Benchmark on real-time long-range aircraft detection for safe RPAS operations

Víctor Alarcón[1], Pablo Santana[1], Francisco Ramos[1], Francisco Javier Pérez-Grau[1], Antidio Viguria[1], and Aníbal Ollero[2]

[1] Advanced Center for Aerospace Technologies (CATEC), Seville, Spain, vmalarcon@catec.aero,
[2] Robotics, Vision and Control Group (GRVC), Universidad de Sevilla, Seville, Spain

Abstract. The growing market in Remotely Piloted Aircraft Systems (RPAS) and the need for cost-effective "Detect and Avoid (DAA)" systems are critical issues up to date towards enabling safe beyond visual line of sight (BVLOS) operations. In hopes of promoting earlier threat detection on DAA systems, we benchmark several object detection algorithms on multiple graphical processing units for the concrete DAA use case. Two state-of-the-art "real-time object detection" and "object detection" model sets are trained using our CENTINELA dataset, and their performances are compared for a wide range of configurations. Results demonstrate that one-stage architecture YOLO variants outperform ViT on all tested hardware in terms of mean average precision and inference speed despite their architecture complexity gap. Additional resources are available to the reader at *https://github.com/fada-catec/detection-for-safe-rpas-operation*.

Keywords: real-time object detection; convolutional neural networks; visual transformers; unmanned aerial systems; detect and avoid

1 Introduction

1.1 Motivation

Over the last years, the Remotely Piloted Aircraft Systems (RPAS) business has grown significantly. The European RPAS market witnessed a growth of 16.6% according to the compound annual growth rate from 2017 to 2021, surpassing USD 3 billion in 2020 [1]. In addition, the RPAS market is projected to expand rapidly at 21.9% through 2032 in the European region [2].

The changes in regulations made by the European Union Aviation Safety Agency (EASA) support the deployment of RPAS in various sectors, such as infrastructure, agriculture, transport, entertainment, and security [3]. Nevertheless, one of the critical points to reaching the full potential of the RPAS market is the transition from operations within Visual Line Of Sight (VLOS) to Beyond VLOS (BVLOS), which offer a higher added value. The Single European Sky ATM Research (SESAR) Joint Undertaking estimates that approximately 50%

of the professional market will focus on BVLOS operations in rural environments for applications such as linear infrastructure inspection and monitoring, precision agriculture, and surveillance [4]. However, factors such as strict regulations, security and safety concerns, and lack of trained pilots, are anticipated to hamper the market growth during the forecast period.

Thus, the need for a Detect And Avoid (DAA) system for RPAS is introduced, which refers to integrating sensors to recognize the environment and navigate safely regarding unforeseen encounters and potential collisions. The problem is divided into two tasks: detecting objects close to the aerial vehicle, and managing the maneuver to avoid a collision. DAA systems' architecture can vary, and multiple types of sensors, including acoustic, visual or radars, can be used [5]. When performing BVLOS flights, the main roadblock is the absence of cost-effective systems approved by the aeronautical authorities. This is a subject of research activities, and different approaches have been tested up to date, including complex sensors and powerful but heavy computer systems. The difference in terms of cost-effective systems between visual sensors and others is a compelling argument for focusing the work on object detection on images. Given the application, an RGB camera providing high-resolution images (i.e. around 20 megapixels) is needed to acquire images with the maximum availability in terms of field of view and number of pixels for object detection, which tends to be a challenge for objects at distances larger than 1 km. In terms of computational needs, this application demands a computer with enough graphical processing capacity and advanced algorithms to perform real-time object detection.

The key enabler of this work is the lack of references in the state of the art for algorithms and their performances to be embedded on small, cost-effective, portable and energy-efficient single-board computers (SBC). The focus is laid on effectiveness in terms of speed and accuracy for the early detection of manned aircraft using RGB cameras and embedded processes over single board computers. Results allow going deeper into the design of DAA systems to detect distant aircraft in real-time, hence fostering BVLOS operations for RPAS.

