A Comparative Study of Deep Learning-Based Models for Object Detection in Remote Sensing Imagery

Andrew V. Coulson, W. Hoyt Thomas, Caixia Wang*

GeoComputing Laboratory
Department of Geomatics, University of Alaska Anchorage, USA -
(avcoulson, whthomas, cwang12) @alaska.edu

Keywords: Deep Learning, Machine Learning, Object Detection, Artificial Intelligence, Remote Sensing, Imagery.

Abstract

Object detection contributes significantly to advancing image interpretation and understanding. The advent of deep learning-based methods has significantly advanced this field. However, the distinctive characteristics of remote sensing images, including large directional variations, scale differences, and complex and cluttered backgrounds, pose considerable challenges for accurate target detection. In this work, we compare the detection accuracy and processing speed of several state-of-the-art models by detecting palm trees in optical satellite imagery. This work aims to explore how these models, adopted in many remote sensing applications, perform when applied to detect objects in overhead satellite images. Several models are selected from the single-stage and two-stage object detection families of techniques. Additionally, we use the timing results of the sliding window object detector to establish a baseline to compare different approaches. Our experiments demonstrate that two-stage detectors perform better in remote sensing contexts when detecting small, crowded objects, outperforming their single-stage counterparts. Future work includes extending this analysis to additional models, such as the multi-stage object detection family.

1. Introduction

Object detection is one fundamental and challenging problem in computer vision for image interpretation and understanding. The goal of object detection is to develop computational models and techniques to automatically identify (what) and localize (where) real-world entities accurately in an image. In recent years, the widespread adoption of convolutional neural networks (CNN) and GPU-accelerated deep-learning frameworks has led to a fresh perspective on the development of object detection algorithms. Many promising object detection algorithms rely on pre-trained deep learning models as their backbone, leveraging large-scale benchmark datasets to determine the majority of the hyperparameters of the model. The model is then transferred to specific applications by re-learning just the head or a limited depth of the model based on the application data(Hao, 2023). Transportation (pedestrian detection), security (facial recognition), medical (tumor identification), energy (site monitoring), government (disaster relief), and many other industries have successfully applied object detection models in image understanding tasks (Jiao et al., 2019). Compared to images in computer vision tasks that are captured from the ground perspective, remote sensing images (Fig. 1) collected from an overhead perspective by satellite or aircraft present unique challenges that pose issues when straight-forward transferring deep learning-based object detection methods. These challenges include different perspectives (bird view vs. ground view), significantly larger image sizes, and fewer details of target objects on remote sensing images (Li et al., 2020).

Significant efforts in photogrammetry and remote sensing have been made to develop deep learning-based object detection models, initially designed for computer vision tasks, to detect features (man-made or natural) from satellite or airborne imagery. For example, Chen et al., 2014 significantly improved upon deep convolutional neural networks (DNNs) to detect vehicles in satellite images by splitting outputs from the highest feature-extracting layers and passing them through additional convolution and pooling layers of varying sizes. This hybrid approach performed well when detecting objects with high variance in size, which can be an issue when transferring models pre-trained for image classification to remote sensing tasks. To employ object detectors on circular oil tanks, Zhang et al., 2015 used a hybrid approach to include contextual scene information (pipelines, shadows, additional oil tanks, and other oil infrastructure). After a line segment and elliptical arc detector were used to select initial object candidates, a buffered region around candidates was passed through a pre-trained CNN to improve classification. The results from CNN were used to filter out ellipsoids detected in non-oil tank storage areas and their experimental metrics demonstrated effective performance of detecting oil storage tanks in satellite images. Moreover, numerous developments have been integrated into commercial software, such as ArcGIS Pro, to streamline their utilization through user-friendly graphical interfaces. Nevertheless, the optimal selection of a model for detecting various types of objects remains unclear.

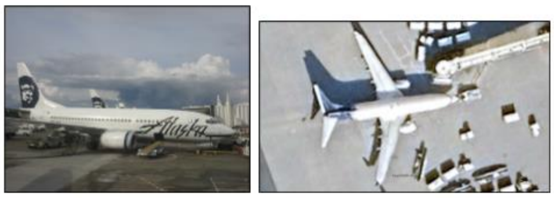
Figure 1. Left: Ground view (courtesy to Bill Abbot, Flickr CC BY-SA 2.0); Right: Overhead (bird's eye) view (Google Earth).

