

OBJECT DETECTION APPLICATION ON DIOR DATASET BY USING YOLO v5 ARCHITECTURE

Abdulkali Kocabasa, Yue Yao

TU Braunschweig

ABSTRACT

You only look once (YOLO) is a state-of-the-art object detection method which is extremely fast and accurate. In this project, we applied YOLOv5 on the DIOR dataset. We tried to go deep in the algorithm of YOLO and train variant models (YOLOv5s and YOLOv5m). Then we evaluated the performance of models with some metrics and observed their difference in various classes. In conclusion, from visualization of detection results we compared different models of YOLO.

Index Terms— YOLO, DIOR, OBJECT DETECTION

1. INTRODUCTION

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.[2] Object detection is a computer vision technique that allows us to identify and locate objects in an image or video. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them. Imagine, for example, an image that contains two cats and a person. Object detection allows us to at once classify the types of things found while also locating instances of them within the image. [1] By using the information obtained in this way, human life is made easier and new developments are paved the way for applications in various fields such as faster decision-making of systems, more precise operation of robots, safe travel of unmanned vehicles and the ability of camera systems to distinguish relevant faces and objects.

Many different classification and object identification systems used in this field, such as Fastest DPM, R-CNN Minus R, Fast R-CNN and Faster R-CNN VGG-16, reuse classifiers or localizers to perform detection. They apply the

model to an image at multiple locations and scales. High-scoring regions of the image are considered detections. YOLO (you only look once) uses a different approach. By applying a single neural network to the entire image, the network divides the image into specific regions and estimates bounding boxes and probabilities for each region. These bounding boxes are weighted according to the estimated probabilities. These differences make YOLO the fastest system developed in this field, which attracts our interest and encourages us to do more research in this area. In this study, we use the DIOR dataset with its large number of instances and diversity on the related classes will provide a basis for our system to deliver higher accuracy results.

2. MOTIVATION

In the final project of the deep learning course conducted at TU Braunschweig, the real-time object identification method using YOLOv5 technique is applied to the DIOR dataset and the results are aimed to lay the foundation for future projects and research focused on object identification for remote sensing datasets. Object identification, which has many applications such as electric vehicles, autonomous driving, unmanned aerial and ground vehicles, and many industrial and defense systems, is currently used to identify the location, position, and related information of multiple or single objects in different classes because of processing high-resolution images obtained with advanced lidar and camera systems. This study will review the work done in this area and a test application will be performed on the DIOR dataset using the YOLO technique. With this study, we aim to gain knowledge about these systems that are widely used today.

3. MATERIALS AND METHODS

The materials and methods used in this work are DIOR dataset and YOLO version 5. Since the state of the art of YOLOv5 provides impressive performance features, it makes itself a good family of compound-scaled object detection models. On the other hand, DIOR has many instances and is a relatively big dataset to work with, so combination of YOLO's performance and DIOR's rich

variety makes this work a good example on remote sensing projects.

3.1. Dataset-DIOR

There are many different datasets used in the field of object identification. TAS(Things and Stuff) was published at the beginning of 2008. Then there are 8 different datasets that follow it. The most notable of these are the DOTA and DIOR datasets shared in 2017 and 2018. Each of these datasets has different category numbers and types, different category sample counts, different sample resolutions and different annotation paths. Indeed, the DIOR dataset, which is superior to the others in terms of both number and diversity of data, has 20 different categories and 23463 sub-samples. In total, it has 190288 instances. Each image has a width of 800 pixels. Some of the other features that distinguish the DIOR dataset from others are as follows:

- 1-A large range of object size variations
- 2-Rich image variations
- 3-High inter-class similarity and intra-class diversity.

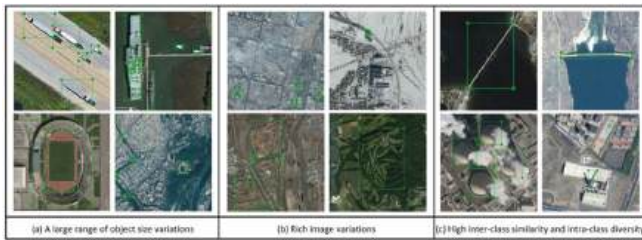


Figure 1: Characteristic of DIOR Dataset[2]

For better understanding the dataset, some statistics are made. From Figure 2 and 3 it can be known that the amount and distribution of each class in the DIOR dataset are extremely imbalanced. For instance, a ship appears in 2706 images 62537 times, which means on average an image contains 23 ships. Meanwhile, chimney shows only in 853 samples for 1680 times. This imbalance may cause different detection precision.

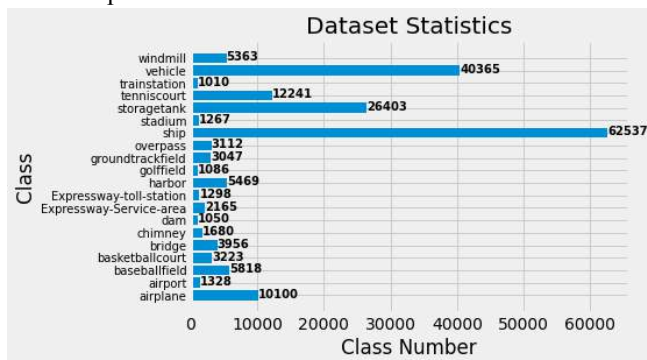


Figure 2: the Instance Number of Each Class

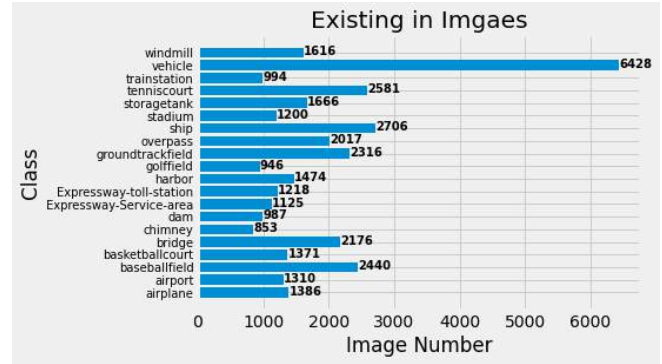


Figure 3: the Image Number including Each Class

3.2. Model Technique-YOLO

As YOLO represented, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, YOLO frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.[2] Represented architecture is extremely fast compared to the other architectures such as 100Hz DPM, Fastest DPM, R-CNN Minus R and Faster R-CNN VGG-16. The YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives in the background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.[2] That is why YOLO is the desired object detection architecture of this work.

