Deep Learning Final Project

Michael Bennett
Department of Economics
Yale University
New haven, CT
m.bennett@yale.edu

Ruiqi Zhang
Department of Biostatistics
Yale University
New haven, CT
ruiqi.zhang@yale.edu

Yawen Wei
Department of Biostatistics
Yale University
New haven, CT
yawen.wei@yale.edu

Ziang Li
Department of Biostatistics
Yale University
New haven, CT
ziang.li@yale.edu

1 Introduction/Motivation

1.1 What is the problem you are solving?

The problem at hand is to evaluate the effectiveness of state-of-the-art object detection models, such as YOLO and Faster R-CNN, for oriented bounding box (OBB) object detection in high-resolution satellite imagery. This requires identifying how well these models can detect various objects in real-world and virtual datasets and understanding their performance under diverse conditions. Moreover, we're exploring the viability of virtual-to-real transfer learning to enhance detection accuracy and reliability.[1]

1.2 What need does it solve?

Accurate object detection in satellite imagery is vital for numerous applications like disaster management, environmental monitoring, urban planning, and defense. By comparing model performance on the DOTAv1.5 dataset (real-world imagery) with the VALID-Det dataset (virtual imagery), this project addresses the need to understand how well these models generalize across varying data sources. Evaluating virtual-to-real transfer learning can reveal new insights into how simulated data can help improve object detection performance in practical scenarios. Ultimately, this work aims to refine object detection strategies to provide more reliable, accurate solutions for real-world challenges, supporting better decision-making across different sectors.

2 Background/Related Work

In the field of object detection, various frameworks have been developed to address the challenges posed by diverse application scenarios and data complexities. Two notable architectures are YOLO (You Only Look Once) and Faster R-CNN (Region-based Convolutional Neural Network). They stand out due to their significant advantages in real-time performance and high accuracy, respectively.

2.1 YOLO Series

The YOLO series was initially designed for real-time object detection. The latest iterations improve performance by optimizing the architecture and training strategies. YOLO's CSP (Cross Stage Partial

[1] Codes used for project available at https://github.com/michaelbennett99/CPSC452_project.

Networks) backbone reduces computational complexity while improving feature extraction, and the SPP (Spatial Pyramid Pooling) module fuses features at different scales for better detection across object sizes. An anchor-free detection head adds flexibility, but the framework struggles with occluded and small objects, limiting accuracy in some scenarios.

2.2 Faster R-CNN

Faster R-CNN builds on Fast R-CNN with a Region Proposal Network (RPN) that generates high-quality candidate regions, improving efficiency and accuracy. It excels in precision, particularly for small objects, but requires significant computational resources, limiting its real-time performance.

2.3 SSD (Single Shot MultiBox Detector)

SSD uses a single-shot detection strategy with multi-scale feature maps and default anchor boxes, achieving high-speed detection suitable for real-time scenarios like drone monitoring. However, its detection accuracy lags behind Faster R-CNN in complex occlusion scenarios.

2.4 RetinaNet

RetinaNet introduces Focal Loss to address class imbalance and improve detection of small objects. Its multi-scale feature fusion with ResNet and FPN (Feature Pyramid Network) enables complex object detection, though it demands high computational power.

2.5 EfficientDet

EfficientDet utilizes EfficientNet as a backbone, combining it with BiFPN (Bidirectional Feature Pyramid Network) for efficient multi-scale feature fusion. Compound scaling strategies balance network depth, width, and resolution for more efficient detection. However, it requires further optimization for ultra-large-scale datasets.

2.6 Cascade R-CNN

Cascade R-CNN uses a cascaded detection strategy for better classification and localization accuracy through multi-level cascading. It excels in complex detection but is computationally intensive with long inference times.

2.7 Deformable DETR (Deformable Detection Transformer)

Deformable DETR improves upon the standard DETR by combining the attention mechanism with deformable convolution for complex object detection. While the attention mechanism improves model flexibility, it increases training and inference times.

2.8 Key Metrics

There are several key metrics which are used in object detection to assess the performance of models. We summarise them here:

1. Intersection over Union (IoU): When two bounding boxes overlap, we can quantify the degree to which they overlap by IoU, which is

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}.$$

   Two boxes that overlap completely would have an IoU of 1, while two boxes that don't overlap have an IoU of 0.