* Corresponding author

In this study, we assessed four widely used deep learning-based object detection models accessible in ArcGIS Pro to provide insights into the process of selecting the most suitable model for different remote sensing object detection applications. The models under evaluation include the Single Shot Detector (Liu et al., 2016), You Only Look Once v3 (Redmon & Farhadi, 2018; (Redmon et al., 2016), RetinaNet (Lin et al., 2017), and Faster R-CNN (Ren et al., 2015). Of these, Single Shot Detector (SSD), You Only Look Once v3 (YOLOv3), and RetinaNet are all single-stage detectors, while Faster R-CNN is a two-stage detector that has separate steps for identifying candidate regions of the image before the object detection step.

The two-stage detector identifies regions of different objects in the image (region proposals, thus the "R" in R-CNN) at its first stage. A second stage follows to determine what (if any) classes those objects belong to using a convolutional neural network (CNN). In essence, the two-stage detector effectively turns the object detection task into image segmentation followed by image categorization. Various algorithms can be applied to the first task, generating candidate regions (Girshick et al., 2014), however, the need to perform two separate and distinct tasks indicates that two-stage models would generally be slower than single-stage models. Faster R-CNN (Ren et al., 2015) achieves much higher speeds by using a neural network for detecting candidate regions that share convolutional layers (and their resulting feature maps) with the neural network performing object classification.

Single-stage detectors skip the step of detecting the candidate regions. Instead, they start with a default set of anchor boxes and use bounding box regression (Girshick et al., 2014) to output information about how to modify the height, width, and center location of the boxes to capture the target objects. All three single-stage models assessed in this study divide the image (or later feature maps) into a grid and generate many default bounding boxes (a.k.a. anchor boxes) at each grid cell. The boundaries of these anchor boxes can be outside of their parent grid cell. In YOLO, the starting position and aspect ratio of these anchor boxes are determined during training, while in SSD and RetinaNet the starting position is always the center of the grid cell, and only the default aspect ratio is determined during training.

All three single-stage models use bounding box regression to modify the anchor boxes' position and dimensions to capture a target object. In SSD, one network performs these bounding box regression calculations and the confidence that the box contains each of the target classes included in the model. In YOLO and RetinaNet, a separate subnetwork calculates the bounding box regression along with the confidence that the final box contains an object, while another calculates the confidence that a grid cell contains each of the classes included in the model. The final output is the bounding box location and dimensions, and the final confidence scores for each class are calculated by multiplying the confidence that the bounding box contains an object by the confidence scores that the bounding box belongs to each of the classes in the model. If the class with the highest final confidence score is greater than the threshold value defined when running the model, an object of that class is detected. To capture objects of different sizes relative to the image, SSD repeats its analysis of anchor boxes at several different scales of the starting image, while RetinaNet repeats its analysis of anchor boxes on convolutional feature maps at different levels of pooling(Lin et al., 2017; Liu et al., 2016; Redmon et al., 2016; Redmon and Farhadi, 2018). In general, single-stage object detection models perform better in remote sensing contexts. (Li et al., 2020)

2. Fine-tuning based Method

We adopted the workflow proposed by Julia Lenhardt (Lenhardt, 2023) to evaluate the aforementioned four popular deep learning-based object detection models. Each model performs coconut palm trees detection from the same multispectral (red, green, and blue) aerial imagery of a plantation in Kolovai, Tonga (Giovando, 2017) using the same manually digitized training data. All four models are based on transfer learning techniques (Zhuang et al., 2021), where pre-trained models are fine-tuned on the object detection data for the target domain. These pre-trained models have been trained for tasks different from coconut palm tree detection. For example, the backbone model, Resnet34 (He et al., 2016), used in the Single Shot Detector is 34 layers deep and was trained on the ImageNet Dataset that has more than 1 million images and 1000 classes. The architectures of deep learning models are typically structured as layers, each learning different features. Initial lower layers of the network learn very generic features, while the higher layers focus on very task-specific features. This hierarchical architecture allows us to utilize the pre-trained network by excluding its final layer which produces the final output in the pre-trained model. Furthermore, we can selectively retrain specific preceding layers to fine-tune the model for the target task, thereby achieving better performance and adaptability. Using our work (Wang and Coulson, 2023) for oil spill detection as an example, MobileNet (Howard et al., 2017) was used as the backbone model to detect oil spills in images by retraining the last 22 layers of the MobileNet model on the new, task-specific data. The weights in all previous layers are left locked and remain unchanged during the training process. This fine-tune-based approach requires significantly fewer resources compared to training the model from scratch. It capitalizes on prior learning from extensive datasets, reducing the training required to optimize the model for a specific task.

