# OBJECT DETECTION FROM DIOR DATASET WITH YOLOV5 AND YOLOV8

*Raissa Magana Ugarte*

r.magana-ugarte@tu-bs.de
Technical University of Braunschweig

*Karam Mawas*

k.mawas@tu-bs.de
Technical University of Braunschweig

**ABSTRACT**

This research proposal explores the use of the DIOR Dataset for advancing object detection in optical remote sensing images. We employ the YOLO framework from version 5 and version 8 models developed by Ultralytics. Our objective is to develop a robust model capable of accurately detecting objects across various classes and environmental conditions. The results from both models will be compared to observe the different performance of both models with variation of hyperparameters, optimizer functions as well as different number of epochs. The evaluation on the DIOR validation dataset includes measuring Mean Average Precision (mAP), Recall, and Precision. The outcomes contribute to significant advancements in object detection for optical remote sensing, providing insights into domain-specific challenges. Leveraging the DIOR dataset enhances visual perception and recognition capabilities, supporting practical solutions for real-world applications in environmental monitoring.

***Keywords:*** *Object Detection, Object Localization, DIOR Dataset, Deep Learning, Computer Vision*

## 1. INTRODUCTION

Object Detection is a computer vision task in which the goal is to detect and locate objects of interest in an image or video. The task involves identifying the position and boundaries of objects in an image and classifying the objects into different categories.

The state-of-the-art methods can be categorized into two main types:

- One-stage methods prioritize inference speed, models like YOLO, SSD and RetinaNet.
- Two-stage methods prioritize detection accuracy, including Faster R-CNN, Mask R-CNN.

Object detection in optical remote sensing images plays a crucial role in various applications, including environmental monitoring, urban planning, agriculture, and disaster management. The accurate and efficient detection of objects, such as buildings, vehicles, and natural features, is essential for extracting valuable information from these large-scale images.

The topic of object detection in optical remote sensing images holds significant technical and applied attractions, making it a compelling and captivating research endeavor. From a technical perspective, object detection in remote sensing images presents unique challenges and opportunities that set it apart from traditional computer vision tasks. Unlike standard RGB images, remote sensing data often comes in the form of multi-spectral or hyperspectral imagery, which captures information across various wavelengths, enabling a deeper understanding of the Earth's surface [1]. Navigating through this vast and complex data requires innovative algorithms and models capable of extracting meaningful insights from diverse spectral information.

In this project, we aim to utilize the DIOR (Detection in Optical Remote sensing) dataset [2], a comprehensive and publicly available benchmark designed explicitly for object detection in optical remote sensing imagery. The DIOR dataset offers a diverse collection of images, encompassing a wide range of geographical locations and capturing objects of varying sizes, orientations, and contextual complexities.

Through this study, we intend to advance the state-of-the-art in object detection for optical remote sensing images, contributing to the broader advancement of computer vision in the field of remote sensing. Furthermore, our findings are expected to facilitate better understanding and utilization of the DIOR dataset as a benchmark for researchers and practitioners in the remote sensing community. Ultimately, this work seeks to enhance visual perception and recognition capabilities, fostering more accurate and efficient object detection solutions for a wide array of real-world applications.

## 2. DIOR DATASET

The DIOR Dataset is a large-scale, publicly available benchmark for object Detection in Optical Remote sensing images, which named as DIOR. Its main objective is to be used to develop and validate data-driven methods to classify and detect multiple different objects from a Satellite Imagery/Optical Remote Sensing image.


**Fig. 1.** Example images taken from the proposed DIOR dataset, which were obtained with different imaging conditions, weathers, seasons, and image quality, adapted from [1].

Each image has 800×800 pixels with 3 channels each and the spatial resolutions range from 0.5m to 30m.The dataset contains 23463 images and 192472 instances, covering 20 object classes, as depicted in Fig. 1, namely: (1) Golf field, (2) Expressway-toll-station, (3) Vehicle, (4) Train station, (5) Chimney, (6) Storage tank, (7) Ship, (8) Harbor, (9) Airplane, (10) Ground track field, (11) Tennis court, (12) Dam, (13) Basketball court, (14) Expressway-Service-area, (15) Stadium, (16) Airport, (17) Baseball field, (18) Bridge, (19) Windmill, (20) Overpass.