2. Precision: The fraction of identified objects that align with the ground truth.
3. Recall: The fraction of ground truth objects that are identified.
4. mAP: Mean Average Precision is the average of class-wise average precision over all classes.

2.9 Summary

Each of these models balances speed and accuracy differently. Real-time detection models like YOLO and SSD excel in speed but are limited in handling small or occluded objects. High-precision models such as Faster R-CNN and RetinaNet deliver superior accuracy at the cost of computational resources, impacting real-time performance. More advanced models like EfficientDet, Cascade R-CNN, and Deformable DETR enhance detection accuracy further but require higher computational resources, making them difficult to deploy on resource-constrained devices. Ultimately, finding the right balance remains a challenge, especially when translating model performance from benchmark datasets to real-world applications.

In the rest of this report, we select Faster R-CNN and YOLO to examine more deeply by training on established challenge datasets in the

3 Model

3.1 You Only Look Once: YOLO

We used the YOLOv8 (*1*, *2*) model to detect cars, as well as some other objects, from the DOTA (*3*) and VALID (*4*) datasets. YOLO is a one-stage object detection model that re-frames object detection as a single regression problem. Unlike other methods such as Fast R-CNN (*5*), it only needs to make once pass of the data to produce a variable number of bounding boxes. As a result, it is able to reason globally about images, it is well optimised end-to-end for detection performance, and has good performance in transfer learning. However, it struggles to accurately detect small objects. These strengths and weaknesses, as well as its support for Oriented Bounding Boxes, which are crucial in aerial image object detection, make it an interesting candidate for object detection of vehicles on from aerial images.

3.1.1 Architecture

Generally, the YOLO model splits the input image into an $S \times S$ grid, and delegates responsibility for detecting an object to the grid cell the center of that object falls into. Each grid cell then predicts B bounding boxes, and a confidence score for each box, which reflect the model's confidence that the box contains an object and how accurate the predicted box is. Each oriented bounding box is formed of 5 parameters, center x coordinate x, center y coordinate y, width w, height h and rotation θ, as well as its confidence.

Each grid cell also predicts C conditional class probabilities, which give the probability that a class is in a grid cell conditional on there being an object in the grid cell. Even if multiple bounding boxes are predicted, there is only one set of class probabilities per grid cell. These probabilities are then multiplied with individual box confidence scores to find the class-specific confidence scores for each box. Figure 1 illustrates this process.

Another key feature of YOLO, as a single stage object detector, is a non-maximum suppression (NMS) algorithm. NMS is used to identify and remove redundant bounding boxes, where the model produces multiple boxes for a single object. While this is uncommon for small objects with large grid due to the structure of the model, sometimes large objects on relatively small grids can be attributed to multiple grid cells, and thus multiple boxes are produced over them. Non-maximum suppression detects predicted bounding boxes with high IoU and suppresses them.

The basic architecture of YOLO is that of a CNN with two fully connected layers at the end which output the grid-wise bounding boxes and confidence scores. An illustrative example of the YOLO architecture for YOLOv1 can be seen in Figure 2. The loss function for YOLO is comprised of three main terms, the box loss which penalises bounding box existence and coordinate error, the class loss which penalises class error, and the distribution focal loss, which is used to deal with class imbalances which occur when dealing with very rare objects.

3.2 Faster R-CNN network with ResNet50

We used the Faster R-CNN network with "ResNet50" backbone with FPN to detect the cars from both COCO dataset and DOTA dataset. COCO datasets contains large scales of photos for object

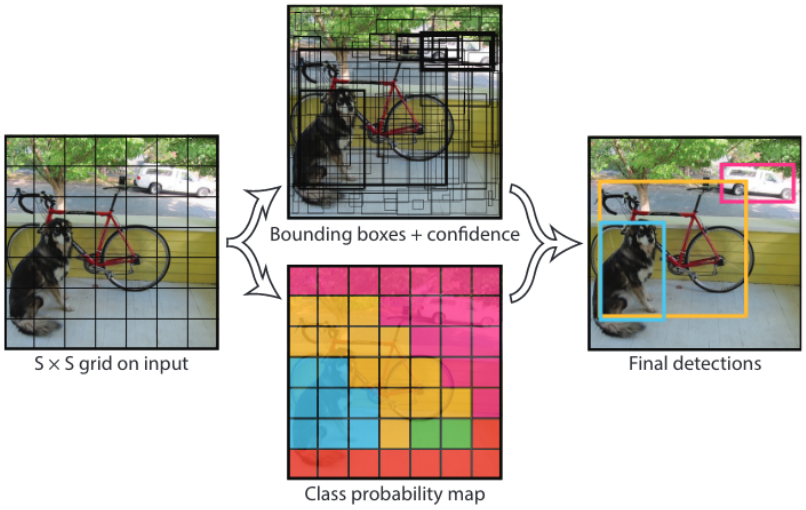

Figure 1: YOLO Bounding Box Production Algorithm (Image from (*1*))