3. Performance Metrics

This work utilized the similar accuracy metrics as in the COCO challenge (COCO Consortium, 2015) to evaluate the model performance for object detection. They include Precision P, Recall R, $F1$ score, and Average Precision AP, calculated from detection results. We adopt the same metric definitions as outlined in the work by Padilla et al. (2020).

Three possible detection results can be found in the model output:

- True positives (TP): Correct prediction of ground truth;
- False positives (FP): Incorrect prediction where no ground truth was present;
- False negatives (FN): Undetected ground truth.

To determine if a prediction is correct or not, we compare the bounding box (BB) of the predicted object and the BB of the ground truth by calculating the intersection of the two BBs over their union (IOU, Figure 2), which is based on the coefficient of similarity of two sets of data (Jaccard, 1901). An IOU of 1 signifies a perfect agreement between the predicted BB and ground-truth BB. In practice, a specific threshold, t, is defined based on the application where an IOU $< t$ is considered incorrect and an IOU $\geq t$ is considered correct.

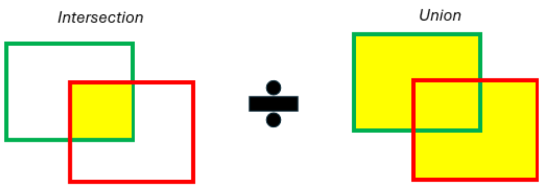

Figure 2. Intersection over Union (IOU) derived from ground truth (green) and predicted (red) BB.

Precision *P*, is a measure of how many true positives were identified and assesses how many detected objects are relevant. P is defined as:

$$P = \frac{\text{TP}}{\text{TP+FP}} = \frac{\text{TP}}{\text{total detections}}. \tag{1}$$

Recall *R*, is a measure of how well the model detects true positives and assesses how relevant are the detected objects. R is defined as:

$$R = \frac{\text{TP}}{\text{TP+FN}} = \frac{\text{TP}}{\text{total ground truths}}. \tag{2}$$

F1 score, the weighted average of precision and recall, is a composite metric commonly used to represent the model performance. F1 is defined as:

$$\text{F1} = \frac{P+R}{0.5*(P+R)}. \tag{3}$$

A robust model would obviously need to have very high true positives and very low false negatives. However, a model with higher precision (fewer FP) may miss many positives, resulting in a higher FN and thus a lower recall, *R*. Conversely, a model has more positives, resulting in higher recall, but the FP may also increase, yielding a low precision. (Padilla et al., 2020). Therefore, there is a trade-off between precision and recall when selecting confidence values for prediction results. F1 score and average precision (AP) are two metrics to measure how well a model approaches this ideal of perfection by incorporating both precision and recall into a single number. Considering the curve generated by the precision vs recall, a good model will have a high area under the curve (AUC), indicating high precision and high recall. However, the precision-recall curve often zigzags which makes calculating the AUC difficult. Instead, interpolating is implemented to smooth out the curve by replacing the precision values at each recall level with the maximum precision value to the right of it (Figure 3).