VOC 2012 test	mAP	ave	bike	bird	boat	bottle	bus	car	cat	cow	chair	class	dog	horse	motorbike	person	sheep	sofa	train	to
MR-CNN-S3RG-DETA [1]	73.9	83.2	82.9	76.6	87.8	82.7	79.4	71.2	85.8	85.0	79.1	42.2	87.0	83.4	84.7	78.3	43.3	75.4	83.3	74.0
HypoNet-VGG	71.6	84.2	78.5	73.6	53.6	53.7	78.7	70.8	87.7	89.6	14.9	52.1	86.0	81.7	85.3	81.8	48.6	73.5	79.4	65.7
HypoNet-AP	71.3	84.1	78.3	73.3	53.5	53.6	78.6	70.6	87.5	89.5	14.9	52.1	85.8	81.6	83.2	81.8	48.4	73.2	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	53.8	43.4	79.1	73.3	89.4	89.4	75.5	57.0	87.5	80.9	81.0	74.7	41.8	73.5	88.3	67.2
MR-CNN-S-CNN [1]	70.7	83.0	78.6	71.5	55.1	57.7	78.0	71.9	84.6	90.5	56.1	61.1	85.5	79.9	81.7	78.4	41.0	69.0	81.2	77.7
Faster R-CNN [1]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.8	40.1	72.6	80.9	81.2
DEEP-ENS-COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.8	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	88.8	75.9
Net [19]	68.8	82.8	79.0	71.6	52.5	53.7	74.1	69.0	84.8	46.0	74.3	55.1	85.0	81.3	79.5	72.2	38.9	72.4	85.5	68.1
Fast R-CNN [1]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.2	80.5	80.8	72.0	35.1	68.3	65.7	80.4
UMCHLPG-STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.8	82.7	86.6	35.8	82.7	77.1	79.9	68.7	41.4	69.0	80.9	72.0
NUS-NN	63.8	80.2	75.8	61.9	45.7	43.0	70.3	67.6	80.7	41.9	60.7	51.7	78.2	75.2	76.9	85.1	38.0	68.3	58.9	63.3
BodyLearning [7]	63.2	73.0	74.2	61.3	43.7	42.7	68.2	66.8	80.2	40.0	70.0	49.8	79.0	74.5	77.9	94.9	33.3	67.9	53.7	68.7
NUS-NN	62.4	77.8	73.1	62.6	39.5	43.5	69.1	66.4	78.9	39.1	68.1	30.0	77.2	71.3	78.1	84.7	38.4	66.9	56.2	66.9
R-CNN VGG-BB [10]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	34.6	38.5	67.2	46.7	82.0	74.8	78.0	85.2	33.6	65.4	54.2	67.4
R-CNN VGG [12]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	61.6	81.1	35.7	64.3	43.9	80.8	71.6	74.0	80.9	30.8	63.4	52.9	63.5
YOLO	57.0	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	64.8	48.5	77.2	72.3	71.3	83.5	28.9	52.2	54.4	73.9
Feature L1 [14]	56.3	74.6	69.1	54.4	38.1	31.1	65.2	62.7	69.7	30.8	56.0	41.6	70.0	64.8	71.1	82.2	31.3	61.1	46.1	67.7
R-CNN BB [10]	51.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.8	59.3
SDS [14]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.4	58.7
R-CNN [10]	49.6	64.1	63.8	46.1	29.4	27.9	56.6	57.0	63.9	26.5	46.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	59.6

Table 1: PASCAL VOC 2012 Leaderboard[2]

Another perspective on this work can be the combination of Fast R-CNN+YOLO to perform better detection as it can be seen on Table 1. Column vectors represent the class examples from the related work.

Specifically YOLOv5 is chosen because of its wide range of application areas, still adding new features, and has a wide range of examples with tutorials. That makes it easy to modify and learn in a limited time. YOLOv5 mainly works with the pictures having 32x pixel sizes hence DIOR is a good choice because of its size 800 by 800 pixels.

3.3. YOLO and Datasets

The YOLOv5 community mentions that a dataset where people seek to have better results should fit some requirements. It does not promise a successful run and results but a convenient way to work with. These features are:

3.3.1. Images per class

The dataset should have more than 1.5k images per class.[6]

3.3.2. Instances per class

The dataset should have more than 10k instances (labeled objects) per class total.[6]

3.3.3. Image variety

It must be representative of the deployed environment. For real-world use cases, it is recommended to have images from different times of day, different seasons, different weather, different lighting, different angles, different sources (scraped online, collected locally, different cameras) etc.[6]

3.3.4. Label consistency

All instances of all classes in all images must be labeled. Partial labeling will not work.[6]

3.3.5. Label accuracy

Labels must closely enclose each object. No space should exist between an object and its bounding box. No objects should be missing a label.[6]

3.3.6. Background images

Background images are images with no objects that are added to a dataset to reduce False Positives (FP). We recommend about 0-10% background images to help reduce FPs (COCO[7] has 1000 background images for reference, 1% of the total).[6]

3.4. Model Selection and Yolo Terms

Model selection is one of the most important steps in deep learning. For the sake of a successful training process, the model needs to be chosen wisely. In this work, we focused on using one of the commonly known models in YOLOv5. These models are provided ready to use in the YOLOv5 github repository.

