How High can you Detect? Improved accuracy and efficiency at varying altitudes for Aerial Vehicle Detection

Rafael Makrigiorgis*, Christos Kyrkou*, Panayiotis Kolios*

Abstract—**Object detection in aerial images is a challenging task mainly because of two factors, the objects of interest being really small, e.g. people or vehicles, making them indistinguishable from the background; and the features of objects being quite different at various altitudes. Especially, when utilizing Unmanned Aerial Vehicles (UAVs) to capture footage, the need for increased altitude to capture a larger field of view is quite high. In this paper, we investigate how to find the best solution for detecting vehicles in various altitudes, while utilizing a single CNN model. The conditions for choosing the best solution are the following; higher accuracy for most of the altitudes and real-time processing (> 20 Frames per second (FPS)) on an Nvidia Jetson Xavier NX embedded device. We collected footage of moving vehicles from altitudes of 50-500 meters with a 50-meter interval, including a roundabout and rooftop objects as noise for high altitude challenges. Then, a YoloV7 model was trained on each dataset of each altitude along with a dataset including all the images from all the altitudes, and several training and evaluation experiments were conducted. Overall, the best method for achieving optimal trade-off between accuracy and inference speed is to training on a mixed dataset of multiple altitudes and tune the inference speed to the smaller image size.**

I. Introduction

Unmanned aerial vehicles (UAVs) have been increasingly used in a variety of applications, such as power infrastructure inspection [1], search and rescue [2], traffic monitoring [3], and emergency monitoring [4]. One of the key tasks in these applications is real-time object detection, which involves identifying and classifying objects in images or video streams captured by UAVs. Real-time object detection is essential for many UAV applications, as it enables the UAV to perform tasks such as tracking targets, avoiding obstacles, monitoring, and making decisions based on the environment.

However, real-time object detection using UAVs poses several challenges, particularly when it comes to small objects, such as vehicles, in aerial images. In addition, the motion of UAVs and the high-resolution aerial images can further complicate the object detection task. One major challenge is the limited resolution of the images, which can make it difficult to distinguish small objects or to identify fine details. This can be especially problematic at higher altitudes, where objects may appear even smaller due to the increased distance from the camera.

*KIOS Research and Innovation Center of Excellence (KIOS CoE) E-mail:{makrigiorgis.rafael, kyrkou.christos, kolios.panayiotis} @ucy.ac.cy

Another challenge is the presence of noise and distractions in the images, such as reflections, shadows, and occlusions. These can interfere with the object detection algorithms and make it difficult to accurately identify and classify the objects. In addition, the quality of the images can vary widely depending on factors such as the weather, lighting conditions, and the stability of the UAV. This can also impact the performance of the object detection algorithms. Furthermore, all these issues are amplified if the resolution is not optimal.

To address these challenges, it may be necessary to use advanced image processing techniques to enhance the resolution and clarity of the images, or to use machine learning algorithms to filter out noise and distractions. Additionally, it may be necessary to develop new object detection algorithms that are specifically designed to handle the unique characteristics of aerial image datasets obtained from UAVs. Therefore, there is a pressing need for effective and efficient algorithms that can reliably detect and classify small objects in aerial images captured by UAVs.

In this paper, we investigate the best strategy for aerial object detection that addresses the challenge of detecting vehicles at various altitudes in real-time on embedded devices. We collected a comprehensive dataset of aerial images capturing moving vehicles at altitudes ranging from 50 to 500 meters, and trained a state-of-the-art YOLO model, YoloV7 [5], on each altitude dataset as well as a mixed-altitude dataset. Our key contribution is a training strategy that involves training the mixed-altitude dataset on a higher input size resolution to learn a larger set of features, and then performing inference detection using a smaller resolution size to increase inference performance while maintaining high accuracy. Our experimental results demonstrate that this approach achieves over 95% mAP@50% at 21 FPS on an Nvidia Xavier NX, making it highly suitable for real-time aerial object detection in diverse altitude scenarios. Moreover, we provide publicly available datasets used in this paper for further research. Our findings contribute to the field of aerial object detection by providing insights into the best strategy for detecting vehicles at different altitudes while maintaining real-time processing on embedded devices, and highlight the potential of using a single CNN model for accurate and efficient object detection in aerial images.