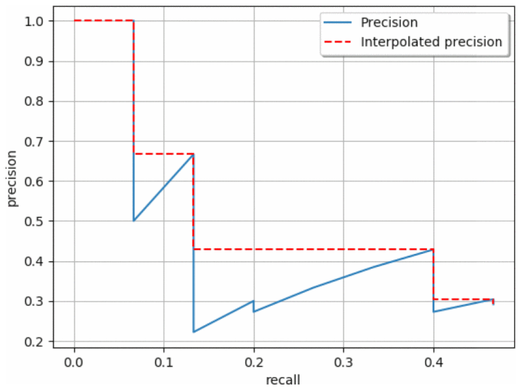

Figure 3. A precision-recall curve with interpolated points (Padilla et al., 2020)

Afterwards, using all-point interpolation the AP can be defined according to the work of Padilla et. Al. (2020) as:

$$AP = \sum_n (R_{n+1} - R_n) * P_{interp}(R_{n+1}), \tag{4}$$

where P_{interp} (R_{n+1}) is the maximum precision whose recall value is greater than or equal to R_{n+1}, and n is the number of every recall level. A higher AP will signify a better model. Moreover, to evaluate the predictive validity of each model, a cross-validation method (Browne, 2000) is adopted by repeating the assessment after reversing the roles of the training and verification data sets.

In addition to the above metrics, we compare the time of each model taken to train and detect coconut palm trees under the same computation environment as a simple measure for computational efficiency.

4. Experiments and Discussion

We employed ArcGIS Pro 3.1.1 with the deep learning framework libraries (3.1) installed to assess the four models for coconut palm tree detection. The experiments were carried out on a desktop PC featuring a 12th-generation Core i9 Intel CPU, and one Nvidia 3050 GPU (8GB).

The training data used to fine-tune the pre-trained models consisted of 1,198 manually identified palm trees (Fig. 4) from seven areas selected in the work of Lenhardt (2023) from the multispectral (red, green, and blue) aerial imagery of a plantation in Kolovai, Tonga (Giovando, 2017). These training data were used to create 3,088 training image chips of 448 by 448 pixels, with a stride length of 128 pixels, each including at least one of the training palms. Training image chips without any training palms were discarded.

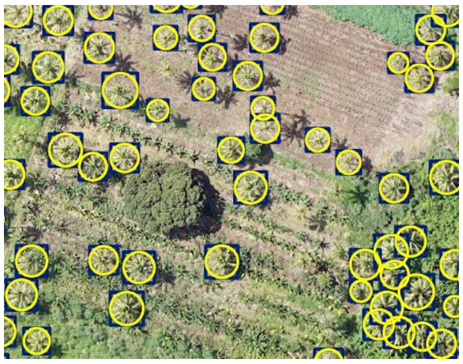

Figure 4. An example of some training data.

The pre-trained models (Resnet 34 for SSD, DarkNet 53 for YOLOv3, Resnet50 for RetinaNet, and Resnet50 for FasterRCNN) were fine-tuned on the palm trees as a single class for 50 epochs and a batch size of 8. All other parameters, including learning rate, were left at the default values in the ArcGIS geoprocessing tool. Palm trees were detected using the fine-tuned models with a 0.2 confidence threshold. Postprocessing with the Non-maximum suppression (NMS) technique was applied to filter duplicate detections and select the most relevant detected objects. The accuracy of the results was assessed by comparing the outputs from each model to a test data set of 787 manually identified palm trees, located in the seven test areas adjacent to but not overlapping the areas of the training data (Figure 5). All detected objects in the output that intersected the test areas were exported and compared to the test data based on performance metrics. We use 0.5 as the threshold *t* for IOU to determine a detected object to be a correct detection (*TP*) of one of the palm trees in the test data (ground truth).

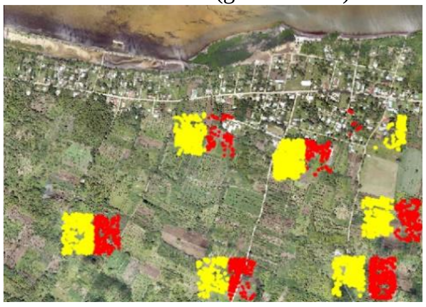
Figure 5. Test data (red) from seven test areas adjacent to the areas of training data (yellow)

Model outputs were scored based on their number of true positives (*TP*, the model identified a palm tree where there was a palm tree), false positives (*FP*, the model identified a palm tree where there was not a palm tree), and false negatives (*FN*, the model did not detect a palm tree where there was one). Figure 6 illustrates that FasterRCNN outputs the highest *TP* and lowest *FN* among the four detectors. SSD and RetinaNet produce relatively high FP, indicating they identified more incorrect palm trees where there were no palm trees. YOLOv3, on the other hand, yields the lowest FP but performs less effectively in terms of *TP* and *FN*. Based on these scores, the precision and recall metrics were derived (Table 1).

