EVALUATION OF DIFFERENT YOLO MODELS ON DIOR DATASET

Vigneshwar Karuppiah Ramanathan

M.Sc. Computational Sciences in Engineering
Technische Universität Braunschweig

Amulya Harihara Narayana Rao

M.Sc. Computational Sciences in Engineering
Technische Universität Braunschweig

ABSTRACT

The importance of remote sensing object detection in real-time applications is predominant, particularly in the areas of defense, disaster management and environmental monitoring and the computational resources here are limited for edge devices. Therefore, we need to have a tradeoff between accuracy and speed. This research focuses on evaluation of performance of YOLO models, a well-established single stage detector on DIOR dataset. Our objective is to develop a robust model to efficiently detect various classes. This study evaluates different YOLOv8 and YOLO-NAS models under various hyperparameters and regularization techniques. Furthermore, data augmentation alongside ensemble methods have been employed to increase detection performance. The evaluation metrics we used includes Mean Average Precision (mAP), Recall, Precision, inference time. The results indicate that it's possible to use YOLO for efficient object detection on edge devices with minimum latency.

Index Terms— *Object Detection, Deep Learning, DIOR Dataset, Remote Sensing*

1. INTRODUCTION

Object detection in computer vision is the process of identifying and localizing objects within digital images or videos. This allows us to attain Pixel-level detections, which can be accurately mapped to real-world coordinates. That is, it allows us to assess the geographical position and detected object dimensions. This ability to accurately detect and localize objects in real time has significant real-world implications.

Object detection in remote sensing has wide range of applications across various fields. Despite this application, object detection in remote sensing faces several unique challenges and also opportunities. These includes, variations in lighting conditions, weather, occlusions, diverse angle of view and many more. The important problem for deep learning-based methods is inadequate imagery dataset.

In this study, we utilize the DIOR (Detection in Optical Remote sensing) dataset, a large-scale, publicly available [1]. The DIOR dataset offers a large range of object size variations, holds rich image variations in viewpoint, appearance, illumination, translation, weather, seasons, scales, etc. Another important characteristic is that it has high inter-class similarity and intra-class diversity, thus making it much challenging.

All these complexity impacts the real time performance of the model, especially for time sensitive tasks where even small delay could result in critical consequences. This study aims to investigate and assess effective object detection models which have the potential for real time processing, while being tailored for edge devices with limited computational resources.

Two categories of conventional object detection methods are: one stage and two-stage detectors. Two-stage detectors, such as Faster R-CNN, Mask R-CNN prioritize accuracy over speed. It first generates region proposals then perform classifications and localizations. In this way we attain high precision but at an expense of long processing time, which doesn't serve well in real time applications and limited resources. On the other hand, single stage detectors such as YOLO, SSD and RetinaNet predicts bounding boxes and class labels directly in one pass through the network. This makes these models ideal for real time detection. This enables faster detection with minimal latency on edge devices.

In this work, we evaluate the performance of YOLOv8 and YOLO-NAS models on the DIOR dataset. The aim is to analyze how effective these models in real time applications. we aim to further boost the detection performance of underrepresented classes, addressing key limitations in the dataset. Our experimental results are compared against the baseline metrics reported in the DIOR dataset paper.

2. DIOR DATASET

The DIOR (Detection in Optical Remote Sensing) dataset, introduced by Ke Li et al., is a large-scale, publicly available benchmark for object detection in optical remote sensing imagery.

It contains 23,463 images and 192,472 object instances, covering 20 object classes. Each category includes approximately 1,200 images. The object classes include airplane, airport, baseball field, basketball court, bridge, chimney, dam, expressway service area, expressway toll station, harbor, golf course, ground track field, overpass, ship, stadium, storage tank, tennis court, train station, vehicle, and windmill.

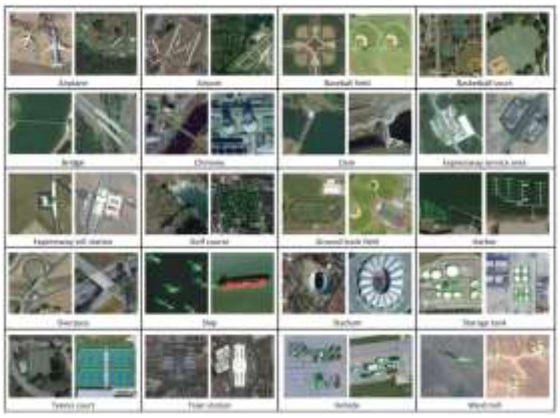


Fig 1: Examples for each class from DIOR Dataset

The images in the dataset are 800×800 pixels in size, with spatial resolutions ranging from 0.5 meters to 30 meters. The dataset was collected from Google Earth by experts in earth observation interpretation and exhibits significant variations in object instance sizes and offers rich image variations with high inter-class similarity and intra-class diversity.