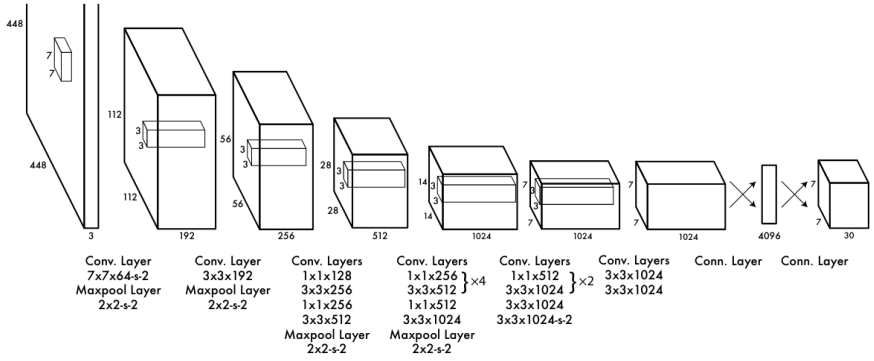

Figure 2: YOLOv1 Architecture (Image from (*1*))

detection, segmentation, and captioning. On the other hand, DOTA datasets contains satellite images annotated by an oriented bounding box. ResNet50 is a type of ConvNet/DCNN with the capacity to avoid vanishing gradient problems using the residual connection.It effectively avoids the vanishing gradient problem through its use of residual connections, enabling the training of deeper network architectures. FPN enhances the model's ability to manage multi-scale feature maps by maintaining consistent dimension sizes across different scales, thus forming a feature pyramid that is crucial for detecting objects at various scales. (He et al.)

Since we are using pretrained Faster R-CNN network with ResNet50 model, the convolution layer is fixed. There are 5 trainable backbone layer in the model. We are using 3 as default. There is a COCO V1 weights for backbones in the model, but the default pretained weights for backbones is chosen since we want to compare the difference performance on both dataset. We create horizontal bounding box annotations for the pretained model on both datasets.

The primary component of our model is the Region Proposal Network (denoted by RPN). RPN scans the feature maps and outputs region proposals that are likely containing objects. This strategy significantly reduces the number of locations where the search region needs to be, allowing the model to concentrate on probable object locations rather than performing conducting an expansive search. The regions of interest (RoI) outputted from RPN are then converted to a fixed sized feature set through the RoI Align operation. This operation ensures that all RoIs produce feature sets of same

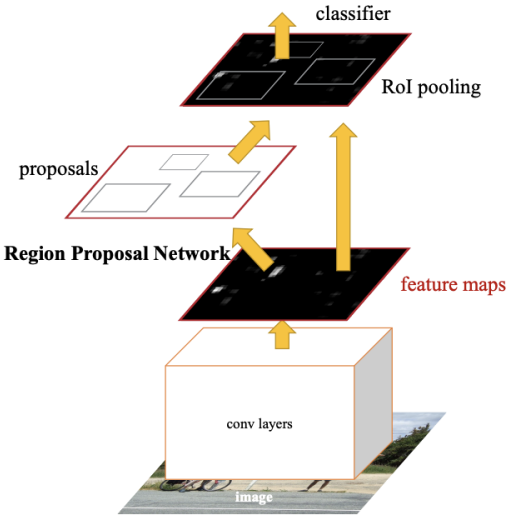

Figure 3: Faster R-CNN Schematic (Ren et al.)

size, preserving object's spatial information. The last step is creating feature vectors from each RoI (Ren et al.). These vectors are then passed through two components: the Bounding Box Regressor and a Classifier. The regressor refines the bounding box coordinates while the classifier determines whether a car exists in these bounding boxes.

