Iraqi Journal of Science, 2025, Vol. 66, No. 10, pp: 4545 – 4565
DOI: 10.24996/ijs.2025.66.10.39

ISSN: 0067-2904

An Optimized Deep Learning Model for Tiny Object Detection in UAV Imaging

Wael Yahya Yaseen*, Azal Monshed Abid, Sawsen Abdulhadi Mahmood
Department of Computer Science, College of Education, Mustansiriyah University, Baghdad, Iraq

Received: 30/6/2024 Accepted: 2/10/2024 Published: 30/10/2025

Abstract

Recent advancements in deep learning models-based Unnamed Aerial Vehicle (UAV) object detection technologies have garnered significant interest in smart cities. The tiny object detection task is still a crucial challenge in research due to variant image resolution and training sample size. The primary aim of this paper is to realize precision and model generalization in multi-scale object detection tasks. An optimization of the YOLOv8 (You Only Look Once version 8) deep learning model was carried out to detect 16 classes of tiny objects in UAV imagery data with the assistance of the transfer learning technique. The optimization method's workflow consists of two main procedures; the first procedure aimed to fine-tune the YOLOv8 model's hyperparameters and adopted the Rectified Linear Unit (ReLU) activation function in the model architecture instead of the Sigmoid Linear Unit (SiLU) activation function for feature map generation. Afterword, the fine-tuned YOLOv8 model is optimized further by an open-source optimization workspace. Open Visual Inference & Neural Network Optimization (OpenVINO) to accelerate the training/inference performance along with getting more accurate detection of tiny objects in the input imagery samples. The proposed framework's performance evaluation was conducted using the Dataset for Object Detection in Aerial Images DOTA-v1.5 dataset. The DOTA dataset has been augmented and balanced to generate a customized dataset to mitigate the effect overfitting problem and get better detection accuracy. The results of the experiment showed a significant improvement in small object detection, achieving a 23.67% increase in inference speed while maintaining a higher detection accuracy.

Keywords: Deep learning , Oobject detection , YOLOv8 , hyperparameters tuning , ReLU activation function , OpenVINO.

نموذج تعلم عميق محسّن لاكتشاف الأجسام الصغيرة في صور الطائرات بدون طيار

وائل يحيى ياسين*, ازل منشد عبد, سوسن عبد الهادي محمود
قسم علوم الحاسبات , كلية التربية, الجامعة المستنصرية, بغداد, العراق

الخلاصة

في الآونة الأخيرة، اكتسب التطور الكبير في تقنيات الكشف عن الأجسام في الطائرات بدون طيار بأستعمال نماذج التعلم العميق اهتمامًا واسع النطاق في المدن الذكية. لا تزال مهمة الكشف عن الأجسام الصغيرة تشكل تحديًا حاسمًا في البحث بسبب دقة الصور المتغيرة وحجم عينات التدريب. أن الهدف الأساسي من هذه الورقة هو تحقيق الدقة وتعميم النموذج في مهمة الكشف عن الأجسام متعددة المقاييس. تم إجراء تحسين لنموذج التعلم العميق YOLOv8 للكشف عن 16 فئة من الأجسام الصغيرة في بيانات صور الطائرات بدون طيار

*Email: waeelyahya@uomustansiriyah.edu.iq

بمساعدة تقنية التعلم بالنقل. يتألف سير عمل طريقة التحسين من إجراءين رئيسيين؛ يهدف الإجراء الأول إلى ضبط المعلمات الفائقة لنموذج YOLOv8 واستعمال دالة تنشيط ReLU في بنية النموذج بدلاً من دالة تنشيط SiLU لتوليد خرائط الميزات. بعد ذلك، تم تحسين نموذج YOLOv8 المعدل بدقة أكبر من خلال تطبيق OpenVINO لتسريع أداء التدريب / الاستدلال جنبًا إلى جنب مع الحصول على كشف أكثر دقة للأجسام الصغيرة في عينات الصور المدخلة. تم تحقيق تقييم أداء الإطار المقترح باستعمال مجموعة بيانات DOTA المتاحة. تم تعزيز مجموعة بيانات DOTA وموازنة هذه المجموعة لتوليد مجموعة بيانات مخصصة للتخفيف من مشكلة الإفراط في التجهيز والحصول على دقة اكتشاف أفضل. أظهرت نتائج تجارب النتائج تحسنًا في اكتشاف الأشياء الصغيرة مع زيادة سرعة الاستدلال بنسبة 23.67% والحفاظ على دقة اكتشاف أعلى.

1. Introduction

The identification and localization of small objects in optical images and videos pose a formidable challenge. Contemporary generic object detection methods, despite being state-of-the-art, often struggle to achieve accurate results in this context. The presence of small objects in real-world scenarios is frequently attributable to a significant camera-object distance. Due to their limited spatial coverage in the input image, which typically accounts for less than 5% of the total area, the information gleaned from these diminutive regions may lack the richness required for effective decision-making. The surge in popularity of object detection methods can be attributed to the advancements in deep learning and the enhanced computational capabilities of GPUs, facilitating the faster and more efficient training of deep neural networks (DNNs) in recent years [1].

The significance of small object detection (SOD) in machine vision stems from two primary reasons. The first is that many real-life situations need to find things that are far away. The second is that SOD is very hard to use because images of small objects tend to be noisy, blurry, and not very informative [2]. Deep learning methodologies [3] for object detection can be categorized into two principal types: two-stage object detectors and one-stage object detectors. Two-stage object detectors, exemplified by Faster R-CNN [4], delineate the detection process into two sequential stages: the region proposal stage and the detection stage. The former is responsible for generating regions where objects may be present, producing both the spatial locations of these regions and an object score (binary, indicating the presence or absence of objects). Subsequently, in the detection stage, candidate region proposals undergo classification into different classes, yielding class probabilities and, optionally, refined region locations. While two-stage detectors achieve state-of-the-art performance, their operational speeds are typically sluggish [5]. In contrast, one-stage object detectors, exemplified by YOLO [6] and SSD [7], consolidate region proposal and detection into a singular deep neural network. Utilizing pre-defined anchors with diverse scales and ratios densely tiled on the image, initial regions are determined.

- The detectors then identify regions likely to contain objects. Despite their faster processing speeds, one-stage detectors generally exhibit less accuracy compared to their two-stage counterparts. Research was conducted on both one-stage and two-stage detectors. The findings indicated that, on average, one-stage detectors exhibit greater speed, while two-stage detectors tend to be more accurate [8]. This research work aims to optimize the pre-trained YOLOv8 deep learning for tiny object detection tasks in UAV (Unnamed Aerial Vehicle) images with higher performance and accurate detection. The main contributions of this research could be summarized as follows:

- Optimizing the YOLOv8 deep learning model for multi-scale object detection using two optimization schemes and a custom dataset.

- Fine-tuning of the model's hyperparameters during the training workflow to improve the model performance in terms of detection accuracy and obtain a more generalized model.
- Adopting the open source optimization workspace OpenVINO (Open Visual Inference & Neural Network Optimization) over the fine-tuned model to accelerate the training/inference performance along with getting higher accurate detection of tiny objects, including 16 classes in the input imagery samples.
- Generate a custom and balanced dataset from the public and original DOTA (Dataset for Object Detection in Aerial Images Version 1.5) dataset to employ in the training, validation, and inference mods of the optimized YOLOv8 model.

The rest of this paper is organized as follows: Section 2 provides a review of the related work in the same field of small object detection using deep learning and UAV technology. Section 3 describes the proposed methodology for developing and training the proposed deep learning model. Section 4 presents and discusses the experimental results. This section provides a detailed description of the dataset used, the performance evaluation metrics, and the obtained results. Finally, Section 5 presents the conclusions and discussion.

2. Related Work

2.1 UAVs Objects Detection

Practically, the object detection tasks require accurate training models using vast quantities of labeled samples. The training process for small object detection in aerial imagery has become more complex due to the varying sizes of objects in the acquired images. Different satellite operators use different hardware and image processing software, making it challenging to obtain a perfectly labeled dataset suitable for training a model capable of object detection anywhere. Many studies have suggested different models and adopted variant methods to build the best possible object detection model with a limited training dataset and minimal time consumption.

For example, Shulin Li et al. [9] designed a neural network to fuse multi-layer feature extraction for vehicle detection in UAV traffic videos. Their network has a convolutional layer to reduce feature size, a deconvolutional layer to increase feature size, and several fully connected layers for detection, according to the VGG-16 model. Their method also combined the detection and tracking for network optimization and higher detection speed. The authors built a self-collected UAV traffic video dataset, and the best accuracy obtained was 67.4% mAP. The model's generalizability is limited due to the self-collected dataset. The complexity of combining detection and tracking could affect real-time performance.

Alexey Bochkovskiy et al. [10] presented an object detection system designed for real-time application using the YOLOv4 model, which incorporated the CSPDarknet53 backbone, SPP and PAN in the neck, and YOLOv3 in the head. Their model combined the Bag of Freebies (BoF) and Bag of Specials (BoS) techniques for both the backbone and the detector. The obtained results were 43.5% AP on the MS COCO dataset. However, the low evaluation metric AP on the MS COCO dataset may render their work less applicable in real-time compared to current models.

