Avrupa Bilim ve Teknoloji Dergisi
Sayı 33, S. 40-54, Ocak 2022

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 33, pp. 40-54, January 2022

Research Article

Moving Object Detection in Video under Different Weather Conditions Using YOLO and Faster R-CNN Algorithms

Abdulghani Mawlood A.ghani Abdulghani[1], Gonca Gokce Menekse Dalveren[2*]

[1] Atılım University, Graduate School of Natural and Applied Sciences, Ankara, Turkey, (ORCID: 0000-0002-5642-0245), abdulghani.abdulghanimawloodaghani@student.atilim.edu.tr

[2*] Atılım University, Faculty of Engineering, Department of Software Engineering, Ankara, Turkey, (ORCID: 0000-0002-8649-1909), gonca.menekse@atilim.edu.tr

(First received 21 October 2021 and in final form 23 January 2022)

(**DOI:** 10.31590/ejosat.1013049)

ATIF/REFERENCE: Abdulghani, A. M. A., Menekse Dalveren, G. G. (2022). Moving Object Detection in Video under Different Weather Conditions Using YOLO and Faster R-CNN Algorithms. *European Journal of Science and Technology*, (33), 40-54.

Abstract

Today, computer vision technology has been continuing to develop in many different areas such as object detection from video, motion tracking and object identification. It is difficult to detect moving objects such as pedestrians, cars, buses, and motorcycles in bad weather conditions or at night, especially in adverse weather conditions such as rain, fog, and snow. In this study, two types of object detection algorithms were utilized to identify objects under different conditions. These types can be identifed as one-step object detection algorithms and the two-step object detection algorithms. In this study, from the one-step algorithms YOLOv3 and YOLOv4, and from the two-step object detection algorithms Faster R-CNN were preferred. These algorithms were trained to identify four objects (pedestrians, car, bus, motorcycle) in bad weather and low light conditions. Then, comparisons of these three algorithms in all conditions was performed. By comparing YOLO versions (v3 and v4), it was aimed to observe the performance differences between these versions. By comparing YOLO and Faster R-CNN algorithms, the differences between the types one-step and two-step algorithms were evaluated. The algorithms were trained with open-image datasets. According to the results, YOLOv4 had the highest performance at 40,000 iterations, 72% mAP, and 63% recall. YOLOv3 has achieved the best result at 36,000 iterations, 65.53% mAP, and 54% recall, Faster R-CNN has achieved the best result at 36,000 iterations, 51% mAP, and 49% recall. To conclude, YOLOv4 performed the best compared to YOLOv3 and Faster R-CNN.

Keywords: Object detection, YOLOv4, YOLOv3, Faster R-CNN

YOLO ve Faster R-CNN Algoritmalarını Kullanarak Farklı Hava Koşullarında Videoda Hareketli Nesne Algılama

Öz

Günümüzde bilgisayarlı görü teknolojisi videodan nesne algılama, hareket izleme ve nesne tanımlama gibi birçok farklı alanda gelişmeye devam etmektedir. Kötü hava koşullarında veya geceleri, özellikle yağmur, sis, kar gibi olumsuz hava koşullarında yayalar, arabalar, otobüsler, motosikletler gibi hareketli nesneleri tespit etmek zordur. Bu çalışmada, nesneleri farklı koşullar altında tanımlamak için iki tür nesne algılama algoritması kullanılmıştır. Bu türler, tek adımlı nesne algılama algoritmaları ve iki adımlı nesne algılama algoritmaları olarak tanımlanabilir. Bu çalışmada tek adımlı algoritmalardan YOLOv3 ve YOLOv4 ve iki adımlı nesne algılama algoritmalarından Faster R-CNN tercih edilmiştir. Bu algoritmalar, kötü hava ve düşük ışık koşullarında dört nesneyi (kişi, araba, otobüs, motosiklet) tanımlamak için eğitilmiştir. Daha sonra bu üç algoritmanın her koşulda karşılaştırılması yapılmıştır. YOLO sürümleri (v3 ve v4) karşılaştırılarak bu sürümler arasındaki performans farklılıklarının gözlemlenmesi amaçlanmıştır. YOLO ve Faster R-CNN algoritmaları karşılaştırılarak tek adımlı ve iki adımlı algoritma türleri arasındaki farklar değerlendirilmiştir. Algoritmalar, açık görüntü veri kümeleri ile eğitilmiştir. Sonuçlara göre, YOLOv4, 40,000 yineleme, %72 mAP ve %63 geri çağırma ile en iyi performansa sahip olmuştur. YOLOv3, 36,000 yineleme, %65.53 mAP ve %54 geri çağırma ile en iyi sonucu elde etmiş ve Faster R-CNN, 36,000 yineleme, %51 mAP ve %49 geri çağırma ile en iyi sonucu elde etmiştir. Sonuç olarak, YOLOv4, YOLOv3 ve Faster R-CNN'ye kıyasla en iyi performansı göstermiştir.

Anahtar Kelimeler: Nesne Tespiti, YOLOv4, YOLOv3, Faster R-CNN

[*] Corresponding Author: gonca.menekse@atilim.edu.tr

1. Introduction

Human brain, which is the most sophisticated and accurate among living things, gets the right data to identify, recognize, and link things together when we look at an image or footage and try to find objects inside. During the day, it also operates in the same way, and it detects objects around us. On the other hand, the ability of computers to detect and recognize objects in computer vision is facilitated by Artificial Intelligence (AI). Object detection consists of classification and localization. Convolutional Neural Network (CNN) is an AI deep learning technology that can make detection more accurate and instantaneous. Today, computer vision technology has been developed and continues to develop in many different areas such as object detection from video, motion tracking and object identification (Havuç et al., 2021). In computer vision, detecting a moving object and tracking motion is very important. For this purpose, different algorithms have been developed (Havuç et al., 2021).

Object detection algorithms are generally either one-step or two-step algorithms. Two activities are involved as well: object classification, in which different colors of objects are classified; and object localization, in which objects are located by drawing a bounding box around the detected object. It is possible to use classifiers, or feature extractors like Darknet (Joseph Redmon & Farhadi, 2017), VGG-16 (Simonyan & Zisserman, 2015) and others to denote classification and localization. There are two types of algorithms for object detection: the first one is the Region-based Convolutional Neural Network (R-CNN) (Girshick, 2015; Girshick et al., 2014; Ren et al., 2017) which comprises of two-step detection algorithms, and the second one is one-step detection algorithm like, Single-Shot Detection (SSD) algorithm and You Only Look Once (YOLO) algorithm.

In two-step detection, the red box on the left of Figure 1 indicates the object, which requires an algorithm that identifies the boundaries first and then classes each boundary separately (Figure 1). The Fully Connected (FC) layers designated by the two red boxes on the right, can be named as Dens layer (Huang et al., 2017). The regression process used to identify and mark the position of the object is represented by red box no.1. The soft-max process, represented by red box no.2, is used to predict the name of an object, and processes are separated from one another in different layers.