The dataset is divided into two distributions, training-validation data containing 11725 images and test data containing 11728 images. Dataset provides two annotation folders, one containing Horizontal bounding boxes and the second one containing vertical bounding boxes. The annotations are available in PASCAL-VOC format. The horizontal annotation folder is used and have been transformed to YOLO text format for the experimentation.

Fig. 2 reports the number of object instances per class for the train-val dataset. The object classes of ship and vehicle have higher instance counts, while the classes of train station, stadium, dam, golf field, airport and expressway toll station have lower instance counts.

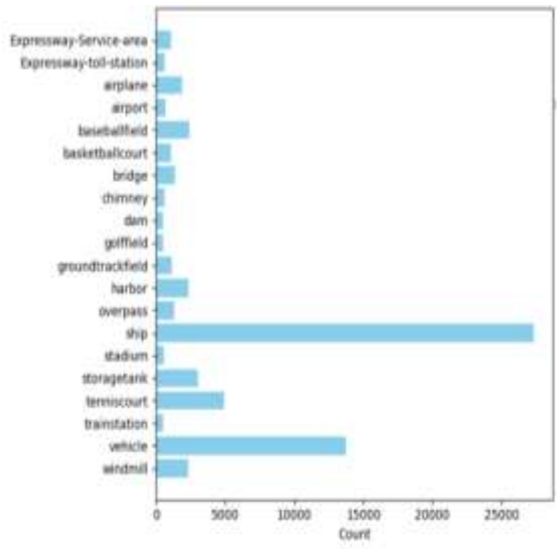


Fig 2: Object Instance Counts in train-val Dataset

Fig. 3 report the same for the test dataset. The object classes of storage tank, vehicle and airplane are present more frequently when compared to the train-val dataset.

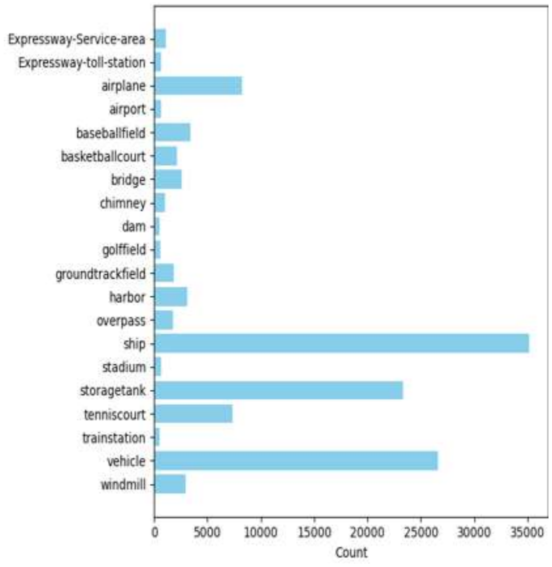


Fig 3: Object Instance Counts in test Dataset

The diversity in object sizes makes this dataset particularly useful for real-world applications. As shown in Fig. 4, the dataset maintains a good balance between small and large-sized instances. Additionally, the substantial differences in object sizes across categories increase the complexity of the detection task. This requires detectors to be highly flexible, as they must be capable of accurately detecting both small and large objects simultaneously.

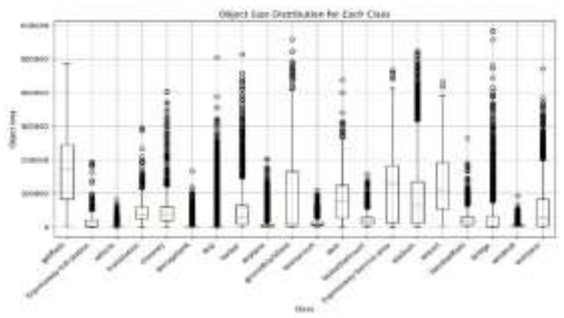


Fig 4: Object Size Distribution for Each Class

The DIOR dataset offers a challenging yet balanced benchmark for object detection tasks in optical remote sensing. Its diverse object sizes and varying category distributions make it an essential resource for evaluating and improving detection algorithms in real-world scenarios.

