CHAPTER 1
INTRODUCTION

1.1 Background :

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Object detection is considered one of the most challenging problems in this field of computer vision, as it involves the combination of object classification and object localization within a scene. The most challenging part is real time detection with high accuracy and fast response. Object detection involves drawing a boundary box around the detected object and classifying it based on the classes of data it is trained on.

1.2 Problem Statement :

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classification, which is defined as predicting the class of the image. A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object).

1.3 Objectives :

The objectives of object detection are:

i. To describe a scene as precisely as a human being.
ii. Detection of defects/anomalies in certain contexts such as automatic video surveillance.
iii. Detection, recognition and tracking of pedestrians, obstacles, potholes, traffic participants, etc. in real time, for self-driving / autonomous vehicles.
iv. Detection and recognition of hand gestures for Human Computer Interaction (HCI).

CHAPTER 2

LITERATURE REVIEW

The survey throws light on the key aspects of the papers studied and also highlights their positive and negative points:

SL. NO	AUTHOR, TITLE AND YEAR OF PUBLICATION	METHODOLOGY	ADVANTAGES	LIMITATIONS
1.	Joseph Redmon, Santosh Divvala, Ross Girshick Ali Farhadi **You Only Look Once: Unified, Real-Time Object Detection** – May 2016.	i) Resizes the input image to 448*448. ii) A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation using S x S grid on input. iii)Processes images in real time at 45 frames per second (fps).	i)YOLO is extremely fast. Since, frame detection is a regression problem it does not require a complex pipeline. ii) YOLO achieves more than twice the mean average precision of other real-time systems. iii) YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM	YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. Also YOLO makes arbitrary guesses on boundary boxes.

			and R-CNN by a wide margin.	
2.	Joseph Redmon, Ali Farhadi **YOLO9000: Better, Faster, Stronger** – December 2016.	i)YOLO9000 also known as YOLOv2 is an improvised version of YOLO algorithm and is capable of detecting over 9000 object categories. ii)Each prediction includes 4 parameters for the boundary box, 1 box confidence score (objectness) and 20 class probabilities. i.e. 5 boundary boxes with 25 parameters: 125 parameters per grid cell.	i)Improves accuracy by moving the class prediction from the cell level to the boundary box level. ii)YOLOv2 has added batch normalization in convolution layers. This removes the need for dropouts and pushes mAP up 2%.	Difficult to detect objects when too many objects are overlapped.
3.	Hyeok-June Jeong, Kyeong-Sik Park, Young-Guk Ha **Image Preprocessing for Efficient Training of YOLO Deep Learning Networks**. – January 2018.	This paper implements a pre-processor for YOLO object detector. It uses the following 4 steps: i) Image picker: This subsystem randomly picks out objects from a prepared crawled image set.	Since the images are pre-processed the performance of YOLO is better when compared to the performance without pre-processing.	The response / detection time is more due to the pre-processing of images. Also this methodology cannot be used for real time systems.

		ii) Scale Modifier: This subsystem reduces the size of objects cropped in the Image picker to an appropriate size. iii) Image Maker: In this step, the modified object image is pasted into base images. Base images with similar backgrounds and sizes. iv)Annotation Creator: This subsystem annotates the position and size of newly placed objects in the base image.		

CHAPTER 3
METHODOLOGY

3.1 Details of Implementation :

The object detection using You Only Look Once (YOLO) algorithm is been implemented in different versions by modifying the network structure of the original YOLO algorithm to improve accuracy, speed and to apply to real time scenarios.

3.1.1 YOLO :

YOLO is refreshingly simple as shown in figure 1.1.A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

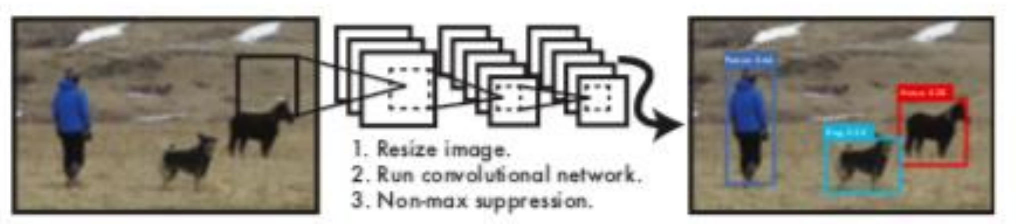


Figure 1.1

YOLO unifies the separate components of object detection into a single neural network. Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons glob- ally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real- time speeds while maintaining high average precision.

This system divides the input image into an S × S grid as shown in figure 1.2. If the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object.

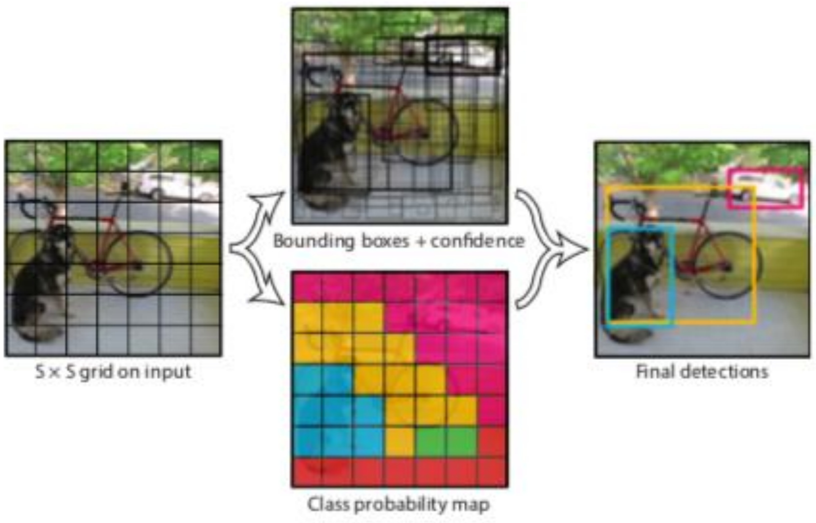


Figure 1.2

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 also how accurate it thinks the box is that it predicts. Formally we define confidence as Pr(Object) $*$ IOU$^{\text{truth}}$ If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the centre of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities, Pr(Class¡ |Object). These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