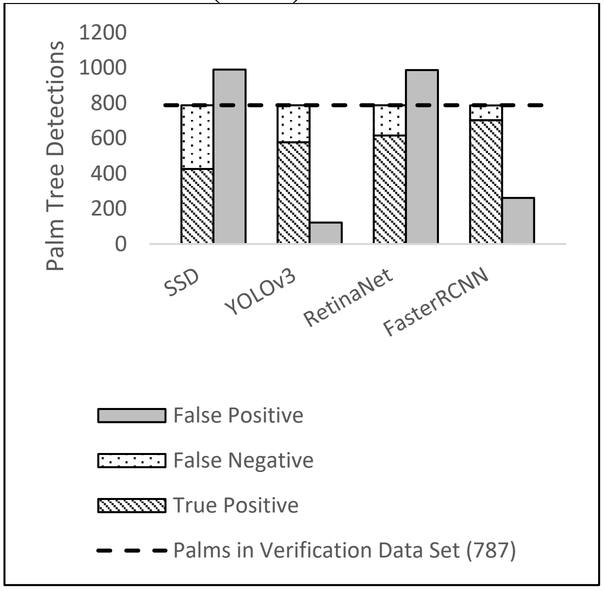

Figure 6. Comparison of model outputs of the four object detection models.

As shown in Table 1, the two-stage detector FasterRCNN performed the best, achieving the highest recall (0.893) and second-highest precision (0.729). Conversely, the single-stage detector YOLOv3 boasted the best precision (0.825) but ranked third in recall (0.733). When evaluating a combination of recall and precision, FasterRCNN outperformed YOLOv3 overall, as evidenced by F1 scores (0.803 vs. 0.777) and AP values (0.754 vs. 0.621). SSD exhibited the poorest performance across all metrics, with the lowest precision (0.301), recall (0.541), F1 (0.387), and AP (0.260) among the four object detectors. Nonetheless, it's worth noting that SSD boasts the fastest training and inference times.

By swapping the roles of the training and verification data sets, we performed cross-validation on the four object detectors. The results, presented in Table 2, verified the same relative rankings of the four models observed in Table 1.

Model	***P***	***R***	**F1**	**AP**	***t*₁**	***t*₂**
SSD	0.301	0.541	0.387	0.260	**1:18**	**0:15**
YOLOv3	**0.825**	0.733	0.777	0.621	1:45	**0:15**
RetinaNet	0.384	0.781	0.515	0.494	1:30	0:20
FasterRCNN	0.729	**0.893**	**0.803**	**0.754**	1:55	0:26

Table 1. Comparison of model outputs of 4 object detection models using identical parameters and training data (training (t_1) and detection(t_2) are in hours and minutes).

Model	***P***	***R***	**F1**	**AP**	***t*₁**	***t*₂**
SSD	0.336	0.538	0.413	0.292	**1:01**	0:17
YOLOv3	**0.863**	0.740	0.796	0.663	1:14	**0:15**
RetinaNet	0.441	0.790	0.566	0.511	1:13	0:20
FasterRCNN	0.742	**0.866**	**0.799**	**0.778**	1:33	0:27

Table 2 Results of cross-validation on the four detectors (training (t_1) and detection(t_2) times are presented in hours and minutes).

5. Conclusions

The superior performance of the two-stage detector (FasterRCNN) in this study aligns with the findings of (Groener et al., 2019), who observed that two-stage object detectors, specifically FasterRCNN, outperformed single-stage detectors in detecting small, crowded objects, such as cars in their study or palm trees in ours. Interestingly, their research reported a much smaller performance gap between RetinaNet and FasterRCNN compared to our results. For example, they noted an AP of 0.661 for RetinaNet and 0.685 for FasterRCNN, whereas our results showed values of 0.494 and 0.754, respectively. While we maintained consistency in most training and detection hyperparameters across different models in our study, this suggests the potential for optimizing hyperparameters tailored to specific models. Further research is required to define these optimizations, which may deviate from default settings in commercial software like ArcGIS Pro. We also found that increasing the training batch size to better take advantage of available hardware capacity actually decreased model performance, possibly due to problems with the automatic learning rate selection in the ArcGIS tool.