3. YOLO MODELS

You Only Look Once (YOLO) is a single stage, real time object detection model introduced in the year 2015 in the famous research paper "You Only Look Once: Unified, Real-Time Object Detection" [2]. The object detection problem is framed as a regression task. A single neural network predicts bounding boxes and class probabilities directly from images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

The primary motivation behind YOLO models is their ability to provides a balance between accuracy and speed. This characteristic makes it highly efficient and suitable for applications requiring low latency and real-time performance. With variations in diverse YOLO models, each iterations offers improvements and applies to different challenges.

In this research work, we trained and evaluated two YOLO models: YOLOv8, developed by Ultralytics and YOLO NAS, released by Deci AI. Both models aim to improve on the limitations of its predecessors. A series of experiments were conducted for different protocol on DIOR dataset. These protocols explored different models, hyperparameters and optimization techniques, with the goal of improving detection performance while maintaining real time abilities.

3.1 MODEL ARCHITECTURE

Since 2015, the YOLO family has undergone significant evolution, leading to advancement in various aspects. The first iteration of YOLO was YOLOv1, a simple architecture along with a novel full-image one-shot regression approach. YOLOv1 architecture comprises 24 convolutional layers followed by two fully-connected layers that predict the bounding box coordinates and probabilities. It was pretrained and validated on the ImageNet 2012 dataset. However, while YOLOv1 performed faster than any object detector, the localization error was larger compared with state-of-the-art methods such as Fast R-CNN at that time [3].

Over the years, there were many updates and new network architecture. YOLOv8 was released in January 2023 by Ultralytics [4]. YOLOv8 provided five versions: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l and YOLOv8x. YOLOv8 supports multiple tasks like object detection, segmentation, pose estimation, tracking, and classification.

CSPDarknet53 is the backbone along with an anchor-free approach, which reduces the complexity of bounding box predictions. This version has an improved Non-Maximum Suppression (NMS) speeding it up and making it more reliable especially as far as small items are concerned. There's also improvement for detecting objects at different scales and resolution, by introduction of combined implementation of Feature Pyramid Network (FPN) and Path Aggregation Network (PAN).

Yolov8 was pretrained and evaluated on the COCO dataset with an input image resolution of 640 x 640. CIoU, Distribution Focal Loss (DFL) are used for bounding box loss and binary cross entropy for classification loss. YOLOv8 model's performance and their specifications are defined in Table 1.

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 1. Performance and parameter specification for the five models available for YOLOv8

YOLO-NAS is a state-of-the-art object detection model released by Deci in May 2023. It builds upon the popular YOLO family of algorithms. It combines real-time detection performance with high accuracy, using neural architecture search (NAS) to optimize its structure automatically. This enables YOLO-NAS to achieve superior object detection capabilities while maintaining efficient inference, making it ideal for applications requiring both speed and precision.

AutoNAC enabled the development of the YOLO-NAS architecture and its variants (YOLO-NAS-S, YOLO-NAS-M, and YOLO-NAS-L) by systematically exploring optimal model configurations. It combined core architectural elements from various YOLO variants with Deci's innovative neural components. The result is a set of advanced architectures that leverage the strengths of YOLO while incorporating cutting-edge improvements from Deci's research.

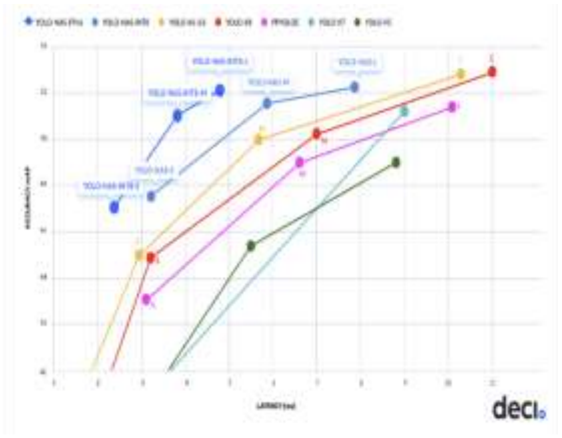


Fig 5: Performance of Yolo-NAS on COCO Dataset

A key differentiating factor if Yolo-NAS is the employment of Quantization. Quantization refers to making the neural network more compact by reducing the bit-representation of numerical values. This process helps to decrease the network's size, making it more efficient for deployment on edge devices. YOLO-NAS employs a hybrid quantization method to mitigate information loss and maintain performance. The hybrid quantization method applies quantization in a non-uniform manner across a network architecture, consideration what areas of the model to apply quantization to and what level of quantization is required. Additionally, the model incorporates an attention mechanism to enhance focus on the parts of the image that contain the target object, improving detection accuracy. [6]