At test time we multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

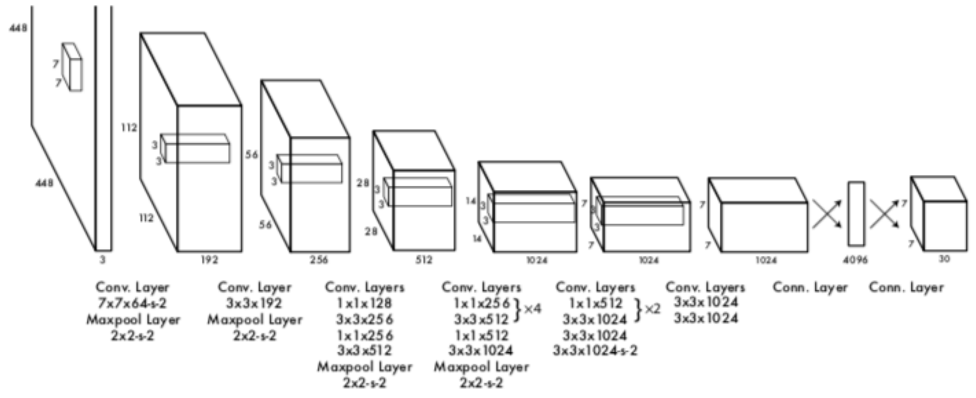


Figure 1.3

YOLO has been implemented as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

Our network architecture is inspired by the GoogLeNet model for image classification. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1 × 1 reduction layers followed by 3 × 3 convolutional layers. The full network is shown in figure 1.3.

We also train a fast version of YOLO designed to push the boundaries of fast object detection. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

The final output of our network is the 7 × 7 × 30 tensor of predictions.

3.1.2 YOLOv2 :

YOLOv2 is an improvised version of the YOLO algorithm to detect more than 9000 classes of images.

For YOLOv2, we first fine tune the classification network at the full 448 × 448 resolution for 10 epochs on ImageNet. This gives the network time to adjust its filters to work better on higher resolution input. We then fine tune the resulting network on detection. This high resolution classification network gives us an increase of almost 4% mAP (Mean Average Precision).

YOLOv2 removes the fully connected layers from YOLO and use anchor boxes to predict bounding boxes. First we eliminate one pooling layer to make the output of the net- work's convolutional layers higher resolution. We also shrink the network to operate on 416 input images instead of 448×448. We do this because we want an odd number of locations in our feature map so there is a single centre cell. Objects, especially large objects, tend to occupy the centre of the image so it's good to have a single location right at the centre to predict these objects instead of four locations that are all nearby. YOLO's convolutional layers downsample the image by a factor of 32 so by using an input image of 416 we get an output feature map of 13 × 13.

3.1.3 YOLO – R :

YOLO-R is a modified version of YOLOv2 designed for pedestrian detection.

First, three Passthrough layers were added to the original YOLO network. The Passthrough layer consists of the Route layer and the Reorg layer. Its role is to connect the shallow layer pedestrian features to the deep layer pedestrian features and link the high and low resolution pedestrian features. The role of the Route layer is to pass the pedestrian characteristic information of the specified layer to the current layer, and then use the Reorg layer to reorganize the feature map so that the currently-introduced Route layer feature can be matched with the feature map of the next layer .The three Passthrough layers added in this algorithm can well transfer the network's shallow pedestrian fine-grained features to the deep network, enabling the network to better learn shallow pedestrian feature information. This paper also changes the layer number of the Passthrough layer connection in the original YOLO algorithm from Layer 16 to Layer 12 to increase the ability of the network to extract the information of the shallow pedestrian features. The improvement was tested on the INRIA pedestrian dataset. The experimental results show that this method can effectively improve the detection accuracy of pedestrians, while reducing the false detection rate and the missed detection rate, and the detection speed can reach 25 frames per second.