Zhuang-Zhuang Wang et al. [11] combined the image super-resolution method with a feature texture transfer (FTT) module. For multi-scale feature fusion, dense blocks were used in the Darknet53 model, as well as a combination of SPPnet and PANnet. The detection accuracy, based on mean average precision (mAP), was 89.91. The self-built dataset for small-target detection in university classrooms served as the basis for the model evaluation. Practically, the reliance on a self-built dataset makes it challenging to assess the model's performance on standard benchmarks.

Onur Can Koyun et al. [12] proposed a two-stage object detection framework called "Focus-and-Detect" to address the challenging issue of small object detection in aerial images. Their proposed system is divided into stages for focus and detection. It uses a Gaussian Mixture Model (GMM) to create focus areas and an incomplete box suppression (IBS) method to avoid truncation effects. Their proposed framework was evaluated using the VisDrone 2021 dataset and obtained an average precision (AP) of 54.16%. However, the two-stage framework increases the computational complexity, which could hinder its practical use in real-time scenarios.

Shiyi Tang et al. [13] introduced an improvement of the YOLOv5 model to enhance models' performance and reduce computation costs by using an additional prediction head (SODH) to focus on small objects. Furthermore, the researchers adopted the involution block (CFFI) between the backbone and the neck to enhance channel information in feature maps. The VisDrone 2019 dataset was used, achieving 36.95% mAP. They achieved lower detection accuracy compared to our framework, indicating that the improvements may not significantly enhance the performance of small object detection tasks.

Xiaohui Guo et al. [14] combined two main models, the cross-scale aggregation module (CAM) and the dual relationship module (DRM), to gather semantic data and improve the data that had already been gathered. The method's effectiveness was evaluated using two datasets, VisDrone2021-DET and SeaDronesSeeV2, achieving 16.10% AP and 45.31% AP, respectively. They obtained the AP metric, which is relatively low, particularly on VisDrone 2021-DET, indicating that the model may struggle with detecting small objects or complex scenes.

Baokai Liu et al. [15] used the dilated convolution mish (DCM) module in the YOLOv3 network's backbone network to improve feature expression by combining feature maps with various receptive fields. The datasets used are the MS COCO2017, VOC2007, and VOC2012 datasets, achieving 88.76% mAP, 90.02% mAP, and 47.1% AP, respectively. The higher computational costs make this research work less suitable for resource-constrained environments.

Xiang Yuan et al. [16] have designed a two-stage framework, including a coarse-to-fine RPN (CRPN) and feature imitation (FI) branch. In this way, they aimed to enhance the model's response to current instances and construct a dynamically optimized feature bank for imitation learning. The datasets used for model evaluation were SODA-D and SODA-A, which specified 30.75%AP and 34.4%AP, respectively. However, the relatively low AP values suggested that the model might not generalize well to diverse datasets.

Shaoyu Chen et al. [17] created a two-step light-weight detection method that includes high-resolution feature maps, sparsely connected convolution (SCConv) to reduce the amount of work that needs to be done, early-stage feature enhancement in the backbone, and a way to fix the issue of feature misalignment. The backbone was MobileNetV3, which increased the computation in early stages to preserve detailed information crucial for small object recognition. Using the COCO dataset, they achieved 35.5% AP.

Aduen Benjumea et al. [18] modified the YOLOv5 model in a planned manner by changing the backbone with ResNet and DenseNet, the neck architecture, the number of anchors, the learning rate, the model width, and the redirection of feature maps. The obtained detection accuracy was 0.9605% mAP when adopting a dataset of annotated cones from the perspective

of an autonomous racing car. The extensive modifications might not translate well to other datasets or general object detection tasks.

Chien-Yao Wang et al. [19] introduced the programmable gradient information (PGI) method to address feature extraction and spatial transformation issues in deep networks. They also proposed a lightweight model named generalized efficient layer aggregation network (GELAN) that leverages gradient path planning and depth-wise convention operators. Their model validation was based on the MS COCO dataset, which specified 0.72% mAP-50. However, the complexity introduced by gradient path planning may result in longer training times.

Ao Wang et al. [20] presented the YOLOv10 model, which employs consistent dual assignments for NMS-free training and a holistic efficiency-accuracy model design. YOLOv10 achieved state-of-the-art results across various model scales, demonstrating significant improvements over previous versions. Specifically, YOLOv10-X achieved an AP of 54.4% on the COCO dataset. They achieved a reasonable detection accuracy of AP, which is considered a significant improvement, but the gains may not be enough to justify upgrading from earlier YOLO versions.

2.2 YOLOv8 Deep Learning Model

- The series of YOLO models represents a breakthrough for object detection systems in real-time, which employed a single-stage detector. The main concepts of an object detection network consist of three main components: the backbone, neck, and head. The backbone often functions in conjunction with a convolutional neural network (CNN) to extract essential features from input images of varying scales. The neck layers are refining these features to enhance spatial and semantic information. The head layers use these filtered features to produce inferences for object detection. The introduction of the YOLOv8 model marks a notable advancement in object detection and localization technologies. YOLOv8 incorporates pyramid network architecture, which enabled the model to effectively identify the multi-scale objects in the input imagery data. In contrast to its predecessors, YOLOv8 maintains a foundation in CNN construction but introduces significant enhancements, including:
- Adoption of a new and efficient Darknet-53 backbone.
- Implementation of the structure of FPN+PAN in the neck architecture to improve feature combinations obtained from various levels of the model backbone.
- Replace the head architecture with PANet [21] structure to address occlusion and multi-scale sample issues.
- Integrating supervised and unsupervised learning techniques in the training procedure.

The main concept of the YOLOv8 model is an anchor-free model with a decoupled head, which allows an independent processing of classification, regression, and object detection tasks. This design optimizes each branch's focus on its specific task, ultimately improving the model's overall accuracy. In the output layer, a sigmoid activation function is used for object detection, indicating the likelihood of an object being present within the bounding box. Class probabilities are determined using the softmax function. YOLOv8 uses CIoU [22] and DFL [23] for bounding box loss and binary cross-entropy for classification loss. These methods have been shown to improve the ability to detect objects, especially smaller ones [23] [24]. The primary workflow of the YOLOv8 model involves dividing the input image into a grid of units or cells. For each unit, YOLOv8 estimates a set of bounding boxes, along with the class probabilities for each bounding box. Figure 1 illustrates the main architecture of the YOLOv8 deep learning model.

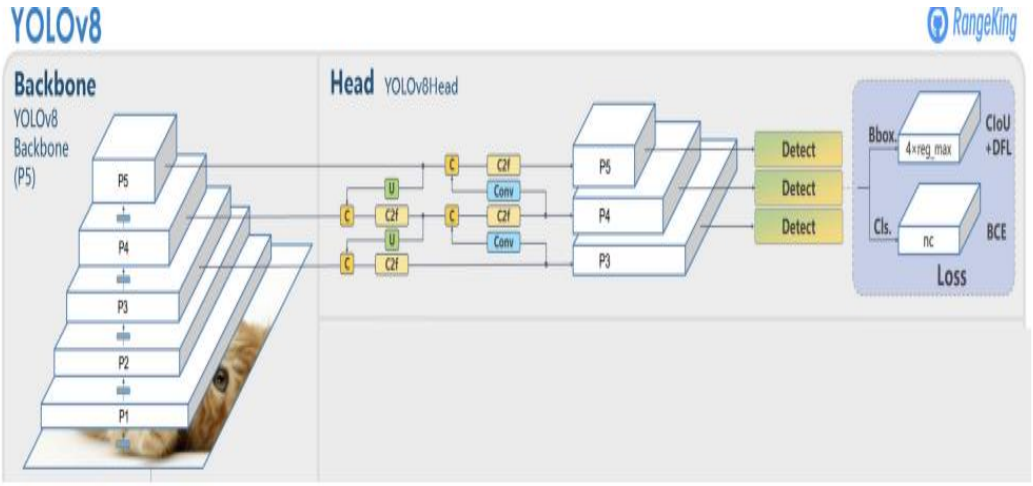

Figure 1: YOLOv8 architecture [24]

3 Research Methodology

The main objective of this research is to optimize the pre-trained YOLOv8 deep learning model to be able to detect tiny objects in UAV images with accurate and accelerant inference. Tiny object detection tasks in UAV images with real-time resides a challenging problem due to discrepancy of object sizes, aspect ratios, deduction speed, and noisy environment. To address these issues and find a reasonable solution, an optimized YOLOv8 deep learning model has been introduced, implemented, and experimented with in this research. The optimization workflow aimed to effectively detect and categorize specific types of small objects in the acquired images, as well as speed up the inference operation to make the model more suitable for real-world applications, taking into account the challenge of a low proportion of pixels associated with the target object in the input image. Furthermore, the framework addresses and achieves the detection of small objects that are not prominently visible in the training samples. The first phase of the proposed framework includes generating a custom dataset by preparing an augmented and balanced dataset, then converting the dataset samples into YOLO format for training, validation, and modeling. Afterward, the pre-trained-YOLOv8 model is trained on the custom dataset, along with hyper-parameter adjustments. To do this, the SiLU (sigmoid linear unit) activation function [25] has been replaced by the ReLU (rectified linear unit) activation function [26] to enhance feature map generation. Finally, we further optimize the generated model using the OpenVINO [27] open source optimization workspace. Figure (2) illustrates the main structure of the proposed framework.