1.2 Related work

Compared to traditional Computer Vision (CV) techniques, Deep Learning (DL) achieves greater accuracies in tasks such as image classification, semantic segmentation and object detection. Since DL approaches are trained rather than deterministically programmed, applications using this approach can exploit the tremendous amount of data available in today's systems, and often require less expert analysis and hand-tuning. DL also provides superior flexibility because the models can be re-trained using custom datasets for other use cases, contrary to CV algorithms, which tend to be more domain-specific [6]. That is the main reason to go deeper into DL than classic CV approaches.

Real-time small object detection using high-resolution RGB images and DL algorithms is an open issue in DAA systems, as well as in other applications such as autonomous driving [7] or video surveillance [8]. In terms of accuracy, recent research on object detection has already achieved significant progress,

but it is still challenging when the target appears as a small percentage of the entire image. It is common to find works that face the problem of detecting objects using Convolutional Neural Networks (CNN), such as Faster R-CNN [9] or Single Shot Detector [10]; however, input images are downsampled using pooling layers, hence targets could easily be filtered out in the final feature map. Other approaches try to solve this issue by using innovative architectures based on Transformers with Attention [11] [12] or Feature Pyramid Networks (FPN) [13]. Although these methods address this problem and improve state-of-the-art performance in small object detection, their time complexity is higher for accurate real-time detection.

It is necessary to prepare detectors that address our specific scenario to ensure a reasonable accuracy in our application. An extensive aerial object detection dataset with objects at various scales is presented at [14]. Still, its great amount of largely-sized items deters the detection of smaller ones, making it impossible to evaluate the performance of detectors on this issue. Other aerospace-related works aim to compare different algorithms in real-time small object detection, as presented in [15] and [16], but both are focused on a different scenario.

The absence of an extensive collection of images with aircraft flying far from the visual sensor makes it difficult to focus the research activities, as there are no results from previous works. As a scientific novelty, this work compares supervised neural network architectures for object detection by training models using a proprietary dataset, where targets are small aircraft flying at least 6000 feet away. In addition, multiple computing boards with different processing capabilities are used to test the performance of the trained models.

The rest of the paper is structured as follows. In section 2, the selected DL object detection algorithms are presented. Section 3 describes the software tools, the hardware platforms, the dataset and the performance metrics used in the experiments. In section 4, the results of each model evaluation are presented, and the implications of the results are discussed. Finally, section 5 presents the study's main conclusions and future lines of work.

2 Object Detection Algorithms

In order to solve the DAA problem using DL algorithms, one of the critical aspects is choosing a network architecture that optimally balances detection accuracy and computational cost. While the nature of the DAA problem favors lightweight DL models to approach real-time performance, detecting small-sized aeroplanes at lengthy distances demands high-resolution imagery in which targets might cover only a few pixels.

Taking into account [17], some architectures have been selected regarding two state-of-the-art benchmarks called "Object Detection" and "Real-Time Object Detection". In the first one, a general ranking prioritizing inference accuracy is presented, while on the other one, a fast time inference while maintaining a base level of accuracy is given. Despite the absence of inference times in the "Object Detection" benchmark, our work is set to provide enough inference speed due

to fast in-time hardware evolution and cost decrease of processing components. These rankings have been tested and validated using the MS COCO dataset [18], one of the most widespread free dataset used for new DL strategies.

The selected architectures are based on the You Only Look Once (YOLO) family, initially defined in [19], and the Visual Transformers (ViT) neural network types [20]. YOLOv architectures are considered the fastest DL object detection methods due to their architecture which prioritizes inference speed by producing the output without intermediary region proposals. ViT architectures are focused on a different paradigm since they avoid recurrence and rely entirely on an attention mechanism to establish global dependencies between input and output, without using sequence-aligned convolutions, prioritizing accuracy over inference speed. For each type of architecture, a simple and a more complex neural network have been selected, with the idea of testing them on multiple board computers with different Graphics Processing Units (GPU).

In terms of CNNs, YOLOv4 [21] variants are considered the fastest real-time neural network architectures for object detection with a base level of accuracy. YOLOv5 [22] architecture builds on YOLOv4 including some improvements to run better off-the-shelf; though no research paper has been made available, its promising experimental results have motivated its inclusion in our study. YOLOv4-tiny and YOLOv5N are the light versions considered, while YOLOV4-P6 and YOLOv5S are the implementations for more complex computers.