In this study, we chose not to perform more post-processing on the output of these detectors besides non-maximum suppression. However, further post-processing to remove detected object features based on geometry or other characteristics would likely reduce the FP score and thus improve precision for all models. Even though AP is the optimal metric for the model performance, the relative importance between recall and precision will depend on the priorities of the specific application.

Acknowledgments

This work was supported by the National Science Foundation under Award 2119689.

References

Browne, M.W., 2000. Cross-Validation Methods. *Journal of Mathematical Psychology,* 44(1), 108–132.

Chen, X., Xiang, S., Liu, C.L., Pan, C.H., 2014. Vehicle detection in satellite images by hybrid deep convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 11, 1797–1801.

COCO, Detection Evaluation, 2015. https://cocodataset.org/#detection-eval (accessed 3.8.24).

Giovando, C., 2017. Kolovai UAV4R Subset (OSM-Fit). OpenAerialMap. https://tiles.openaerialmap.org/5a28639331eff4000c380690/0/5b1b6fb2-5024-4681-a175-9b667174f48c/wmts (accessed 4.20.23).

Girshick, R., Donahue, J., Darrell, T., Malik, J., Berkeley, U.C., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation, *In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 580–587. doi.org/10.1109/CVPR.2014.81

Groener, A., Chern, G., Pritt, M., 2019. A Comparison of Deep Learning Object Detection Models for Satellite Imagery, 2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR). doi.org/10.1109/AIPR47015.2019.9174593

Hao, J., 2023. Deep learning-based medical image analysis with explainable transfer learning, 2023 International Conference on Computer Engineering and Distance Learning, CEDL 2023. Institute of Electrical and Electronics Engineers Inc., pp. 106–109.
doi.org/10.1109/CEDL60560.2023.00029

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Las Vegas, NV, pp. 770–778. doi.org/10.1109/CVPR.2016.90

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. doi.org/https://doi.org/10.48550/arXiv.1704.04861

Jaccard, P., 1901. Etude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin de la Societe Vaudoise des Sciences Naturelles 37, 547–579. doi.org/10.5169/seals-266450

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., Qu, R., 2019. A survey of deep learning-based object detection. *IEEE Access*, 7, 128837–128868.

Lenhardt, J., 2023. Use Deep Learning to assess palm tree health. ESRI, Inc. learn.arcgis.com/en/projects/use-deep-learning-to-assess-palm-tree-health/ (accessed 3.3.24).

Li, K., Wan, G., Cheng, G., Meng, L., Han, J., 2020. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing,* 159, 296–307.

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P., 2017. Focal Loss for Dense Object Detection, *In: Proceedings of the IEEE International Conference on Computer Vision*. IEEE Computer Society, pp. 2980–2988. doi.org/10.1109/TPAMI.2018.2858826

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. SSD: Single shot multibox detector, in: Computer Vision -- ECCV 2016. Springer Verlag, pp. 21–37. doi.org/10.1007/978-3-319-46448-0_2/FIGURES/5

Padilla, R., Netto, S.L., da Silva, E.A.B., 2020. A Survey on Performance Metrics for Object-Detection Algorithms, *In: 2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, pp. 237–242. doi.org/10.1109/IWSSIP48289.2020.9145130

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection, *In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 779–788. doi.org/10.1109/CVPR.2016.91

Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement.
doi.org/https://doi.org/10.48550/arXiv.1804.02767

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, in: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 28.

Wang, C., Coulson, A., 2023. Fine Tuning MobileNet Neural Networks for Oil Spill Detection, I-GUIDE Forum 2023. New York. doi.org/10.5703/1288284317671

Zhang, L., Shi, Z., Wu, J., 2015. A Hierarchical Oil Tank Detector with Deep Surrounding Features for High-Resolution Optical Satellite Imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(10), 4895–4909.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q., 2021. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109, 43–76.