II. Related Work

Aerial image object detection has been an active area of research in the computer vision community in recent years. There have been numerous approaches proposed to tackle this problem, ranging from traditional machine learning methods to more recent deep learning approaches.

One of the early works in this field is the approach of Ajmera et al. [6], which presents a survey of object detection techniques in aerial images, including both traditional machine learning methods and deep learning approaches. With the advent of deep learning, several approaches have been proposed that leverage convolutional neural networks (CNNs) for object detection in aerial images. One such approach is the work of Tayara et al. [7], which proposes a CNN-based method for object detection in high-resolution aerial images. Another approach is the work of Chalavadi et al. [8], which proposes a multi-scale CNN for object detection in aerial imagery. Zhao et al. [9] provide a review of deep learning approaches for object detection in aerial images, including a discussion of the challenges and limitations of these methods. Unfortunately, all of these either focus on satellite imagery or do not focus on real-time performance on an embedded device.

Recently, there has been a lot of interest in using unmanned aerial vehicles (UAVs) for vehicle detection, with many approaches based on deep learning [10], [11]. However, this task presents some challenges, such as the need for computational efficiency and the detection of small-sized objects, as vehicles in aerial images often appear as small objects. To address the problem of small object detection in aerial images, some approaches have employed a two-stage Faster R-CNN framework with an Inception V2 backbone [12]. However, this method has shown a decrease in the balance between accuracy and computational intensity, making it unsuitable for video processing applications. To reduce processing time, other approaches have suggested searching for subregions in the image [13], [14], [15]. Another work is the work of Lang et al. [16], which proposes a method for fast and accurate object detection in high-resolution remote sensing images using a lightweight YOLO-like object detector. More recent works have used single-shot convolutional neural networks for better inference time, such as the YOLOv3 [17] network for top-view vehicle detection [18]. There has also been some effort to develop smaller networks for use on lightweight, embedded processing platforms [19], [20], although not all of these are designed for aerial imaging. In general, the Faster R-CNN and YOLO families of algorithms are commonly used for vehicle detection, and for real-time traffic monitoring, the YOLO family of networks [21], [17], [22] is particularly relevant due to their lower computational complexity. Hossain et al. [23] proposed a deep learning-based framework for object detection and segmentation in aerial images. Their proposed framework achieved state-of-the-art results on several aerial image datasets.

On the dataset side, previous work has focused on ground-level datasets for object detection and analysis. Examples include the KITTI Vision Benchmark [24] and the Microsoft COCO dataset [25], which contain a wide range of object categories. While these datasets provide a useful baseline for evaluation, they are not ideal for UAV-based object detection and analysis tasks due to the unique characteristics of aerial imagery. In addition to ground-level datasets, there are a few existing aerial imagery datasets for object detection and analysis with UAVs. Examples include the Stanford Drone Dataset [26], which contains images of cars, pedestrians, and cyclists, and the UAVid dataset [27], which contains images of pedestrians, vehicles, and buildings. UAVDT [28] is a large-scale challenging UAV Detection and Tracking benchmark which includes different altitudes and angles for vehicles. For our analysis, we needed datasets that include various altitudes and all of the above datasets did not include this requirement. Similarly to our work though, Kouris et. al in [29] proposed altitude aware approach for detecting vehicles using UAV while publishing their dataset as well but unfortunately their dataset consists of only a few samples.

Overall, there has been a significant amount of research on object detection and analysis for aerial imagery datasets for embedded devices. However, there is still a need for research that focuses on the application of deep learning techniques to aerial imagery datasets on embedded devices. The goal of this paper is to fill the gap by proposing a training strategy for object detection in aerial imagery datasets that is simultaneously robust to different altitudes and also suitable for embedded devices. With a focus on detecting vehicles at different altitudes, including both small and large vehicles of the same class.

