AirWall: Malicious Drones Detection using YOLO

Alexey Tuzikov
Stanford University
atuzikov@stanford.edu

Abstract

We have explored implementation of YOLOv9 model to the increasingly important and urgent task of malicious drone detection. For that we use a rare dataset of (mostly) small object drone images in the sky taken from the ground during various day/season/weather conditions. Apart from the baseline YOLOv9 model we experiment with scaling the image during trainign and applying Slicing Aided Hyper Inference for better small object detection. The baseline delivers quite good results, but SAHI shows better performance, possibly because of better feature learning trough enchanced representation of small drones, to which the dataset is skewed. Further work might layer connections further for adapting FPN to small object detection.

1. Introduction

Proliferation of drones is changing the landscape of possible terrorist and malicious attacks. Recently, drones carrying explosives were used to attack civilian objects in places as diverse as Saudi Arabia[1], Burkina-Faso [2] and Russia[3]. In many such cases, the result was huge economic cost of destroyed infrastructure[4], disruption of airport operations[5] and loss of civilian lives[6]. Unlike airplanes, drones often do not require runway and other complex infrastructure for launch, and so can be launched from any place, including from the territory of the country they are aimed to attack. Heavier drones can travel long distances (sometimes up to 1000 kilometers) without being detected. Current detection systems were built for detecting planes - large, metallic, flying quite high objects, whereas drones are often made from materials that are hard for radars to detect, are quite small, and are able to fly at low altitudes below the reach of radars. However, visual systems systems may be particularly well suited for detecting malicious drones.

In this proiect we aim to build a system that would detect drones using images from CCTV. Speficifally, the model has image as inpute and uses CNN (Convulational Neural Network) to predict presence of a drone and a rectangular bounding box around it.

2. Related work

2.1. General object detection

Before proliferation of deep learning-based detection methods, Deformable parts models (DPM)[7] were state of the art, and used a sliding window approach where the classifier is run at evenly spaced locations over the entire image, comparing each window's features with pre-trained features of objects.

With the rebirth of Convolutional neural nets in 2012[8], deep learning-based methods diverged into two branches: 1) Two-stage detectors, and 2) One-stage detectors.

Two-stage detectors started with R-CNN[9] that extract a set of object proposals (object candidate boxes) by selective search [10]. Search generates potential bounding boxes, a CNN extracts features, an SVM scores the boxes, a linear model adjusts the bounding boxes, and non-max suppression eliminates duplicate detections. Each stage of this complex pipeline must be precisely tuned independently and the resulting system is very slow, taking more than 40 seconds per image at test time [11]

The techniques were further refined by avoiding repeated computing feature maps using Spatial pyramid pooling [12]. Fast R-CNN[11] enabled us to simultaneously train a detector and a bounding box regressor under the same network configurations, while Faster R-CNN [13] introduced a Region Proposal Network (RPN) to propose regions of interest. Further introduction of Feature Pyramid Networks (FPN) [14] improved detection by running a detector several layers deep, but the family of methods still falls short of real-time detection speed.

The first one-stage detector in the deep learning era, YOLO [15], applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region simultaneously. Single Shot MultiBox Detector (SSD) [16] introduced the multi-reference and multiresolution detection techniques, which significantly improves the detection accuracy of a one-stage detector, especially for some small

objects. DETR [17] applied transformers architecture to object detection.

2.2. Object detection applied to drones

There have been some attempts to build video-based drone detection systems. Most of them build on top of YOLO, SSD or Faster R-CNN models. [see [18] for a review of attempts]

2.3. Small object detection

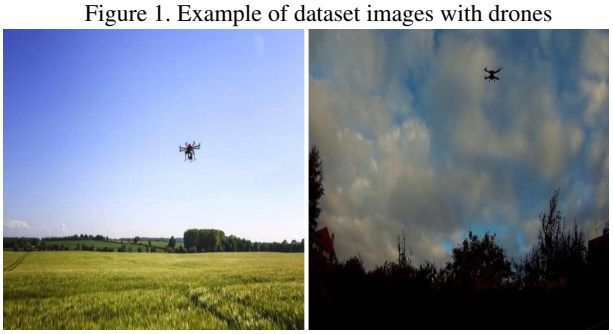
For the task at hand, many drones can be quite far away from a CCTV and thus be quite small relative relative to the image size. The current prevailing feature extractors usually down-sample the feature maps to diminish the spatial redundancy and learn high dimensional features, which unavoidably extinguishes the representation of tiny objects.