YOLO-NAS was initially trained on the Object 365 dataset, which contains 2 million images across 365 categories. An additional 125,000 images were incorporated through a COCO pseudo-labeled dataset using a semi-supervised technique.[10] The training process utilizes Knowledge Distillation (KD) to reduce computational resources by training a simpler student model to emulate the performance of the original model. Additionally, Distribution Focal Loss (DFL) was applied to modify the focal loss function, helping the model manage variability in target object sizes more effectively. [9]

3.2 BUILDING BLOCKS OF YOLO

The YOLO algorithm is designed around several key components, which makes it one of the popular object detection models. The following components are used in the above-mentioned YOLO models:

3.2.1 Anchor Free Approach

This approach simplifies the model by discarding predefined anchor boxes. Instead, the model predicts fewer boxes by directly learning bounding box coordinates. This results in faster detection and reduced computational complexity.

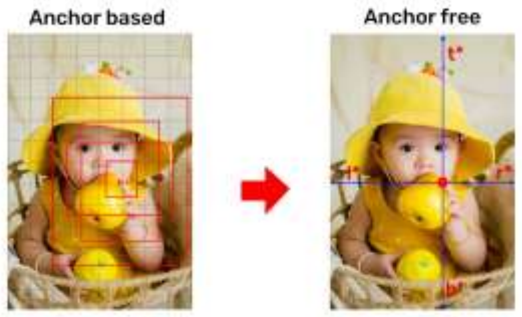


Fig 6. Comparison between anchor-based and anchor-free object detection approaches [5].

3.2.2 Feature Pyramid Network (FPN)

FPN enables multi scale detection, that is, it allows YOLO to detect objects at different scales by combining low-resolution and high-resolution feature maps This is particularly useful for detecting small objects or objects that appear at varying scales within an image.

This enables the model to detect small, medium, and large objects more effectively. The architecture allows for efficient propagation of detailed features and semantic information through the network

3.2.3 Network Architecture Search

Network Architecture Search (NAS) is a sub-field within automated machine learning (ML) that focuses on the development of models capable of automatically designing and configuring deep neural network architectures. This approach allows for the efficient discovery of high-performing architectures by considering various factors such as accuracy, speed, and resource efficiency. A key advantage of using NAS techniques is the ability to streamline machine learning processes through automation, significantly reducing the human effort and time required in research and development. Examples of architectures designed or optimized using NAS include MobileNetV3 and EfficientDet. However, due to the resource-intensive nature of NAS, its use has traditionally been limited to a smaller number of organizations with extensive computational capabilities. One example of a proprietary NAS technology is AutoNAC, developed by Deci, designed to efficiently explore a wide range of architectural configurations and structures. It systematically evaluates various block types, the number of blocks, and channel allocations to optimize neural network architectures.[9]

3.2.3 Non-Max Suppression (NMS)

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 [3]. 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. Algorithm 1 describes the procedure. Fig 2 shows the typical output of an object detection model containing multiple overlapping bounding boxes and the output after NMS.

Input: Set of predicted bounding boxes **B**, Confidence scores ***S***, IoU threshold ***τ***, Confidence threshold ***T***
Output: Set of filtered bounding boxes ***F***

```
1.  Initialize $F \leftarrow \emptyset$
2.  Filter the boxes: $B \leftarrow \{b \in B \mid S(b) \geq T\}$
3.  Sort the boxes $B$ by their confidence scores
    in descending order
4.  while $B \neq \emptyset$ do
5.      Select the box b with the highest
        confidence score
6.      Add b to the set of final boxes,
        $F: F \leftarrow F \cup \{b\}$
7.      Remove b from the set of boxes
        $B: B \leftarrow B - \{b\}$
8.      for all remaining boxes $r$ in $B$ do
9.          Calculate the IoU between $b$ and $r$:
            $iou \leftarrow IoU(b,r)$
10.         if $iou \geq \tau$ then
11.             Remove $r$ from the set of boxes
                $B: B \leftarrow B - \{r\}$
12.         end if
13.     end for
14. end while
```