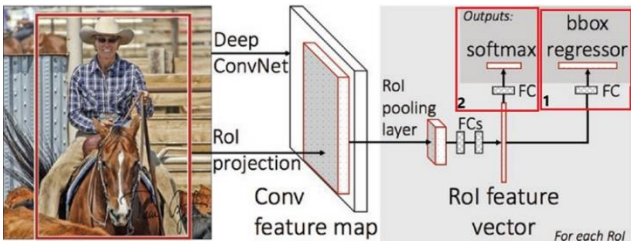

*Figure **Hata! Belgede belirtilen stilde metne rastlanmadı.**. The Fast R-CNN Architecture* (Ren et al., 2017)

Since the final output has two layers, such algorithms are named two-step algorithms. These layers, however, take more time to extract the bounding box and class names and place them on the photo or video. As a result, algorithms such as R-CNN (Simonyan & Zisserman, 2015), Fast R-CNN (Girshick, 2015), and Faster R-CNN (Ren et al., 2017) are not recommended for real-time detection.

One-step detection algorithms such as YOLO (Joseph Redmon et al., 2016; Joseph Redmon & Farhadi, 2017, 2018), and SSD (Ning et al., 2017), on the other hand, have only one tensor output. The one-step object detection, as opposed to the two-step detection, has a single layer output. Both classification and prediction are performed at the same layer, and the output would resemble [P, bx, by, bh, bw, c1, c2, c3.... cn] (Figure 2)(Corovic et al., 2018). This is referred to as the output tensor of YOLO. The denotations (bx, by, bh, bw) depict the bounding box in four dimensions, and (c1- cn) represent the classes. If the network detected a car for the first time, it would denote it as c1. The second car would be designated as c2, and the process would be repeated for the remaining cn. Essentially, one-step detection collects all of the output results in a single layer, where the bounding box and class prediction occur concurrently. As a result, one-step object detection is faster than two-step object detection. Figure 3 depicts the YOLO architecture's one-step object detection (Joseph Redmon & Farhadi, 2017), in which the output layer is gathered in a single location and shows how the YOLO algorithm's output layer looks when compared to the Fast R-CNN's layers, which have two output layers as shown in Figure 1.

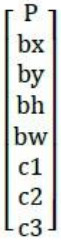

Figure 2. YOLO Output Tensor (Corovic et al., 2018)

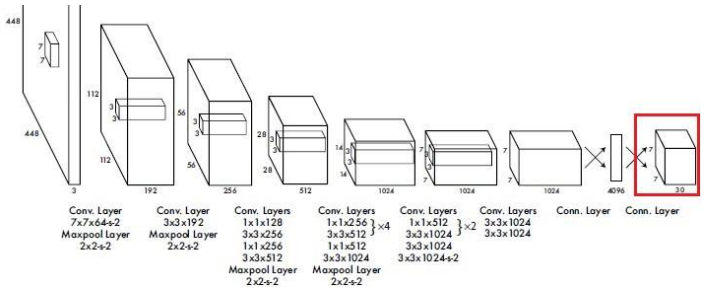

Figure 3. YOLO Architecture (Joseph Redmon & Farhadi, 2017)

Object detection technology has failed to perform well in bad weather conditions, particularly in snowy and foggy weather (Corovic et al., 2018; Ren et al., 2017). This is primarily due to the obstruction caused by snowflakes, fogs, or even darkness, which may result in a decrease in detection accuracy. The main problem is the failure to detect objects like person, car, bus, motorcycle in bad weather (rainy, foggy, snowy) and low light conditions (night) and the goal of this study is to detect various objects in different conditions and provide comparision results for the algorithms YOLOv3, YOLOv4 and Faster R-CNN. For this purpose, one-step algorithms (YOLOv3 and YOLOv4) and a two-step algorithm (Faster R-CNN) was trained to identify four objects [person, car, bus, motorcycle] in bad weather (rainy, foggy, snowy) and low light (at night) conditions. Then, comparisons of these three algorithms in all conditions was performed. By comparing YOLO versions v3 and v4, it is aimed to observe how different results the new version produces from the old version. Also, the performance of the new version in

detecting objects in bad weather conditions compared to the previous version. By comparing YOLO and Faster R-CNN algorithms, the performance differences between one-step and two-step algorithms were evaluated.

The rest of the paper is organized as follows. In section 2 Background of the Study is given. Section 3 consists of the Material and Method. Results of the study were given in Section 4. A Discussion was provided in Section 5 and the Limitations and Future Work Studies mentioned in Section 6.

2. Background of the Study

2.1. Two-Step Object Detection Algorithms

R-CNN is an earlier object detection algorithm developed by Ross Girshick and published in 2014 (Girshick et al., 2014). The name R-CNN arose from the fact that it combines region proposals with the CNN network. It includes three models: the first generates region proposals, the second takes these regions as input and applies a large CNN network to extract fixed-length features as a vector for each region, and the third is the Support Vector Machine (SVM). Figure 4 shows how the R-CNN algorithm works. It starts with the input images and then extracts 2000 regions to calculate the features for each region using a massive CNN. Following that, SVM is used to classify these regions.

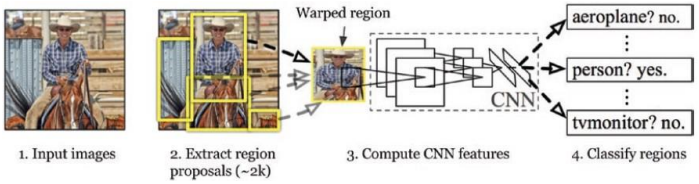

Figure 1. How the R-CNN Algorithm Work (Girshick et al., 2014)

R-CNN was the first algorithm for object detection, it had several issues with processing time delay. It was, however, very accurate. The R-CNN extract nearly 2000 region proposals from each image, each using 2000 convolutional neural networks, which is causing delays.

Fast R-CNN is an improved version of the R-CNN algorithm thand it was created by Ross Grishick and published in 2015 (Girshick, 2015). This model outperforms the previous version (He et al., 2015). It employs VGG-16 (Simonyan & Zisserman, 2015) for classification, which improves the model's accuracy. The Fast R-CNN processes all of the data (image) at once, whereas the R-CNN divides the image into many regions before processing it.

Faster R-CNN is an improved version of the Fast R-CNN algorithm, which was developed in 2016 by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun (Ren et al., 2017). They increased training speed and detection accuracy by combining the Fast R-CNN and Region Proposal Network (RPN) into a single network and using shared convolutional RPN. It employs the VGG-16 (Simonyan & Zisserman, 2015) detection system, but the region proposal model remains unchanged from the Fast R-CNN, and the detection rate capacity of this version can reach up to 5 FPS. RPN shares convolutional layers at test time rather than generating region proposals again. This enables it to use the output region to create a new region. Furthermore, the use of anchor boxes with varying scales allows the algorithm to detect objects over a wide range, which was not possible in the previous version (Huang et al., 2017). This algorithm's enhancements have proven to be a cost-effective and efficient solution for improving object detection capability (Ren et al., 2017).