Cheng et al 2023[19] provide a taxonomy of small object detection methods, which consists of 6 main approaches: 1) sample-oriented (data augmentation, etc); 2) attention-based 3) feature-imitation (enriching the regional features of small objects) 4) context-modelling 5) scale-aware (scale-specific detectors, feature fusion) 6) focus and detect. In this paper we focus on the first branch - data augmentation.

3. Data

We use a "Real World Object Detection Dataset for Quadcopter Unmanned Aerial Vehicle Detection"[20]. The dataset is specifically designed for detection of flying drones from the ground in real-life circumstances with varying time of the day, season and landscape backgrounds. The set contains 51446 train and 5375 (640x480) images and 55539 bounding boxes in PASCAL-VOC format, divided into train and test sets. See Figure 1.

Figure 1. Example of dataset images with drones



In the training dataset, there are more small objects than large ones: 41% of objects are small (area < 1024), 36% are medium ($1024 <$ area < 9216), and 24% are large (area > 9216) as specified by the COCO challenge (Figures 2, 3).

The training set doesn't contain negative examples (without drones), but the valid and test sets do. However, the negative examples in the test set seem to have quite different distributions from the positive examples, as they contain images of interior, faces and those few exterior negative images there are are taken during a sunny summer day in the European latitude (see Figure 4).

Figure 2. Heatmap of object (drone) box center locations (x,y) and height and width of their boxes

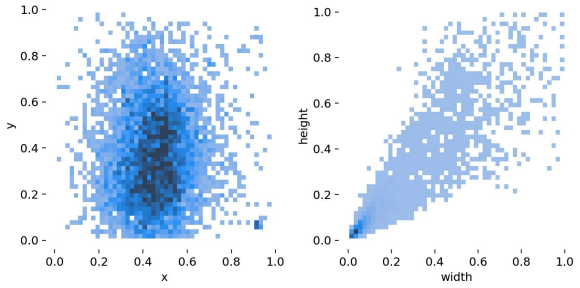
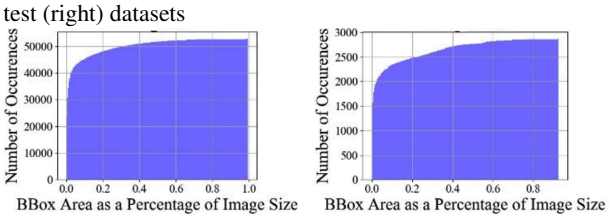


Figure 3. Cumulative distribution of box areas for train (left) and test (right) datasets



Due to infrastructure and training time limitation we randomly select 6592 training images from the train set and 4098 validation images from the test set (2625 or 64% of which have drones and the rest are negative examples) for the purpose of our analysis.

We pre-process the dataset by stretching the resulution to 640x640 and converting to YOLOv9 format.

Figure 4. Negative examples (without drones) in the test set have quite different features from the positive ones, as the contain images of interior, faces, and exteriors are sunny summer days

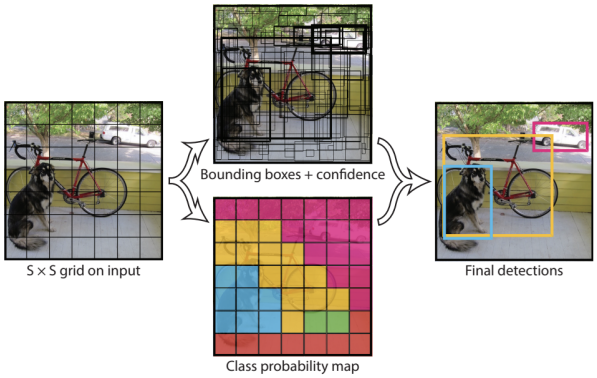
4. Methods

4.1. Baseline

For the baseline, we use the YOLOv9 (small) model[21], which is a state-of-the-art CNN based image detection model that "predicts bounding boxes and class probabilities directly from full images in one evaluation".