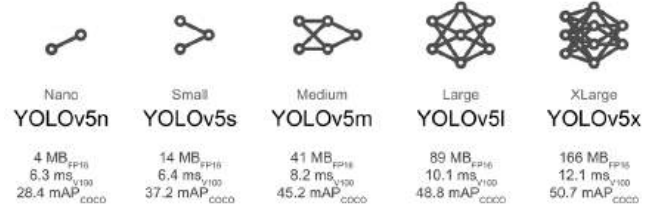


Figure 4: Ready to use models of YOLOv5[6]

3.4.1. Learning rate

Learning rate value can be chosen manually by modifying the hyperparameter folder in YOLO where it is stored under /yolov5/data/hyps/hyp.scratch-low.yaml. All of the hyperparameters are stored in this folder. By default, the learning rate is initialized by 0.01. However, it will be decreased by the multiplication of 0.01 which ends up being 0.0001 for only the last epoch. It is independent from how many epoch the developer sets, only the last will have this small learning rate.

3.4.2. Transfer learning

Transfer learning is one of the famous techniques to have fast and good results. Yolo has different types of ready to use models in their repository. These models are trained by the COCO[7] dataset by the image size of 640x640. There are several types of transfer learning options, these models are called YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x. Every one of these models has a different number of layers For a better understanding of the performance of these models, the following Table 2 can be used.

Model	size (pixels)	mAP ^{val} 4.5.0.15	mAP ^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Table 2: YOLOv5 model comparison[6]

Model	Depth Multiple	Width Multiple
YOLOv5n	0.33	0.25
YOLOv5s	0.33	0.5
YOLOv5m	0.67	0.75
YOLOv5l	1	1
YOLOv5x	1.33	1.25

Table 3: Depth and width multiple of different YOLOv5 models

The different kinds of YOLOv5 models have the same block structure. They are different by 2 parameters: depth multiple and width multiple. Figure 5 demonstrates the structure of YOLOv5s, which is the basic model of YOLOv5. The 2 parameters applied in it are both 1. The actual number of layers in models is obtained by multiplication of the depth multiple and basic number of various layers. And the width multiple controls the number of output channels before the last output. The 2 parameters decide the size of the models and the computer power requirement.

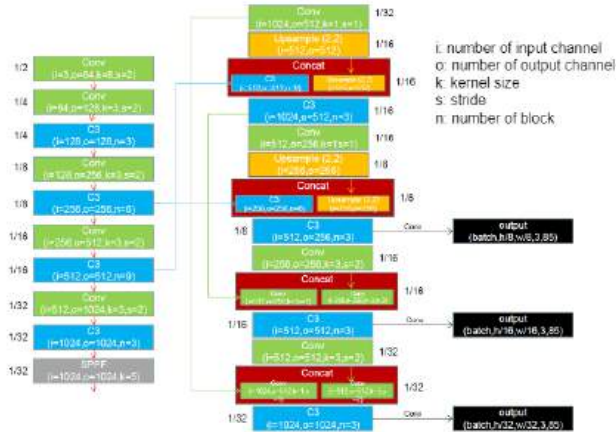


Figure 5: The basic structure of YOLOv5 models[8]

In our project, we used YOLOv5s after considering speed and the number of parameters that needed to be trained. On the other hand, YOLOv5x and YOLOv5l should be considered to have better mAP results in the future. Since the time for this project is limited, one should consider the performance and training time.

Another good feature of the YOLOv5 is that we may freeze the number of layers as we wish. That allows us to decrease the number of parameters that need to be trained.

3.4.3. Albumentation

Albumentation is a widely used method to overcome overfitting and lack of data. It is used for the training data

in YOLOv5. By default, Blur, MedianBlur, ToGray, Clahe, tile_grid_size are applied. The probability of all of these albumentation methods being used is 0.01.

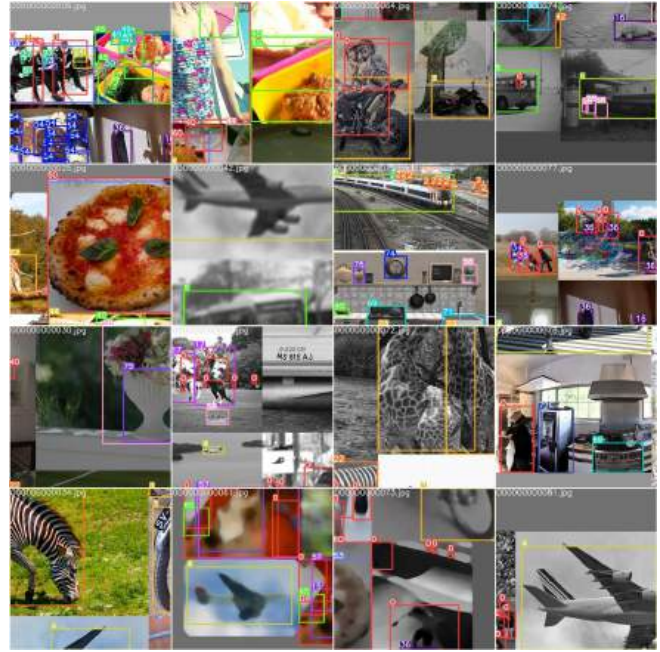


Figure 6: Albumented training images in mosaic

This image shows that some of the training images are albumented. Gray scale pictures can be seen relatively ebay compared to the other albumentations. This picture is created by the YOLOv5 itself by using mosaic functionality.[6]

3.4.4. Visualization tools

There are several tools which might be used in YOLOv5 however, we only used tensorboard for interface and result.png which is provided by YOLOv5 after training is done.