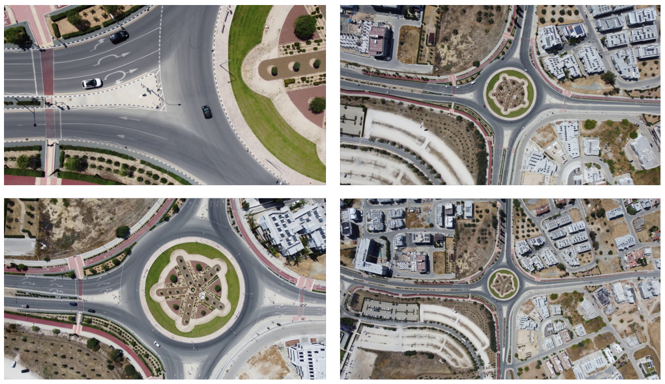
Fig. 1: Examples of images in the dataset. Top Left: 50 meters, Bottom Left: 150 meters, Top Right: 300 meters, Bottom Right: 500 meters.

III. Datasets and Proposed Methodology

A. Datasets

In the absence of any other datasets for altitude evaluation, aside from the one published in [29], it was deemed necessary to create custom datasets for various altitudes in the same location to suit our requirements. To have a thorough evaluation as per the detection in various altitudes a location near the University of Cyprus campus was decided to capture the footage data in need. The location selected is a roundabout outside the university campus. Near the roundabout, a lot of structures can be found which can be seen above 100 meters of altitude due to the field of view of the camera, and while flying at even higher altitudes even more and more buildings are visible. These buildings also have solar panels and water tanks on the rooftops, which makes our detection task even harder from a higher altitude since they can be easily confused by the algorithm as vehicles due to their rectangular shapes. Furthermore, near the roundabout, a university parking lot can be found which had several parked vehicles. Figure 1 depicts 4 images taken from the dataset with altitudes of 50,150,300 and 500 meters. The pictures emphasize the difference in the field of view and the size of the objects and the differences on visible features as the altitude increases.

The data were captured using a small UAV developed by DJI, the Mavic 2 Enterprise. The videos were obtained by flying the UAV on top of the region of interest, in a single day while adjusting the altitude of the UAV. After obtaining these videos, images were extracted from those videos with a step rate of 1 image per 5 frames, while all the videos were captured at a framerate of 30 frames per second. All the data were captured during a sunny day so the images are bright and include minimal shadows. Furthermore, upon extracting the images, all the data were annotated using only a single class labeled as 'Car' since there were only a few instances of heavy vehicles or buses. For each altitude, which varies from 50 to 500 meters with a 50 meters step as mentioned above, the data were split into 3 sets, training, validation, and testing with a split of 75%, 15%, and 15%. Table I depicts information about each dataset as per the number of images and a number of annotations that were included in each dataset. By observing this table, the first thing you notice is that upon increasing the altitude, the number of vehicles drastically increases. This is expected since the area of field of view visible by the UAV's camera is bigger as the altitude increases, capturing more vehicles that are moving or being parked around the selected location. Also, all the datasets have almost the same amount of images, which makes sense since the duration for each data collection experiment was about the same. Furthermore, all the images were kept at their original resolution of 3840x2160 which makes it even harder to detect objects due to the high resolution of the images. The average size of the vehicles in pixels for each of the datasets can be seen in Table II. As seen in the table, the average size of the vehicles in pixels in the lower altitude starts from a percentage of 0.5% of the image and goes down to 0.008% of the image which is a significant 62.5x decrease in size.

B. *YoloV7*

YoloV7 is the latest version of the Yolo series of networks, which was published in 2022. YoloV7 method claims to have much better performance than its predecessors achieving up 161 FPS on a Tesla V100. The main contributions that make it better than its predecessors are the Extended Efficient Layer Aggregation (E-ELAN), Model scaling techniques, and the trainable bag of freebies including re-parameterization planning and auxiliary head coarse-to-fine head loss. These all-new features intrigued us to move on and proceed with YoloV7 as our main training model for this particular evaluation. Following is a brief explanation of these new features.