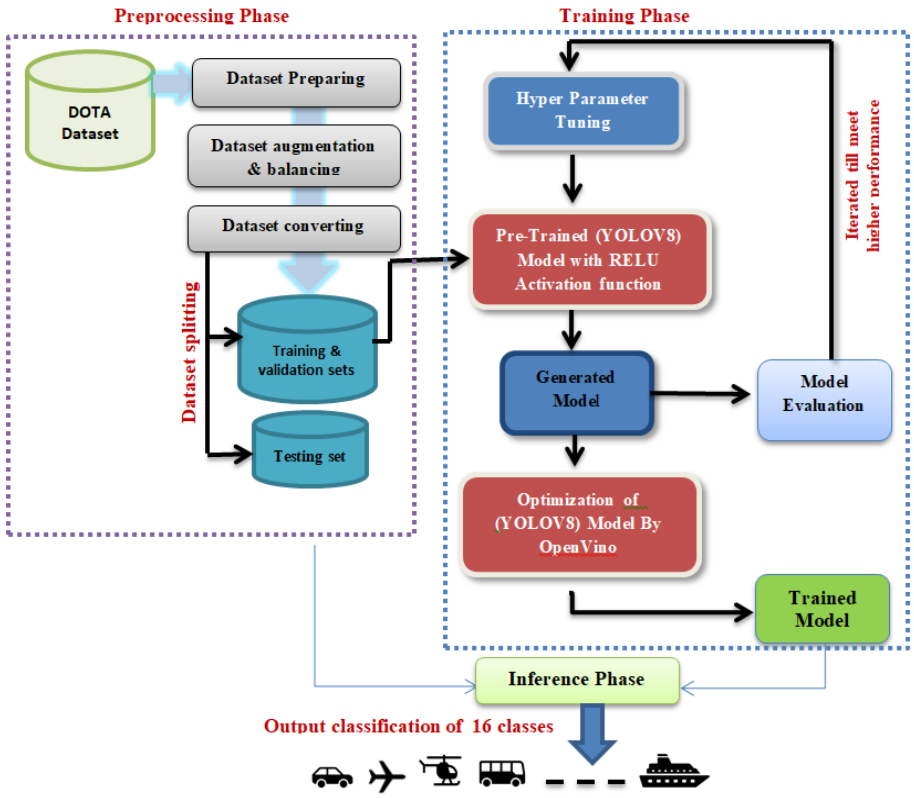

Figure 2: Diagram of the proposed framework for tiny objects detection based on optimized YOLOv8 deep learning model

3.1 Preprocessing phase

The preprocessing phase employed in the proposed framework involves dataset preparation, dataset augmentation, dataset balancing, converting dataset samples into YOLO format, and the dataset splitting process, as illustrated in the following subsection:

3.1.1 Dataset preparing

In order to accomplish the training of the YOLOv8 model, it is essential to prepare training samples and establish corresponding annotations for each sample. Accordingly, an annotation file must be generated for each training image, encompassing the class of object target as well as bounding box coordinates that surround the object target within the interval [0, 1]. This research uses the DOTA dataset [28] to train and evaluate the optimized YOLOv8 model. In order to make images aligned with the training requirements of the YOLOv8 model, the dataset's original files are reorganized. This adaptation involved restructuring dataset samples into train, val, and test folders to store images and related annotation/label files. Additionally, .yaml configuration file was created to specify the dataset's paths, class names, and the number of classes. The dataset was further divided into three subsets: training, validation, and testing, with specific data splitting ratios selected previously for model evaluation workflow.

3.1.2 Dataset augmentation and balancing

Commonly, the overfitting problem occurs throughout the training of deep learning models due to the lake of training samples and the unbalanced distribution of samples for each class in the dataset used. In practice, an imbalanced dataset consists of an unequal number of samples related to different classes. Thus, when there is an imbalance class, the classifier tends to be more biased towards the major class, which leads to poor classification for the minor class. Therefore, the proposed framework employs the dataset's augmentation and balancing techniques to overcome these issues and produce more precise predictions for tiny objects with

varying angled placements of target objects. The public dataset DOTA-v1.5 is comprised of 16 object classes with various sample representations for each class as shown in figure (3-a). The original dataset involved 2,806 samples with unbalanced classes' distribution. To handle the unbalanced distribution of dataset samples, we have applied five augmentation techniques over dataset samples, such as horizontal and vertical flipping, brightness adjustment, erasing, and gray filters. Taking into consideration a specific emphasis on smaller classes and given the high density of small objects within a single image and the frequent occurrence of closely positioned or overlapping bounding boxes. The applied augmentation techniques were adopted to preserve the integrity of these bounding boxes without disrupting their locations. Thereby, dataset samples are successfully expanded into 6,059 samples with reasonable and evenhanded distribution over entire classes, as illustrated in figure (3-b). By addressing the imbalance dataset, the augmented balanced dataset provides an effective foundation for training models with more generalization and performance.

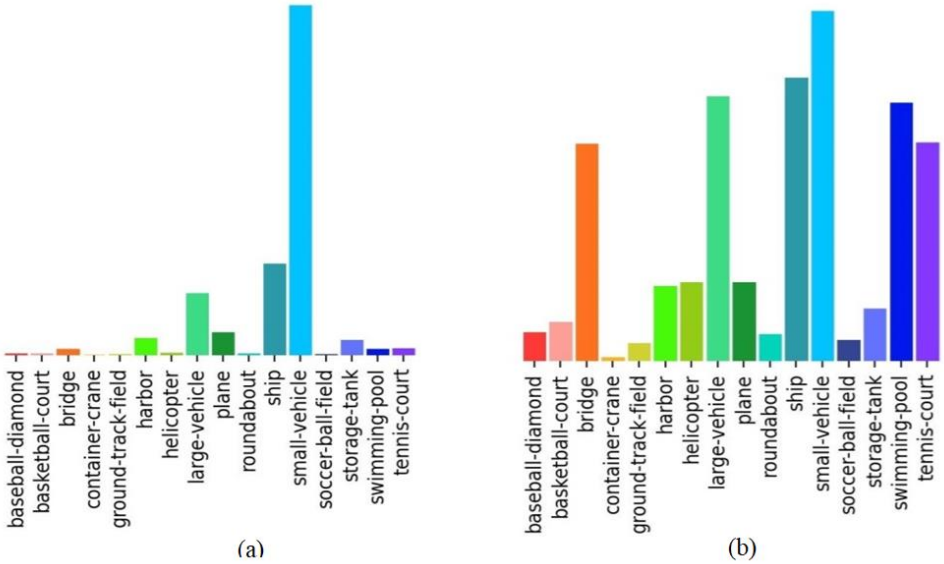

Figure 3: Classes distribution of DOTA-v1.5 dataset in: a) original dataset b) augmented and balanced dataset.

3.1.3 Converting Dataset Samples into YOLO format

In this phase, an annotation process is applied to dataset samples (images) to meet the training requirements of the YOLOv8 model. The annotation process involves converting the original format into YOLO format. The original annotation format includes comprehensive details, such as object category, bounding box coordinates, scores, truncation, and occlusion. To create YOLO annotation format for each sample, we preserved the most important information about object target as object category and excluded unrelated regions. In this way, a streamlined annotation structure compatible with the YOLO format is generated. Furthermore, the bounding box coordinates surrounding the object target were adjusted to conform with YOLO's context, taking into consideration image size. Making sure that the YOLO format was generated for every sample in the dataset was important for making sure that the annotations matched what the model expected. This made it easier to train and test the YOLOv8 model on the DOTA dataset.

3.2 Model Training

After the preprocessing phase, we have to import and train the YOLOv8 deep learning model over the preprocessed dataset to generate a robust detector to detect and classify the tiny objects in the unseen samples. This section will detail the implementation of the training process,

which involved conducting several experiments to optimize the YOLOv8 model and enhance the performance of the small object detection system on the DOTA [28] dataset. The YOLOv8 model has a high trade-off between speed and detection accuracy. First, we instantiate the YOLOv8 model with pre-trained weights and new hyperparameter configurations to train it on the preprocessed dataset. It's worth noting that the input sample size for the YOLOv8 network is (640×640). The input images for the model are in RGB color space and normalized into the range [0, 1]. The dimensions of the small objects in all training images do not surpass 12×12 pixels. Second, the pre-trained model will go through the whole training dataset a certain number of times, using the training images and their labels to identify the target objects at each time. This is called a supervised learning workflow. Third, we have implemented the hyperparameter tuning technique in the validation mode to enhance the pre-trained model's performance in detecting small objects in the input images. The goal of the fine-tuning technique is to adjust key model settings, improve model convergence, generalization, and higher object detection accuracy, as well as speed up the training process. By fine-tuning the model hyperparameters, we can adapt the model expertise to specific target tasks, thereby saving valuable time and computational resources. In this research, we have focused on the essential models' hyperparameters illustrated in Table 1. The default setting of hyperparameters in the YOLOv8 deep learning model provides a robust starting point for object detection tasks.