Algorithm 1. NMS Algorithm

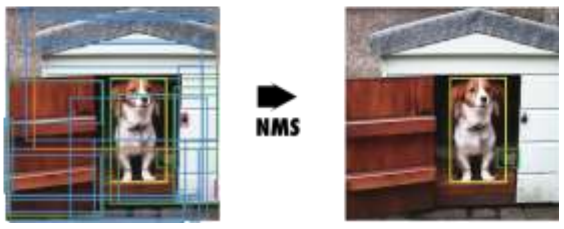


Fig 7. NMS.

a) Shows the typical output of an object detection model containing multiple overlapping boxes.
b) Shows the output after NMS

4. METRICS AND DATA AUGMENTATION

For evaluating object detection models, several key metrics are used, with **Mean Average Precision (mAP)** being the primary metric for comparing performance of different models and tuning strategies. mAP calculates the average precision across all classes and IoU thresholds, providing an overall measure of the model's performance.

Intersection over Union (IoU) measures how well the predicted bounding boxes align with ground truth boxes. Based on IoU, metrics such as True Positive (TP) and False Positive (FP) are derived.

Precision measures the accuracy of the model's positive predictions, which indicates how well the model distinguishes true objects from false positives. **Recall** measures the model's ability to detect all relevant objects in the dataset.

mAP@50 evaluates the model's performance at an IoU threshold of 0.50, while **mAP@50-95** provides a more comprehensive evaluation by averaging AP scores across IoU thresholds from 0.50 to 0.95. These metrics give a detailed view of the model's ability to accurately detect objects across various classes and thresholds.

Data augmentation was applied to address under-represented classes compared to the test dataset. The under-represented classes included storage tank, vehicle, and airplane. Augmentation techniques were implemented using the albumentations library and the pipeline included horizontal flip (p=0.5), random brightness and contrast adjustments (p=0.2), random snow (p=0.2), and random fog (p=0.2). A total of 3,341 images were transformed using these augmentations, and the augmented images along with their corresponding labels were integrated into the full dataset.

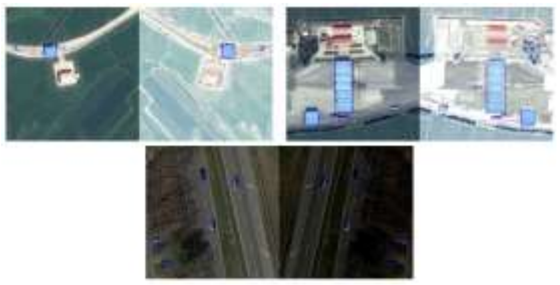


Fig 8: Examples with augmented images

5. EXPERIMENTAL SETUP AND RESULTS

This project is divided into two distinct parts, each focusing on different object detection models and methodologies: one assesses YOLOv8 performance while the other deals with Neural Architecture Search (NAS) using YOLO-NAS. Separate experimental protocols were employed for each model, The specific methodologies, experimental setups and results will thus be described in subsequent sections.
The dataset distribution used in this work was split into training, validation, and test sets based on the distribution followed in DIOR dataset paper. The dataset contains 11,738 images in the test set, which was used exclusively for inference. The remaining 11,725 images were used as the training and validation set, with 80% allocated for training and 20% for validation. This split was essential for the model to train on a significant portion of the dataset while having enough validation data to monitor performance during training.

5.1 YOLOv8 Initial Experimental Setup

Initial set of experiments focused on training the model with different learning rates, optimizers and model sizes to determine the best configuration. In total, nine experiments were conducted including a sample run. The models trained includes the Nano and Medium variant of YOLOv8 with Adam and Stochastic Gradient Descent (SGD) optimizers along with learning rate of 0.01 and 0.001.

The model showed consistent improvement over time, with decreasing loss curves for bounding box, classification and DFL losses across training and validation. Table 2 represents the model performance during inference, showcasing the model's generalization ability on unseen data—an essential aspect of any object detection task.

Experiment 07 achieved a mAP@50 of 0.806 and mAP@50-90 of 0.594, with a precision of 0.851 and recall of 0.732. These results indicate that the model effectively generalized when applied on test set without overfitting.