E-ELAN (Extended efficient layer aggregation networks) is a method for designing efficient neural network architectures. It builds upon the ELAN (efficient layer aggregation) architecture by adding expand, shuffle, and merge cardinality to the computational blocks in the architecture. This allows for the continuous enhancement of the network's learning ability without disrupting the original gradient path. E-ELAN uses group convolution to expand the channel and cardinality of the computational blocks and shuffles and concatenates the feature maps calculated by each block into groups. The goal of E-ELAN is to improve the inference speed of the network while maintaining a stable state in the architecture.

Model scaling is a technique used to adjust various aspects of a model to meet the specific requirements of an application. These aspects can include the width (number of channels), depth (number of stages), and resolution (input image size) of the model. In traditional concatenation-based architectures like ResNet or PlainNet, it is difficult to analyze the effects of different scaling factors independently. YOLOv7 introduces **Compound Model Scaling** for concatenation-based models to address this issue. Compound model scaling allows for the maintenance of the optimal structure of the model while adjusting its depth and width factors. It does this by changing the output channel of a computational block when scaling its depth factor and adjusting the transition layers with the same level of change for the width factor.

Planned re-parameterized convolution (RepConvN) is a technique that was developed to improve the performance of RepConv in VGG architectures. However, when applied directly to ResNet or DenseNet, it results in a significant loss of accuracy. In YOLOv7, RepConvN is used in the architecture to avoid identity connections when replacing a convolutional layer with residual or concatenation by re-parameterized convolution.

The YOLO architecture consists of a backbone, neck, and head. The head contains the predictions made by the model. YOLOv7 introduces the use of multiple heads, inspired by the Deep Supervision technique. The head responsible for the final output is called the lead head, while the head used to assist in training the middle layers is called the **auxiliary head**. YOLOv7 also introduces a Label Assigner mechanism that generates soft labels by considering the network's prediction results and the ground truth together. Soft labels are more reliable than traditional hard labels, which are generated solely based on given rules and the ground truth because they also consider the quality and distribution of the prediction output.

C. Analysis Process

After capturing and preparing the datasets, we conducted training, and overall performance analysis. In general, for any training conducted on all datasets, including the mixed dataset, was trained using the YOLOv7 network with default hyperparameters and augmentations. The training augmentations included mosaic, mixup, HSV enhancements, scaling, flipping, and rotating translations. All models were trained for 60 epochs using pretrained weights on COCO dataset.

Our overall process consisted of three steps. The first step of our process involved training each dataset separately using an input size of 640x640, which is usually the default input size for when comparing yolov7 models, in order to evaluate the performance of each model on each individual dataset. In the second step, we focused on the mixed altitude dataset and trained it using various resolution sizes. These new models were then evaluated on all datasets and more explanation and results can be found in Section IV-B. Finally, in the performance evaluation step, we compared the performance and accuracy of each model from the second step using various input sizes. The description and results of this evaluation are seen in detail in Section IV-C.

TABLE I: Information about the number of images and annotations per dataset.

	Images				**Labels**			
Dataset	**Train**	**Valid**	**Test**	**Total**	**Train**	**Valid**	**Test**	**Total**
50m	156	20	19	195	311	48	49	408
100m	296	36	37	369	1126	135	152	1413
150m	293	36	37	366	2026	272	272	2570
200m	296	36	37	369	2112	248	262	2622
250m	294	36	37	367	3978	489	503	4970
300m	296	36	37	369	7514	920	954	9388
350m	296	36	37	369	12826	1590	1603	16019
400m	294	36	37	367	24086	2916	2968	29970
450m	295	36	37	368	38119	4658	4784	47561
500m	324	36	45	405	59819	6662	8268	74749
Mix	2839	344	360	3543	151911	17938	19815	189664