Adam Optimizer [29], known for its adaptability and efficiency, was employed in a new setting of hyperparametes to obtain a more generalization model. The initial learning rate was adjusted to 0.001 to facilitate smoother convergence, while the batch size was reduced to 16 to enhance the model's ability to capture intricate object details. Furthermore, the checkpoint save frequency (save) and early stopping patience (patience) were modified to align with the dataset's characteristics. Table 1 provides a comprehensive view of hyperparameter settings in the proposed framework in addition to the default settings and the refined choices employed.

Table 1: Hyperparameters tuning of YOLOv8 deep learning model

Parameters	Default setting	Tuned setting	Tuned setting
Epochs	100	150	Class loss gain (scaled with pixels)
Patience	50	5	Epochs waiting for no noticeable enhancement for early ending the training
Batch	32	16	Number of images per single batch (-1 for AutoBatch)
Imgsz	640	640	Size of input images
Optimizer	Auto	Adam	Optimizer used
Ir0	0.01	0.001	Initial learning rate
Weight_decay	0.0005	0.0005	The weight decay of the Optimizer (5e-4)
Warmup_epochs	3.0	3.0	Warmup epochs (fractions are acceptable)
Box	7.5	7.5	Box loss gain
cls	0.5	.05	Class loss gain (scaled with pixels)

3.3 Model Optimization

Our framework employs an optimization technique to enhance the performance of YOLOv8 model-based small object detection, accelerating the detection task's run time and making it more suitable for real-time applications. These optimization techniques rely on the ReLU activation function and the OpenVINO framework.

3.3.1 Activation Function Replacement

The decision to replace the SiLU (Sigmoid Linear Unit) activation function with ReLU (Rectified Linear Unit) in our model stems from both theoretical and practical considerations. SiLU, defined as:

$$\boldsymbol{SiLU}(\boldsymbol{x}) = x\left(\frac{1}{1+e^{-x}}\right) \quad (1)$$

It involves complex mathematical operations such as exponentiation and division, which are computationally expensive. On the other hand, ReLU, defined as:

$$\mathbf{ReLU}(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x}) \quad (2)$$

It requires only a simple comparison and is computationally efficient. This simplicity significantly reduces the computational load during both the forward and backward passes of neural network training and inference as the activation function is replaced in every layer of the model. SiLU integrates the sigmoid function with a linear term to sustain a non-linear transformation and scale activations, aiding in the identification of intricate patterns and relationships within the data. In contrast, ReLU is a simpler activation function that introduces non-linearity by setting all negative values to zero and keeping positive values unchanged. This simplicity can significantly speed up computations, as ReLU's operations are less computationally intensive compared to SiLU's sigmoid component. However, this reduction in computational complexity can also lead to a decrease in the model's ability to capture intricate features, potentially reducing accuracy because it does not model complex data relationships as effectively as SiLU. Thus, while ReLU improves speed, it may sacrifice some of the model's capacity for nuanced data representation.

As a result, using ReLU instead of SiLU can lead to faster processing times and lower latency, making the system more suitable for real-time applications where speed is critical.

3.3.2. Leveraging OpenVINO for Model Speedup

To this end, OpenVINO (Open Visual Inference and Neural Network Optimization) is utilized in the proposed framework, which serves as a cross-platform deep learning toolkit developed by Intel. As a pivotal component in the realm of artificial intelligence (AI), OpenVINO specializes in optimizing neural network inference with a universal deployment approach across various Intel hardware platforms. OpenVINO toolkit works as a transformer or bridge between common deep learning platforms, such as TensorFlow, ONNX, and hardware components of Intel, permitting users to exploit their computational resources [27]. The OpenVINO toolkit accelerates model inference by optimizing deep learning models for Intel hardware. This is achieved through a variety of key techniques. Firstly, OpenVINO performs layer fusion, combining multiple neural network layers into single, more efficient operations, thereby reducing the overall number of computations. Second, it uses model quantization to convert high-precision 32-bit floating-point weights to lower-precision 8-bit integers. This makes the model smaller and speeds up inference without sacrificing too much accuracy. Additionally, OpenVINO utilizes hardware-specific optimizations, leveraging the full potential of Intel CPUs, integrated GPUs, and VPUs (Vision Processing Units) for parallel processing. These optimizations collectively result in substantial performance gains, enabling faster and more efficient model inference, which is particularly beneficial for deployment in resource-constrained environments and edge devices. With its focus on optimizing deep neural network models, OpenVINO accelerates the deployment of accurate and efficient models, adapting

them to various hardware architectures seamlessly [27]. Figure (4) illustrates the main architecture of the OpenVINO API.

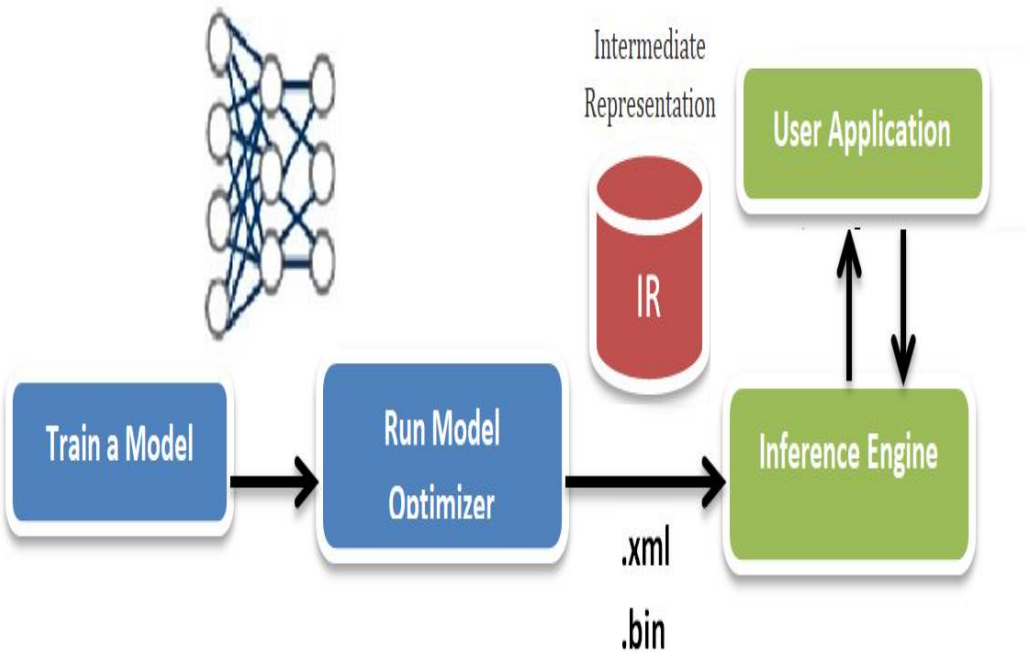

Figure 4: An illustration of OpenVINO API architecture [27]

3.4 Model Evaluation

To assess the performance of the trained model, we have partitioned the dataset into training, validation, and testing sets. Specifically, 20% of the entire dataset was allocated for the validation process, serving to quantify the model's accuracy and generalization capabilities. The validation dataset played a crucial role in evaluating the model's performance at each epoch. To thoroughly evaluate YOLOv8's performance, we set the number of training epochs to 150. This choice provided valuable insights into the models' behavior throughout training iterations, offering a comprehensive understanding of their convergence and overall performance.

4 Experimental Results

In this section, we demonstrate and argue about several experiments implemented during the evaluation process, with corresponding results obtained from variant executions of the proposed framework. Extensive experiments have been attained to evaluate the performance of the proposed framework. The dataset used for training and testing the model is DOTA [28], described below in Table 2. The experiments show how well the model works with the chosen optimization methods, such as YOLOv8 model generalization and convergence with hyperparameter tuning, ReLU activation function use, and YOLOv8 model performance with the OpenVINO toolkit. Furthermore, we conducted a quantitative and qualitative comparison study during the performance evaluation process, demonstrating the superior performance of the proposed framework over state-of-the-art frameworks. We implemented all the experiments using a V100 GPU in Google Colab for training and a PC with a 13th Gen Intel(R) CoreTM i5-13400F 2.50 GHz processor for the computation speed benchmark.

Table 2: Description of DOTA dataset

Dataset Name	DOTA (Dataset for Object detection in Aerial Images)
Number of Categories	16 common categories
Number of Images	6,059 images
Number of Instances	160, 691 instances
Image Size	Range from 800 × 800 to 20,000 × 20,000 pixels
Dataset Splits	The proportions of the training set, validation set, and testing set are 70%, 20%, and 10%, respectively
Annotation	Instances in DOTA images are annotated by experts in aerial image interpretation by arbitrary (8 degrees of freedom) quadrilateral

4.1 Dataset Used

To undertake the training, validation, and testing of the proposed fine-tuned YOLOv8 deep learning model for small object detection, we leveraged the DOTA [28] dataset, which consists of 6,095 image samples and 160,691 instances. The DOTA dataset samples include various types/classes of objects (16 classes) captured by UAVs with different resolutions, altitudes, and weather conditions. This dataset was partitioned into three main sets: training, validation, and testing, with partition ratios of 4241, 2121, and 606 images for each set, respectively. Actually, the annotations for both object tracking and object detection tasks are available for 16 distinct object categories, including baseball-diamond, basketball-court, bridge, container-crane, ground-track-field, harbor, helicopter, large-vehicle, plane, roundabout, ship, small-vehicle, soccer-ball-field, storage tank, swimming pool, and tennis court, as illustrated in Table 3. Figure (5) demonstrates samples of UAV images of the DOTA dataset. In terms of the object detection task, the dataset includes annotations with bounding boxes and object categories. These annotations furnish class label and bounding box coordinates for each target object. Table 3 shows the number of instances for each class.