The dataset contains: a training and validation set of images together in JPEG format (50%), a test set of images in JPEG format (50%), three text files with the image numbers for each training, validation and test set, and the .xml files for two types of annotations: Horizontal Bounding Boxes and Oriented Bounding Boxes. For this project, we will be using the Horizontal Bounding Boxes, since that is the format that the frameworks accept.

One of the challenges presented with this dataset is the imbalance of the presented classes as presented in Fig. 2. With a contrast of the most present classes: (7) Ship with 62533 instances, (3) vehicle with 40363, (6) storage tank with 26403. And the least present classes: (4) train station with 1010 instances, (12) dam with 1050, (1) golf field with 1086. This has to be handled carefully so that the algorithm learns to equally identify all classes, and not be biased to the ones with more presence in the dataset.

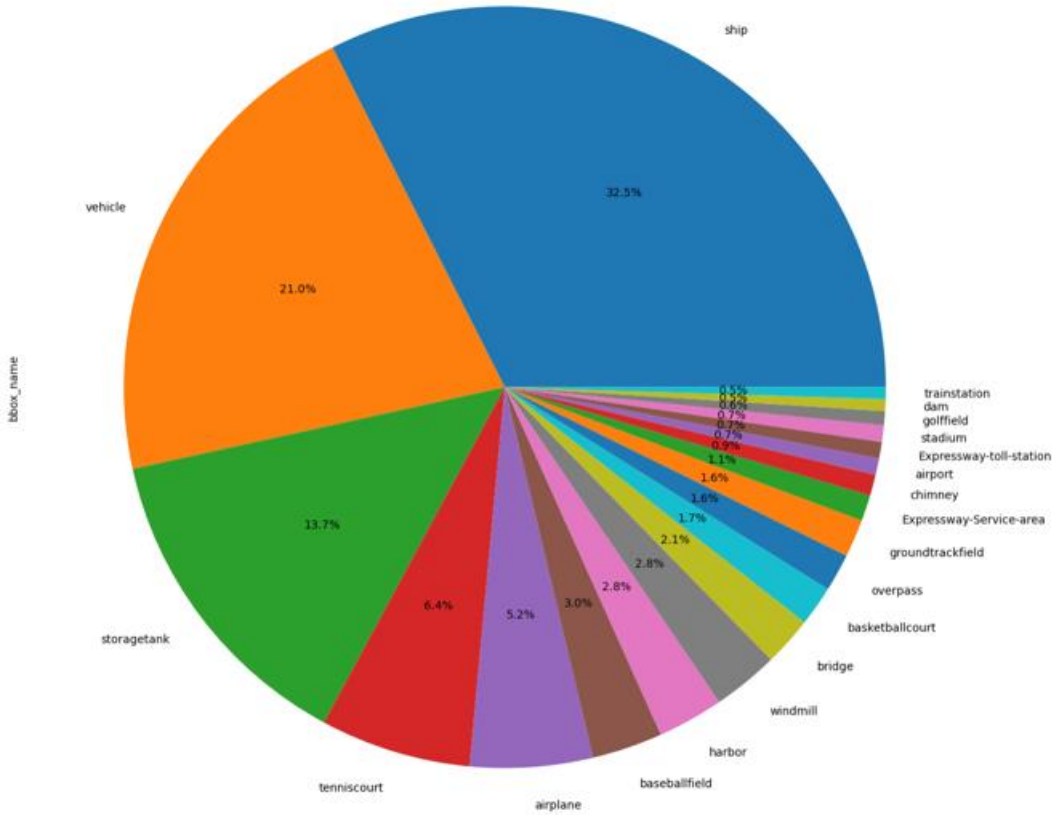


**Fig. 2.** Frequency distribution of objects per class in %

## 3. APPROACH

You Only Look Once (YOLO) is a state-of-the-art, real-time object detection algorithm introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their famous research paper "You Only Look Once: Unified, Real-Time Object Detection" [4].