The pedestrian detection process of the YOLO network model is shown in figure 2.1:

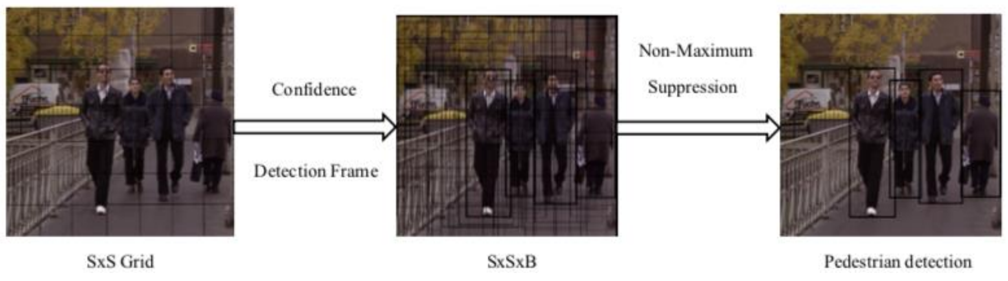


Figure 2.1

i) First divide the image into a grid of SxS .If the pedestrian is in a grid, the grid is responsible for detecting the pedestrian .Each grid predicts B detection boxes and the confidence of these detection boxes, The number of detection frames for each picture is SxSxB.

ii) Each detection box has 5 predicted values (X, Y, W, H, Conf).Where X,Y is the offset of the centre of the prediction box relative to the cell boundary, and W and H are the ratios of the predicted box width to the entire image, and Conf represents the confidence of the detection box.

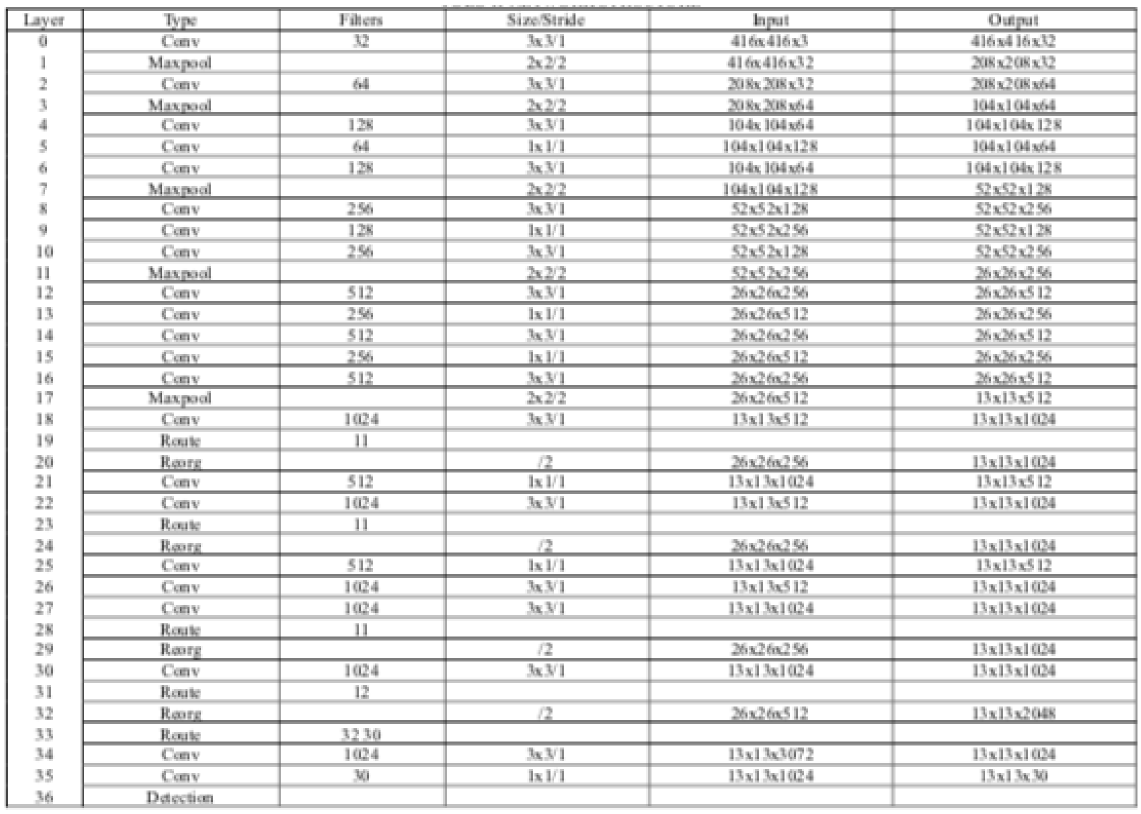
iii) Each grid predicts the pedestrian's conditional probability Pr(class|object), provided that the known grid contains pedestrians.

iv) At the time of detection, the conditional probability is multiplied by the predictive value of different detection box confidences to obtain the confidence score for each detection box pedestrian category. These pedestrian types also include the probability of pedestrians appearing in the detection frame and check the match between the box and the pedestrian goal. The confidence score formula as,

$$\text{Conf(class)Prclass}\,IOU^{\text{Ttuth}}$$

In the formula Prclass represents the probability of pedestrians appearing in the grid, and IOU^{Truth} indicates the overlapping ratio of the area between the predict box and the ground truth box. Pred indicates the area of the prediction box, and Truth indicates the area of the ground truth box. The larger the IOU, the higher the accuracy of pedestrian detection. The final output vector of each picture through the network is SxSxBx[X,Y,W,H,Conf,Conf(class)].

Three Passthrough layers have been added to the original YOLO v2 network, and the number of layers connected to the Route layer in the Passthrough layer in the original YOLO v2 algorithm has been improved from the 16th layer to the 12th layer. This constitutes a new YOLO-R network structure, which allows the improved network to further improve pedestrian detection accuracy. The network structure diagram of YOLO-R is shown in figure 2.2.