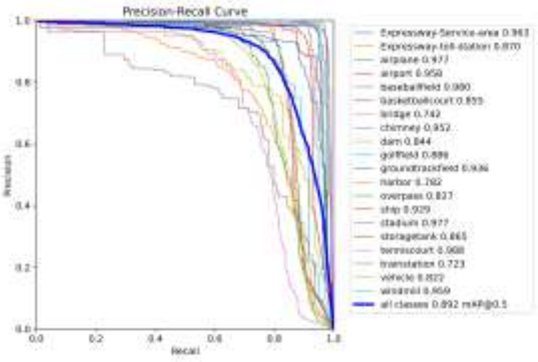
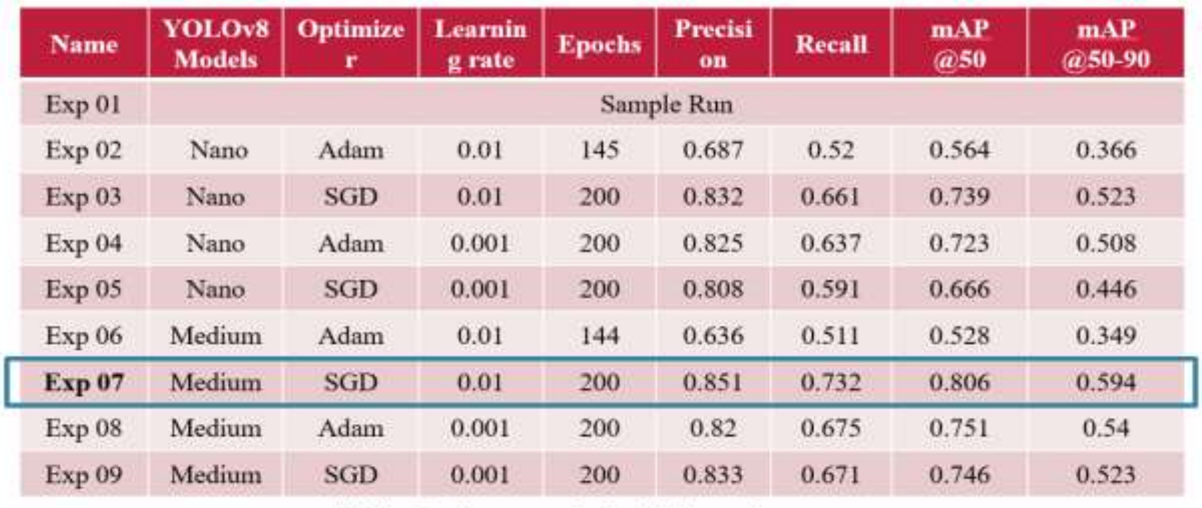


Fig 9. Precision-Recall Curve

The **Precision-Recall Curve** in Figure 9 highlights the individual class performances. While the overall performance was robust, some object classes - **bridge**, **dam**, **overpass**, **train station**, and **vehicle** showed relatively lower performance in the above Precision-Recall curve with low individual Average Precision (AP). These classes are also underrepresented classes in the training set. In the following section we focus on techniques on improving these challenging and underperforming classes.



Name	YOLOv8 Models	Optimizer	Learning rate	Epochs	Precision	Recall	mAP @50	mAP @50-90
Exp 01	Sample Run							
Exp 02	Nano	Adam	0.01	145	0.687	0.52	0.564	0.366
Exp 03	Nano	SGD	0.01	200	0.832	0.661	0.739	0.523
Exp 04	Nano	Adam	0.001	200	0.825	0.637	0.723	0.508
Exp 05	Nano	SGD	0.001	200	0.808	0.591	0.666	0.446
Exp 06	Medium	Adam	0.01	144	0.636	0.511	0.528	0.349
Exp 07	Medium	SGD	0.01	200	0.851	0.732	0.806	0.594
Exp 08	Medium	Adam	0.001	200	0.82	0.675	0.751	0.54
Exp 09	Medium	SGD	0.001	200	0.833	0.671	0.746	0.523