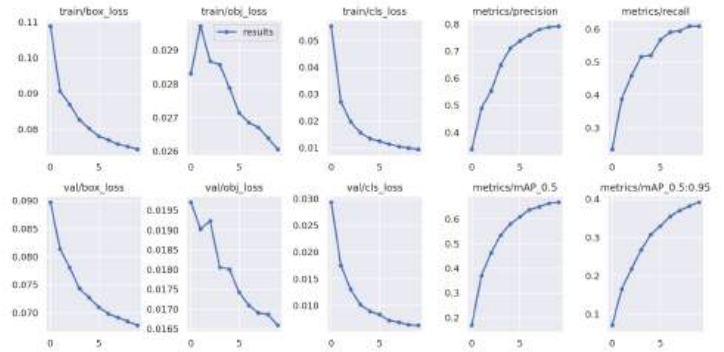


Figure 7: Typical result graphs of YOLOv5

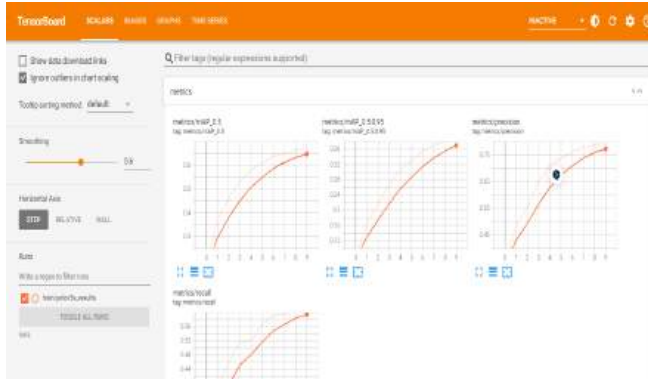


Figure 8: Typical result graphs of Tensorboard[9]

3.4.5. Hyperparameters

There are plenty of hyperparameters in YOLOv5. Here are some of them:

`lr0: 0.01` # initial learning rate (SGD=1E-2, Adam=1E-3)

`lrf: 0.01` # final OneCycleLR learning rate (`lr0 * lrf`)

`momentum: 0.937`

`weight_decay: 0.0005`

`warmup_epochs: 3.0`

`warmup_momentum: 0.8` # warmup initial momentum

`warmup_bias_lr: 0.1` # warmup initial bias lr

`box: 0.05` # box loss gain

`cls: 0.5` # cls loss gain

`cls_pw: 1.0` # cls BCELoss positive_weight

`obj: 1.0` # obj loss gain (scale with pixels)

`obj_pw: 1.0` # obj BCELoss positive_weight

`iou_t: 0.20` # IoU training threshold

`anchor_t: 4.0` # anchor-multiple threshold

anchors: 3 # anchors per output layer (0 to ignore)

`fl_gamma: 0.0` # focal loss gamma

`hsv_h: 0.015` # image HSV-Hue augmentation (fraction)

`hsv_s: 0.7` # image HSV-Saturation augmentation (fraction)

`hsv_v: 0.4` # image HSV-Value augmentation (fraction)

`degrees: 0.0` # image rotation (+/- deg)

`translate: 0.1` # image translation (+/- fraction)

`scale: 0.5` # image scale (+/- gain)

`shear: 0.0` # image shear (+/- deg)

`perspective: 0.0` # image perspective (+/- fraction)

`flipud: 0.0` # image flip up-down (probability)

`fliplr: 0.5` # image flip left-right (probability)

`mosaic: 1.0` # image mosaic (probability)

`mixup: 0.0` # image mixup (probability)

`copy_paste: 0.0` # segment copy-paste (probability)

3.4.6. Cfg

Cfg is the model structure if no transfer learning is being used during the process. One might create one by default or use the given transfer learning models but without their weights. The models are called the same as it is given in the transfer learning section. The models can be described separately and can be modified and customized.

3.4.7. Batch

To decide how big a batch will be used in every epoch can be decided by the user; however, it is mainly restricted by the features of the hardware. In google colab, hardware features are better than GPU and CPU of daily laptops however, it has its limits. Since DIOR dataset is a large dataset, we can not exceed a batch number of 128 therefore a batch of 64 is used in the project.

3.4.8. Epochs

Number of epochs used during training is up to us however, the right number of epochs needs to be determined. To do so, some mock training runs are tested and 50 epochs are decided to be used. Unfortunately, there is no early stop feature in YOLOv5 however, the best and last weights are always updated after every epoch. This lets us use the resume feature of YOLOv5. Resume basically refers to using the last epoch to start a new training session. This makes the process even better and controllable. Moreover, the project has been developed not only in one training run but several.

3.4.9. Activation function

Activation function can be chosen for different layers by using `common.py` file in the YOLOv5. For common usage LeakyReLU activation function is used for the hidden layers and Sigmoid for the final detection layer. The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0

and 1, sigmoid is the right choice. The function is differentiable. [10]

3.4.10. Lost function

In the YOLO family, a compound loss is calculated based on objectness score, class probability score, and bounding box regression score.

Ultralytics have used Binary Cross-Entropy with Logits Loss function from PyTorch for loss calculation of class probability and object score.

We also have an option to choose the Focal Loss function to calculate the loss. You can choose to train with Focal Loss by using `fl_gamma` hyper-parameter. [16]

3.4.11. Optimizer

For optimization functions in YOLOv5, we have two options SGD and Adam. In YOLOv5, the default optimization function for training is SGD. It can be modified by using “—adam” command-line argument.

3.4.12. Bounding Boxes

Boundary boxes can be in two different forms: Oriented bounding box and vertical boundary box. Yolo uses a vertical boundary box by default.