Layer	Type	Filters	Size/Stride	Input	Output
0	Conv	32	3x3/1	416x416x3	416x416x32
1	Maxpool		2x2/2	416x416x32	208x208x32
2	Conv	64	3x3/1	208x208x32	208x208x64
3	Maxpool		2x2/2	208x208x64	104x104x64
4	Conv	128	3x3/1	104x104x64	104x104x128
5	Conv	64	1x1/1	104x104x128	104x104x64
6	Conv	128	3x3/1	104x104x64	104x104x128
7	Maxpool		2x2/2	104x104x128	52x52x128
8	Conv	256	3x3/1	52x52x128	52x52x256
9	Conv	128	1x1/1	52x52x256	52x52x128
10	Conv	256	3x3/1	52x52x128	52x52x256
11	Maxpool		2x2/2	52x52x256	26x26x256
12	Conv	512	3x3/1	26x26x256	26x26x512
13	Conv	256	1x1/1	26x26x512	26x26x256
14	Conv	512	3x3/1	26x26x256	26x26x512
15	Conv	256	1x1/1	26x26x512	26x26x256
16	Conv	512	3x3/1	26x26x256	26x26x512
17	Maxpool		2x2/2	26x26x512	13x13x512
18	Conv	1024	3x3/1	13x13x512	13x13x1024
19	Route	11			
20	Reorg		/2	26x26x256	13x13x1024
21	Conv	512	1x1/1	13x13x1024	13x13x512
22	Conv	1024	3x3/1	13x13x512	13x13x1024
23	Route	11			
24	Reorg		/2	26x26x256	13x13x1024
25	Conv	512	1x1/1	13x13x1024	13x13x512
26	Conv	1024	3x3/1	13x13x512	13x13x1024
27	Conv	1024	3x3/1	13x13x1024	13x13x1024
28	Route	11			
29	Reorg		/2	26x26x256	13x13x1024
30	Conv	1024	3x3/1	13x13x1024	13x13x1024
31	Route	12			
32	Reorg		/2	26x26x512	13x13x2048
33	Route	32 30			
34	Conv	1024	3x3/1	13x13x3072	13x13x1024
35	Conv	30	1x1/1	13x13x1024	13x13x30
36	Detection				

Figure 2.2

In the YOLO-R network, a Passthrough layer is added before the 21st layer, It is composed of a Route layer (19th layer) and a Reorg layer (20th layer), combine the features of the maxpooling shallow layer features in the 11th layer with the deep layer features in the 22nd layer. The Passthrough layer was added before the 25th layer and consisted of a Route layer

(23rd layer) and a Reorg layer (24th layer),combine the features of the maxpooling shallow layer features in the 11th layer with the deep layer features in the 25th layer. The Passthrough layer was added before the 30th layer and consisted of the Route layer (28th layer) and the Reorg layer (29th layer),combine the features of the maxpooling shallow layer features in the 11th layer with the deep layer features in the 30th layer. And the original YOLO network structure in the 31rd layer of the Route layer from the original 16th to the 12th layer, in order to extract more shallow layer feature information and the deep layer information to do fusion.

3.1.4 YOLO-LITE :

YOLO-LITE, a real-time object detection model developed to run on portable devices such as a laptop or cell phone lacking a Graphics Processing Unit (GPU).

In YOLO-R, a spatial refinement module is introduced as shown in Fig to enhance the performance of YOLOv2, then present a selective dropout channel pruning approach to accelerate and compress the refined model.

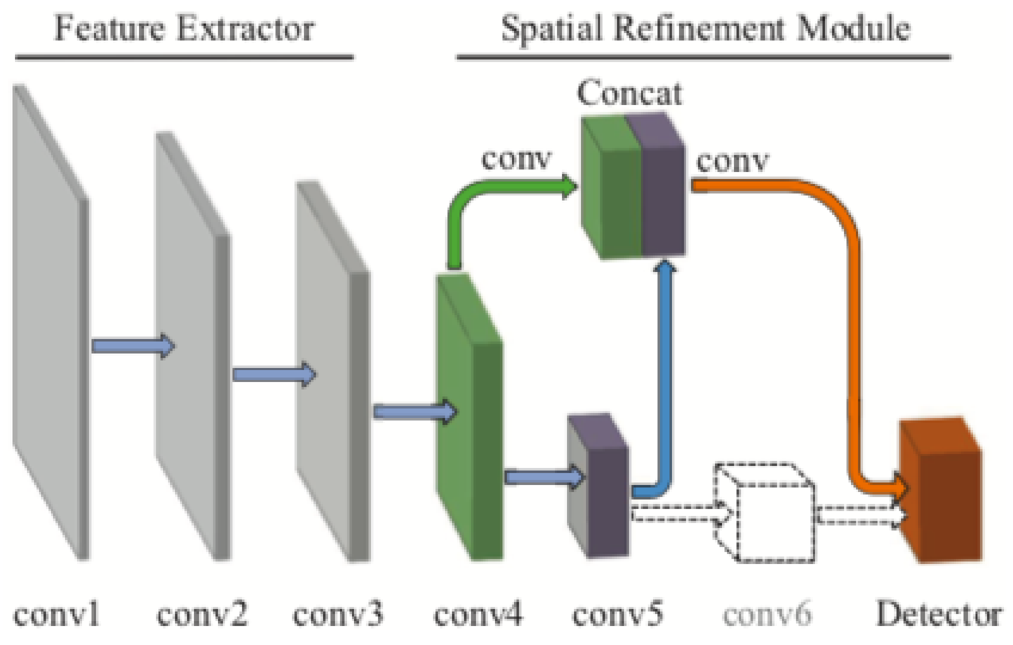


Figure 3.1

To improve the ability of gesture recognition for YOLOv2, we propose a spatial refinement module, as shown in Figure 3.1. First, we remove the last maxpooling layer and the following conv6 to make the network's top feature map higher resolution. The larger size of top feature maps always corresponds to more information and more anchors for gestures, and both of them can improve the performance. We also cut the channels of the convolutional layers in the detector by half, in order to adapt to the channels of the top feature maps.