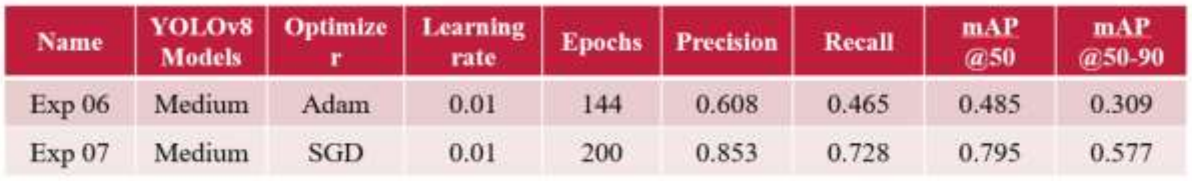
Table 2. Inference metrics for initial experiment setup

During model training, Experiment 07 emerged as one of the top-performing models. The key performance metrics for Experiment 07 during training include a precision of 0.886, recall of 0.749, and a mean Average Precision (mAP) of 0.892 for mAP@50 and 0.713 for mAP@50-90. The validation set metrics are used for selecting the optimal hyperparameters across different model.

5.2 YOLOv8 Tuning Experimental Setup

The first part of tuning involves applying the regularization technique of data augmentation on training dataset. This was done to address the underperforming classes and generalize better. The data augmentation was conducted for the best and worst performing experiments, **Experiment 06** and **Experiment 07**.

The model configuration of these two experiments were used for model training. During inference, in Experiment 07 there's a slight drop in mAP@50 from 0.806 to 0.795 and mAP@50-95 from 0.528 to 0.485. Although the mAP decreased after augmentation, precision and recall of the model remained consistent.



Name	YOLOv8 Models	Optimizer	Learning rate	Epochs	Precision	Recall	mAP @50	mAP @50-90
Exp 06	Medium	Adam	0.01	144	0.608	0.465	0.485	0.309
Exp 07	Medium	SGD	0.01	200	0.853	0.728	0.795	0.577

Table 3. Inference metrics after augmentation

In the second part of the tuning process, ensemble methods were employed to further improve model performance. Ensemble learning combines predictions from multiple models to enhance the overall detection accuracy by leveraging the strengths of each individual model. the predictions from two independently trained YOLOv8 models were combined using a concatenation strategy followed by **Non-Maximum Suppression (NMS)**. This method ensured that redundant bounding boxes were removed, and only the most confident predictions were retained. Two ensemble configurations were tested and the choice of these specific ensembles was guided by the performance of the individual experiments.

In Ensemble 1, Experiment 3 and Experiment 7 was chosen for their complementary strengths and diversity in architecture. Here our focus was to provide complementary strengths without being too similar. In Ensemble 2, the top 2 performing classes in test dataset was selected. The performance of the two ensembles was evaluated across three different IoU thresholds: **0.4**, **0.5**, and **0.6**.

Ensemble Model	IoU Threshold	Precision	Recall	mAP 50
1	0.4	0.6949	0.733	0.48
2	0.4	0.6944	0.7362	0.477
1	0.5	0.6961	0.7348	0.483
2	0.5	0.6956	0.7382	0.482
1	0.6	0.6965	0.7359	0.486
2	0.6	0.6958	0.7393	0.485

Table 4. Inference metrics of ensemble methods

There's no significant improvement in model performance, as measured by mAP in both augmentation and ensemble approach. Therefore, advanced augmentation techniques or class specific tuning might be required. In the ensemble prediction, we might need to experiment with different model combinations or use different ensemble techniques, such as Weighted Box Fusion (WBF), Soft NMS, etc.

5.3. YOLO-NAS Experimental Setup

For our experiments, we utilized pre-trained weights available in SuperGradients, Deci's library. Fine-tuning of the model was conducted using the Trainer provided by the SuperGradients library. We employed a dataloader from SuperGradients, specifically the COCO-detection-YOLO-format dataloader, for our project. Annotations were converted from the PASCAL-VOC format to YOLO text format to ensure compatibility.

The training parameters for our models were set as follows: the maximum number of epochs was defined as 50. We utilized the PPYoloELoss function as the loss criterion to guide the model's optimization. Two different optimizers, Adam and SGD, were employed to train the YOLO-NAS-S, YOLO-NAS-M, and YOLO-NAS-L models, each with the aim of evaluating performance across different optimization strategies. The metric monitored during training was mean Average Precision (mAP) at the intersection over union thresholds of 0.50 to 0.95. Learning rate scheduling was managed using a cosine annealing strategy to adjust the learning rate throughout the training process.