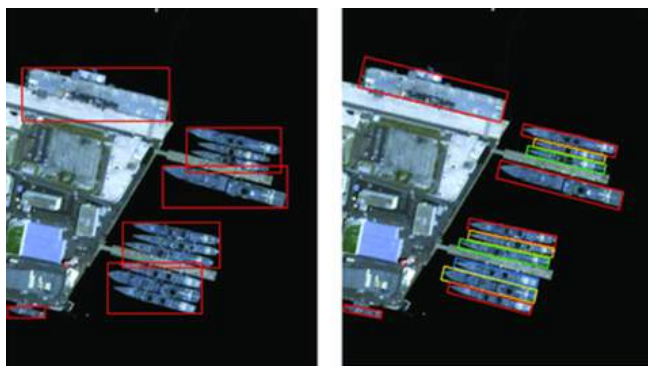


Figure 9: Difference between oriented and horizontal boundary boxes [11]

3.4.13. Confidence level of bbox

In a nutshell, object detectors predict the location of objects of a given class in an image with a certain confidence score. Locations of the objects are defined by placing bounding boxes around the objects to identify their position.

Therefore, a detection is represented by a set of three attributes:

1- Object class (i.e. person)

2- Corresponding bounding box (i.e. [63, 52, 150, 50])

3- Confidence score (i.e. 0.583 or 58.3%)

Similarly, the performance evaluation of the object detection model is done based on:

1- A set of ground-truth bounding boxes representing the rectangular areas of an image containing objects of the class to be detected

2- A set of detections predicted by a model, each one consisting of a bounding box, a class, and a confidence value.

3.4.14. Anchor box

Anchor box is just a scale and aspect ratio of specific object classes in object detection. The FPN (Future Pyramid Network) has three outputs and each output's role is to detect objects according to their scale. For example:

P3/8 is for detecting smaller objects.

P4/16 is for detecting medium objects.

P5/32 is for detecting bigger objects.

So when you're going to detect smaller objects you need to use smaller anchor boxes and for medium objects you should use medium scale anchor boxes. One can see it on following image as well:

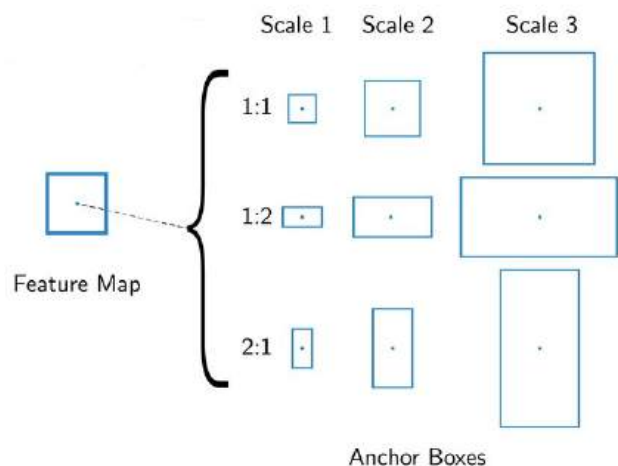


Figure 9: Anchor Boxes [12]

The fact that why do we use smaller anchor boxes for bigger feature maps is because, during downsampling the feature maps if you downsample it many times you may lose smaller objects, that's why bigger feature maps are used for smaller anchor boxes to detect smaller objects.

3.4.15. Backbone

A convolutional neural network that aggregates and forms image features at different granularities. CSPNet solves the problems of repeated gradient information in large-scale backbones, and integrates the gradient changes into the feature map, thereby decreasing the parameters and FLOPS (floating-point operations per second) of model, which not only ensures the inference speed and accuracy, but also reduces the model size.[13]

3.4.16. Neck

A series of layers to mix and combine image features to pass them forward to prediction. Secondly, the YOLOv5 applied a path aggregation network (PANet) as its neck to boost information flow. PANet adopts a new feature pyramid network (FPN) structure with enhanced bottom-up path, which improves the propagation of low-level features. [13]

3.4.17. Head

Consumes features from the neck and takes box and class prediction steps. The head of YOLOv5, namely the YOLO layer, generates 3 different sizes (18×18 , 36×36 , 72×72) of feature maps to achieve multi-scale prediction, enabling the model to handle small, medium, and big objects. [13]

3.5. Coding Environment

Lastly, the model training process will be completed through Tensorflow using the Google Colab interface, and the relevant data samples and evaluation results can be saved using Tensorboard and saved for later review. We aim to expand the recognition range of the model by increasing the variety of data to be used in relevant data generation and augmentation techniques. As a result of the study, we aim to achieve an accuracy of approximately 85%.

With the power of Google Colab, we are able to use the Tesla T4 or Tesla P100-PCIe GPU. They provide approximately 16 GB GPU memory and their peak performance is around 12 TFlops. With the 25 CPU RAM, every epoch takes about 20 minutes for YOLOv5s and YOLOv5m models.

4. LIMITATIONS

There are several limitations of the training process. One of them is the RAM. To be able to cache images in the training environment we approximately need 36 GB for training and 4.5 GB for the validation set. Unfortunately, the RAM provided from the google colab is limited to 25 GB for google colab subscription. This makes the process not possible to run on the RAM however, YOLOv5 has another approach to cache images. This is called disk caching. The cache files are created once disk caching starts. This process

takes a little longer time compared to the RAM caching but overall it fastens the process of the training compared to the non cache training. Time consumed without cache processing is approximately 80 minutes per epoch. This means with cache processing, we reduce the time by quarter of the non cache processing. Unfortunately we are not able to run YOLOv5l and YOLOv5x models since they are bigger models compared to the YOLOv5s and YOLOv5m because of the RAM limitation.

Another limitation of the training system is related to the batch number. Bigger batch numbers allow us to observe better correlations but it is also limited with the hardware that is used. When 128 batch is tried by us, that returns a VRAM(Visual RAM from GPU) failure and the system immediately stops execution of the related cell in google colab.