Second, we replace the passthrough layer with a down- sampling convolutional layer to obtain better fine-grained feature maps. The original passthrough layer directly stack the feature maps from conv5 on the conv6 to form expanded feature maps. But it makes the shallower features which have more parameters domain the high-level features, which makes the model not robust. In this paper, inspired by the downsampling building block in ResNet, we downsample the feature maps in conv4 by a 1 × 1 convolutional layer that have a stride of 2. And finally we concatenate the fine-grained features with the feature maps in conv5, which provide the access for the detector to the abundant context-information.

3.1.5 YOLOv3 :

YOLOv3 is the newest version of the YOLO algorithm for real time detection. Here, YOLOv3 has been used to detect traffic participants in real-time.

YOLO v3 algorithm consists of fully Convolutional Neural Network (CNN) and an algorithm for post-processing outputs from neural network. CNNs are special architecture of neural networks suitable for processing grid-like data topology. On Figure 4.1 is presented the overview of YOLO v3 algorithm.

This algorithm starts with extraction single image from video stream, in a next step extracted image is resized to 416x416 and that represent input to Yolo network.

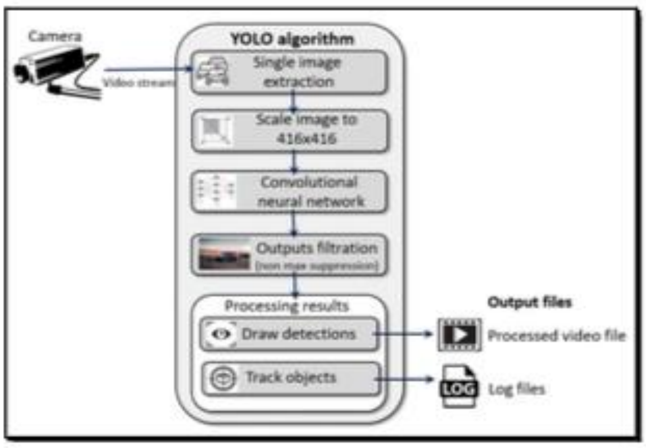


Figure 4.1

As it is shown on Figure 4.2, YOLO v3 neural network consist of 106 layers. Besides using convolutional layers, its architecture also contains residual layers, upsampling layers, and skip (shortcut) connections.

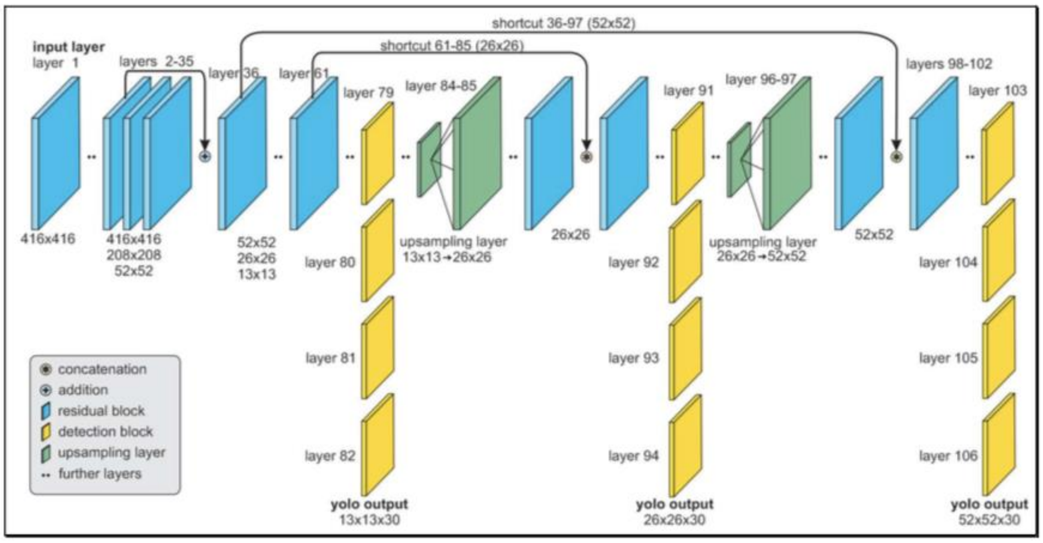


Figure 4.2

CNN takes an image as an input and returns tensor which represents:

- Coordinates and positions of predicted bounding boxes which should contain objects,
- A probability that each bounding box contains object,
- Probabilities that each object inside its bounding box belongs to a specific class.

The detection is done on the three separate layers, whose input dimensions (width and height) are 13x13, 26x26 and 52x52. Object detection done at 3 different scales addresses [9] the issue of older YOLO neural network architectures, the detection of the small objects. Output tensors from those detection layers have the same widths and heights as their inputs, but depth is defined as:

$$\text{depth} = (4 + 1 + \text{class probabilities}) \times 3,$$

where 4 is the number of bounding box properties such as width (b_w), height (b_h), x and y position of the box (b_x, b_y) inside the image, 1 is the probability that box contains the detectable object (p_c) and class probabilities for each of the classes (c_1, c_2, ..., c_5) as shown in Figure 4.3.