2.2. One-Step Object Detection Algorithms

The two-step detection method can be more accurate than the one-step detection method. The speed that one-step detection provides, on the other hand, will make it more appealing. Because it lacked the necessary speed, the Faster R-CNN was unable to work with real-time detection and videos. For the time being, one-step detection, such as the SSD and YOLO algorithms, is the most recommended detection method, particularly for real-time detection. The one-step detection has only one output layer, which is sometimes referred to as a tensor. SSD is a simple method for training and combining predictions from a doubled features map with multiple resolutions to control objects of varying sizes. The network generates a score for each object in each box; SSD combines all computation into a single network. The SSD accuracy is 74.3 % mAP at 59 FPS for input 300x300 on pascal VOC and nearly 80 % mAP for 512x512 (Ning et al., 2017). It surpasses the Faster R-CNN (Ren et al., 2017), and it uses data augmentation to enhance the accuracy. The Faster R-CNN improves the algorithm by combining the RPN with shared convolutional layers, whereas the SSD combines the RPN and the Faster R-CNN to generate a network that is less complicated, easier to train, more precise, and efficient.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi created YOLOv1 (Joseph Redmon et al., 2016) in 2016. YOLO is the fastest algorithm for real-time detection and video analysis. It can detect multiple objects at the same time and can run at 45 fps (Joseph Redmon et al., 2016). YOLO is made up of a single CNN that predicts multiple bounding boxes as well as the class likelihood for these boxes as a whole. YOLO has several advantages, such as speed; all needed is to run the single network on an image to predict and detect. During training and testing it sees the entire image. When compared to the Fast R-CNN, it produces less than half amount of the background errors. Also, YOLO learns an overall representation of objects that it can apply to new contexts or unexpected input.

A different team (Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao) created YOLOv4 in 2020 (Bochkovskiy et al., 2020). Previously, the Redmond team developed YOLO from version 1 to version 3, and now a new team has designed YOLOv4 (Bochkovskiy et al., 2020). YOLOv4 has a distinctly new structure than YOLOv3 and incorporates YOLOv3 into its structure. It also employs an improved version of the Darknet-53 (CSP-Darknet-53) (Bochkovskiy et al., 2020). YOLOv4 employs new features such as MOSAIC (Bochkovskiy et al., 2020) data augmentation, CIoU loss (Zheng et al., 2020), DIOU NMS (Zheng et al., 2020), SPP-Net (Wang et al., 2020), CSP-Net (Huang et al., 2017), Dens-Net (Huang et al., 2017), Greedy NMS (Hosang et al., 2017), and is capable of achieving 65.7% AP50 with 65 fps in real-time detection (Bochkovskiy et al., 2020). On COCO datasets, YOLOv4 outperforms the other algorithms with AP greater than 43, as shown in Figure 5 (He et al., 2015). Every new version of YOLO improves on speed because the main goal of YOLO is speed with good accuracy.

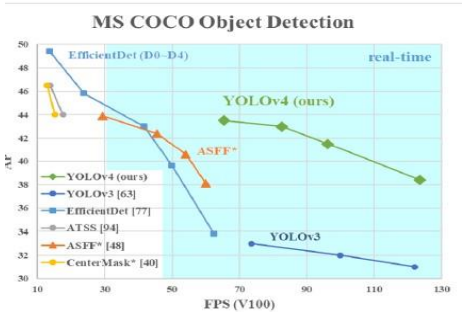

Figure 5. Comparison YOLOv4 with YOLOv3 and Other Algorithms (He et al., 2015)

3. Material and Method

In this study, a comparison was made to better understand the differences between two-step and one-step algorithms and the performance difference between the versions of YOLO algorithm. Because of YOLOv4 produces faster and more accurate results, and it can work at speeds of up to 30 fps. Faster R-CNN was chosen from the two-step algorithms because it is the most recent and improved algorithm in its segment. The open images dataset (Krasin, 2017) was chosen which contains person, car, bus, and motorcycle objects. The reason behind the selection of this dataset is; it contains 16 M bounding boxes for over 600 objects classes, and 1.9M images. It contains labels for many classes and organized into three datasets: a training dataset with 9,011,219 images, a validation dataset with 41,620 images, and a test dataset with 125,436 images. The OIDv4 (Vittorio, 2018) tool was used to download the dataset, and then it was converted to YOLO-Annotation (text file). The dataset of over 10,000 images with 1024 x 768 resolution (the training dataset only) was used. Figure 6 depicts the number of labels for each object and for testing 2825 images was used.

Figure 6. Number of Labels For Every Object in the Training Dataset

3.1. The Structure of YOLOv4

In this study, YOLOv4-Darknet-53 was used and there were two phases: the first was the Training Phase, and the second was the Detection Phase. There were five steps to completing the Training Phase of YOLOv4:

1. Feed the Dataset: The Darknet-53 classifier was used to train YOLOv4; dataset was fed with over 10,000 images labeled [person, car, bus, and motorcycle].
2. Begin the Training: The Darknet-53 was used to train this system, which resizes images to 512 pixels. Then, as shown in Table 1, data augmentation for images were generated. The transfer learning technique was used to augment the data used for YOLOv4 training. A pre-trained model (coco model) was downloaded, which was a very strong model trained with the coco dataset, and it was then used to perform transfer learning for YOLOv4. Tesla P4 as a GPU for training was used to speed up the training.

Table 1. Data Augmentation Used in YOLOv4 and YOLOv3

Data Augmentation	YOLOv4
MOSIAC	Yes
Random Flag	Yes
Photometric	Yes

   - MOSIAC: This type of data augmentation improves the model's accuracy.
   - Random flag: Resizes the input images every 10 iterations, giving the model the ability to detect different sizes of objects while also improving the mAP.
   - Photometric: This is another type of data augmentation that alters the photo's content (saturation, exposure, and hue) to provide the model with more examples from the dataset.
3. Calculate mAP: Every 1000 iterations the Darknet computed the Mean Average Precision (mAP). The training was stopped at 50,000 iterations based on the mAP, Recall, and F1-Score values. YOLOv4 produced the best results after 40,000 iterations.
4. Begin the prediction with the (IoU): The dataset was labeled while the training was running (labeling is drawing bounding boxes around the object that wanted to be detected). These were the truth boxes, as the labels indicate. When training started, the network generated predicted bounding boxes, also known as default boxes, and compared them to ground truth boxes. The default IOU threshold value was 0.5, and if the intersection over Union (IOU) was greater than 0.5, an overlap occurs, resulting in accurate detection. Otherwise, if the value is less than (0.5), good detection is not achieved.
5. Extract the prediction: After the training process was completed, a model was ready which contains all of the predictions for the objects to be detected.