Concerning ViT architectures, DETR [23], Swin-transformer [24] and DyHead [25] have high scores in the benchmark. DAB-DETR and Swin-s are selected as light architectures from DETR and Swin-Transformers, respectively, and deformable DETR, Swin-t and DyHead as the complex architectures. In particular, the proposed ViT arquitectures are the following: a DAB-DETR, its multiscale version DAB-Deformable and the Dynamic Head (DyHead) architectures using the same ResNet50 backbone. In addition, three configurations of the Swin-transformer architecture including the two-stage detector Mask RCNN combined with Swin-T and Swin-S backbones, and a combination of the one-stage detector RetinaNet with Swin-T backbone.

The nature of the proposed DAA system demands a trade-off between accuracy and inference speed, so we consider it essential to compare both available options in the state-of-the-art, which there is no doubt about their relevance to the scientific community regarding the object detection challenge.

3 Experimental setup

This section discusses technical details towards software and hardware selection, dataset construction and model configuration. Related material is available at *https://github.com/fada-catec/detection-for-safe-rpas-operation*.

3.1 Hardware Platforms

Our experiments have been carried out on different hardware platforms to compare the performance of our models in a varied range of specifications. Since

the detection algorithms should optimally be run in real-time, it is interesting to obtain performance metrics using various devices, as such processing can be performed either *in situ* or remotely. In the former case, light single board computers with limited GPUs are deployed in the operating area. In the latter, more powerful computer equipment has been considered taking into account the recent advances in cloud computing and high-bandwidth communications with low latencies, enabling high-end servers for real-time data processing.

Regarding light computing, we target an array of NVIDIA SBCs, which provide great computing capabilities on compact and energy-efficient equipment [26]. These are delivered through different boards with a broad performance range, from which we select the increasingly powerful set consisting of the Jetson Nano, Jetson TX2 and Xavier NX.

In the middle range, we include a laptop with an NVIDIA RTX A2000 GPU. While such card is neither suitable to be deployed *in situ*, nor powerful enough to be considered a cloud computing solution, its study becomes relevant towards early development stages of DL projects, which usually take place on average machines with mid-end GPUs.

Finally, we assess two high-end GPU machines geared with an NVIDIA GeForce RTX 2080Ti and an RTX 3080 GPU, which stand for remote cloud computers. Though commercial servers that process information using DL algorithms typically provide higher capabilities, these still possess a decent amount of power and represent a common self-hosted solution for research teams to have as a shared resource. These two machines will also be used to train our models.

3.2 Software Tools

Regarding software, it is useful to select a common DL framework to execute training and model evaluations. PyTorch [27] has been selected, since it covers all our target models, which are run through intermediary tools that use it as a backend.

One-stage CNN architectures have been supported by three different open-source repositories: ScaledYOLOv4 [28] for YOLOv4-P6, PyTorch_YOLOv4 [29] for YOLOv4-tiny, and Yolov5 [22] from Ultralytics for YOLOv5N and YOLOv5S. On the other hand, the open-source toolbox MMDetection [30] has been used to work with the ViT-NN models: DAB-DETR, DAB-Deformable, DyHead, Mask RCNN Swin-T, Mask RCNN Swin-S and RetinaNet Swin-T.

To test our models within a common set of performance metrics, these are evaluated with the official MS COCO validation API [31], by first using the aforementioned tools to perform inference on a test split of the data, and then using MS COCO's to compare those against its ground-truth labels. Besides overall precision results, MS COCO also computes metrics by dividing the target data in three size groups: small for an area of $32 \times 32px^2$, large for an area over $96 \times 96px^2$, and medium elsewhere. Such size segregation refers to the original images, without regard to the resizing they undergo at inference time.

3.3 Datasets

In order to benchmark the models defined in section 2, we have crafted a proprietary dataset composed of images of small aircraft taken at different field tests, namely the *CENTINELA* dataset. Such field tests took place in five different airfields in which a few distinctive landscapes were captured during flight tests. Natively, these images have a resolution of $5472\times3648\ px^2$. They were manually labelled using [32], being $125\times80\ px^2$ the average size of aircraft labels.