Since we do not have enough resources(GPU) to train the model, we will provide theoretical results here. Theoretically, our model outputs the final detections marked by bounding boxes along with their corresponding confidence scores, which reflect the model's confidence whether it was a correct car detection or not. Once the model was fine-tuned based on COCO and DOTA dataset, it can be generalized to detect cars and other object scales in more dataset. Theoretically, the fine-tuned model should enhance the detection performance in scenes with varying object scales and complexities, thereby improving detection accuracy and reducing false positives. Additionally, the model's theoretical ability to learn discriminating features that capture a wide array of car appearances under different lighting and weather conditions suggests that it would perform well in real-world applications such as traffic monitoring and autonomous driving assistance systems. By leveraging transfer learning principles and the rich feature representations learned from comprehensive and diverse datasets like COCO and DOTA, the model is expected to offer high levels of precision and reliability in detecting cars across different environments.

The workflow of the model(Figure 1): from the input image to the ResNet50 FPN backbone; the RPN; the ROI Align; (Ren et al.) classification and bounding box regression; from raw image data to structured detection.

4 Empirical/Theoretical Results

4.1 Training and Validating YOLOv8-obb on DOTA and VALID

4.1.1 DOTA

DOTA (*3*) is a large scale challenge dataset for object detection in aerial images, with images taken from Google Earth and annotated by aerial imagery experts. We use DOTAv1, which contains 15 common categories, 2,806 images and 188,282 instances, annotated with oriented bounding boxes. The categories include plane, ship, large vehicle, small vehicle, bridge and harbour, among others. We conduct some training runs on the full datasets, and some on a restricted dataset including only the 6 categories mentioned above, removing images which don't have any of these six categories, for compatibility with VALID.

DOTAv1 is easily available as a base dataset in the ultralytics library which provides YOLO, and a custom dataset was written to download DOTAv1.5 to pytorch format. However, this dataset was limited to the horizontal bounding boxes, since lack of native pytorch support for oriented bounding boxes would have increase development times by too much. To restrict DOTAv1 to 6 classes, a custom dataset transformer was written to update the annotations and ultralytics dataset .yaml file. All images were resized to be 640x540 pixels, and the ultralytics OBB default data augmentations were applied to each image when it was reloaded as part of a mini-batch.

4.1.2 VALID

VALID (*4*) is a large scale challenge dataset of virtually generated aerial images and accompanying oriented bounding boxes for a variety of classes, including plane, ship, large vehicle, small vehicle, bridge and harbor, among others. This dataset was restricted to just these 6 simple classes for comparison with YOLO and to reduce the size of the dataset. Images with no valid annotations were excluded.

To use VALID, a custom pytorch dataset was written to download and format the images and annotations, and a torch to ultralytics method was written in this dataset to convert the dataset to the format required by ultralytics. All images were resized to be 640x540 pixels, and the ultralytics OBB default data augmentations were applied to each image when it was reloaded as part of a mini-batch.

4.1.3 Model Versions

The following results resulted from running three different versions of the `yolov8-obb` model. These are the same architecture, but with different numbers of weights. These models were:

- `YOLOv8n-obb`: Smallest model, with 3.1 million parameters and requiring 23.3 GFLOPs per evaluation.
- `YOLOv8s-obb`: Second smallest model, with 11.4 million parameters and requiring 76.3 GFLOPs per evaluation.
- `YOLOv8m-obb`: Medium sized model, with 26.4 million parameters and requiring 280.6 GFLOPs per evaluation.

I ran most models on the YOLOv8s-obb architecture, since this was a good compromise between model size and reduced computational requirements, allowing training batches of 32.

All models were run using automatic optimiser and hyper-parameter settings, which worked well, and training performance was strong, as will be detailed below.

4.1.4 MLOps

While YOLO is considered computationally tractable for an object detection model, training such a model is beyond the capabilities of a CPU on a consumer PC, which became clear after attempting to train the smallest model locally. To overcome this problem, I trained all YOLO models enumerated below on a remote machine accessed through `vast.ai`. I used a single NVIDIA RTX 4070 GPU with 24GB of VRAM for 2 hours on each run in a custom docker container which allowed all necessary packages and environment utilities to be immediately available on the remote machine, and allowed testing of training code on a simulated amd64 architecture before deployment to the training node.