IV. Evaluation

A. Evaluation for each Dataset

We evaluated each trained model on the test set of all datasets. The initial results, as shown in Figure 2, reveal that each model performs better on datasets at altitudes near to the one on which it was trained. Models trained on altitudes less than 150 meters perform poorly on datasets at least 100m higher, and the same pattern is observed for models trained on higher altitudes when evaluating lower altitude datasets. Hence, we cannot expect models trained on particular altitudes to generalize without retraining. The models trained at 200m and 250m altitudes tend to perform better overall but still struggle with datasets above 350m. The models trained at 400m, 450m, and 500m altitudes are not shown in the figure because they had difficulty training even on their own datasets, resulting in a mAP@.50 below 5% for each model meaning the default model and hyperparameters are not sutable for such high altitudes. This occurred because the annotated vehicles in these datasets were significantly smaller than those in the datasets at lower altitudes. Table II shows the average size of vehicles in each dataset and the size of vehicles in various image sizes that can be trained using YOLOv7. This table also helps us determine the average size of pixels that YOLOv7 is able to train. It seems that YOLOv7 struggles to train objects with a size below 49 pixels after resizing. To test this theory and validate our findings, we proceed to train the 400-500m datasets using an image size of 1088x1088. The test was successful, with the models achieving an accuracy of $88 - 95\%$. For fairness, we compare in Figure 2 only models trained on the same 640x640 resolution.

TABLE II: Average Size of Vehicles in pixels for each of the datasets

Dataset AVG Area (Pixels)	50m	100m	150m	200m	250m	300m	350m	400m	450m	500m	mix
Original	43618	13009	5436	3409	2370	1548	1005	770	729	646	7959
640x640	2154	642	268	168	117	76	49	38	36	31	54
1088x1088	6225	1857	776	487	338	221	143	110	104	92	156
1440x1440	10904	3252	1359	852	592	387	251	193	182	161	274
2160x2160	24534	7317	3058	1918	1332	870	565	433	410	363	615

B. Mixed Altitude Evaluation

Our mixed altitude dataset features high-resolution imagery, with many vehicles occupying small pixel areas. Upon resizing the Yolov7 network input to 640x640, these areas become even smaller, as demonstrated by Table. II, second row. Hence, the areas for different sizes such as 1088x1088, 1440x1440, and 2144x2144 were also calculated and they are also included in Table II. Furthermore, these results show that the areas are significantly larger so the next experiment was to train the mixed altitude dataset with a larger input size so that the area of the annotated vehicles would retain more features. Typically, by training a model with a larger input size, the performance of the detector in terms of accuracy increases, but the inference time increases as well. Our goal was to strike a balance between accuracy and performance during training. To achieve this, we conducted experiments using the sizes listed in Table II. We then reduced the inference size to evaluate the impact on both accuracy and FPS.

TABLE III: Performance in terms of speed on the Jetson Xavier NX. The model naming consists of the model name, training size and inference size.

Model	**Pre-process**	**Inference + nms**	**Post-process**	**Total**
yv7_640_416	0.006	0.021	0.001	0.027
yv7_640_512	0.006	0.027	0.013	0.046
yv7_640_640	0.006	0.040	0.003	0.049
yv7_1088_416	0.005	0.022	0.009	0.037
yv7_1088_512	0.0059	0.026	0.009	0.041
yv7_1088_640	0.006	0.04	0.01	0.05
yv7_1088_1088	0.006	0.102	0.00005	0.109
yv7_1440_416	0.06	0.0203	0.0005	0.027
yv7_1440_512	0.0058	0.0264	0.0092	0.041
yv7_1440_640	0.006	0.04	0.0005	0.046
yv7_1440_1088	0.014	0.101	0.011	0.1268

As mentioned in Section III-C, a model was trained using the mixed dataset on each altitude using the following input sizes 1088×1088, 1440×1440, 2144×2144 and we also