In Fig. 1, the left column shows random samples of these images, and each row belongs to a different scenario. The main pitfall is the similarity among samples within the same field test, as each location contains only a different set of landscapes over which the aircraft appear. Since many images are visually similar (except for the small aircraft within the image itself), performing a train-validation-test split by randomly shuffling all our images would lead to all three splits being very similar. Hence, this would yield overly-optimistic yet unrealistic detection performance metrics. On the other hand, splitting the images by field test makes our training set very monotonous, so that trained models can generalize to neither test nor validation sets.

To overcome these two issues, we featurized the images of each field test using a pre-trained low-resolution VGG-16 network, and then used those features to cluster the images using the K-means algorithm. We then shuffled the clusters of all field tests to achieve a 70:15:15 split for train, validation and test subsets respectively. An overview of this splitting strategy can be seen in the rest of Fig. 1, which shows a random preview of samples from different clusters and the three final splits. According to the MS COCO standard sizes, our test split is composed of 189 small, 881 medium and 533 large instances.

3.4 Model configuration

Our detectors must process image sizes of 20 megapixels in real-time. Large image sizes would not only hinder inference speed when deployed, but also increase training costs. Besides being time-wise expensive, training on very big images can also limit the choice of batch size.

Since GPUs have a limited amount of memory, increasing the image size would restrict to training on smaller batches. Small batch sizes detriments training efficiency, as the gradient is not so well averaged across samples. Besides slower processing, this would lead to longer training times, since loss convergence is harder to reach. On the other hand, a big batch size performs less gradient updates per epoch, requiring larger learning rates that may lead to instabilities. The latter is, however, not usually an issue within object detection, where larger batch sizes are advisable [22].

Real-time models were trained on a square resolution of $1280px$. Such image size was deemed enough for the labelled aircraft not to become unrecognisable pixel blobs, while allowing the batch size to be higher than one picture in our training computer for our heaviest target model. After this resize, average label

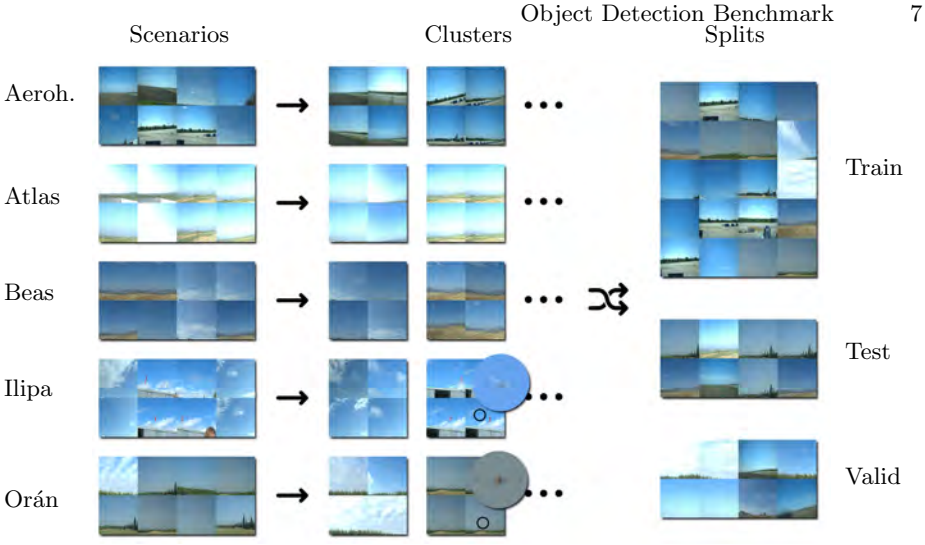

Fig. 1. *CENTINELA* dataset construction overview. Images gathered from different scenarios (left) were clustered by similarity (middle). Those clusters were finally shuffled to assemble three landscape-rich yet heterogeneous splits (right).

size approached $30\times30px^2$. Following best practices [22], the batch size was set as big as possible after the image size was established.