The authors frame the object detection problem as a regression problem instead of a classification task. The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

In this project, the trained models belong to YOLOv5 and YOLOv8, both developed by Ultralytics, and also the most efficient algorithms for object detection in the YOLO family. By stablishing different parameter configurations in the YOLO models during training for this project, we get to observe the different performance of the algorithms on the DIOR dataset.

### 3.1. Architecture of the models

The YOLO models have their architectural base in the first version. YOLOv1 was the first official YOLO model. It used a single convolutional neural network (CNN) to recognize objects in an image and was relatively fast compared to other object recognition models. However, it was not as accurate as some of the two-stage models at the time. The next versions of YOLO have enhanced the features from the first version to boost the performance or improved the model for a specific purpose, like YOLO9000.

YOLOv1 has overall 24 convolutional layers, four max-pooling layers, and two fully connected layers. The first 20 convolutional layers from the backbone of the network and, with the addition of an average pooling layer and a single fully connected layer, it was pre-trained and validated on the ImageNet 2012 dataset. During inference, the final four layers and 2 FC layers are added to the network; all initialized randomly.

YOLOv5 used the EfficientDet architecture, based on the EfficientNet network, and several new features and improvements, such as hyperparameter optimization, integrated experiment tracking and automatic export to popular export formats, to achieve improved object

detection performance. The models available for YOLOv5 and their specifications are defined in Table 1.

| Model | Size (pixels) | mAP 50-95 (val) | Params (M) | FLOPs (B) |
|---|---|---|---|---|
| YOLOv5n | 640 | 34.3 | 2.6 | 7.7 |
| YOLOv5s | 640 | 43.0 | 9.1 | 24.0 |
| YOLOv5m | 640 | 49.0 | 25.1 | 64.2 |
| YOLOv5l | 640 | 52.2 | 53.2 | 135.0 |
| YOLOv5x | 640 | 53.2 | 97.2 | 246.4 |

**Table 1.** Performance and parameter specification for the five models available for YOLOv5

There are a series of updates and new convolutions in the YOLOv8 architecture. One of the big changes in this model is the anchor-free-detection, which will be covered in the following sections. The models available for YOLOv8 and their specifications are defined in Table 2.

| Model | Size (pixels) | mAP 50-95 (val) | Params (M) | FLOPs (B) |
|---|---|---|---|---|
| YOLOv8n | 640 | 37.3 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 11.2 | 28.6 |
| YOLOv8m | 640 | 50.2 | 25.9 | 78.9 |
| YOLOv8l | 640 | 52.9 | 43.7 | 165.2 |
| YOLOv8x | 640 | 53.9 | 68.2 | 257.8 |

**Table 2.** Performance and parameter specification for the five models available for YOLOv8

### 3.2. Key features in YOLO

By considering that, i.e. for YOLOv5 and YOLOv8, an official paper for the models has not being released, the detailed functionality of the algorithms is still not confirmed by the authors, but rather empirically proven by the users.

Nevertheless, it is noted that the efficiency of the YOLO models relies on five different features, that even though they are not exclusive from the models, it is their specific combination in this algorithm what makes it have a higher performance than other object detection systems. These features will be presented next and how they define the functioning process of the algorithm.

#### 3.2.1. Anchor Boxes

Anchor boxes are a pre-defined set of boxes with specific heights and widths, used to detect object classes with the desired scale and aspect ratio. They are chosen based on the size of objects in the training dataset and are tiled across the image during detection. Although this feature is no longer used in YOLOv8, it was one of the main points of the YOLO algorithm in the other versions.