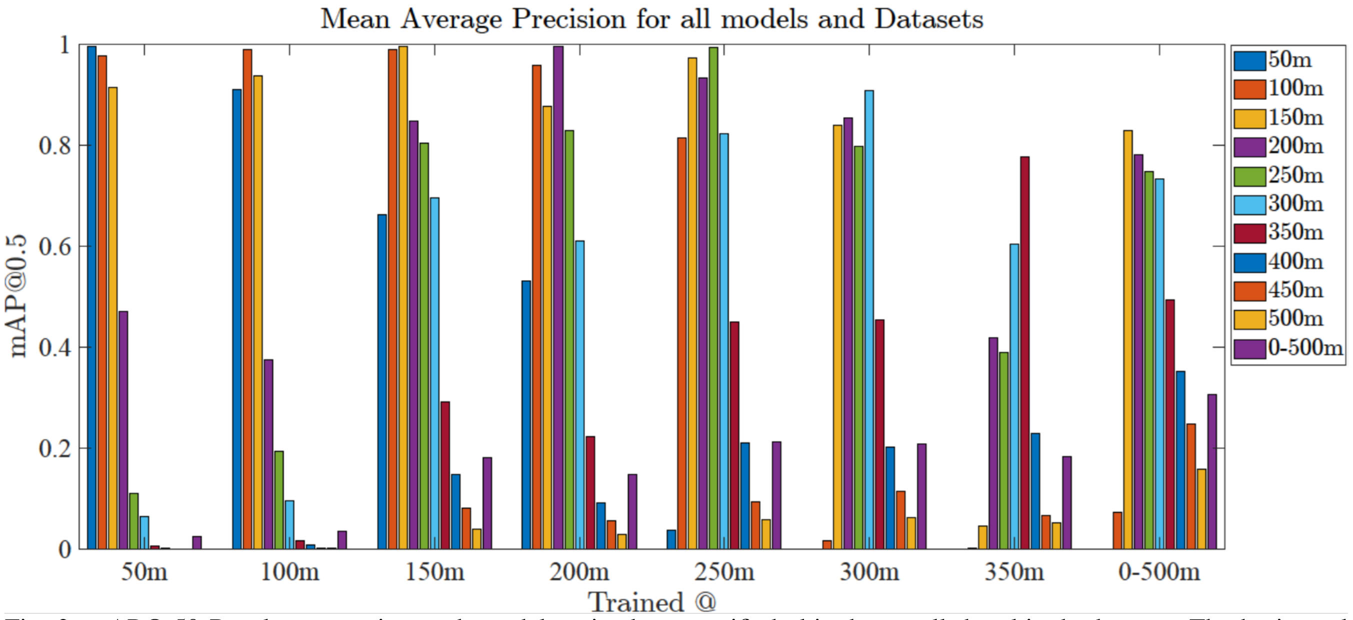

Fig. 2: mAP@.50 Results comparing each model, trained on specified altitude, on all the altitude datasets. The horizontal lines represent the highest mAP for the specific altitude color.

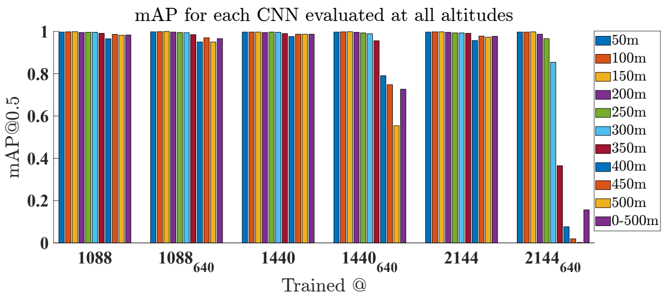

Fig. 3: mAP@.50 Results comparison of the high-resolution input models with evaluation on the initial size and 640x640.

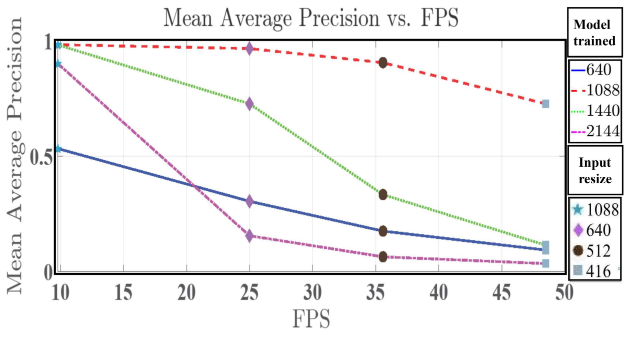

Fig. 4: mAP@.50 vs FPS. Each line represents the model trained with the input size mentioned in the legend on top right. The markers represent the inference input size of 416,512,640 and 1088 represented as a star, a diamond, a circle, and a rectangle respectively. Input resize of 2144,1440 are not shown as they wouldn't run due to memory insufficiency of the Jetson device.