On the other hand, non-real-time models inherit their input resolution from the FPN standard [33], which comes by default in [34]. Because our images are taller than wide, they get resized to a height of $800px$ while keeping their $3:2$ aspect ratio, thus leaving a network input image resolution of $1200\times800px^2$. Such resolution is on par with the one used for our real-time models, still allowing small aircraft to be identifiable with an average label size of $27\times17px^2$. Again, the batch size was set at the highest value allowed by the training hardware.

Table 1 holds a summary of the training setup for all target models.

4 Results and discussion

4.1 Performance Metrics

To obtain an evaluation of the accuracy and the inference speed, we must select the metrics on which to base our conclusions. Object detection challenges typically use their own metrics to evaluate the proposed task. Researchers who want to evaluate their work using different datasets need to either implement their version of the metrics or adhere to an existing standard.

When measuring accuracy, [17] proposes the use of MS COCO as the most popular dataset to establish their ranking, and then evaluates Mean Average Precision (mAP) metrics, a metric calculated with the help of several other metrics such as Intersect over Union (IoU), confusion matrix, precision and recall.

Model	Input-size	Batch-size	Machine
yolov4-p6	1280×1280	2	RTX 2080Ti
yolov5l	1280×1280	4	RTX 2080Ti
yolov4-tiny	1280×1280	24	RTX 2080Ti
yolov5n	1280×1280	28	RTX 2080Ti
swin-t	1200×800	2	RTX 3080
swin-s	1200×800	2	RTX 3080
dyhead	1200×800	3	RTX 3080
retina-swin	1200×800	3	RTX 3080
detr	1200×800	4	RTX 3080
defor	1200×800	1	RTX 3080

Table 1. Input image size (px^2), batch size and training GPU for all target models.

Despite using a proprietary dataset, our work is based on the assumption of results published by this source, so we have considered using the same metrics.

The case of measuring inference speed is different. The most common scenario is to provide the results in completely processed frames per second (FPS), and the inference time per frame in milliseconds (ms). Therefore, FPS and MS are the metrics selected in our work.

4.2 Inference speed comparison

Table 2 shows the results of testing each model on each hardware platform, where the higher speed is highlighted in bold. The results prove that one-stage detectors perform better than two-stage detectors due to their architecture type. YOLOv models are more appropriate for real-time tasks since their inference time is generally lower than those using ViT. Specifically, YOLOv tiny models are faster than others due to their lightweight architecture. By contrast, large YOLOv architectures and ViT models perform poorly on single board computers, or cannot be used due to a lack of memory (e.g. Jetson Nano). Still, executing real-time operations for the studied use case is feasible using medium and high-processing graphic boards.

4.3 Inference accuracy comparison

The results for all models are shown in Table 3, and the highest values are highlighted in bold. Real-time networks lead the accuracy scores, with the most accurate model being YOLOv4-P6. If we focus on comparing the real-time architectures, some behaviours worthy of study are found:

– The size accuracies indicate that smaller objects are harder to detect, and that YOLOv4 models have more difficulty dealing with them versus YOLOv5 variants. An extreme example is YOLOv4-tiny, which gets 0% of mAP on

Name	Height *px*	Nano *ms/fps*	TX2 *ms/fps*	Xavier *ms/fps*	A2000 *ms/fps*	2080Ti *ms/fps*	3080 *ms/fps*
yolov4-P6	1280	–	1600/0.63	670/1.5	87/12	43/23	33/30
yolov5L	1280	–	740/1.4	310/3.2	40/25	23/43	18/56
yolov4-tiny	1280	560/1.8	140/7.1	64/16	8.8/110	**6.0/167**	**4.5/220**
yolov5n	1280	**470/2.1**	**90/11**	**60/17**	**7.8/130**	7.3/140	6.9/150
dyhead	800	–	1700/0.59	1600/0.63	170/5.9	82/12	63/16
swin-s	800	–	2700/0.37	1300/0.77	120/8.3	62/16	46/22
swin-t	800	–	1800/0.56	1400/0.71	120/8.3	68/15	47/21
retina-swin	800	–	1600/0.63	1300/0.77	130/7.7	66/15	45/22
defor	800	–	1800/0.56	1500/0.67	150/6.7	77/13	51/20
detr	800	–	**710/1.4**	**640/1.6**	**74/14**	**38/26**	**28/36**

Table 2. Inference speed per image of all different models on the target hardware.

small objects. On the contrary, the results reflect that YOLOv4 architectures perform faster on large objects.