The compute resources provided by the compute node were more than sufficient, and were a vast improvement over training on a consumer macbook with mps, which proved to be a slowdown relative to pure cpu training.

4.1.5 Results

All the models were run for 2 hours on the above workflow, the full table of results is shown in Table 1. Note that Precision, Recall and mAP50 were all calculated from the validation set with the last epoch weights after training/finetuning for two hours. Using the validation sample for testing here is valid, because the validation sample was not used during training for model selection.

There are a few clear takeaways. Firstly, prediction on VALID is much easier than on DOTA, which is mainly due to the nature of the images. Secondly, YOLOv8s does seem to be the right architecture

Architecture	Dataset	Pretrained	Precision	Recall	mAP50
YOLOv8n	DOTAv1-Full	No	0.60631	0.34389	0.36317
YOLOv8n	DOTAv1-Full	DOTAv1-Full	0.7517	0.45035	0.4975
YOLOv8s	DOTAv1-Full	No	0.69093	0.38877	0.42964
YOLOv8s	DOTAv1-Full	DOTAv1-Full	0.72842	0.48829	0.53598
YOLOv8m	DOTAv1-Full	DOTAv1-Full	0.49384	0.56364	0.41848
YOLOv8s	DOTAv1-6cat	No	0.71199	0.53842	0.56836
YOLOv8s	DOTAv1-6cat	VALID	0.72791	0.52192	0.5664
YOLOv8s	VALID	No	0.9891	0.95477	0.97671
YOLOv8s	VALID	DOTAv1-Full	0.9936	0.96431	0.98349

Table 1: Yolo Model Performance

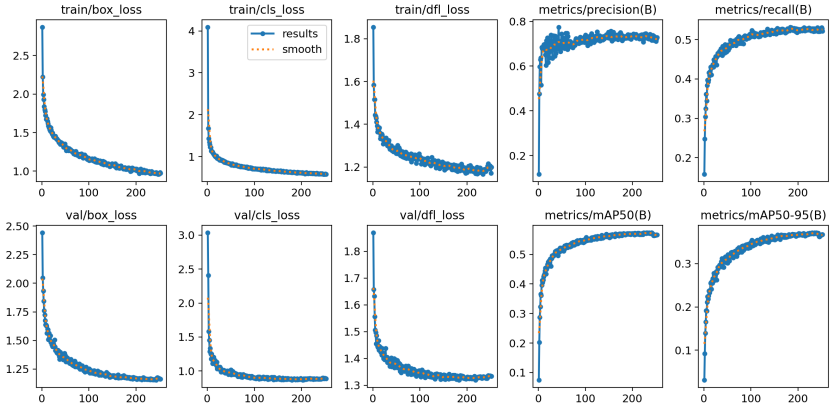

Figure 4: YOLOv8s - DOTAv1-6cat - VALID Results

for our needs in this case. Running YOLOv8m required reducing the minibatch size from 16 to 32 due to GPU memory constraints, which slowed down training considerably. Finally, pretraining on VALID does not seem to produce much benefit for making predictions on DOTA.

Finally, we can examine one training run in detail in figures 4 and 5, for the model of architecture YOLOv8s pretrained on VALID and trained on DOTAv1-6cat.

The results show good training performance and no signs of overfitting on any of the three loss components, while the PR curve shows that object detection is very bad on bridges, best on large vehicles and ok on small vehicles.

4.2 For graduate students, you can use this section to detail what each member of the group focused on

Michael Bennett: I came up with the idea for the project, sourced, collected and formatted the DOTA and VALID datasets, researched the YOLO model in detail and handled all MLOps and training related to the YOLO model. I wrote up the sections on the YOLO model and defined the key quantities in section 2.

Ruiqi Zhang: In our group project, I was responsible for the comprehensive research on existing models, encapsulated in the "Background/Related Work" section of our report. My contribution involved a thorough review and analysis of various object detection frameworks, such as YOLO, Faster R-CNN, SSD, RetinaNet, EfficientDet, Cascade R-CNN, and Deformable DETR. I explored their development history, architectural specifics, and performance characteristics. Additionally, I analyzed the limitations of these models in real-world scenarios. This research laid a solid foundation for the project by identifying key areas where existing methods could be further examined and understood in the context of satellite imagery.