In the Detection Phase, the trained model was used to detect objects in videos or photos. Four steps were listed below for the Detection Phase of YOLOv4:

1. Load the Previously Trained Model: The trained model was loaded into the darknet, and the algorithm's first layer resized the inputted frame to 512 x 512. The algorithm then divided the frame into 512 x 512 grid cells, as shown in Figure 7.

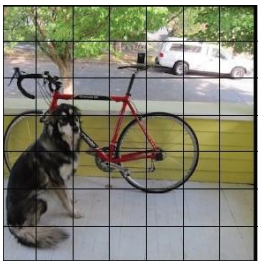

Figure 7. YOLOv4 Divide the Input into S X S Grid Cells (Joseph Redmon et al., 2016)

2. Search for Objects: If any grid cell was the center of an object, that cell was in charge of detecting that object, and each cell generated a confidence

probability and the label coordinates (bx, by, bh, bw) for each object with the class name.

3. Use NMS: Normally, there were good and bad detections, but only the good ones needed to be filtered. The None Maximum Suppression (NMS) method was used for this pupose. YOLOv4 used the Greedy NMS, which was used to eliminate bad and duplicated object detections. The Greedy NMS was the final layer of YOLOv4 and responsible for removing all bad and duplicated detections. Figure 8 depicts how the Greedy NMS operates.

Figure 8. How the Greedy NMS Works (Hosang et al., 2017)

4. Detection Output: Create a tensor or complex vector containing [P, bx, by, bh, bw, c1, c2, c3.... cn], where P represents a flag, indicating that the network recognized the object. It takes a boolean value (0,1) and ignores the other values [bx, by, bh, bw, c]. If it is 0, it means 'do not care state'. [bx, by, bh, bw] represent the four coordinates for labeling the detected objects, with (bx, by) representing the object's center and (bh, bw) representing the object's height and width. Ci denotes the number of classes, and everything is linked together within this tensor (confidence probability, label coordinates and classes names). As a result, all of the previous operations' processes are grouped together within a single layer.

The model's results were the same frame that feded into it. If it is a video, there will be several frames, and the model draws the bounding boxes around the objects. Figure 9 shows examples of detection results, and Figure 10 summarizes the steps of method's two phases in a flowchart.

Figure 9. Detection Result (Output)

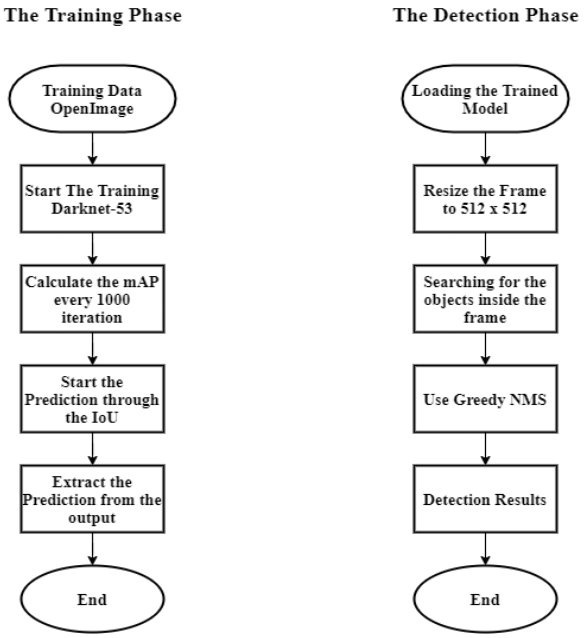

Figure 10. Flowchart For Phase One and Two of the Method

Complete Intersection Over Union Loss (CIoU-Loss) was used for YOLOv4 which is a type of loss function used for YOLOv4, and three equations for calculating the CIOU-Loss are as follows, where b and b^{gt} denote the central points of B and B^{gt} (predicted box B and target box B^{gt}), p(.) is the Euclidean distance, c is the diagonal length of the smallest enclosing box covering two boxes, α is a positive trade-off parameter, and υ mesures the consistency of aspect ratio:

$$L_{CIoU} = 1 - IoU + \frac{\rho^2 + b,b^{gt}}{c^2} + \alpha\upsilon \quad (1)$$ (Zheng et al., 2020)

$$\alpha = \frac{\upsilon}{(1-IoU)+\upsilon'} \quad (2)$$ (Zheng et al., 2020)

$$\upsilon = \frac{4}{\pi^2}\left(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h}\right)^2 \quad (3)$$ (Zheng et al., 2020)

Configurations of YOLOv4;

• Batch size: 64, which means that for each iteration of YOLOv4, 64 images will be used.

• Height (h), Width (w) = 512, the size of the input images, RGB channel = 3.

• Learning-rate = 0.001, set the learning rate.

• Maximum batch size = 50,000, with training for 50,000 iterations.

4. Results

4.1. One-step Algorithms (YOLOv4 and YOLOv3)

Training was performed inside Google-Collab using the Darknet classifier (J Redmon, 2016) for YOLOv3 and YOLOv4, and all of the results were achieved with a Tesla P4 as the GPU.

4.1.1. YOLOv4

After construction of the model, the mAP (Mean Average Precision) of the model needs to be calculated. Therefore, a test dataset was required in order to use the map function within the Darknet system to calculate the model's mAP. At the 50,000

iteration, training was completed. After training, models were tested, the best result found at the 40,000 iteration, and the mAP@50 = 0.72.12, Precision = 0.71, Recall = 0.63, and F1-Score = 0.67 (Table 2). This experiment was carried out with a dataset of 2800 images obtained from Open-Images (Krasin, 2017).

Table 2. Map Calculation for Some Specific Iteration for the YOLOV4 Model

Iteration	mAP@50%	Precision	Recall	F1-Score
20000	71.85	0.67	0.62	0.64
23000	71.11	0.72	0.60	0.66
26000	72.22	0.67	0.63	0.65
27000	71.16	0.64	0.64	0.64
31000	71.55	0.65	0.65	0.65
38000	72.07	0.68	0.64	0.66
40000	**72.12**	**0.71**	**0.63**	**0.67**
45000	70.36	0.68	0.62	0.65
46000	71.17	0.70	0.62	0.66
48000	70.52	0.73	0.60	0.66
50000	69.80	0.70	0.60	0.64

TP (True Positive) represents the good detection of the model is capable to obtain, FP (False Positive) represents the value of errors or the incorrect detection, and FN (False Negative) represents the label that the model was unable to recognize or missed. TP, FP, and FN were metrics used from the 40,000 iterations to calculate the Precision, Recall, and F1-Score.