- Comparing YOLOv5 variants, YOLOv5-L has slightly lower mAP and AP50 scores than YOLOv5-N, which is interesting since the former has a more complex architecture, which should allow it to learn more features and significantly outperform the latter. This result can be explained by a slight overfitting, to which larger networks are more vulnerable, although YOLOv4 models do not share such pattern. In any case, YOLOv5-L is more accurate for higher IoU thresholds.
- Across different box scales, YOLOv5-N seems to perform better with large objects, while YOLOv5-L shows better mAPs on small and medium objects. The best mAP is for the medium size group, the most abundant in our test set.
- Comparing YOLOV4 on large objects, although YOLOv4-P6 is still ahead, the 3 points difference in mAP shows that simpler architectures like YOLOv4-tiny are close in performance to more complex architectures.
- The fact that the accuracies of YOLOv5-N are so close to those of YOLOv4-P6 suggests that small models can keep up with large ones when the number of classes is low.
- YOLOv5-L and YOLOv5-N perform better than YOLOv4 variants on smaller objects, keeping a similar mAP, thus generating a case worth studying.

5 Conclusions and future work

This paper compares the top-ranked neural network architectures according to [17] for detecting small aircraft in high-resolution images in real-time. We present the outcomes of typical real-time-oriented architectures (YOLOv family) and good-accuracy performance-oriented architectures (ViT). The keys of the result

Name	Height *px*	mAP *all*	AP50 *all*	AP75 *all*	mAP *small*	mAP *medium*	mAP *large*
yolov4-P6	1280	30.2	**76.9**	15.1	18.5	30.3	35.4
yolov5L	1280	26.2	68.3	14.7	**23.7**	28.4	27.3
yolov4-tiny	1280	22.2	57.4	13.7	0.00	19.4	32.4
yolov5n	1280	26.7	70.2	14.2	22.7	26.4	30.3
dyhead	800	21.7	55.0	15.7	13.9	21.1	22.6
swin-s	800	**32.4**	60.3	**32.3**	0.00	**33.0**	**39.8**
swin-t	800	23.3	50.7	19.1	0.00	24.4	28.1
retina-swin	800	23.1	58.6	14.0	13.5	23.7	28.5
defor	800	14.2	42.5	7.1	8.4	11.9	20.5
detr	800	16.3	63.1	3.4	9.7	14.3	23.3

Table 3. Average precision metrics (%) for all models on the dataset. Respectively: image height, mean average precision, average precision at IoU 50%, average precision at IoU 75%, plus mean average precisions for small, medium and large bounding boxes.

are based on two standard performance metrics, mAP for inference accuracy and time metrics for inference speed. The results clearly show that real-time algorithms perform better than ViT concerning the use-case related to this work, represented by our proprietary dataset.

Proposing the use of ViT architectures was motivated by the good results of other scientific researchers according to their practical use cases. However, the poor results presented compared to YOLOv architectures will be subject of future work, as some possibilities are open to study. Increasing the quality and extension of the dataset, updating the input data configuration of architectures in the training phase, or combining backbones with more precise features extractors are good examples of the open paths to continue this work. These proposed items are also valid for real-time object detection networks, as some discussed behaviours are not entirely comprised.

Finally, as DL techniques are subject to continuous update, the emergence of new architectures has to be considered to be included in the benchmark, as they could improve the results of real-time object detection metrics in terms of accuracy and inference speed, offering better performances in the use case proposed in this work.

Acknowledgements. This work has been partially supported by the OMICRON project, funded by the EU H2020 programme under grant agreement 955269, and CEL.IA, a Cervera Network for applied artificial intelligence, funded by the Spanish government through CDTI (CER-20211022).

References

[1] Graphical Research. *Europe Commercial Drone Market Forecast 2027*. [Online; accessed 30 Aug. 2022]. URL: https://www.graphicalresearch.com/

industry - insights / 1016 / europe - commercial - drone - unmanned - aerial - vehicle-UAV-market.