YOLO divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S × S × (B*5 + C) tensor.

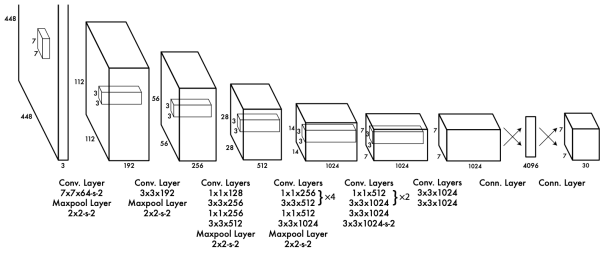
Figure 5. High-level YOLO atchitecture



Then, YOLO uses non-maximal suppression to retain only the largest prediction, to avoid duplicate predictions between adjacent cells.

The Original YOLO network architecture has 24 convolutional layers followed by 2 fully connected layers (see Figure 6).

Figure 6. YOLO original architecture



During training, the following loss function is optimized:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{\text{obj}} \Big((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \Big) +$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{\text{obj}} \Big((\sqrt{w_i} - \sqrt{\hat{w}_i})^2 +$$

$$(\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \Big) + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{\text{obj}} \Big(C_i - \hat{C}_i \Big)^2 +$$

$$+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{\text{noobj}} \Big(C_i - \hat{C}_i \Big)^2 +$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i,j}^{\text{obj}} (p_i(c) - \hat{p}_i))^2$$

Where $\mathbb{1}_i^{\text{obj}}$ encodes whether any object appears in cell i. Note that if there is no object in a cell we do not consider any loss from the bounding box coordinates predicted by that cell. In this case, there is no ground truth bounding box so we only penalize the associated probabilities with that region.

YOLOv5 and higher use a two-stage object detector, and seperates the model into the input, backbone, neck, and head sections. The backbone model is a feature extractor that uses a classification model such as VGG16. The neck is a feature aggregator that collects feature maps from differing stages from the backbone model. The head is an object detector that reports whether a particular grid has an object in it but not necessarily what class of object it is. YOLOv9 introduces a few improvements over previous versions of YOLO, such as Programmable Gradient Information (PGI) and the concept of Reversible function, that aid in preserving essential data across the network's depth, ensuring more reliable gradient generation and, consequently, bettesr model convergence and performance.

Figure 7. YOLOv5 architecture

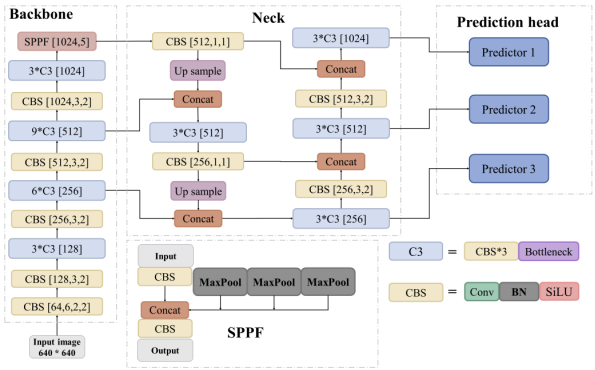
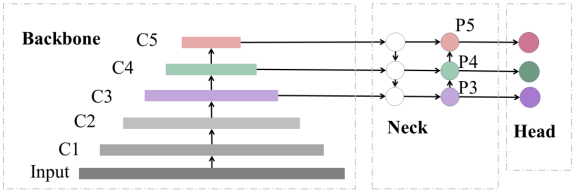


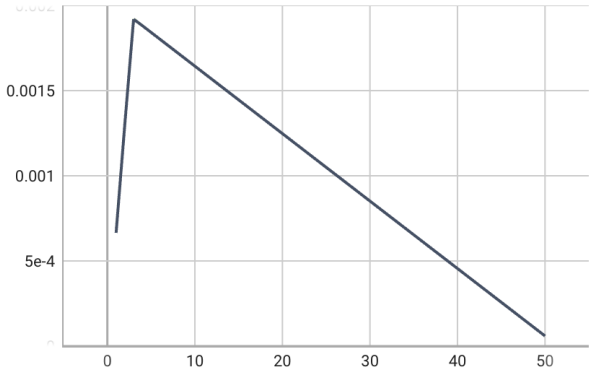
Figure 8. FPN and PANet structure of YOLO starting from v5.