Yawen Wei: In our project, I am responsible for building up the pseudocode for the Faster R-CNN model with ResNet50 and its writeup parts in the report. It covers the schematics of the pretrained

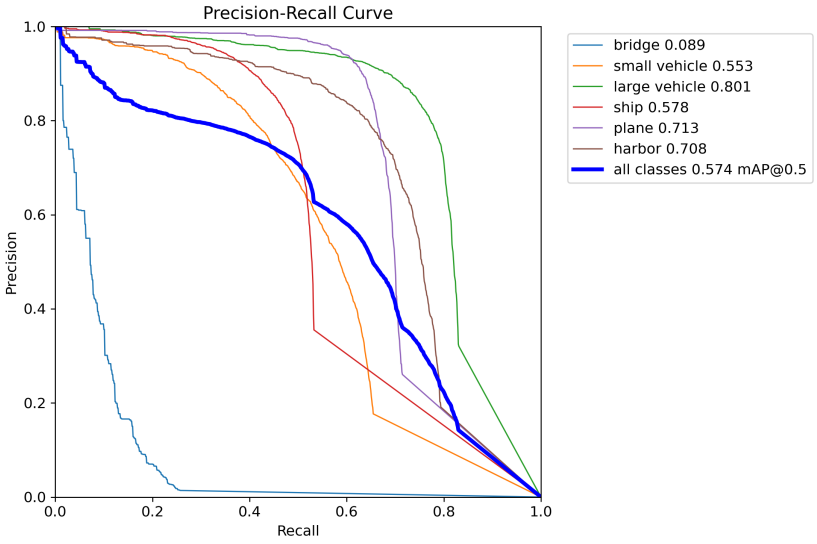

Figure 5: YOLOv8s - DOTAv1-6cat - VALID

Faster R-CNN, how we implement the fine-tuning model on COCO datasets and DOTA datasets, and generalization if we have theoretical results.

Ziang Li: In our group project, I played a role in organizing the LaTeX format and ensuring integration of the contributions from each member. I structured the document to highlight the complexities of our deep learning approach, refining the presentation of model architectures, training methodologies, and empirical results.

5 Conclusions/Future work

It was interesting to learn about state of the art methods for object detection. In the future, we would like to deeply examine more models. Additionally, in the context of the YOLO model, it would be interesting to do more analysis of transfer learning capabilities and vary the size of the input images more, to see whether that makes it easier for the model to pick up small objects like cars, and whether it can handle bridges better in such a situation.

References

References

1. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, June 8, 2015, (2024; `https://arxiv.org/abs/1506.02640v5`).
2. P. Jiang, D. Ergu, F. Liu, Y. Cai, B. Ma, *Procedia Computer Science*, The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19 **199**, 1066–1073, (2024; `https://www.sciencedirect.com/science/article/pii/S1877050922001363`) (Jan. 1, 2022).
3. J. Ding *et al.*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**, 7778–7796, arXiv: `2102.12219[cs]`, (2024; `http://arxiv.org/abs/2102.12219`) (Nov. 1, 2022).
4. L. Chen *et al.*, presented at the 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 2009–2016, (2024; `https://ieeexplore.ieee.org/document/9197186`).
5. R. Girshick, *Fast R-CNN*, Apr. 30, 2015, (2024; `https://arxiv.org/abs/1504.08083v2`).

[1] Cai Z, Vasconcelos N. Cascade r-cnn: Delving into high quality object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6154-6162.

[2] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.

[3] Jiang P, Ergu D, Liu F, et al. A Review of Yolo algorithm developments[J]. Procedia computer science, 2022, 199: 1066-1073.

[4] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer International Publishing, 2016: 21-37.

[5] Tan M, Pang R, Le Q V. Efficientdet: Scalable and efficient object detection[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 10781-10790.

[6] Wang Y, Wang C, Zhang H, et al. Automatic ship detection based on RetinaNet using multi-resolution Gaofen-3 imagery[J]. Remote Sensing, 2019, 11(5): 531.

[7] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." ArXiv.org, 2015, arxiv.org/abs/1512.03385v1.

[8] "Object Detection Using PyTorch Faster RCNN ResNet50 FPN V2." DebuggerCafe, 7 Nov. 2022, debuggercafe.com/object-detection-using-pytorch-faster-rcnn-resnet50-fpn-v2/.

[10] Ren, Shaoqing, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." ArXiv.org, 2015, arxiv.org/abs/1506.01497.