Training metrics were continuously logged and monitored using Weights and Biases, providing detailed insights into the model's performance and progress. After training, the models were evaluated on the test set to assess their final performance and effectiveness.

Table 5 summarises the performance metrics of various YOLO-NAS Models trained with Adam and SGD optimizers after epochs. Model trained with SGD consistently outperforms the models trained with Adam optimizer across all model sizes in terms of mAP, F1 Score, Precision, and Recall.

Model	mAP	F1	Precision	Recall
Yolo-NAS S Adam	0.45864	0.05346	0.02857	0.62784
Yolo-NAS S SGD	0.4961	0.07475	0.0405	0.64904
Yolo-NAS M Adam	0.45846	0.05637	0.03031	0.62941
Yolo-NAS M SGD	0.54484	0.086	0.04662	0.68706
Yolo-NAS L Adam	0.46853	0.05881	0.03166	0.63515
Yolo-NAS L SGD	**0.54588**	**0.09203**	**0.05013**	**0.69102**

Table 5. Run Summary after 50 Epochs

Table 6 represents the model performance on Test Data. The Yolo-NAS L (Large) model with SGD achieved the highest performance metrics, indicating its superior capability in object detection tasks among the tested configurations.

Model	mAP	F1	Precision	Recall
Yolo-NAS S Adam	0.42318	0.06944	0.03884	0.56913
Yolo-NAS S SGD	0.47623	0.08584	0.04825	0.60808
Yolo-NAS M Adam	0.42106	0.06771	0.03775	0.56921
Yolo-NAS M SGD	0.52334	0.09432	0.05262	0.64669
Yolo-NAS L Adam	0.43365	0.06817	0.03791	0.58749
Yolo-NAS L SGD	**0.53234**	**0.10027**	**0.05641**	**0.65682**

Table 6. Model Performance on Test Data

6. EVALUATION AND CONCLUSION

In this research, YOLOv8 and YOLO-NAS models are compared with results from the benchmark paper for DIOR dataset. The best overall mAP@0.50 is reported as 66.1% for RetinaNet and PANet, while YOLOv3 follows with 57.1%. There is good performance on specific categories of YOLOv3 like airplane (72.2%), ship (87.4%), storage tank (68.7%), vehicle (87.3%) and tennis court (48.3%), but it struggles to perform well in other more difficult classes such as bridge or train station.

Our experiments show that YOLOv8 Medium (Experiment 07) significantly outperformed YOLOv3, achieving an overall mAP@0.50 of 80.6%. The main specific classes that improved include airplane (86.9%), ship (92.3%), and vehicle (93.6%). However, similar to YOLOv3, lower performance was seen in the bridge and train station classes and remain challenging classes.

On the other hand, YOLO-NAS Larger had a mAP value of 53.23% placing it second best in terms of storage tanks, and tennis courts overall still lags behind YOLOv8 implying more optimizations are necessary. In conclusion, YOLOv8 has shown great progress compared to YOLOv3 especially when it comes to high precision areas; however, there is a possibility for its improvement on some difficult object classes within specific limitations.

To assess real-time applicability, a video was prepared using the test data. The predictions were run using the computing cluster, less than 25 ms/frame was the predicted time by the model hence proving that it can be used for real-time object detection tasks. The prediction time on an Intel i5 CPU was under 200 ms/frame suggesting that even with limited hardware, the models could still give reasonably fast inferences.

7. References

[1] Object Detection in Optical Remote Sensing Images: A Survey and A New Benchmark https://arxiv.org/abs/1909.00133

[2] Real-Time Flying Object Detection with YOLOv8 https://arxiv.org/abs/2305.09972

[3] You Only Look Once: Unified, Real-Time Object Detection https://arxiv.org/abs/1506.02640

[4] Juvan R Terven et. Al. "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS" https://arxiv.org/html/2304.00501v6

[5] https://learnopencv.com/yolox-object-detector-paper-explanation-and-custom-training/

[6] Ashish Vaswani et. Al. "Attention Is All You Need" https://arxiv.org/abs/1706.03762

[7] https://github.com/Deci-AI/super-gradients

[8] https://roboflow.com/model/yolo-nas

[9] https://richmondalake.medium.com/yolo-nas-uncovered-essential-insights-and-implementation-techniques-for-machine-learning-engineers-87ee266b37f6

[10] https://medium.com/codex/decis-yolo-nas-next-generation-model-for-object-detection-8ccb7f2013a7