Table 3: Number of instance for each class in DOTA dataset

#	Class name	Number of instances	#	Class name	Number of instances
1	baseball-diamond	2,325	9	large-vehicle	20,952
2	basketball-court	3,071	10	plane	7,615
3	bridge	17,664	11	roundabout	2,140
4	container-crane	214	12	ship	23,368
5	ground-track-field	1,444	13	small-vehicle	25,115
6	harbor	5,874	14	soccer-ball-field	1,599
7	helicopter	6,360	15	storage-tank	4,771
8	swimming-pool	20,270	16	tennis-court	17,909

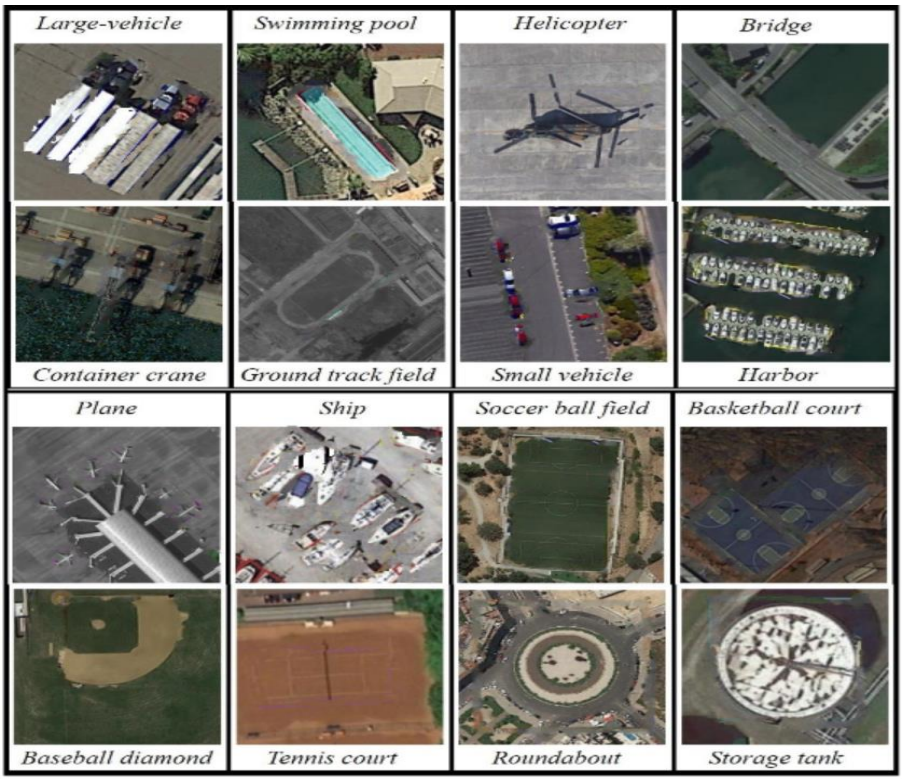

Figure 5: UAVs images samples of DOTA dataset [28]

4.2 Performance Evaluation Metrics

The performance assessment of this research was carried out using standard evaluation metrics that are utilized for object detection approaches as precision, recall, F1-score, and average precision (AP@.50), which gives the average precision of objects detected with at least 50% IoU (intersection of union) between the detected and the ground-truth bounding boxes, and AP@[0.5, 0.95], which is the average AP when the IoU goes from 50% to 95% in 5% steps.

Precision (P): Precision measures the amount of correct positive predictions out of all the positive predictions done by a system. Precision focuses on the accuracy of positive predictions. It refers to how many of the occasions predicted to be positive are actually positive [30].

$$precision(p) = \frac{tp}{tp+fp} \quad (3)$$

Where *tp* represents the number of true positives (properly expected positive instances), *fp* represents the number of false positives (instances incorrectly expected as positive when they are actually negative).

Recall (R): is defined as the ratio of true positive detections to the sum of true positives and false negatives, and it measures the proportion of actual positive instances that are correctly identified by the model [30].

$$Recall(R) = \frac{tp}{tp+fn} \quad (4)$$

F1-Score (F1): The F1-Score is the harmonic mean of precision and recall metrics. It delivers a balanced measure that reflects both precision and recall. F1-Score combines precision and recall into a single metric. It is particularly useful when there's a need to balance the trade-off between making accurate positive predictions and not missing any positive instances [30].

$$F1 = 2 * \frac{(Precision*Recall)}{(Precision+Recall)} \quad (5)$$

Average Precision (AP): Average precision is a metric used in specific tasks, such as information retrieval. It quantifies the area under the precision-recall curve (PR curve). The AP

score summarizes the system's performance across different classification thresholds. Higher values of AP indicate better overall performance [31].

4.3 Experiments Details and Implementation

In this section, we have presented the implementation details of four experiments conducted for the purpose of model training/validation, model optimization process, and model detection of 16 classes of tiny objects in UAV images. The experiments are performed to evaluate the performance of the proposed optimized YOLOv8 model, which is dedicated to small object detection tasks in terms of accuracy and speed measurements. In order to exhibit the effectiveness of the proposed optimized model, three evaluation metrics have been employed, including mean average precision (mAP@50), mean average precision (mAP50@95), and inference speed, as elaborated below:

Experiment 1: YOLOv8 with original DOTA dataset

In this experiment, the pre-trained YOLOv8 model was imported and carried out to detect and classify 16 classes of objects on the original DOTA dataset, which consists of 2,806 image samples. The training process was intended to run over 150 epochs and stopped prematurely at epoch 68. The reason for early stopping is the fact that no more improvement in accuracy was obtained through training at the last 5 epochs. The early stopping parameter (patience) was set to 5 epochs, with a batch size of 4 and a checkpoint save frequency of 2. Each epoch has taken an average of 5 minutes to complete. This experiment served as the baseline for model performance on the DOTA dataset with default activation functions. The entire duration of training the model in this experiment was approximately 5 hours and 40 minutes. Figure (6) demonstrates the training loss, validation loss, precision/recall, and mAP@50 and mAP50@95 performance evaluation metrics. Obviously, all the evaluation metrics were unacceptable with the original dataset size and unbalanced distribution of classes. So, to make the YOLOv8 model better at finding small objects in imagery data, we need more data samples and a more even distribution of objects across dataset samples. This experiment was done using a Nvidia RTX 3080 GPU.

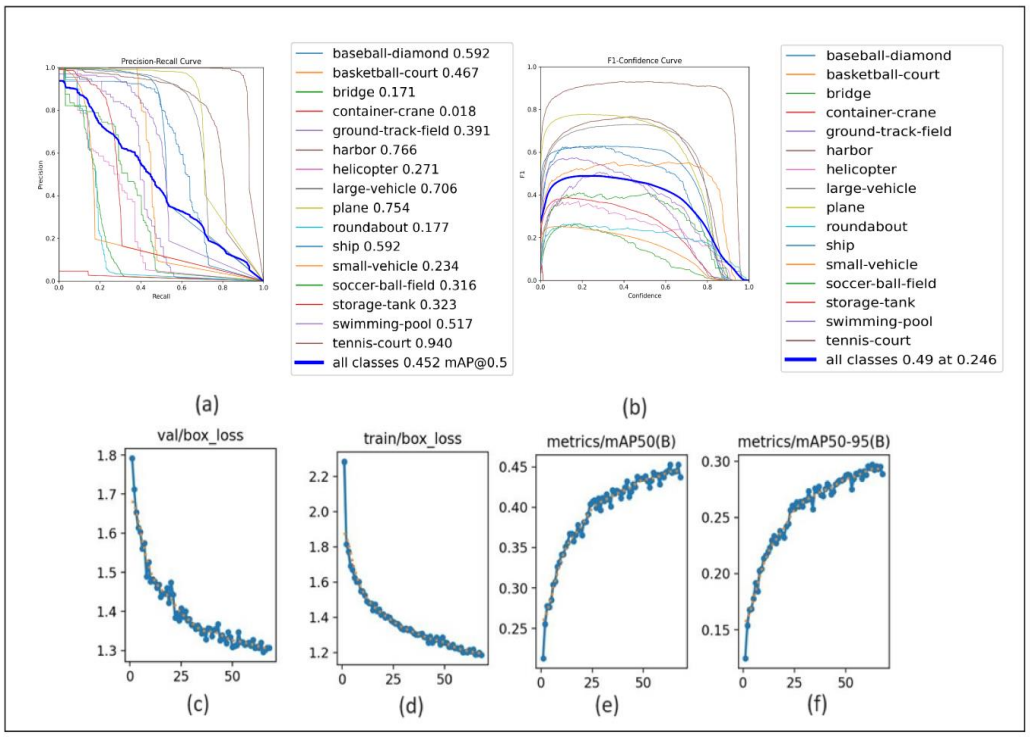

Figure 6: Evaluation of pre-trained YOLOv8 model on DOTA dataset; (a) precision, (b) F1-confdence, (c) training loss, (d) validation loss, (e) mAP50, (f) mAP50-95.