A. TP = 6551, FP = 2678, FN = 3806

B. Precision = $\frac{TP}{TP+FP}$ = 0.71

C. Recall = $\frac{TP}{TP+FN}$ = 0.63

D. F1-Score = $2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$ = 0.67

E. mAP@0.5= $\frac{The\ summation\ for\ the\ AP\ of\ all\ objects}{Number\ of\ objects}$ = 72.12

The mAP threshold 0.5 was used to filter all detections greater than 50% for each object (Table 3). Every object [person, car, bus, and motorcycle] had the AP. This network (YOLOv4) has 162 layers and 90,259 billion floating point operations per second. The 40,000 iteration model was chosen to test the object detection. The detection was performed on a large number of videos, at various times of day and night, and under various weather conditions, such as (snowy, foggy, rainy). YOLOv4 can run at 42 fps for 30 fps videos with a resolution of 1920x1080 pixels. With a threshold of 0.5. For YOLO detection, there were four different colored labels: pink for people, blue for cars, green for buses, and yellow for motorcycles (Figure 11).

Table 3. AP for each Object at each Iteration

Iteration	mAP@50%	F1-Score%	Objects	AP%
20000	71.85	0.64	Person Car Bus M-cycle	47.87 64.15 61.12 83.92
31000	71.55	0.65	Person Car Bus M-cycle	48.49 63.30 90.51 83.92
38000	72.07	0.66	Person Car Bus M-cycle	46.40 67.96 89.69 84.24
40000	**72.12**	**0.67**	Person Car Bus M-cycle	46.79 67.09 90.97 83.64
46000	71.17	0.66	Person Car Bus M-cycle	45.06 65.70 90.42 83.49

Figure 11. Detection Result for YOLOV4 at Daylight

Figure 12 indicates the negative aspect of YOLOv4 detecting the Long truck as a Bus.

Figure 12. Wrong Detection of the Long-Track as a Bus With YOLOV4

Figure 13. Detection Results for YOLOV4 at Night

Figure 13 depicts the detection results of YOLOv4. There were some errors at night, such as large or long cars detected as buses and a traffic light detected as a person, or a car as a person. Figure 14 depicts incorrect detection results.

Figure 14. Wrong Detection and Errors of YOLOV4 at Night

Figure 15. Yolov4 Detection Results Under Heavy Rain Condition

Figure 15 depicts the detection results of YOLOv4 in rainy weather for both night and daylight. As shown in Figure 15, YOLOv4 was able to identify cars in heavy rain at night or in low light, and some detections occurred with almost no visibility.

Figure 16 shows some examples of YOLOv4 incorrect detections in heavy rain; it can be seen that the empty part of the photo was detected as a person. YOLOv4 detected the truck and identified it as a bus on the right side of the image.

Figure 16. Wrong Detection for YOLOv4 in Heavy Rain at Night

Figure 17 depicts the detection results of YOLOv4 in foggy conditions.

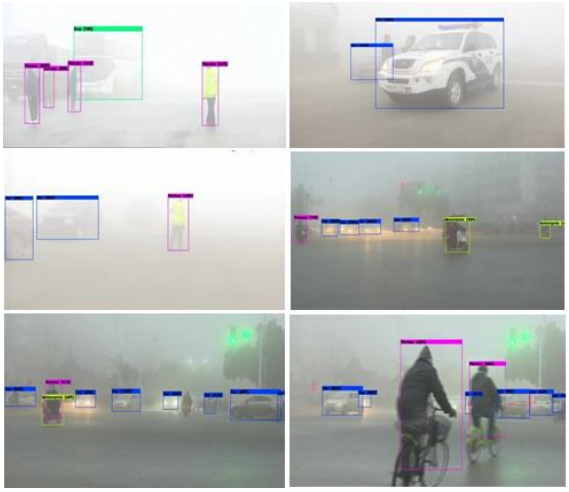

Figure 17. YOLOv4 Detection Results in Foggy Weather

There were some incorrect detections, such as identifying a part of a truck as a car and another truck as a bus. The images in Figure 18 depicted the detection errors. As shown in Figure 18, YOLOv4 identified two objects as a bus. The first object from the left (in the red box) was initially identified as a truck, but YOLOv4 changed it to a bus.

Figure 18. Wrong Detections for YOLOv4 in Foggy Weather

During a snowfall, YOLOv4 was able to detect objects. Figure 19 depicts photos of YOLOv4 detections in snowy conditions.

Figure 19. YOLOv4 Detection Result in Snow Weather

YOLOv4 has good detection in bad weather, but when there is heavy snow, YOLOv4 makes mistakes and misidentifies buses as cars.

Table 4. Detection Results for Every Object, in a Different Environment for YOLOv4

Weather conditions	Objects Names	AP%
Daylight	Person	94
	Car	100
	Bus	100
	Motorcycle	88
Night	Person	91
	Car	100
	Bus	95
	Motorcycle	85
Rainy	Person	-
	Car	95
	Bus	98
	Motorcycle	85
Foggy	Person	70
	Car	72
	Bus	97
	Motorcycle	60
Snowy	Person	79
	Car	80
	Bus	91
	Motorcycle	-

Table 4 shows the AP for each object (person, car, bus, motorcycle) in various weather conditions and throughout the day (daylight, night, rain, fog, snow). A demonstration of how different environments can make it difficult to detect objects, particularly in foggy or snowy conditions was presented. The detection results (AP %) for the mentioned objects were given in the table above for the three different weather conditions and throughout the day. The detection result of the object in the middle of the frame is represented by AP %. The higher detection results were gathered in the daylight scenario. Lower detections were discovered in the snowy and foggy weather. The two obstacles had the highest reported AP value throughout the obstacles (car and bus). The objects with the lowest AP value were the person and the motorcycle. YOLOv4 was successful in detecting all of the aforementioned objects at various times of day and in challenging weather conditions. This is yet another discovery that demonstrates the effectiveness of the YOLOv4 one-step object detection algorithm.

4.1.2. YOLOv3

YOLOv3 models were tested as the same with YOLOv4 models. The training was ended at 43,000 iterations and the mAP was calculated. Table 5 shows the results of selected models with high mAP, recall, and F1-Score. According to Table 5, the model with the 36,000 iterations produced the best results. It has the greatest Recall, F1-Score, and mAP. The same dataset (2800 images) was used for the test as previously used for testing YOLOv4.

Table 5. mAP Calculation for some Specific Iteration for the YOLOv3 Model

Iteration	mAP@50%	Precision	Recall	F1-Score
22000	65.05	0.72	0.53	0.61
25000	64.91	0.72	0.52	0.61
28000	64.95	0.69	0.55	0.61
34000	65.01	0.73	0.53	0.61
36000	**65.53**	**0.73**	**0.54**	**0.62**
39000	64.89	0.72	0.53	0.61
40000	64.73	0.76	0.50	0.61
41000	65.10	0.75	0.52	0.61
43000	64.32	0.71	0.54	0.61

TP, FP, and FN were used at 36,000 iterations to evaluate the model and calculate the Presision, Recall, and F1-Score, with the following equation:

A. TP = 5543, FP = 2021, FN = 4814

B. Precision = $\frac{TP}{TP+FP}$ = 0.73

C. Recall = $\frac{TP}{TP+FN}$ = 0.54

D. F1-Score = $2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$ = 0.62

E. mAP@0.5= $\frac{The\ summation\ for\ the\ AP\ of\ all\ objects}{Number\ of\ objects}$ = 65.52

YOLOv3 was able to work with 37 FPS, by using 30 FPS videos with 1920 x 1080 resolution. As can be seen from Table 6, YOLOv3 was able to detect each object (person, car, bus, motorcycle) with the threshold 0.5, and it can be seen that at 36,000 iterations of YOLOv3 was able to detect person with 36.79% of AP, car with 61.51%, bus with 84.77%, and motorcycle with 79.03% of AP, and YOLOv3 107 layers.