used the one trained on 640×640 from the earlier experiment. Then, the evaluation of each model was performed with all the datasets to test out whether the accuracy is any better than the 640 × 640 mixed dataset model. Surprisingly, the mAP@0.50 for all the models performed significantly high, achieving scores above 90% as seen in Figure 3. This figure also depicts the mAP@0.50 of the same models for a resized inference of 640 × 640. Here is the part where we start to see a decline in the performance for the higher resolution such as 1440 and especially for the 2144 model. Also, the performance drop, as expected, is seen in the higher altitude datasets since they are the datasets with the least vehicles average area of pixels. Having these results, we are one step away from confirming that having a higher training input size may result in better overall performance, the only thing left to check is the actual influence of the inference speed.

C. Jetson Performance Evaluation

Accuracy and inference speed are both crucial factors in evaluating the performance of UAVs in real-time applications. High accuracy is necessary to ensure reliable and trustworthy predictions, while fast inference speed is needed to meet the demands of real-time scenarios where time is of the essence. Both accuracy and speed must be balanced in order to achieve optimal performance in real-time applications. To test the performance of the models we used an Nvidia Jetson Xavier NX as it is a device commonly used on UAVs. We tested the performance using a video with around 900 frames in its original 2k resolution. Detection was performed on each frame of the video for each model, and we calculated the average time in milliseconds for the pre-processing of the frames, the inference and non-maximum suppression (NMS),