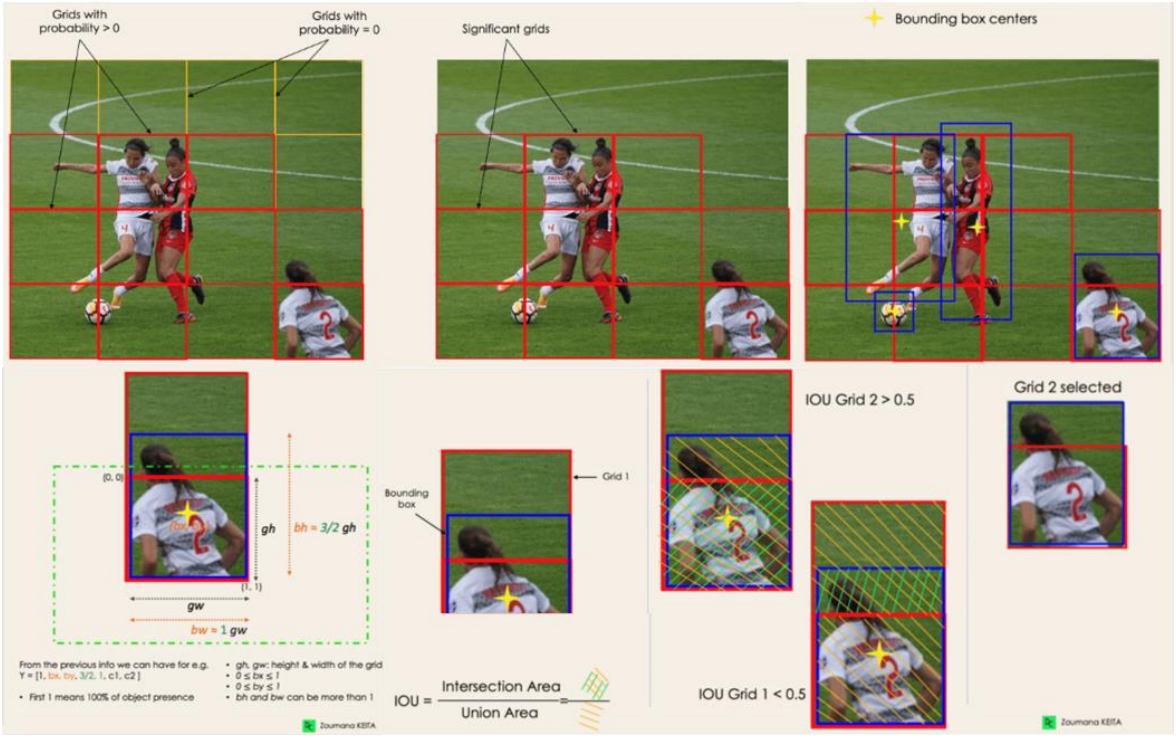


**Fig. 2.** Example images taken from the proposed DIOR dataset, which were obtained with different imaging conditions, weathers, seasons, and image quality.

#### 3.2.2. Residual Blocks

This first step starts by dividing the original image into NxN grid cells of equal shape. Each cell in the grid is responsible for localizing and predicting the class of the object that it covers, along with the probability/confidence value.

#### 3.2.3. Bounding Box Regression

YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

$$Y = [pc, bx, by, bh, bw, c1, c2]$$

- pc corresponds to the probability score of the grid containing an object.
- bx, by are the x and y coordinates of the center of the bounding box w.r.t. the enveloping grid cell.
- bh, bw correspond to the height and the width of the bounding box w.r.t. the enveloping grid cell.
- c1 and c2 correspond to the two classes Player and Ball.

#### 3.2.4. Intersection over Union (IoU)

Most of the time, a single object in an image can have multiple grid box candidates for prediction, even though not all of them are relevant. The goal of the IOU (a value between 0 and 1) is to discard such grid boxes to only keep those that are relevant.
The user defines its IOU selection threshold, which can be, for instance, 0.5. Then, the algorithm computes the IOU of each grid cell which is the Intersection area divided by the Union Area. It ignores the prediction of the grid cells having an IOU $\leq$ threshold and considers those with an IOU > threshold.