Experiment 2: YOLOv8 with augmented and balanced DOTA dataset

In this experiment, the pre-trained YOLOv8 model is imported and run on an augmented and balanced dataset to detect and classify 16 classes of objects. The dataset size has been increased to 6,059 data samples using the augmentation technique. In addition, we have balanced the number of target objects for each class over the whole dataset. The dataset balancing process helps to prevent the model from being biased towards one or a major class, preserving rightful representation and further reliable estimations. The training process was intended to run over 150 epochs but stopped prematurely at epoch 108 with no more improvement of performance at the last 5 epochs. The early stopping parameter (patience) was also set to 5 epochs, the batch size of 4, and a checkpoint save frequency of 1. On average, each epoch took about 10 minutes to complete. This experiment served as the new baseline for model performance on the augmented and balanced DOTA dataset. The training process in this experiment took a total of 20 hours and 40 minutes. Figure (7) demonstrates the training and validation losses, precision/recall, mAP@50, and mAP50@95 performance evaluation metrics. As shown in Figure (7), the detection accuracy was improved from 0.49 to 0.59 based on the Map50 metric. Adopting augmentation and class balancing of dataset samples leads to this performance improvement. This experiment was done using a Nvidia Tesla V100 GPU.

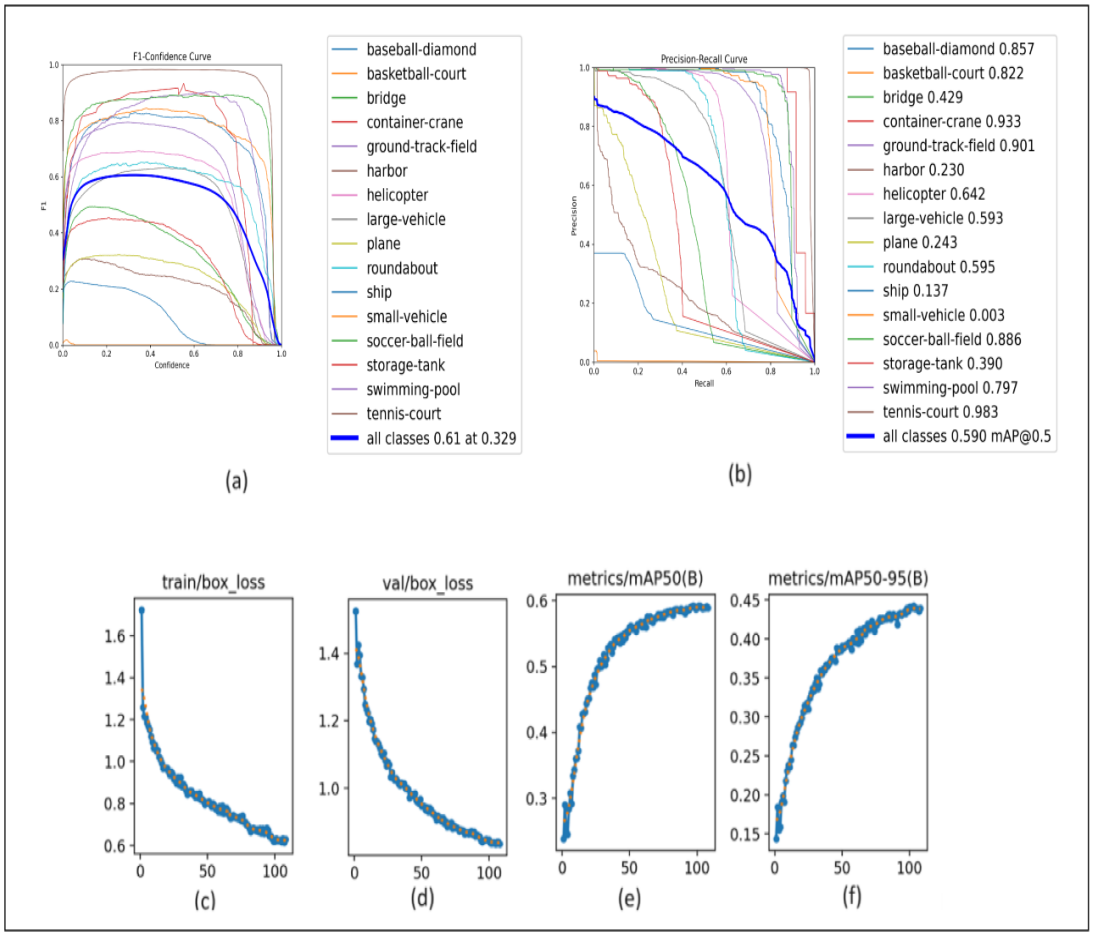

Figure 7: Evaluation of YOLOv8 model on custom DOTA dataset: (a) F1-confdence, (b) precision, (c) training loss, (d) validation loss, (e) mAP50, (f) mAP50-95.

Experiment 3: YOLOv8 with ReLU activation function, augmented and balanced DOTA dataset

In this training experiment, the ReLU activation function was employed and adopted in terms of model optimization. The training process was accomplished on the customized and

balanced DOTA dataset. The training workflow was intended to run over 150 epochs but stopped early at epoch 85, with a batch size of 4 and a checkpoint save frequency of 1 epoch. The early stopping patience was set to 5 epochs. On average, each epoch took about 10 minutes to complete. This experiment lasted for approximately 20 hours. Based on the findings of this experiment, the accuracy was a little bit decreased by 1%. However, the run time was much faster in the inference stage. Figure (8) demonstrates the training loss, validation loss, precision/recal, and mAP@50 and mAP50@95 performance evaluation metrics. This experiment was done using a Nvidia Tesla V100 GPU.

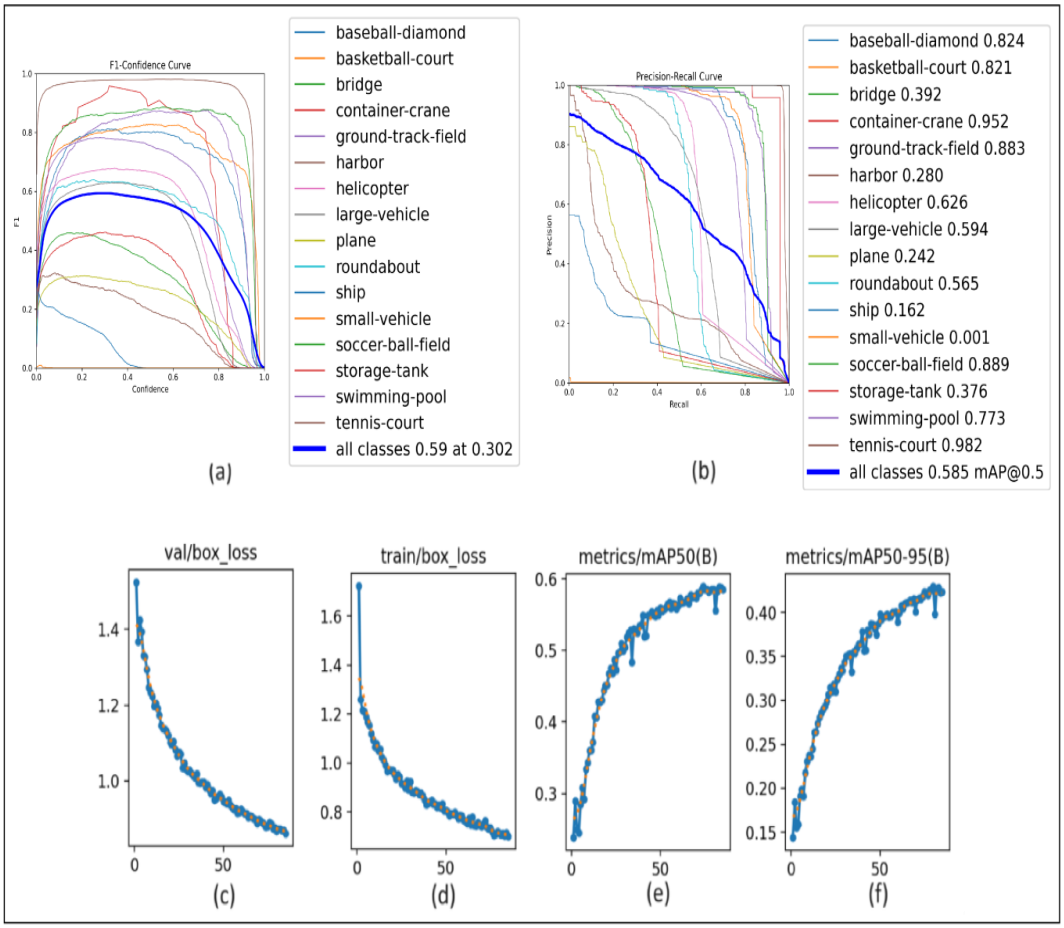

Figure 8: Evaluation of optimized YOLOv8 model on custom dataset: (a) F1-confdence, (b) precision, (c) training loss, (d) validation loss, (e) mAP50, (f) mAP50-95.

When we tested the model on unseen test data, it achieved a 0.573 mAP, indicating that the model achieved a good generalization, as the accuracy did not significantly decrease compared to the validation data. Figure 9 demonstrates the precision/recall and f1 performance evaluation metrics. Figure 10 shows samples of the detected objects in the testing images.

