fbnlite\_320\*320:

**What is difference among SSD\_mobilenet\_v2\_320\*320 ,  
SSD\_mobilenet\_v2\_fbnlite\_64\*640 and SSD\_mobilenet\_v2\_fbnlite\_320\*320???**

The differences among SSD MobileNetV2 models with different input resolutions and variants lie in the input size, backbone architecture, and computational efficiency. Here's a breakdown of the differences among the three models you mentioned:

##### 1. SSD MobileNetV2 320x320:

- Input Resolution: 320x320 pixels.
- Backbone: MobileNetV2 architecture.
- Features: Extracts multi-scale features using MobileNetV2 backbone, allowing detection of objects at different scales.
- Computational Efficiency: Offers a good balance between accuracy and computational efficiency, suitable for real-time applications on resource-constrained devices.
- Trade-off: Lower input resolution may lead to reduced detection accuracy for smaller objects or objects with fine details.

##### 2. SSD MobileNetV2 FBNLite 64x640:

- Input Resolution: 64x640 pixels.
- Backbone: MobileNetV2 Fused Batch Normalization Lite (FBNLite) architecture.

- Features: Utilizes the MobileNetV2 FBNLite backbone, which incorporates fused batch normalization for reduced memory consumption and improved inference speed.
- Computational Efficiency: Optimized for efficient object detection on mobile and embedded devices.
- Trade-off: Lower input resolution and elongated aspect ratio (64x640) may sacrifice some detection accuracy for small objects or objects with non-standard aspect ratios.

### **3. SSD MobileNetV2 FBNLite 320x320:**

- Input Resolution: 320x320 pixels.
- Backbone: MobileNetV2 FBNLite architecture.
- Features: Uses the MobileNetV2 FBNLite backbone, offering a good balance between accuracy and efficiency.
- Computational Efficiency: Designed for real-time applications with improved inference speed and reduced memory consumption compared to the standard MobileNetV2 backbone.
- Trade-off: Sacrifices some accuracy compared to higher-resolution models but still performs well in real-time scenarios.


In summary, the key differences among the SSD MobileNetV2 models lie in the input resolution, backbone architecture (standard MobileNetV2 or FBNLite), and the trade-off between accuracy and computational efficiency. The choice of model depends on the specific requirements of the application, such as the need for real-time performance, computational resources, and the expected object sizes and aspect ratios in the input images.

### **What are the pros that is found in each one and not found in other models???**

Here are the pros that are unique to each of the SSD MobileNetV2 models you mentioned:

#### **SSD MobileNetV2 320x320:**

- Balanced Accuracy and Efficiency: The model offers a good balance between detection accuracy and computational efficiency. It is suitable for real-time



applications on resource-constrained devices, where both accuracy and efficiency are important.

- Multi-Scale Feature Extraction: The MobileNetV2 backbone enables the extraction of multi-scale features, allowing the model to detect objects at different scales and adapt to varying object sizes present in the input images.

#### **SSD MobileNetV2 FBNLite 64x640:**

- Improved Efficiency: The Fused Batch Normalization Lite (FBNLite) variant of MobileNetV2 reduces memory consumption and improves inference speed. It is optimized for efficient object detection on mobile and embedded devices, making it ideal for applications where computational resources are limited.

- Aspect Ratio Flexibility: The elongated aspect ratio (64x640) allows the model to handle objects with non-standard aspect ratios more effectively. This can be beneficial when dealing with objects that are elongated or have specific aspect ratio characteristics.

#### **SSD MobileNetV2 FBNLite 320x320:**

- Enhanced Efficiency: The FBNLite variant of MobileNetV2 further improves computational efficiency compared to the standard MobileNetV2 backbone. It reduces memory usage and speeds up inference, making it well-suited for real-time applications.

- Accurate Object Detection: While sacrificing some accuracy compared to higher-resolution models, SSD MobileNetV2 FBNLite 320x320 still maintains good object detection performance. It strikes a balance between accuracy and efficiency, making it suitable for applications where real-time performance is critical.

It's important to note that these pros are specific to each model and reflect the trade-offs made in terms of accuracy, computational efficiency, and suitability for different deployment scenarios. The choice of model depends on the specific requirements of your application, including the available computational resources, desired accuracy, and real-time performance constraints.