#### 3.2.5. Non-Max Suppression (NMS)

Non-Maximum Suppression (NMS) is a post-processing technique used in object detection algorithms to reduce the number of overlapping bounding boxes and improve the overall detection quality. Object detection algorithms typically generate multiple bounding boxes around the same object with different confidence scores. NMS filters out redundant and irrelevant bounding boxes, keeping only the most accurate ones.

### 3.3. New features in YOLOv5

The backbone of YOLOv5 is Darknet53, a new network architecture that focuses on feature extraction. The neck in the architecture aggregates and refines the features extracted by the backbone. YOLOv5's head consists of three branches, each predicting a different feature scale. Each head produces bounding boxes, class probabilities, and confidence scores.

Finally, the network uses Non-maximum Suppression (NMS) to filter out overlapping bounding boxes.

### 3.4. New features in YOLOv8

YOLOv8 supports a full range of vision AI tasks, including detection, segmentation, pose estimation, tracking, and classification.

This model introduces anchor-free detection in the YOLO architecture. This is when an object detection model directly predicts the center of an object instead of the offset from a known anchor box. The advantage of anchor-free detection is that it is more flexible and efficient, as it does not require the manual specification anchor boxes, which can be difficult to choose and can lead to suboptimal results in previous YOLO models such as version 1 and version 2.

The Focal Loss is included in the loss function, which was already introduced in the version 7, to tackle the possible imbalance from the classes in different datasets. We will also find an improvement for detecting objects at different scales and resolutions, by substituting the Cross-Stage Partial with a combined work of Feature Pyramid Network (FPN) and Path Aggregation Network (PAN). For Object Detection, both YOLOv5 and YOLOv8 were pretrained on the COCO detection dataset with an image resolution of 640.

## 4. EXPERIMENTAL SETUP

In this project, a total of 10 experiments were conducted for different model sizes of each version: Nano, Small and Medium, as the following table indicates. As for the distribution of the dataset, it was decided to divide the DIOR dataset differently than what it is suggested in the paper and in the folders: 50% of the images for training and validation and 50% for testing. Instead, the split was calculated according to the number of instances, rather than the number of images, as an attempt to tackle the class imbalance problem presented with the DIOR dataset. Hence, our training set contains 45% of the images (10590), the validation set includes 6387 taken randomly from the training set and the testing set includes 50% (11738), which is the same as the original testing dataset.

Table 3. states the different configuration for each training hyperparameters with different models and YOLO version. All the hyperparameters states indicates that there is difference from the default hyperparameters otherwise the default values are kept.

Three different comparison groups among the obtained training models are performed to evaluate the effect of the model size, YOLO versions as well as the hyperparameters. The first group (i), colored in cyan, is setup to compare the effect of the model size. Thus, Nano-v5 is compared against Nano-v8, etc. The second group (ii), in dark blue, is built up to monitor the effect of epochs' numbers (five and 50) along with the learning rate step independently from the model size namely: Yolov5-small of five epochs against 50 epochs as well as Yolov8-small of five epochs against 50 epochs. Lastly, the third group (iii), in light purple, is configured to evaluate the effect of the optimizer function (SGD) over the different YOLO versions for different model size.

| Model size | V5 | | V8 | |
|---|---|---|---|---|
| **Nano** | epochs: 5<br>batches: 16<br>seed: 42<br>optimizer: Adam<br>image_size: 800<br>lr: 1e-2 | | epochs: 5<br>batches: 16<br>seed: 42<br>optimizer: Adam<br>image_size: 800<br>lr: 1e-2 | epochs: 10<br>batches: 16<br>optimizer: SGD<br>image_size: 800<br>lr: 1e-2 |
| **Small** | epochs: 5<br>batches: 16<br>seed: 42<br>optimizer: Adam<br>image_size: 800<br>lr: 1e-2 | epochs: 50<br>batches: 16<br>seed: 42<br>optimizer: Adam<br>image_size: 800<br>lr: 1e-2 | epochs: 5<br>batches: 16<br>seed: 42<br>optimizer: Adam<br>image_size: 800<br>lr: 1e-2 | epochs: 50<br>batches: 16<br>seed: 42<br>optimizer: Adam<br>image_size: 800<br>lr: 1e-3 |
| **Medium** | epochs: 5<br>batches: 16<br>seed: 42<br>optimizer: Adam<br>image_size: 800<br>lr: 1e-2 | epochs: 10<br>batches: 16<br>optimizer: SGD<br>image_size: 800<br>lr: 1e-2 | epochs: 5<br>batches: 16<br>seed: 42<br>optimizer: Adam<br>image_size: 800<br>lr: 1e-2 | |