In addition to the previous limitations, one of the most challenging parts of the training process is the scanning part. YOLOv5 needs to confirm how much of the training and validation data is provided and actually there in the given folder. This process takes too long when the data is stored in the google drive. This is because of the internet needed and the process between google colab and the google drive. Solution of this problem is providing an external link for google colab to download the dataset as a rar or zip file and extracting the files directly in the google colab directory. This allows google colab to use its own computational power without being forced to use the internet connection. Moreover, this method brings us another limitation which is the bandwidth limitation of the cloud providers such as google drive and dropbox. However, by defining different accounts for data solves the problem temporarily.

5. EVALUATION

Using the DIOR dataset and taking the YOLOv5 architecture as a basis, this study aims to lay the foundation for future studies by comparing the information learned and the outputs obtained with previous studies in this field and to gain depth about object identification systems. To evaluate the study of YOLOv5 structure on DIOR dataset, mAP values from the validation set will be used. mAP can be explained as (mean Average Precision) that uses other dataset related IOU, Precision, Recall, Precision Recall Curve and AP. .

5.1. IOU

Object detection models predict the bounding box and category of objects in an image. Intersection Over Union (IOU) is used to determine if the bounding box was

correctly predicted. The IOU indicates how much bounding boxes overlap. This ratio of overlap between the regions of two bounding boxes becomes 1.0 in the case of an exact match and 0.0 if there is no overlap.

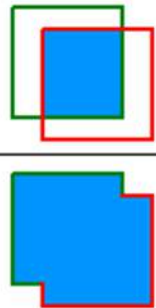
$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


Figure 10: IOU [14]

In the evaluation of object detection models, it is necessary to define how much overlap of bounding boxes with respect to the ground truth data should be considered as successful recognition. For this purpose, IOUs are used, and mAP50 is the accuracy when IOU=50, i.e., if there is more than 50% overlap, the detection is considered successful. The larger the IOU, the more accurate the bounding box needs to be detected and the more difficult it becomes. For example, the value of mAP75 is lower than the value of mAP50.

5.2. Precision and Recall

Precision is the ability of a model to identify only the relevant objects. It answers the question: What proportion of positive identifications was correct? A model that produces no false positives has a precision of 1.0. However, the value will be 1.0 even if there are undetected or not detected bounding boxes that should be detected.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

Figure 11: Precision [14]

Recall is the ability of a model to find all ground truth bounding boxes. It answers the question: What proportion of actual positives was identified correctly? A model that produces no false negatives (i.e. there are no undetected bounding boxes that should be detected) has a recall of 1.0. However, even if there is an “overdetection” and wrong bounding box are detected, the recall will still be 1.0.

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}}$$

Figure 12: Recall [14]

5.2.1. Precision-recall curve

The Precision Recall Curve is a plot of Precision on the vertical axis and Recall on the horizontal axis.

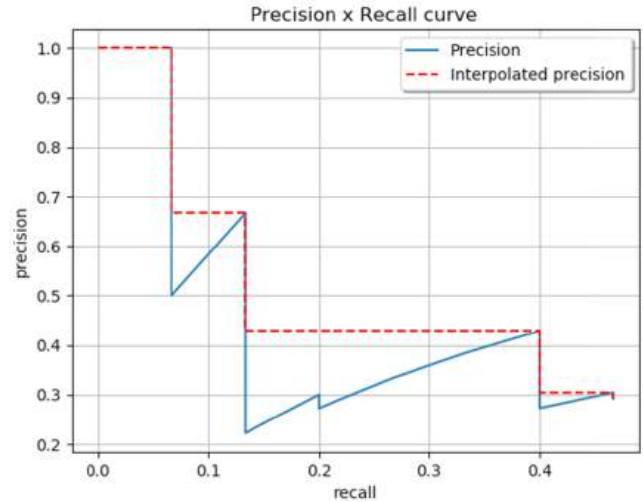


Figure 13: Precision recall curve[14]

There is a threshold for object detection. Increasing the threshold reduces the risk of over-detecting objects but increases the risk of missed detections. For example, if threshold=1.0, no object will be detected, Precision will be 1.0, and Recall will be 0.0. On the other hand, if threshold=0.0, an infinite number of objects will be detected, Precision will be 0.0, and Recall will be 1.0. Conversely, if threshold=0.0, an infinite number of objects will be detected, Precision will be 0.0, and Recall will be 1.0.

In the case of a good machine learning model, over-detection will not occur even if the threshold is reduced (Recall is increased), and Precision will remain high. Therefore, the higher up the curve to the right in the graph, the better the machine learning model is.

5.3. AP

When comparing the performance of two machine learning models, the higher the Precision Recall Curve, the better the performance. It is time-consuming to actually plot this curve, and as the Precision Recall Curve is often zigzagging, it is subjective to judge whether the model is good or not.

A more intuitive way to evaluate models is the AP (Average Precision), which represents the area under the curve (AUC) 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.

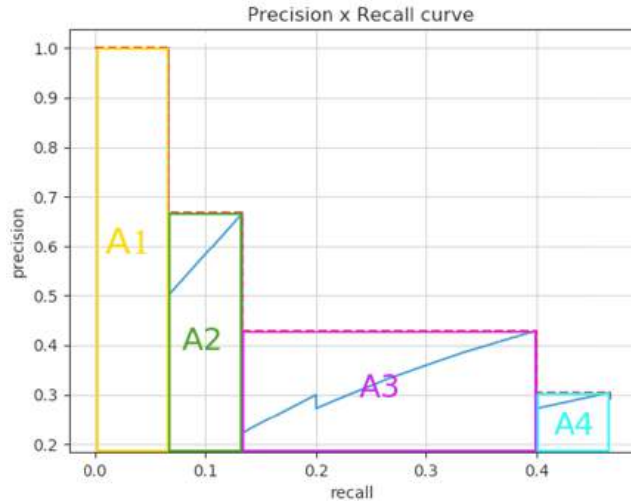


Figure 14: Precision recall curve[14]

5.4. mAP

The mAP is an average of the AP values, which is a further average of the APs for all classes.