That sum is multiplied by 3, because each of the cells inside the grid can predict 3 bounding boxes. As the output from the network, we get 10 647 bounding box predictions.

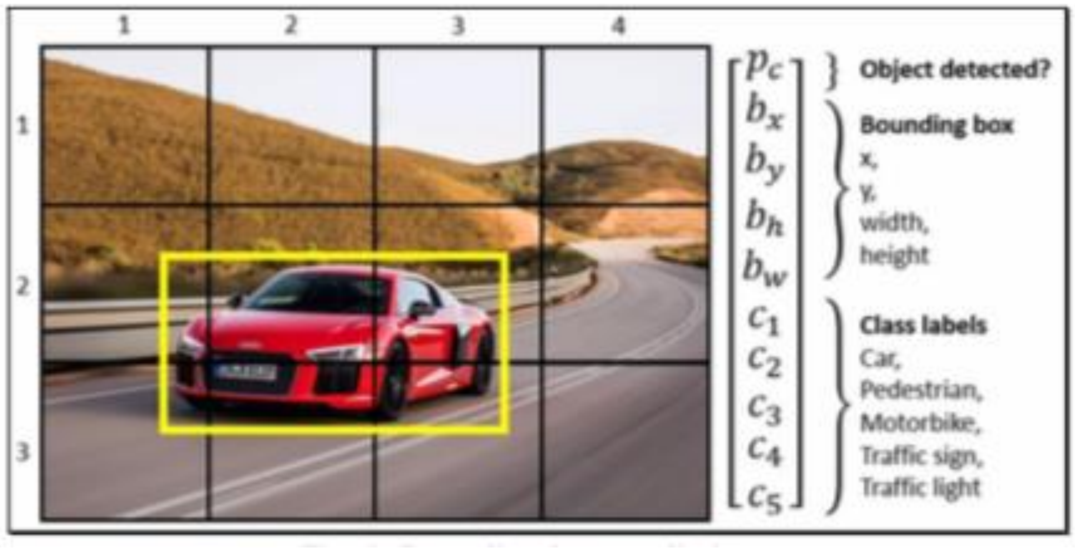


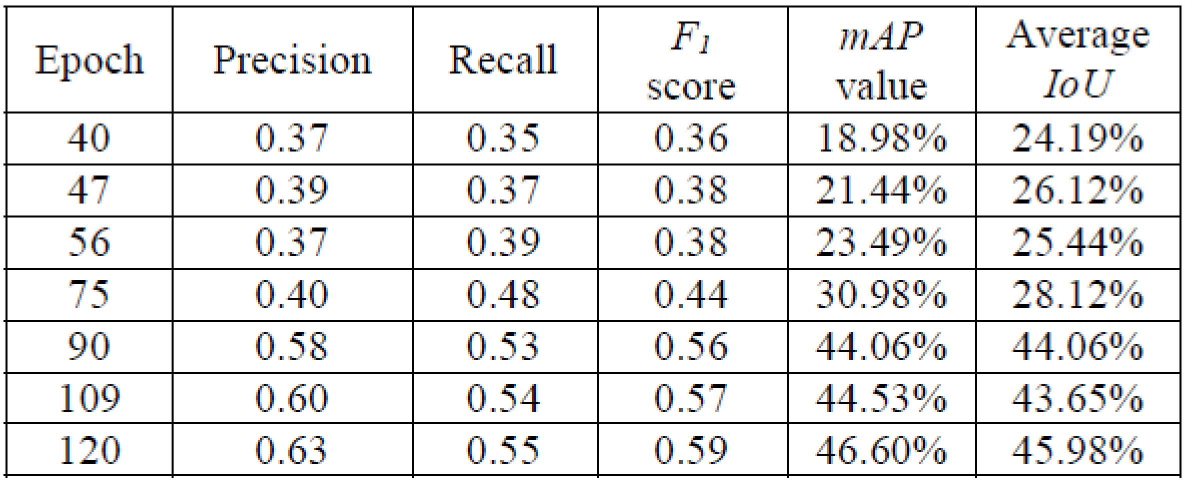
Figure 4.3

This network has an ability to simultaneously detect multiple objects on the single input image. Features are learned during the network training process when the network analyses the whole input image and does the predictions. In that way, the network has knowledge about the whole scenery and objects environment, which helps the network to perform better and achieve higher precision results comparing to the methods which use the sliding window approach. The concept of breaking down the images to grid cells is unique in YOLO, as compared to other object detection solutions. The input image is divided into an S x S grid of cells where each grid cell can predict 3 bounding boxes.

Predictions whose p_C is lower than 0.5 are ignored and that way, most of the false predictions are filtered out. Remaining bounding boxes are usually prediction of the same object inside the image.

CHAPTER 4
RESULTS AND ANALYSIS

The neural network was trained for 120 epochs and that took two weeks on the previously specified hardware. Key values that were regularly checked during the training were: the loss function value, precision, recall, mAP and average IoU. As seen from the Figure 5.1, all of those values were improving until the 120th epoch, when the model started overfitting and then the training was stopped.