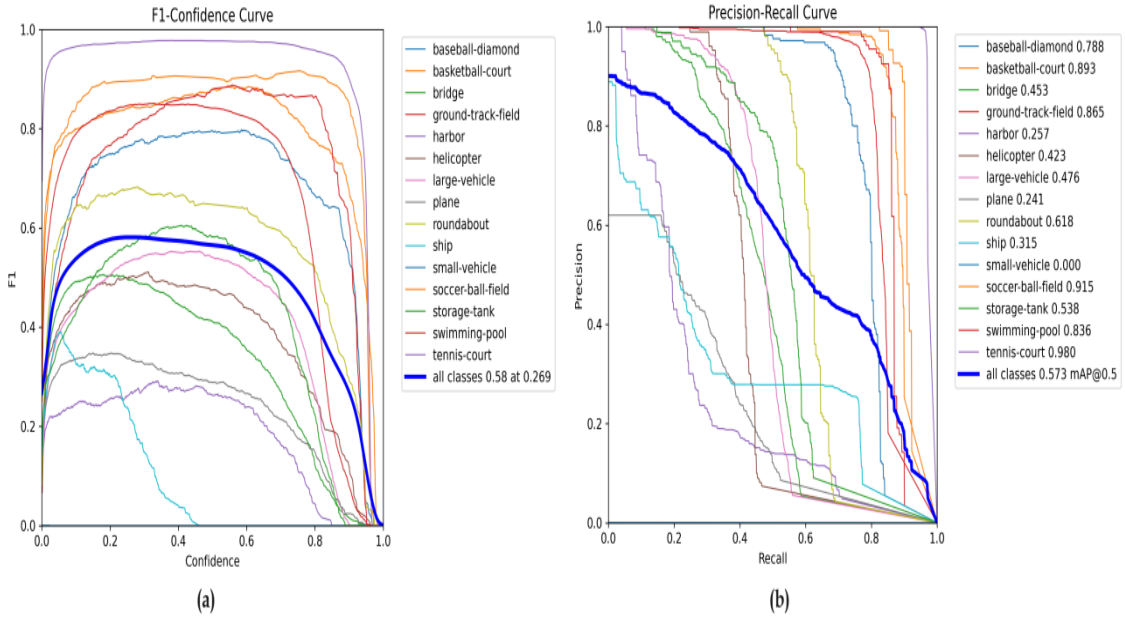

Figure 9: Evaluation of Yolov8 with ReLU activation function model on the test subset of DOTA dataset: (a) f1-confdence, (b) precision.

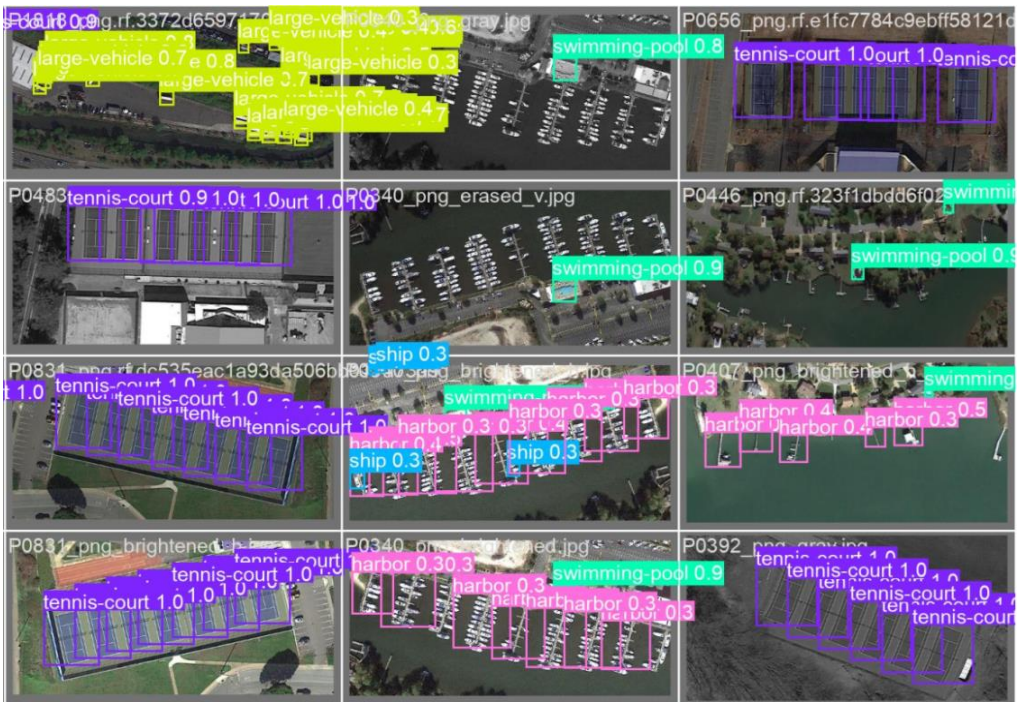

Figure 10: Shows samples of the detected objects in the testing DOTA images.

Exterminate 4: YOLOv8 with ReLU activation function, OpenVINO, augmented and balanced DOTA dataset

In order to make the optimized model even faster in the inference mode, an OpenVINO toolkit was adopted and exploited in our framework along with the previous setting of the YOLOv8 model (mentioned in experiment 3) and a custom dataset. Using the benchmark function in YOLOv8, the OpenVINO file was generated to test the inference speed. The results revealed a 23.67% increase in inference speed, which is faster than the base line model setting mentioned in Experiment 2. The detection accuracy was the same when the model used the ReLU function. Figure (11) demonstrates the obtained results compared to the original model in terms of

accuracy and inference speed. The accuracy is measured based on the mAP50 metric, where a higher value is better and the speed in terms of the time it takes the model to process each image in milliseconds (ms), where lower is better, and table 4 shows the accuracy obtained for each class in the mAP50 metric. This experiment was done using 13th Gen Intel Core i5-13400F at 2.50 GHz.

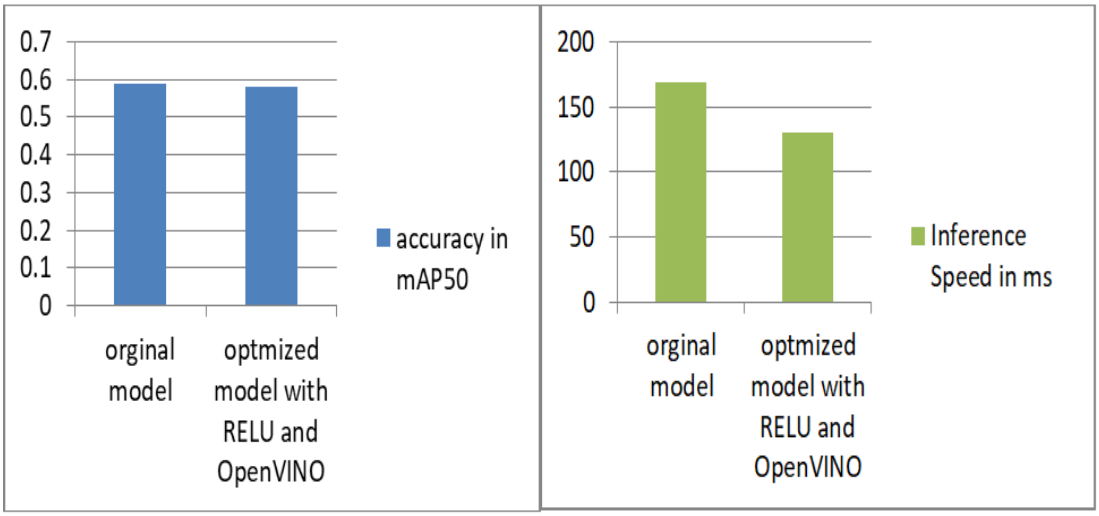

Figure 11: Original model compared to the model optimized with ReLU and OpenVINO together

Table 4: Shows the accuracy obtained for each class in mAP50.

#	Class name	accuracy	#	Class name	accuracy
1	baseball-diamond	0.592	9	large-vehicle	0.706
2	basketball-court	0.467	10	plane	0.754
3	bridge	0.171	11	roundabout	0.177
4	container-crane	0.018	12	ship	0.592
5	ground-track-field	0.391	13	small-vehicle	0.234
6	harbor	0.766	14	soccer-ball-field	0.316
7	helicopter	0.271	15	storage-tank	0.323
8	swimming-pool	0.517	16	tennis-court	0.940

4.4. Comparison Study

In order to show the performance of the proposed framework compared to the other export formats for the YOLOv8 model, a comparison study based on evaluation metrics such as average precision metric (AP) and inference speed is achieved and illustrated in Table 5. Based on our experimental results, we observed a significant improvement in the detection accuracy and inference speed measurements of the optimized YOLOv8 deep learning model, reaching up to 23.67% faster while maintaining nearly the same accuracy. This demonstrates the efficacy of utilizing the ReLU activation function and OpenVINO to enhance the framework's inference speed, thereby enhancing its suitability for real-world applications. Furthermore, we demonstrated that the proposed small object detection framework outperformed the state-of-the-art methods using the optimized YOLOv8 model, as shown in Table 6.