[2] Fact.MR. *Europe Drone Market Outlook (2022-2032)*. [Online; accessed 30 Aug. 2022]. URL: https://www.factmr.com/report/europe-drones-market.

[3] EASA. *Civil drones (unmanned aircraft)*. [Online; accessed 30 Aug. 2022]. URL: https://www.easa.europa.eu/domains/civil-drones.

[4] Single European Sky ATM Research. *European Drones Outlook Study*. [Online; accessed 30 Aug. 2022]. URL: https://www.sesarju.eu/sites/default/files/documents/reports/European_Drones_Outlook_Study_2016.pdf.

[5] Jorge Mariscal-Harana et al. "Audio-based aircraft detection system for safe rpas bvlos operations". In: *Electronics* 9.12 (2020), p. 2076.

[6] Kohei Arai and Supriya Kapoor. "Advances in computer vision". In: *Conference proceedings CVC*. Springer. 2019, p. 104.

[7] Yingfeng Cai et al. "YOLOv4-5D: An Effective and Efficient Object Detector for Autonomous Driving". In: *IEEE Trans. Instrum. Meas.* 70 (), pp. 1–13.

[8] Sudan Jha et al. "Real time object detection and trackingsystem for video surveillance system". In: *Multimed. Tools Appl.* 80.3 (), pp. 3981–3996.

[9] Shaoqing Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[10] Wei Liu et al. "SSD: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[11] Xingkui Zhu et al. "TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios". In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)* (2021), pp. 2778–2788.

[12] Jing Lian et al. "Small object detection in traffic scenes based on attention feature fusion". In: *Sensors* 21.9 (2021), p. 3031.

[13] Ziming Liu et al. "HRDNet: high-resolution detection network for small objects". In: *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2021, pp. 1–6.

[14] Gui-Song Xia et al. "DOTA: A large-scale dataset for object detection in aerial images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3974–3983.

[15] Xian Sun et al. "FAIR1M: A benchmark dataset for fine-grained object recognition in high-resolution remote sensing imagery". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 184 (2022), pp. 116–130.

[16] Yi Wang et al. "Remote sensing image super-resolution and object detection: Benchmark and state of the art". In: *Expert Systems with Applications* (2022), p. 116793.

[17] Papers with Code. *A free and open resource with Machine Learning papers, code, datasets, methods and evaluation tables*. [Online; accessed 30 Aug. 2022]. URL: https://paperswithcode.com.

[18] Tsung-Yi Lin et al. "Microsoft COCO: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[19] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[20] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[21] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "Scaled-yolov4: Scaling cross stage partial network". In: *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*. 2021, pp. 13029–13038.

[22] Ultralytics. *YOLOv5*. [Online; accessed 30 Aug. 2022]. URL: https://github.com/ultralytics/yolov5.

[23] Nicolas Carion et al. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[24] Ze Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.

[25] Xiyang Dai et al. "Dynamic head: Unifying object detection heads with attentions". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 7373–7382.

[26] NVIDIA. *Sistemas integrados NVIDIA para las máquinas autónomas de la próxima generación*. [Online; accessed 30 Aug. 2022]. URL: https://www.nvidia.com/es-es/autonomous-machines/embedded-systems.

[27] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[28] Wong Kin-Yiu. *ScaledYOLOv4*. [Online; accessed 30 Aug. 2022]. URL: https://github.com/WongKinYiu/ScaledYOLOv4.

[29] Wong Kin-Yiu. *PyTorch_YOLOv4*. [Online; accessed 30 Aug. 2022]. URL: https://github.com/WongKinYiu/PyTorch_YOLOv4.

[30] Kai Chen et al. "MMDetection: Open mmlab detection toolbox and benchmark". In: *arXiv preprint arXiv:1906.07155* (2019).

[31] cocodataset. *COCO API*. [Online; accessed 30 Aug. 2022]. URL: https://github.com/cocodataset/cocoapi.

[32] Roblox. *ImageLabel*. [Online; accessed 30 Aug. 2022]. URL: https://developer.roblox.com/en-us/api-reference/class/ImageLabel.

[33] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[34] OpenMMLab. *mmdetection*. [Online; accessed 30 Aug. 2022]. URL: https://github.com/open-mmlab/mmdetection.