Epoch	Precision	Recall	F_1 score	*mAP* value	Average *IoU*
40	0.37	0.35	0.36	18.98%	24.19%
47	0.39	0.37	0.38	21.44%	26.12%
56	0.37	0.39	0.38	23.49%	25.44%
75	0.40	0.48	0.44	30.98%	28.12%
90	0.58	0.53	0.56	44.06%	44.06%
109	0.60	0.54	0.57	44.53%	43.65%
120	0.63	0.55	0.59	46.60%	45.98%

Figure 5.1

Because of the too many small and occluded, but labeled objects in the dataset, precision and recall values were not reaching high values as expected. Common false detections found in the outputs of the neural network were further investigated on the smaller custom dataset and video. Most of the undetected objects were in heavy traffic situations where one cell in the detection grid would be responsible for detecting more than three objects. In the remaining cases, some of the objects were undetected because they were occluded by other detected objects. However, in both previous cases, all of the closest objects to the camera's position (vehicle) were successfully detected and classified as shown in Figure 5.2 and Figure 5.3.

Because of this accuracy, the safety of vehicles and passengers will not be put in danger at any moment if using this algorithm for the detection of traffic participants.

The YOLO algorithm processed input video stream from the dashboard camera with the frame resolution 1920 x 1080 px at the average of 23 frames per second using previously described hardware. That confirmed the statement that YOLO algorithm can be used for the real time video processing.

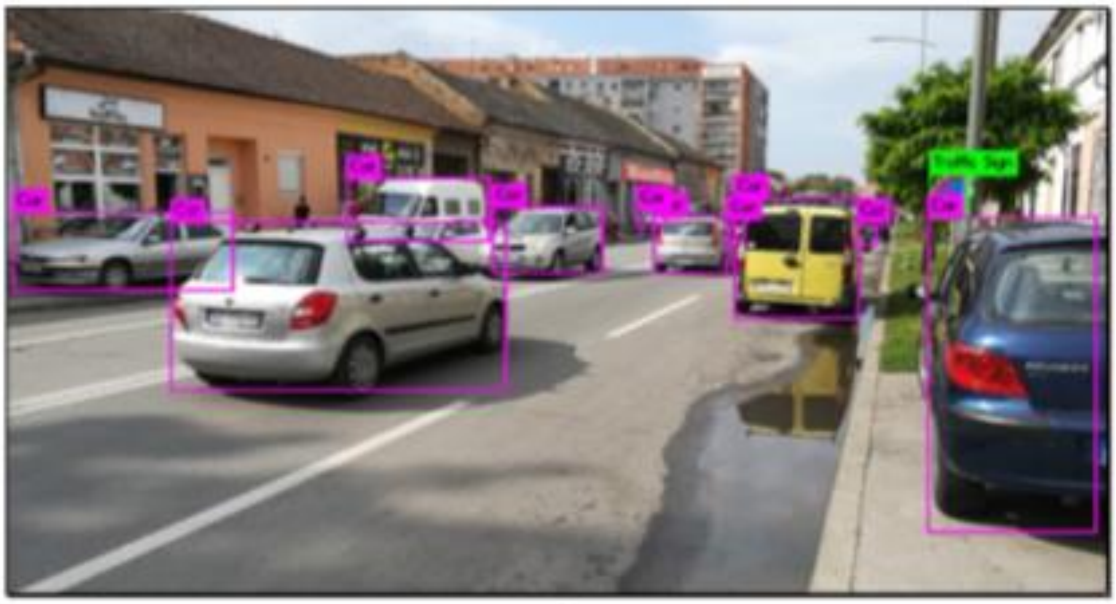


Figure 5.2

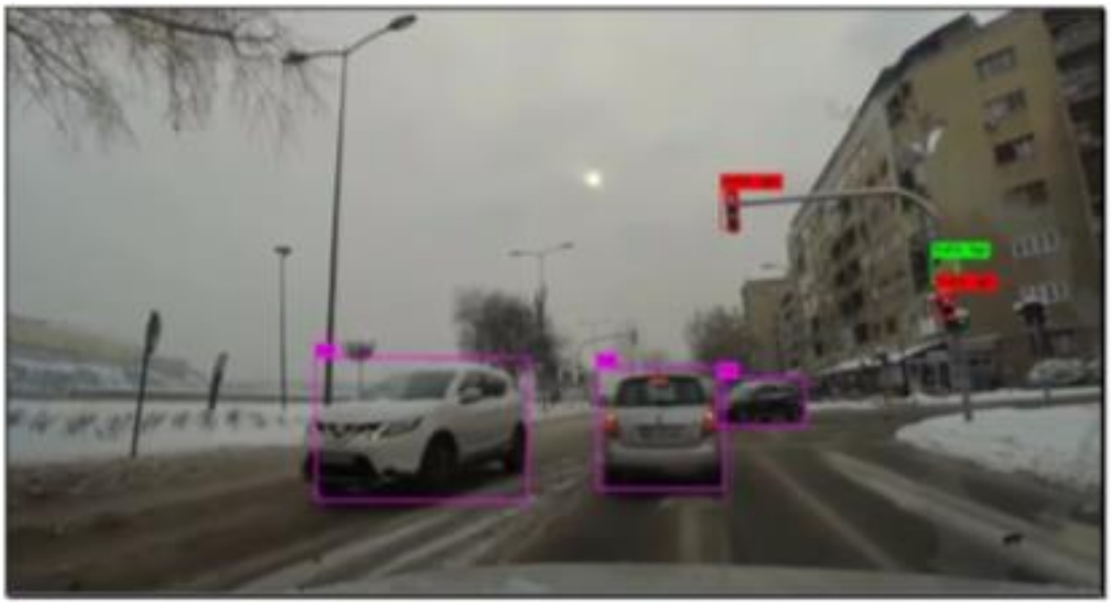
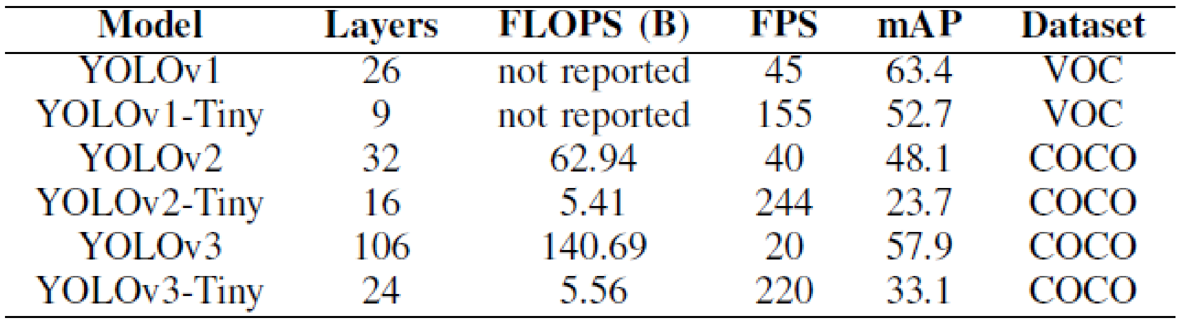