6. RESULTS

In this project, two different models of YOLOv5 are used. Training, validation and testing results are represented in the upcoming sections. Comparison between the two models is made by using validation mAP values, confusion matrices and detected objective pictures.

6.1. Class Validation Comparison

Class	Images	Instances	P	R	mAP50	mAP50-95:
all	2346	18702	0.75	0.674	0.737	0.514
airplane	2346	1094	0.901	0.902	0.943	0.723
airport	2346	138	0.686	0.348	0.504	0.251
baseballfield	2346	590	0.898	0.895	0.930	0.794
basketballcourt	2346	300	0.698	0.907	0.944	0.815
bridge	2346	374	0.525	0.388	0.43	0.235
chimney	2346	174	0.89	0.679	0.926	0.776
dam	2346	117	0.61	0.427	0.519	0.257
expresswayservicearea	2346	217	0.752	0.447	0.506	0.316
expresswaytollstation	2346	113	0.78	0.752	0.93	0.655
golffield	2346	457	0.712	0.58	0.675	0.434
groundtrackfield	2346	119	0.575	0.193	0.261	0.103
harbor	2346	320	0.738	0.834	0.843	0.624
overpass	2346	342	0.713	0.602	0.672	0.394
ship	2346	5859	0.849	0.936	0.953	0.81
stadium	2346	131	0.757	0.780	0.8	0.584
storagetank	2346	2319	0.834	0.867	0.91	0.651
tennis court	2346	1168	0.872	0.936	0.96	0.857
trainstation	2346	110	0.455	0.364	0.349	0.172
vehicle	2346	4228	0.702	0.546	0.674	0.424
windmill	2346	532	0.847	0.887	0.929	0.53

Table 4: Validation result overview for YOLOv5s

Class	Images	Instances	P	R	mAP50	mAP50-95:
all	2346	18702	0.805	0.797	0.828	0.618
airplane	2346	1094	0.928	0.937	0.965	0.786
airport	2346	138	0.857	0.652	0.761	0.402
baseballfield	2346	590	0.914	0.917	0.95	0.836
basketballcourt	2346	300	0.904	0.913	0.948	0.875
bridge	2346	374	0.602	0.519	0.57	0.341
chimney	2346	174	0.899	0.92	0.95	0.853
dam	2346	117	0.745	0.701	0.705	0.382
expresswayservicearea	2346	217	0.809	0.816	0.869	0.564
expresswaytollstation	2346	113	0.783	0.796	0.86	0.733
golffield	2346	457	0.689	0.698	0.737	0.523
groundtrackfield	2346	119	0.737	0.613	0.648	0.403
harbor	2346	320	0.771	0.863	0.87	0.706
overpass	2346	342	0.726	0.69	0.768	0.52
ship	2346	5859	0.881	0.951	0.905	0.66
stadium	2346	131	0.752	0.855	0.846	0.682
storagetank	2346	2319	0.857	0.888	0.927	0.69
tennis court	2346	1168	0.896	0.947	0.967	0.895
trainstation	2346	110	0.667	0.545	0.595	0.369
vehicle	2346	4228	0.717	0.616	0.722	0.469
windmill	2346	532	0.885	0.912	0.946	0.589

Table 5: Validation result overview for YOLOv5m

Compared to the small and medium model of YOLOv5, all of the classes have higher mAP50 values in the medium model. It can be observed that YOLOv5s has 0.737 means of average precision. On the other hand, YOLOv5m has 0.828. This shows that when the model training time and computational load is considered. Using YOLOv5m is a better choice. These tables show that some of the classes such as tennis court, basketball court, ships and windmill do not differ a lot for the both models. On the other hand, train station, dam, bridge and airport classes increased their mAP values seriously. For example, the train station mAP value increased from 0.349 to 0.595. This shows that using more layers and parameters for the training provides a better solution for DIOR dataset.

6.2. Confusion Matrix

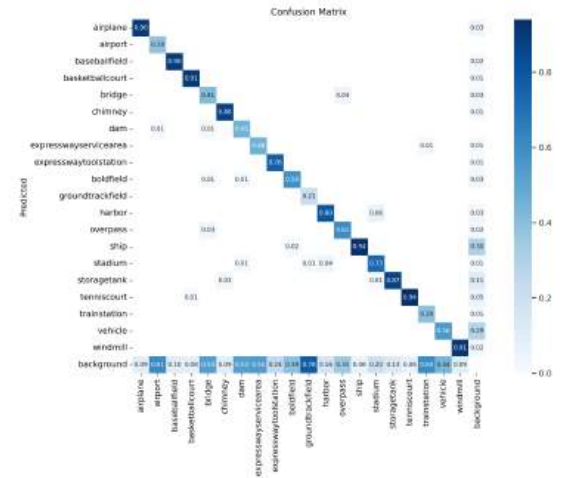


Figure 15: Confusion matrix overview for YOLOv5s

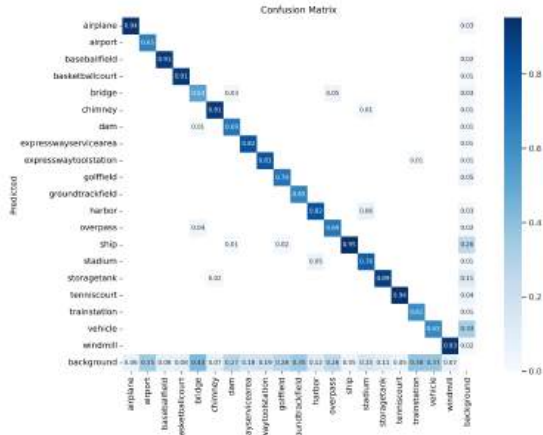


Figure 16: Confusion matrix overview for YOLOv5m

Confusion matrix shows how each of the classes correlates with the background of the pictures. That makes the object detection fail normally. If the background value of the class is high, the related object might not be found by the model. From the given graphs, it can be seen that the YOLOv5m model has better background- object awareness than the YOLOv5s model.