Table 6. AP for each Object at each Iteration for YOLOv3

Iteration	mAP@50%	F1-Score	Objects	AP%
25000	64.91	0.61	Person	36.90
			Car	60.72
			Bus	84.92
			M-cycle	77.11

28000	64.95	0.61	Person Car Bus M-cycle	37.86 61.14 84.71 76.08
34000	65.01	0.61	Person Car Bus M-cycle	36.66 61.70 84.29 77.38
36000	**65.53**	**0.62**	Person Car Bus M-cycle	36.79 61.51 84.77 79.03
39000	64.89	0.61	Person Car Bus M-cycle	36.52 60.80 84.75 77.84
41000	65.10	0.61	Person Car Bus M-cycle	37.60 60.23 84.14 78.42
43000	64.32	0.61	Person Car Bus M-cycle	37.10 60.81 84.21 75.18

Figure 20. Detection for YOLOv3 at Daylight

The detection results for YOLOv3 for the daylight are shown in the Figure 20.

As can be seen in Figure 21, there were some misdetections of YOLOv3. It recognizes the long truck as a bus, the bus as a car, and the car as a motorcycle, as well as the blue label for a car, the green label for a bus, the yellow label for a motorcycle, and the pink label for a person.

Figure 21. Wrong Detection of YOLOv3

Figure 22. Detection of YOLOv3 at Night with Dash Cam

Figure 22 depicts the detection results for YOLOv3 at night; this video was recorded using a dash-cam for a car; YOLOv3 detects only cars and buses. It could not detect person or motorcycles. Also, in some scenarios, it detected one or two objects per frame.

In Figure 23, another scenario with a fixed camera was shown, and the detection of YOLOv3 was poor; it could only detect one or two cars per frame, and there was no detection of the bus.

Figure 23. Bad detection of YOLOv3 at Night

As shown in Figure 24, YOLOv3 made incorrect detections by detecting a car as a motorcycle in the right photo (with yellow label) and detected a truck as a bus in the left photo (with green label), and YOLOv3 was unable to detect a person at night.

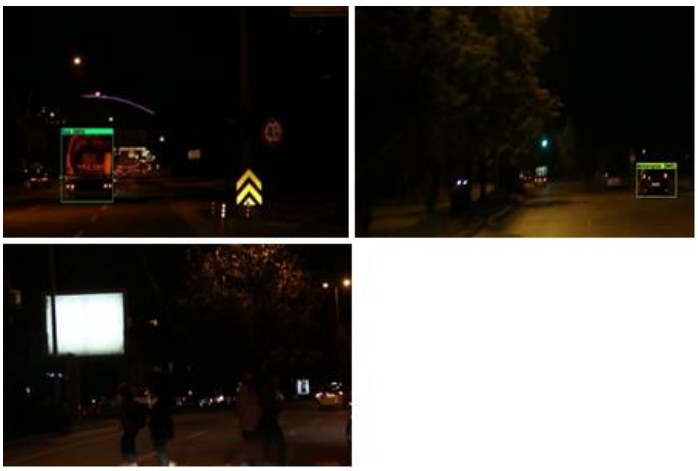

Figure 24. Wrong Detection of YOLOv3 at Night

YOLOv3 performance was not good in rainy conditions; when there was heavy rain and night, it detected nothing; however, when there was enough light, it could work properly and detected objects (Figure 25).

Figure 25. YOLOv3 Detection in Rainy Weather Condition

In scenarios such as heavy rain, YOLOv3 failed to detect anything because the raindrops obscured the vision, as shown in the bottom two images of Figure 25. In another instance, YOLOv3 detected a car and recognized it as a bus. YOLOv3 was able to detect all of the objects despite the foggy conditions (Figure 26).

Figure 26. Detection of YOLOv3 in Foggy Weather Condition

YOLOv3 also produced incorrect detection results, such as detecting the truck and misidentified it as a bus in the snow (Figure 27). YOLOv3 was unable to detect the bus or a person riding a motorcycle, and in some frames detected only one car.

Figure 27. YOLOv3 Detection in Snowy Weather Condition

Table 7. Detection Results for Every Object, in Different Environment for YOLOv3

Weather conditions	Objects Names	AP%
Daylight	Person	70
	Car	94
	Bus	70
	Motorcycle	70
Night	Person	0
	Car	96
	Bus	70
	Motorcycle	0
Rainy	Person	-
	Car	94
	Bus	85
	Motorcycle	70
Foggy	Person	70
	Car	72
	Bus	97
	Motorcycle	72
Snowy	Person	65
	Car	70
	Bus	75
	Motorcycle	-

In Table 7, there were three main columns for the detection of AP percent, which represents the detection for the object in the middle of the frame, Mini-AP percent, which represents the minimum detection for the object, and Max-AP percent, which represents the maximum detection for the object. In the daytime, YOLOv3 was able to detect all of the objects with an AP of 70% and the car with an AP of 94%, whereas at night, YOLOv3's detection was poor. It could only detect cars and buses, with a 96% success rate for cars and a 70% success rate for buses. However, it was unable to detect a person or a motorcycle at night. In rainy weather, YOLOv3 detected objects, with a 94% success rate for cars, an 85% success rate for buses, and a 70% success rate for motorcycles.

4.2. Two-step Algorithm (Faster R-CNN)

The Faster R-CNN algorithm was trained for 36,000 iterations and achieved mAP 0.51, Precision 0.5832, Recall 0.4987, and F1-Score 0.5376 as shown in Table 8.

Table 8. mAP Calculation for Some Specific Iteration for the Faster R-CNN Model

Iteration	mAP@50%	Precision%	Recall%	F1-Score%
7000	0.4185	0.4709	0.4391	0.4544
15000	0.4749	0.5401	0.4711	0.5032
19000	0.4933	0.5591	0.5991	0.5784
25000	0.4965	0.5631	0.5899	0.5761
28000	0.5040	0.5759	0.4927	0.5310
32000	0.5103	0.5809	0.4961	0.5351
36000	**0.5105**	**0.5832**	**0.4987**	**0.5376**

Table 8 shows the metrics for each iteration of the Faster R-CNN algorithm training: mAP, Precision, Recall, and F1-Score. It can be seen that the highest value of mAP and Precion was taken at iteration 36,000. Iteration 19,000 produced the highest value of Recall. These values are obtained by evaluating the existing model on a test dataset of 28,000 images. This evaluation method calculated the mAP directly from the AP of the objects and Precision and recall without calculating the model's TP, FP, and FN.