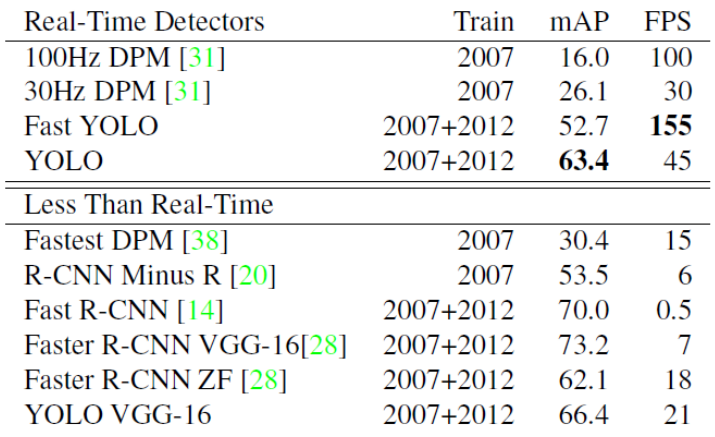
Figure 5.3



Model	Layers	FLOPS (B)	FPS	mAP	Dataset
YOLOv1	26	not reported	45	63.4	VOC
YOLOv1-Tiny	9	not reported	155	52.7	VOC
YOLOv2	32	62.94	40	48.1	COCO
YOLOv2-Tiny	16	5.41	244	23.7	COCO
YOLOv3	106	140.69	20	57.9	COCO
YOLOv3-Tiny	24	5.56	220	33.1	COCO

Figure 5.4

The performance of each version of YOLO is shown in Figure 5.4.



Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Figure 5.5

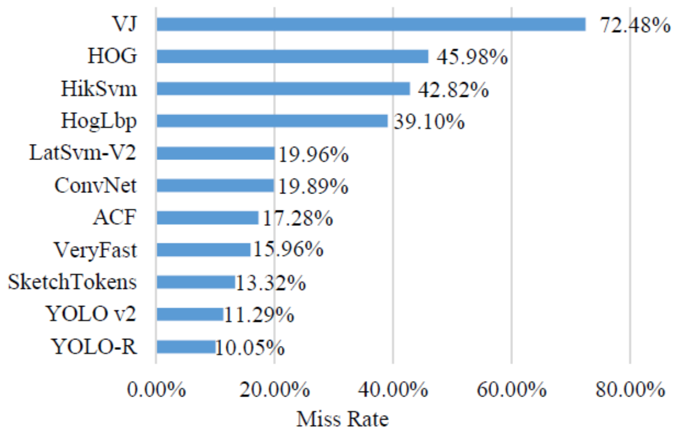


Figure 5.6

The performance and miss rate comparison of YOLO with other object detection algorithms is shown in Figure 5.5 and Figure 5.6.

CHAPTER 5
CONCLUSION

An application of object detection using YOLOv3 algorithm for real – time detection of traffic participants has been presented. The weights of the neural network were initialized using a pertained model trained on the COCO dataset. The neural network was further trained on the Berkley Deep Drive dataset to specialize in the detection of five classes of traffic participants. False detections were investigated on the custom dataset made from images representing different traffic situations in Novi Sad.

The main benefit of the solution proposed is specializing YOLO neural network for the real-time detection and tracking of the multiple classes of traffic participants. The application of YOLO algorithm presented provides a solid base for the object detection module for different applications by modifying the original YOLO algorithm to the required needs.

CHAPTER 6
SCOPE OF FUTURE WORK

- Precision of the algorithm could be improved with training on the bigger and more diverse datasets that cover different weather and lighting conditions.
- The YOLO algorithm could be used in fusion with other sensor's readings to reduce the number of false detections.
- YOLO can be combined with other object detection algorithms such as Region based Convolutional Neural Networks (R-CNN), Deep Learning Neural Networks (DNN) to benefit the accuracy and precision of these algorithms and fast and quick detection of YOLO.
- Look ahead obstacle detection for self – driving / autonomous vehicles.