Table 5: Comparison study according to mAP-50 metric and inference speed in ms Using 13th Gen Intel(R) Core(TM) i5-13400F , 2.50 GHz

Framework/Export Format	mAP-50	mAP50-95	Inference Speed (ms)
PyTorch (original model)	0.590	0.442	168.4
ONNX Runtime [32]	0.581	0.442	159.7
TensorFlow Graph Def [33]	0.581	0.442	420.3
TensorFlow Lite [33]	0.581	0.442	1207.5
YOLOv8 with ReLU	**0.585**	**0.430**	**151.5**
Optmized-YOLOv8 with ReLU and OpenVINO	**0.585**	**0.430**	**129.8**

Table 6: Comparison study of the proposed framework against stat of the art models.

Method	Detection accuracy	Dataset used	Inference speed (ms)	Hardwar
YOLOv9 [34][19]	0.55 mAP	DOTA	73 FPS ≈ 13.62	NVIDIA Tesla T4
YOLOv10 [20]	54.4 AP^{val}	COCO	Unknown	NVIDIA Tesla T4
RTMDe[35]	78.12 mAP	DOTA	121 FPS ≈ 8.26	NVIDIA RTX 3090
YOLOv8 with ReLU	**0.585 mAP50**	**DOTA**	**151.5**	
Optmized-YOLOv8 with ReLU and OpenVINO	**0.585 mAP50**	**DOTA**	**129.8**	13th Gen Intel Core i5-13400F 2.50 GHz

5. Conclusion and Discussion

In this paper, we addressed the problem of small object detection in UAV images based on the YOLOv8 deep learning model and optimized a model for faster inference using ReLU as an activation function and the OpenVINO API to improve the system's inference speed, making it more suitable for real-world applications. Multiscale objects with different resolutions and variant weather conditions pose a significant challenge. We evaluated the performance of the proposed framework using a DOTA dataset, which comprises images captured under varying altitudes, resolutions, and weather conditions. To improve the YOLOv8 model's ability to detect things quickly while maintaining accuracy, we first optimized the model with the ReLU activation function and then optimized it with the OpenVINO API. The experiments yielded remarkable results. Then, optimizing the YOLOv8 model with the ReLU activation function and OpenVINO yielded even better results, achieving a 23.67% speed increment compared to the original experiment. The findings underscore the importance of model activation function choice and optimization APIs for fulfilling higher inference speeds. This work addresses a critical issue in computer vision as well as paves the way for future improvements and innovations in small object detection in UAV images, benefiting a wide range of practical applications.

References

[1] A. M. Rekavandi, L. Xu, F. Boussaid, A.-K. Seghouane, S. Hoefs, and M. Bennamoun, "A Guide to Image and Video based Small Object Detection using Deep Learning : Case Study of Maritime Surveillance." arXive, 2022. doi: https://doi.org/10.48550/arXiv.2207.12926.

[2] Y. Kondo *et al.*, "MVA2023 Small Object Detection Challenge for Spotting Birds: Dataset, Methods, and Results," *Proceedings of MVA 2023 - 18th International Conference on Machine Vision and Applications*. 2023. doi: 10.23919/MVA57639.2023.10215935.

[3] A. S. Mahdi and S. A. Mahmood, "An Edge Computing Environment for Early Wildfire Detection," *Annals of Emerging Technologies in Computing*, vol. 6, no. 3. pp. 56–68, 2022. doi: 10.33166/AETiC.2022.03.005.

[4] J. S. Shaoqing Ren, Kaiming He, Ross Girshick, "Faster R CNN Towards Real-Time Object

Detection with Region Proposal Networks." arXive, 2016. doi: https://doi.org/10.48550/arXiv.1506.01497.

[5] S. K. Jarallah and S. A. Mahmood, "Deep-learning models based video classification: Review," *AIP Conf. Proc.*, vol. 2834, no. 1, p. 50008, Dec. 2023, doi: 10.1063/5.0161553.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem. pp. 779–788, 2016. doi: 10.1109/CVPR.2016.91.

[7] Wei Liu *et al.*, "SSD: Single Shot MultiBox Detector." Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, pp. 21-37, Springer International Publishing, 2016. doi: https://doi.org/10.1007/978-3-319-46448-0_2.

[8] Y. Liu, P. Sun, N. Wergeles, and Y. S. Department, "A survey and performance evaluation of deep learning methods for small object detection." Expert Systems with Applications, Volume 172, 2021, 114602, ISSN 0957-4174, 2021. doi: https://doi.org/10.1016/j.eswa.2021.114602.

[9] S. Li, W. Zhang, G. Li, L. Su, and Q. Huang, "Vehicle Detection in UAV Traffic Video Based on Convolution Neural Network," *Proceedings - IEEE 1st Conference on Multimedia Information Processing and Retrieval, MIPR 2018*. pp. 1–6, 2018. doi: 10.1109/MIPR.2018.00009.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXive, 2020. doi: https://doi.org/10.48550/arXiv.2004.10934.

[11] Z. Z. Wang, K. Xie, X. Y. Zhang, H. Q. Chen, C. Wen, and J. B. He, "Small-Object Detection Based on YOLO and Dense Block via Image Super-Resolution," *IEEE Access*, vol. 9. pp. 56416–56429, 2021. doi: 10.1109/ACCESS.2021.3072211.

[12] O. C. Koyun, R. K. Keser, İ. B. Akkaya, and B. U. Töreyin, "Focus-and-Detect: A small object detection framework for aerial images," *Signal Process. Image Commun.*, vol. 104, p. 16, 2022, doi: 10.1016/j.image.2022.116675.

[13] S. Tang, S. Zhang, and Y. Fang, "HIC-YOLOv5: Improved YOLOv5 For Small Object Detection," 2023, doi: https://doi.org/10.48550/arXiv.2309.16393.

[14] X. Guo, "A novel Multi to Single Module for small object detection." arXive, 2023. doi: https://doi.org/10.48550/arXiv.2303.14977.

[15] B. Liu, F. He, S. Du, J. Li, and W. Liu, "An advanced YOLOv3 method for small object detection," *Journal of Intelligent & Fuzzy Systems*. pp. 1–13, 2023. doi: 10.3233/jifs-224530.

[16] X. Yuan, G. Cheng, K. Yan, Q. Zeng and J. Han, "Small Object Detection via Coarse-to-fine Proposal Generation and Imitation Learning," in 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 2023, pp. 6294-6304. doi: 10.1109/ICCV51070.2023.00581

[17] S. Chen *et al.*, "TinyDet: accurately detecting small objects within 1 GFLOPs," *Science China Information Sciences*, vol. 66, p. 119102, 2023. doi: 10.1007/s11432-021-3504-4.

[18] A. Benjumea, I. Teeti, F. Cuzzolin, and A. Bradley, "YOLO-Z: Improving small object detection in YOLOv5 for autonomous vehicles." 2021. [Online]. Available: http://arxiv.org/abs/2112.11798

[19] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information." 2024. [Online]. Available: http://arxiv.org/abs/2402.13616

[20] A. Wang *et al.*, "YOLOv10: Real-Time End-to-End Object Detection." 2024. [Online]. Available: http://arxiv.org/abs/2405.14458

[21] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 8759–8768, 2018, doi: https://doi.org/10.48550/arXiv.1803.01534.

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

[23] Xiang Li *et al.*, "Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection Xiang." arXive, 2020. doi: https://doi.org/10.48550/arXiv.2006.04388.

[24] RangeKing, "Model structure of YOLOv8 detection models." 2023. [Online]. Available: https://github.com/ultralytics/ultralytics/issues/189#issue-1527158137

[25] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, 2018, doi:

10.1016/j.neunet.2017.12.012.

[26] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)." arXive, 2018. doi: https://doi.org/10.48550/arXiv.1803.08375.

[27] Intel Corporation, "OpenVINO™ Toolkit, [Online], Available: https://github.com/openvinotoolkit/openvino." 2018.

[28] G. S. Xia *et al.*, "DOTA: A Large-Scale Dataset for Object Detection in Aerial Images," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 3974–3983, 2018. doi: 10.1109/CVPR.2018.00418.

[29] Diederik P. Kingma and Jimmy Lei Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," 2017, doi: https://doi.org/10.48550/arXiv.1412.6980.

[30] H. Dalianis, "Evaluation Metrics and Evaluation," *Clinical Text Mining*. pp. 45–53, 2018. doi: 10.1007/978-3-319-78503-5_6.

[31] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms." International Conference on Systems, Signals and Image Processing (IWSSIP), vol. 1, no. 1, Jul. 2020, 2020. doi: 10.1109/IWSSIP48289.2020.9145130.

[32] ONNX Runtime developers, "onnxruntime." https://onnxruntime.ai/, 2021.

[33] P. B. Martín Abadi, Ashish Agarwal, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." 2015. [Online]. Available: https://www.tensorflow.org/

[34] ausk, "Yolov5 yolov8 yolov9 speed test on T4." 2024. [Online]. Available: https://github.com/WongKinYiu/yolov9/issues/178

[35] C. Lyu *et al.*, "RTMDet: An Empirical Study of Designing Real-Time Object Detectors." 2022. [Online]. Available: http://arxiv.org/abs/2212.07784