Learning rate: Learning rate schedule is as follows: For the first epoch we set the learning rate to $7 * 10^{-4}$, then double it for the 2nd epoch, triple for the 3rd, and then linearly decrease it to $6 * 10^{-5}$ until the last 50th epoch (see the Figure 9). If we start at a high learning rate our model often diverges due to unstable gradients.

Optimizer: We use AdamW optimizer that due to momentum and weight decay to speed-up and stabilize learning and improve generalization.

Figure 9. Learning rate schedule by epoch



Mini-batch size: We used minibatch size of 16 as recommended by OpenCV [22], which is a good balance between stability of training and speed in terms of effective utilization of GPU.

Leveraging of existing code: We utilize existing YOLOv9 implementation by Ultralytics [23]. We do not adjust architecture of the model as we explore the data augmentation branch of small object detection field.

4.2. Experiment 1: Higher resolution images for better small object detection

If as we saw in 2.3 the down-sampling to learn high dimensional features leads to extinguishment of the representation of tiny objects, the logical solution is to increase the image size so that small objects occupy more pixels in the image after down-sampling, hopefully improving their representation. Moreover, higher resolution can lead to better small object detection through better spatial context.

The experiment is therefor to increase the image size during training from 640x640 to 960x960.

4.3. Experiment 2: Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection (SAHI)

Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection (SAHI) [24] is a slicing aided inference and fine-tuning pipeline for small object detection.

In finetuning, the dataset is augmented by extracting patches from the images and resizing them to a larger size. During inference, image is divided into smaller patches and predictions are generated from larger resized versions of these patches. Then these predictions are converted back into original image coordinates. Optionally, predictions from full inference can also be added.

Due to time and compute resources limitation we use only the inference option of SAHI, slicing images 5x5.

We utilize SAHI implementation provided by the authors of the original paper [25], although the code is quite buggy and poorly documented.

Figure 10. SAHI: Slicing aided fine-tuning

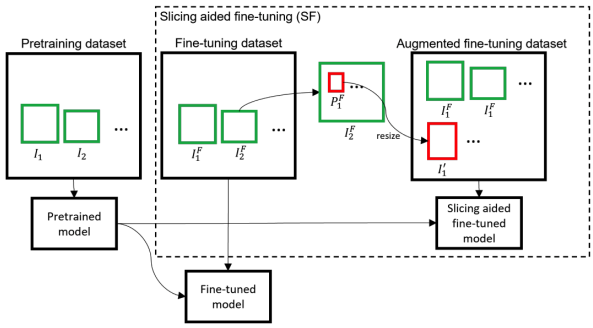
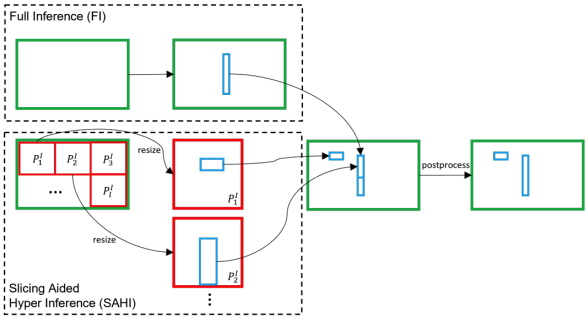


Figure 11. SAHI: Slicing aided hyper inference



5. Results

5.1. Metrics

We use F1 score and mAP as our main performance metrics, defined as follows:

$$F1 = 2\frac{precision * recall}{precision + recall}$$

where

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$mAP = \frac{1}{C}\sum_{C} AP$$

$mAP = \sum_{k=0}^{n-1} (Recalls(k) - Recalls(k+1)) * Precisions(k)$
mAP is a weighted sum of precisions at each confidence threshold where the weight is the increase in recall. In our case there is only 1 class (drone), so $mAP = AP_{drone}$

5.2. Qualitative analysis of errors

We have reviewed errors of the baseline model to confirm that most of them come from failing to recognize small drones or mistaking small objects other than drones as drones (See Figure 12 and Figure 13). It confirmed our hypothesis that small object might indeed be a problem for the baseline.