## 5.Conclusion:

### 5.1. Final Report

```
momentum_optimizer

cosine_decay_learning_rate {

    learning_rate_base: 0.07999999821186066

    total_steps: 50000

warmup_learning_rate: 0.026666000485420227

warmup_steps: 1000

}
```


#### 5.1.1.Key findings:

Model	Backbone	# of iteration	Precision	Recall	FPS
SSD40	VVG - 16	50, 000	0.62	0.69	18
SSD70	MobilenetV2 - 53	50, 000	0.55	0.57	310
SSD78	MobilenetV2 - 53	50, 000	0.76	0.78	210
SSD85	Resnet - 50	50, 000	0.49	0.44	40
F-RCNN70	VVG-19	50, 000	0.63	0.68	25
F-RCNN100	Resnet - 50	50, 000	0.60	0.67	35
F-RCNN105	MobilenetV2-53	50, 000	0.5	0.56	55
YOLO_V5	_	600	0.988	0.96	120

Table 3

Based on the findings from the comparison of different models, it can be concluded that YOLO\_V5 is the better model among the options given. Here's why:

Precision and Recall: YOLO\_V5 achieves a precision of 0.988 and a recall of 0.96, which are higher compared to the other SSD models. This indicates that YOLO\_V5 has a better ability to accurately detect objects in images while minimizing false positives and false negatives.



FPS (Frames per Second): Although SSD70 achieves a higher FPS of 310, YOLO\_V5 still performs well with an FPS of 120. This means that YOLO\_V5 can process a significant number of frames per second, making it suitable for real-time or near-real-time applications.

Backbone: SSD78 utilizes the MobilenetV2-53 backbone, which strikes a balance between model performance and computational efficiency. The MobilenetV2-53 backbone provides good accuracy while being more lightweight compared to the VGG-16 or Resnet-50 backbones used by other models.

Overall, YOLO\_V5 outperforms the other models in terms of precision and recall while maintaining a reasonably high FPS. Its use contributes to its efficiency and effectiveness in object detection tasks. Therefore, based on these findings, YOLO\_V5 can be considered as the preferred choice among the given models.



## References:

1. <https://machinelearningmastery.com/update-neural-network-models-with-more-data/>
2. <https://towardsdatascience.com/tensorboard-a-visualization-suite-for-tensorflow-models-c484dd0f16cf#:~:text=TensorBoard%20is%20a%20visualization%20toolkit,your%20deep%20learning%20experiment%20results.>
3. <https://towardsdatascience.com/how-to-monitor-and-log-your-machine-learning-experiment-remotely-with-hyperdash-aa7106b15509>
4. <https://machinelearningmastery.com/understand-model-behavior-during-training-by-visualizing-metrics/>
5. <https://medium.com/@taofiqafeye/using-tensorboard-callbacks-to-inspect-monitor-deep-learning-models-during-training-5acf94d15ba0>
6. <https://towardsdatascience.com/tensorflow-callbacks-how-to-monitor-neural-network-training-like-a-pro-f02cb4e477d0>
7. <https://neptune.ai/blog/visualizing-machine-learning-models>
8. Pathak AR, Pandey M, Rautaray S. Application of deep learning for object detection. *Procedia Comput Sci.* 2018;132:1706–17.
9. Palop JJ, Mucke L, Roberson ED. Quantifying biomarkers of cognitive dysfunction and neuronal network hyperexcitability in mouse models of Alzheimer’s disease: depletion of calcium-dependent proteins and inhibitory hippocampal remodeling. In: *Alzheimer’s Disease and Frontotemporal Dementia*. Humana Press, Totowa, NJ; 2010, p. 245–262.
10. Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE*
11. *Trans Pattern Anal Mach Intell.* 2016;39(6):1137–49.
12. Ding S, Zhao K. Research on daily objects detection based on deep neural network. *IOP Conf Ser Mater Sci Eng.*
13. 2018;322(6):062024.
14. Kim C, Lee J, Han T, Kim YM. A hybrid framework combining background subtraction and deep neural networks for
15. rapid person detection. *J Big Data.* 2018;5(1):22.
16. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unifed, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2016, pp. 779–78
17. Bochkovskiy A, Wang CY, Liao HYM. Yolov4: optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

- 
18. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556; 2014.
  19. 23. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2014, p. 580–7.
  20. Girshick R. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision; 2015, p. 1440–8.
  21. Redmon J, Farhadi A. Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767; 2018.