The Faster R-CNN algorithm can work with 7 - 10 FPS by using 30 FPS videos with a resolution of 1920 x 1080, and the detections were performed in different weather conditions, just like the YOLO algorithm. Table 9 shows the detection results for Faster R-CNN for each object (person, car, bus, motorcycle). The threshold was set to 0.5, and the algorithm was able to detect a person with 0.0051 of AP, a car with 0.52, a bus with 0.80, and a motorcycle with 0.71 of AP after 36,000 iterations.

Table 9. AP for each Object at each Iteration for the Faster R-CNN

Iteration	mAP@50%	F1-Score	Objects	AP%
7000	0.42	0.45	Person Car Bus M-cycle	0.0035 0.4121 0.7111 0.5470
15000	0.47	0.50	Person Car Bus M-cycle	0.0045 0.4805 0.7628 0.6517
19000	0.49	0.58	Person Car Bus M-cycle	0.0045 0.5053 0.7850 0.6785
25000	0.49	0.58	Person Car Bus M-cycle	0.0046 0.5104 0.7795 0.6917
28000	0.50	0.53	Person Car Bus M-cycle	0.0050 0.5239 0.7883 0.6987
32000	0.51	0.53	Person Car Bus M-cycle	0.0049 0.5295 0.8006 0.7064
36000	0.51	0.54	Person Car Bus M-cycle	0.0051 0.5200 0.8005 0.7171

The Faster R-CNN algorithm was able to detect all of the objects in daylight. Figure 28 depicts the Faster R-CNN algorithm detection during daylight.

Figure 28. Detection of Faster R-CNN at Daylight

Faster R-CNN made some mistakes, misidentifying parts of a car and some cars as motorcycles (Figure 29). A long truck was also detected and identified as a bus. It had trouble at detecting people and failed to detect objects from a long distance.

Figure 29. Wrong Detection for the Faster R-CNN at Daylight

The faster R-CNN could function and detect objects at night, but the recognition was not good enough in some scenarios, particularly for distant objects. Figure 30 depicts the Faster R-CNN algorithm's detection at night.

Figure 30. Detection of the Faster R-CNN at Night

Figure 30 shows the footage from a car dash camera, also there was a footage from a road monitoring camera (Figure 31). The object was not close enough in this video for the Faster R-CNN to detect it.

Figure 31. Wrong Detection of the Faster R-CNN at Night

In the rainy weather, the detection of the Faster R-CNN was poor. Figure 32 depicts such detections made by the Faster R-CNN algorithm.

Figure 32. Detection of the Faster R-CNN in Rainy weather conditions

In Figure 32, the Faster R-CNN algorithm either failed to detect objects like the cars in the images or misidentified some as motorcycles. It was unable to detect buses and motorcycles.

The Faster R-CNN algorithm was able to detect all of the objects despite the foggy weather conditions. Figure 33 depicts some images of the Faster R-CNN detection in foggy conditions.

Figure 33. Detection of the Faster R-CNN in Foggy Weather Conditions

The Faster R-CNN could be able to detect objects in snowy weather conditions, but detection was poor, and it missed some objects during the snowfall. It was able to detect a few, but they were misidentified as different objects. For example, it detected a car and recognized it as a motorcycle. It was also unable to detect objects from a long distance. Figure 34 depicts some images demonstrating the performance of the Faster R-CNN detection in snowy conditions. As can be seen in the last two images of Figure 34, the algorithm made some errors by misidentifying a car and a portion of another as motorcycles.

Figure 34. Detection of the Faster R-CNN in Snowy Weather Condition

4.3. Comparison of YOLO with Faster R-CNN

This section contains comparison results for YOLOv4, YOLOv3, and Faster R-CNN, as well as photos illustrating the differences between each version. In this study, two YOLO versions (YOLOv3 and YOLOv4), as well as the Faster R-CNN, were trained. The best result for YOLOv3 at 36,000 iterations, and the the best result for YOLOv4 at 40,000 iterations. The best output for Faster R-CNN at 36,000 iterations. It is possible to examine each one and determine which one is the best. Table 10 shows the distinction between the YOLO models and the Faster R-CNN model. The 40,000 iteration of the YOLOv4 model has mAP of 72.12, Precision of 71, Recall of 63, and F1-Score of 67. The mAP of the YOLOv3 model with 36,000 iterations is 65.53, Precision is 73, Recall is 54, and F1-Score is 62. Precision was higher in YOLOv3 than in YOLOv4, while Recall, mAP, and F1-Score were higher in YOLOv4. As a result, the detection results of YOLOv4 were superior to those of YOLOv3. However, even when compared with YOLOv3, the Faster R-CNN with 36,000 produced the worst results. YOLOv4 received a higher AP for each object than YOLOv3. YOLOv4 outperformed YOLOv3 in the TP, FP, and FN. Detection results of YOLOv3

Table 10. Difference Between the Metrics of YOLOv4, YOLOv3 Faster R-CNN Models

Alg.	Iteration	Num.of Layers	mAP%	Precision	Recall	F1-Score	Obj.	AP%
YOLOv4	40000	162	72.12	0.71	0.63	0.67	Person Car Bus M-cycle	46.79 67.09 90.97 83.64
YOLOv3	36000	107	65.53	0.73	0.54	0.62	Person Car Bus M-cycle	36.79 61.51 84.77 79.03
Faster R-CNN-Res-Net-50-TF2	36000	50	51.00	0.58	0.49	0.53	Person Car Bus M-cycle	0.01 0.52 0.80 0.71

4.3.1. Detection at night

At night, each algorithm was tested using dash-cam and static camera videos. YOLOv4 was able to detect all objects with the dashcam, whereas YOLOv3 detected only cars and buses. Faster R-CNN detection results were similar to YOLOv3. On the other hand, the Faster R-CNN and YOLOv3 underperform for the static camera. YOLOv4 detected almost all of the cars, as shown in Figure 35. As a result, YOLOv4 is suggested for night detection. Figure 36 demonstrates how YOLOv4 and Faster R-CNN were able to detect people crossing the street at night, whereas YOLOv3 could not.