6.3. Testing Images Detection Comparisons



Figure 17: Detected objects of testing subset for YOLOv5s



Figure 18: Detected objects of testing subset for YOLOv5m



Figure 19: Detected objects of testing subset for YOLOv5s



Figure 20: Detected objects of testing subset for YOLOv5m

From the given graphs, it can be observed that ground truth and detection results are almost the same; however, the medium model provides better detection than the small model. It can also be seen that medium and small models sometimes detect some objects that are not given in the ground truth. This shows that the model has good detection ability but the dataset is not reliable for all image samples inside of it.

performance-oriented model such as YOLOv5 by using the diversity of the DIOR dataset and the advantage it provides for different classes. After the training process is completed, the results of the test images are observed in approximately 11 ms for each image in the small model and 17 ms in the medium-sized model. These data show that compared to other object detection techniques such as faster R-CNN, it can provide much faster results in instant object detection projects and products. Considering the fact that R-CNN models can identify objects within 200 ms[15] on the COCO dataset. That shows YOLOv5s and YOLOv5m models are approximately 9x faster than R-CNN models with respect to selected model and dataset. In practical terms YOLO runs a lot faster than RCNN due its simpler architecture. Unlike faster RCNN, it's trained to do classification and bounding box regression at the same time. When a comparison is made between the YOLO models that are used in this project, it is seen that the medium-scale model gives an average of 6 ms later results in the test data, but the success of the results are higher.

From another perspective, YOLO continues to improve its performance and accuracy through internal version updates. The reason why yolov4 and its previous models are different from yolov5 is that yolov5 is built on a powerful and fast infrastructure such as pytorch. The reason for choosing yolov5 in particular is that we believe that the new version of yolov7, which is not yet used by a large enough community, will have limited solutions to errors and problems. Based on this study, we would like to state that our future object detection projects will continue on this basis. Finally, we would like to thank the readers of this report.

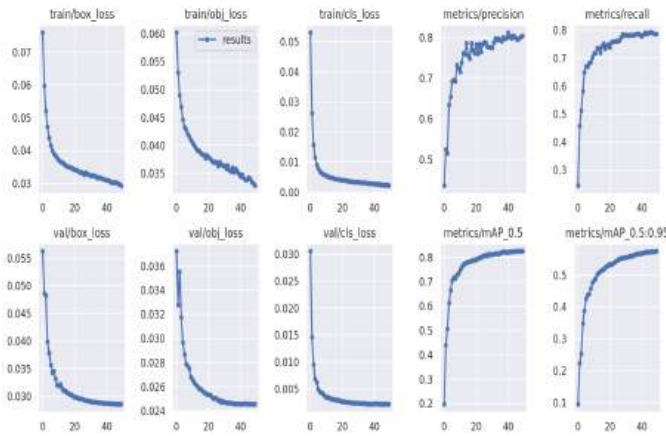


Figure 21: Training metric graph of YOLOv5m

As a result, the YOLO, which stands out with its performance among many different deep learning methods, has improved itself over time and increased its performance and mAP values by producing different versions. In this study, good results were obtained in the mAP values of a

11. REFERENCES

- [1] "ObjectDetectionGuide."
<https://www.fritz.ai/object-detection/>
- [2] "You Only Look Once: Unified, Real-Time Object Detection"
<https://arxiv.org/pdf/1506.02640.pdf>
- [3] "Fast and Accurate Object Detection in Remote Sensing Images Based on Lightweight Deep Neural Network"
<https://www.mdpi.com/1424-8220/21/16/5460#cite>
- [4] "mAP : Evaluation metric for object detection models"
<https://medium.com/axinc-ai/map-evaluation-metric>
- [5] "Object Detection Metrics"
<https://github.com/rafaelpadilla/Object-Detection-Metrics>
- [6] "ultralytics/yolov5"
<https://github.com/ultralytics/yolov5>
- [7] "Microsoft COCO: Common Objects in Context"
[arXiv:1405.032](https://arxiv.org/abs/1405.032)
- [8] "YOLOv5 Basic Models"
https://github.com/yyccR/yolov5_in_tf2_keras
- [9] "Tensorflow: Large-Scale Machine Learning on Heterogeneous Systems"
<https://www.tensorflow.org/>
- [10] "Activation Functions of Neural Networks"
<https://towardsdatascience.com/activation-functions>
- [11] "Arbitrary-Oriented Inshore Ship Detection based on Multi-Scale Feature Fusion and Contextual Pooling on Rotation Region Proposals"
<https://www.researchgate.net/figure/Differences-between-horizontal-bounding-boxes>
- [12] "Anchor Boxes for Object Detection"
<https://stackoverflow.com/questions/70227234/anchor-boxes-for-object-detection>
- [13] "Xu, Renjie & Lin, Haifeng & Lu, Kangjie & Cao, Lin & Liu, Yunfei. (2021). A Forest Fire Detection System Based on Ensemble Learning. Forests"
<https://www.mdpi.com/1999-4907/12/2/217>
- [14] "Object Detection Metrics"
<https://github.com/rafaelpadilla/Object-Detection-Metrics>
- [15] "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"
<https://arxiv.org/pdf/1506.01497.pdf>
- [16] "YOLOv5 Explained and Demystified"
<https://towardsai.net/p/computer-vision/yolo-v5%E2%80%8Aexplained-and-demystified>