**Table 3.** Experiments configurations for a variety of models (Nano. Small, and Medium), and YOLO V5 & V8.

Different training models are trained on different computer configurations as follow:

| | |
|---|---|
| Trained with five epochs (cyan group) | Google Colab. free version.<br>NVIDIA-SMI 525.105.17<br>Driver Version: 525.105.17<br>CUDA Version: 12.0<br>Tesla T4 15360MiB |
| Trained with ten epochs (dark blue group) | Google Colab. Pro<br>Torch 2.0.1<br>CUDA Version: 18.0<br>Tesla V100 – SXM2 – 16GB15360MiB |
| Trained with 50 epochs (light purple group) | Torch 2.0.1<br>CUDA 117<br>Quadro P2000, 5120 MiB |

**Table 4.** PC configuration for different training experiments

## 5. RESULTS

The results are compared based on different metrics namely: Precision (P), Recall (R), mean average precision 50 (mAP50), and mAP50-95.

The precision is calculated as the ratio between the number of Positive samples correctly classified to the total number of samples classified as Positive (either correctly or incorrectly). In other words, precision measures how accurate is the predictions of the model. i.e. the percentage of the model's predictions are correct. While, the recall measures the model's ability to detect Positive samples. Thus, recall measures how good the model finds all the positives. The higher the recall, the more positive samples are detected. On the other hand, average precision represents the area under the curve Precision Recall Curve. The higher the curve is in the upper right corner, the larger the area, so the higher the AP, and the better the machine learning model. Thus, the mAP is the mean average precision computes the average precision value for recall value over 0 to 1. However, mAP50-95 means average mAP over different intersection over union IoU thresholds. While, mAP50 is the average precision at IoU = 0.5.

### 5.1. First group (i), in cyan color, of five epochs

In this group the comparison between the different model size between two Yolo versions namely V5 and V8 for five epochs only.

| Model | V5 | V8 |
|---|---|---|
| Nano | 214 Layers<br>1,790,977 Parameters<br>1,790,977 Gradients<br>02:08:33 Duration | 225 Layers<br>3,014,748 Parameters<br>3,014,732 Gradients<br>01:29:01 Duration |
| Small | 214 Layers<br>7,073,569 Parameters<br>7,073,569 Gradients<br>02:09:12 Duration | 225 Layers<br>11,143,340 Parameters<br>11,143,324 Gradients<br>01:46:36 Duration |
| Medium | 291 Layers<br>20,948,097 Parameters<br>20,948,097 Gradients<br>02:09:52 Duration | 295 Layers<br>25,867,900 Parameters<br>25,867,884 Gradients<br>02:22:53 Duration |

**Table 5.** Model size for v5 and v8 configuration

From the results it is shown that despite of Yolov8 has more layers and parameters but it is faster than Yolov5 except than the medium model. Also, the metrics results of over the validation dataset in table6 shows that Yolov8-Nano has better score except for precision score. This can be explained since the Nano model is the smallest model among the others, thus, it requires fewer steps to converge compared with the bigger models.

| Model | V5 | | | | V8 | | | |
|---|---|---|---|---|---|---|---|---|
| | P | R | mAP50 | mAP50-95 | P | R | mAP50 | mAP50-95 |
| Nano | 0.24 | 0.39 | 0.17 | 0.09 | 0.31 | **0.47** | **0.20** | **0.12** |
| Small | 0.21 | 0.41 | 0.17 | 0.09 | 0.32 | 0.37 | 0.17 | 0.10 |
| Medium | **0.40** | 0.35 | 0.16 | 0.09 | 0.23 | 0.33 | 0.14 | 0.09 |