Figure 12. Examples of baseline model errors:
left: one drone recognized as two (false positive)
right: two drones recognized as one (false negative)

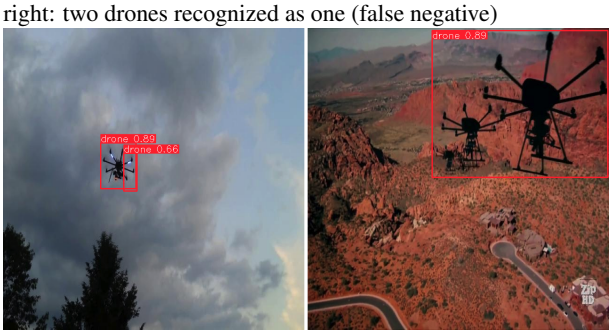


Figure 13. Examples of baseline model false positive mistakes:
left: small street lights mistakenly recognized as a drone
right: a hole in the ground mistakenly recognized as a drone

5.3. Comparison of models

We see that the Experiment 1 of increasing resolution of the image from 640x640 to 960x960 during training leads to worse results, but Experiment 2 of slicing images into 5x5 non-overlapping cells actually improve the results.

Method	mAP@50
Baseline	0.971
Baseline higher img size	0.966
Baseline + SAHI	0.975

Table 1. mAP Results

Figure 14. F1-confidence curves of the SAHI (left) and Baseline (right) methods

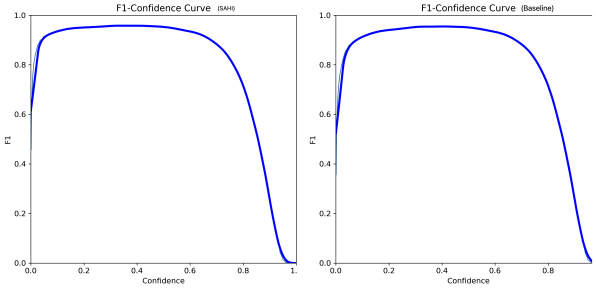


Figure 15. mAP50 over 50 epochs

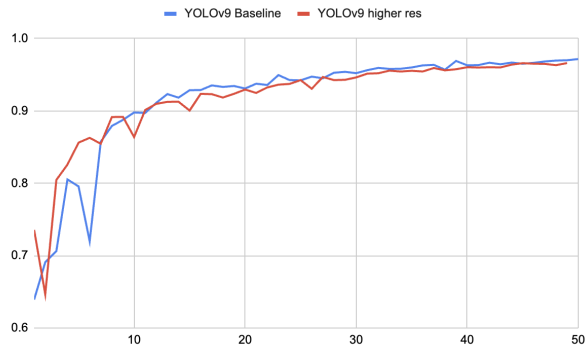


Figure 16. Precision over 50 epochs

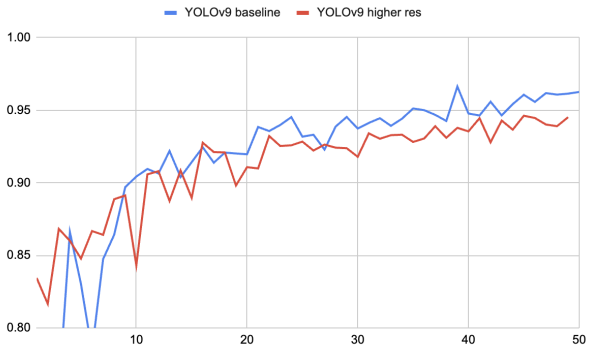


Figure 17. Recall over 50 epochs

5.4. Discussion of results

Failure of experiment 1 (Scaling the image during training 1.5x): If the model is trained on larger images but evaluated on the original resolution (640x640), there may be a mismatch in the feature scales and receptive fields that the model has learned versus what it needs to apply during inference. This discrepancy can lead to suboptimal performance as the model might not generalize well from the higher resolution training data to the lower resolution inference data.