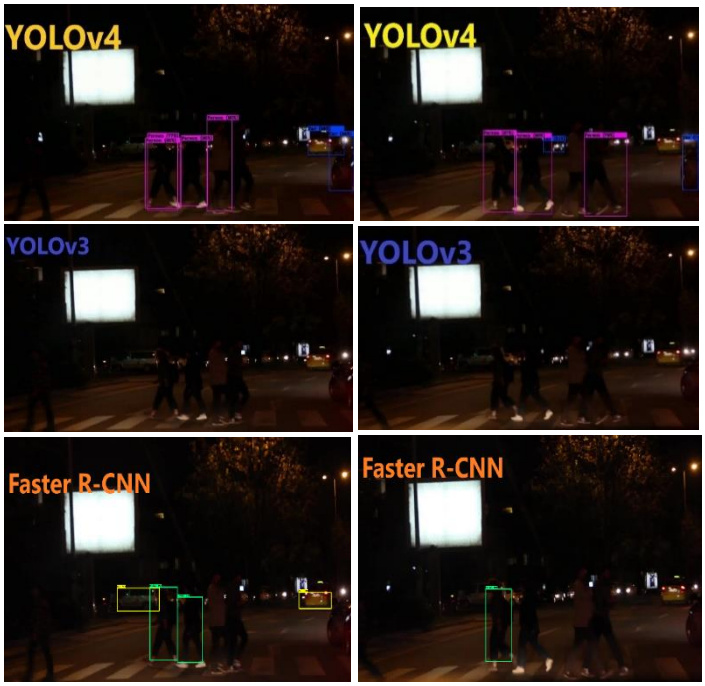

Figure 35. Detection of YOLO, and Faster R-CNN at Night

Figure 36. Person detection for YOLO and Faster R-CNN

4.3.2. Detection in rainy weather condition

Each algorithm was tested with a large number of videos in rainy conditions. However, when it was dark and raining heavily, YOLOv3 was unable to detect anything, whereas YOLOv4 was able to detect some objects. The Faster R-CNN performed poorly, as shown in Figure 37, which depicts images of the scenarios for YOLOv3, YOLOv4, and the Faster R-CNN.

Figure 37. Detection of YOLO, and Faster R-CNN Under Heavy Rain

4.3.3. Comparison of the detection results in snowy weather condition

In snowy weather, YOLOv3, YOLOv4, and Faster R-CNN detection were tested. The detection of YOLOv4 was superior to that of YOLOv3 and Faster R-CNN. YOLOv4 was able to detect objects from a long distance, even when they were covered in snow, such as cars. YOLOv4 detected more objects per frame than YOLOv3 and the detection of the Faster R-CNN was poor as well. On the other hand, YOLOv3 detection outperformed the Faster R-CNN. Figure 38 depicts detection for YOLOv4, YOLOv3, and Faster R-CNN.

Figure 38. Detection of YOLO, and Faster R-CNN in Snowy Weather

5. Discussion

This study aimed to compare three object detection algorithms YOLOv4, YOLOv3 from the one-step algorithms and Faster R-CNN from the two-step algorithms. To compare these algorithms their detection results were used. Four different objects [person, car, bus, motorcycle] tried to be detected during daylight, night and in different weather conditions. Based on the results of this study, YOLOv4 outperformed other two algorithms (YOLOv3, Faster R-CNN) in detecting most of the objects in different conditions. It had the highest AP for all four objects, and its processing time was faster than YOLOv3 and R-CNN, with a speed of 42 FPS. In terms of speed, YOLOv3 came in second with 37 FPS, indicating that it can perform real-time detections. With a speed of 7 – 10 FPS, the Faster R-CNN was the slowest. According to the findings of this study, YOLOv4 is the best choice for bad weather and night detections. This algorithm's speed adds another point to its favor, making it the best choice if speed and accuracy were required for a specific task. According to the results of this study, the performance result differences between the versions of YOLO algorithm was presented. For better understanding the one-step and two-step algorithms YOLO and Faster R-CNN was compared.

6. Conclusion and Future Work

Object detection is a critical field which makes machines to be able to recognize a wide range of objects using AI. In this study, two types of algorithms were examined: the one-step algorithms (YOLOv4, YOLOv3) and the two-step algorithm (Faster R-CNN). These algorithms were used to detect various objects [person, car, bus, motorcycle] in different conditions [day, night, rainy, foggy, snowy]. YOLOv4 was able to work and detect objects in all of the conditions mentioned except when it was dark and raining heavily. The second algorithm (YOLOv3) could not function at night or in snowy conditions. The Faster R-CNN performed the worst of the three because it could not work at night, in snowy or rainy conditions.

As a future work, improvements can be possible regarding the detection of YOLOv4 in bad weather conditions and at night. To improve the detection in bad weather conditions, a new model can be build to enhance the vision. This future prospect model would run to improve vision by removing obstructions such as raindrops on the lens, fogs, or snowflakes. It could also improve the brightness and make it suitable for use at night. By removing the obstructions, the model can perform better, and detection in poor weather and at night can be improved. A data augmentation method to generate raindrops, snowflakes, and fogs on the dataset would improve detection even more. It would provide more examples of bad weather conditions to the algorithm, allowing the model to recognize obstacles and detect more objects in bad weather conditions. The same dataset can be trained with the YOLOR algorithm, which is the most recent version of the YOLO series.

References

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv Preprint ArXiv:2004.10934*. http://arxiv.org/abs/2004.10934

Corovic, A., Ilic, V., Duric, S., Marijan, M., & Pavkovic, B. (2018). The Real-Time Detection of Traffic Participants Using YOLO Algorithm. *2018 26th Telecommunications Forum, TELFOR 2018 - Proceedings*. https://doi.org/10.1109/TELFOR.2018.8611986

Girshick, R. (2015). Fast r-cnn. *Proceedings of the IEEE International Conference on Computer Vision*, 1140–1448.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/CVPR.2014.81

Havuç, E., Alpak, Ş., Çakırel, G., & Baran, M. K. (2021). Derin Öğrenme Vasıtasıyla Masa Tenisi Topu Takibi. *European Journal of Science and Technology*, *27*, 629–635. https://doi.org/10.31590/ejosat.885795

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824

Hosang, J., Benenson, R., & Bernt, S. (2017). Learning non-maximum suppression. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6469–6477.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

Krasin, I. (2017). *OpenImages: A public dataset for large-scale multi-label and multi-class image classification*.

Ning, C., Zhou, H., Song, Y., & Tang, J. (2017). Inception Single Shot MultiBox Detector for object detection. *2017 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2017*, 549–554. https://doi.org/10.1109/ICMEW.2017.8026312

Redmon, J. (2016). *Darknet: Open Source Neural Networks in C*. http://pjreddie.com/darknet/

Redmon, Joseph, Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Redmon, Joseph, & Farhadi, A. (2017). YOLO9000: better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Redmon, Joseph, & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv Preprint ArXiv:1804.02767*. http://arxiv.org/abs/1804.02767

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Vittorio, A. (2018). *Vittorio, Angelo, "OIDv4_ToolKit, Toolkit to download... - Google Akademik*. Toolkit to Download and Visualize Single or Multiple Classes from the Huge Open Images v4 Dataset. github.com/EscVM/OIDv4_ToolKit

Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. *IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition Workshops, *2020-June*, 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU loss: Faster and better learning for bounding box regression. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, *20*, 12993–13000. https://doi.org/10.1609/aaai.v34i07.6999