**Table 6.** Metric results for group l (cyan color) with five epochs

### 5.2. Second group (ii), in dark blue color, of 50 epochs

This comparison is configured to compare the YOLO versions 5 and 8 over long training epochs of 50. However, the learning rate step of YOLO-v8 is ten times less than YOLO-v5. Despite of the gap of learning rate of v8, it overcomes older version in all metrics as it is shown in Table.7. However, YOLO-v5 is more efficient.

Comparing the results in Table 6 & 7 it is clearly seen that increasing the training epochs have a direct correlation to increase the overall performance of both models YOLO v5 & v8. However, the training duration is increased dramatically. Nevertheless, the PC configuration plays a major role here and has its impact over the performance. Since the training of 50 epochs was trained on different configuration in comparison with five epochs configuration.

| Model | V5s– 50epochs \| lr 1e-2 | | | | V8s – 50epochs \| lr 1e-3 | | | |
|---|---|---|---|---|---|---|---|---|
| | P | R | mAP50 | mAP50-95 | P | R | mAP50 | mAP50-95 |
| Small | 0.32 | 0.68 | 0.31 | 0.19 | **0.39** | **0.79** | **0.39** | **0.30** |
| Duration | **258.602 h ~ 11 days** | | | | 290.028 h ~ 12 days | | | |

**Table 7.** Metric results for group (ii) (dark blue) with 50 epochs for YOLO v5 & v8

### 5.3. Third group (iii), in light purple color, of ten epochs

This comparison group is performed to evaluate the effect of the optimizer function. From the Table 8. It is shown that YOLOv8 with Nano model size overcomes the Medium size of YOLOv5. This might be interpreted that since the medium model size is bigger than the Nano one, thus, it requires more training epochs to converge better.

| Model | V5m – 10epochs \| SGD | | | | Model | V8n – 10epochs \| SGD | | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | mAP50 | mAP50-95 | | P | R | mAP50 | mAP50-95 |
| Medium | 0.39 | **0.76** | 0.39 | 0.29 | Nano | **0.90** | 0.25 | **0.57** | **0.48** |

**Table 8.** Metric results for group (iii) (light purple) with ten epochs for YOLO v5-Medium & v8-Nano

However, from metric scores in Table 8., Compared with the other Tables 6 & 7, the SGD optimizer overcomes Adam for both YOLO version 5 & 8 for different model sizes.

## 6. CONCLUSION

In conclusion, DIOR dataset has been widely used in research on object detection in aerial images and has contributed to the development of new object detection algorithms and systems. It has also been used as a benchmark for evaluating the performance of different object detection approaches. However, in addition to the data imbalance, the constant 800 x 800 image size make the dataset not robust for image ratio variation. From the results, the two versions of YOLO (v5, and v8) show solid performance on the DIOR datasets. However, based on our validation results, YOLO v8 seemed to have the best performance out of the three. Also, to have an overall good performance score over than 50 epochs should be trained. Nevertheless, undoubtedly the choice of the optimizer function plays a vital role for having a good performance. As it was concluded that SGD performed better results in compare with Adam. Additionally, having a GPU setup helps to reduce the training time.

## 7. References


[1] A. Shirazy, A. Shirazi and H. Nazerian, "Application of Remote Sensing in Earth Sciences – A Review," *International Journal of Science and Engineering Applications (IJSEA)*, vol. 10, no. 5, p. 45/51, 2021.

[2] K. Li, G. Wan, G. Cheng, L. Meng and J. Han, "Object Detection in Optical Remote Sensing Images: A Survey and A New Benchmark," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 159, no. 0924-2716, pp. 296-307, 2020.

[3] K. He, X. Zhang , S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[4] Redmon, Joseph, et al. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 779-788.