Another reason is that YOLO models use predefined anchor boxes and feature maps at different scales to detect

objects. When the input image size is scaled up significantly, the predefined anchor boxes and feature maps might not align well with the new object sizes and positions. This misalignment can reduce the accuracy of object localization and classification since the anchor boxes and feature maps are no longer optimally tuned for the new image scale.

Relative success of Experiment 2 (SAHI): Why the increase in performance of the experiment over the baseline is not that significant, it may come from better feature learning trough enchanced representation of small drones, to which the dataset is skewed.

6. Conclusion and further work

We have explored implementation of YOLOv9 model to the increasingly important task of drone detection. The baseline model provided quite good performance out-of-the-box. This might be due to the neck that is implemented in YOLO models starting from v5, which is a feature aggregator that collects feature maps from differing stages from the backbone feature extractor model, what allows it to perform well at detecting objects of different sizes, whereas data augmentation methods we explored focus primarily on small objects at the expense of larger ones.

The experiment of data augmentation in the form of increasing image size during training to actually worse results, possibly due to mismatch between train and test feture maps and change in anchor box size and its respective loss due to scaling.

Applying Slicing Aided Hyper Inference to the image slightly improved performance over the baseline, possibly because of better feature learning trough enchanced representation of small drones, to which the dataset is skewed.

Further work might might include pre-training rather than pure inference using the SAHI method, as well as adress the false positive issue by exploring scale-aware feature fusion models that control information that deep layers deliver to shallow layers, for adapting FPN to small object detection.

References

[1] "Two major saudi oil installations hit by drone strike, and u.s. blames iran," *New York Times, https://www.nytimes.com/2019/09/14/world/middleeast/saudi-arabia-refineries-drone-attack.html*.

[2] "Burkina faso: Drone strikes on civilians apparent war crimes," *Human Rights Watch, https://www.hrw.org/news/2024/01/25/burkina-faso-drone-strikes-civilians-apparent-war-crimes*.

[3] "The refinery in slavyansk-on-kuban partially suspended operations after a drone attack," *RIA Novosti, https://ria.ru/20240427/npz-1942684243.html*.

[4] "Russia oil refining curbed by flood as drone damage persists," *Bloomberg, https://www.bloomberg.com/news/articles/2024-04-22/russian-oil-refining-hampered-by-floods-as-drone-damage-persists,*.

[5] "Almost 50 jets fail to land in moscow, redirected to other airports, says aviation agency," *TASS, https://tass.com/economy/1662831*.

[6] "Ukrainian drones kill six, injure 35 in russia's belgorod region, governor says," *Reuters, https://www.reuters.com/world/europe/ukrainian-drones-kill-six-injures-35-russias-belgorod-region-governor-says-2024-05-06/,*.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[9] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 09 2013.

[10] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[11] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 346–361, Springer International Publishing, 2014.

[13] N. Hao and K. Rajakani, "3d object detection from point cloud based on deep learning," *Wirel. Commun. Mob. Comput.*, vol. 2022, jan 2022.

[14] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2016.

[15] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, 2015.

[17] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *ArXiv*, vol. abs/2005.12872, 2020.

[18] R. A. Zitar, M. A. Al-Betar, M. Ryalat, and S. Kassaymeh, "A review of uav visual detection and tracking methods," *ArXiv*, vol. abs/2306.05089, 2023.

[19] G. Cheng, X. Yuan, X. Yao, K. Yan, Q. Zeng, X. Xie, and J. Han, "Towards large-scale small object detection: Survey and benchmarks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–20, 2023.

[20] M. Pawełczyk and M. Wojtyra, "Real world object detection dataset for quadcopter unmanned aerial vehicle detection," *IEEE Access*, vol. 8, pp. 174394–174409, 2020.

[21] C.-Y. Wang, I.-H. Yeh, and H. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *ArXiv*, vol. abs/2402.13616, 2024.

[22] "Learn opencv blog: Fine-tuning yolov9 models on custom dataset,"

[23] "Ultralytics yolov9, url=https://docs.ultralytics.com/models/yolov9/usage-examples,"

[24] F. C. Akyon, S. Onur Altinuc, and A. Temizel, "Slicing aided hyper inference and fine-tuning for small object detection," in *2022 IEEE International Conference on Image Processing (ICIP)*, IEEE, Oct. 2022.

[25] "Github of the authors of the sahi method,"