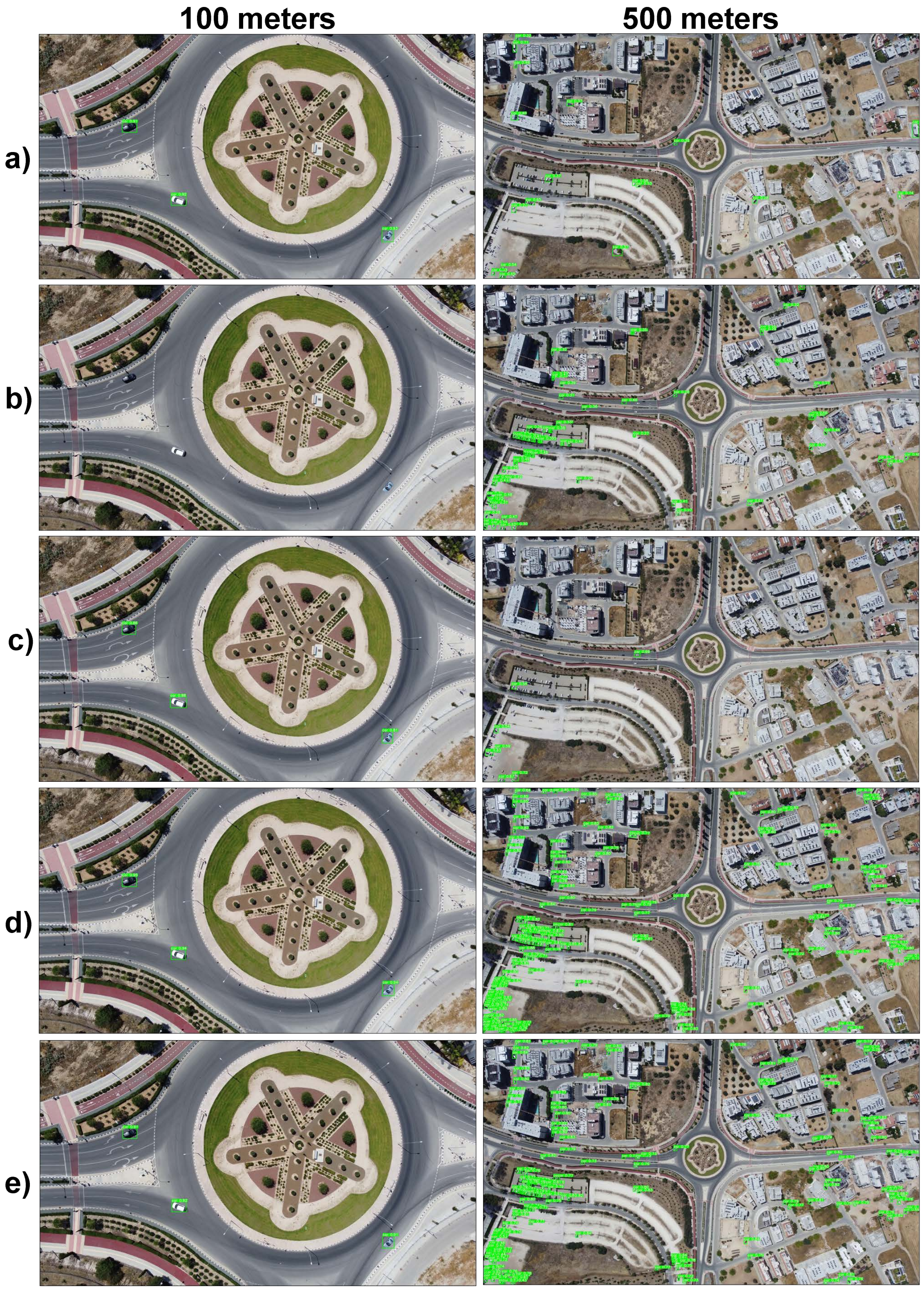

Fig. 5: Detection results examples where each row represents a different model. (a) is trained on the 150-meter dataset with a 640x640 input size, and performed a 640x640 inference. (b),(c),(e) all are trained on the mix-dataset with 640x640, 2144x2144, 1088x1088 input sizes respectively and all performed with 640x640 inference. (d) is trained on the mix-dataset with 1088x1088 input size and performed 1088x1088 inference size

and the post-processing of the detections. Table III shows the results of this performance test for the 640×640, 1088×1088, and 1440×1440 models at their original size and resized to 640x640, 512×512, and 416×416. The results for the 2144 model are not included in this table because Jetson was unable to run the model due to insufficient memory. The 1440 model was also unable to run at its original resolution. All models were converted to TensorRT with floating point 16 (FP16), to achieve the maximum performance possible, as the Jetson devices come with an embedded CUDA graphical processing unit with Tensor cores. The inference batch size was set to 1 for streaming scenarios. As seen in the table, the pre-processing time is similar and uses the central processing unit (CPU) to process a frame. The post-processing time may vary slightly as it depends on how many objects the model detects, even if they are false positives. The post-processing time will be more meaningful when we consider the mean average precision (mAP) of the models, as we do not yet know if the detections are false or true positives. The main thing to consider is the inference and NMS time. From the results, it appears that the inference and NMS time does not increase when the initial training was performed on a higher resolution input, indicating that the inference time mostly depends on the input size of the image.

The final mean average precision (mAP)@0.50 evaluation test was conducted using the mixed dataset test set to draw a final conclusion. This test was performed on all four models trained on the mixed dataset, using a range of input sizes. The results of this evaluation are depicted in Figure 4, with the frames per second on the X-axis and the mAP@0.50 on the Y-axis. Each line represents the model trained with the input size mentioned in the legend on the top right. The markers represent the inference input size of 416,512,640 and 1088 represented as a star, a diamond, a circle, and a rectangle respectively. Input resize of 2144,1440 are not shown as they wouldn't run due to memory insufficiency of the Jetson device.

The model that was trained with a 1088×1088 input resolution outperformed the others in this evaluation. It was able to run at over 20 frames per second for resizing resolutions of 640, 512, and 416 while maintaining a precision of 75% in the worst-case scenario and exceeding 90% for 640 and 512 resizing. This indicates that this model is able to maintain a high level of accuracy and performance even at lower-resolution inputs.

D. Discussion

Figure 5 displays some examples of detections of various trained models for 100 and 500 meters altitude images. The models are as follows, (a) trained on 150 meters dataset with 640x640 input size, inference on 640×640, (b) to (e) are all trained on mix-dataset having (b)(c)(e) with 640×640, 2144×2144, 1088×1088 for training input size respectively and all with 640×640 inference. (d) is trained on 1088×1088 and performed inference on 640×640. As you can see from this figure, it demonstrates our results depicted in previous figures by having the 1088×1088 model perform excellent results on both low and high altitudes even after a 640×640 inference size. After conducting multiple evaluation experiments, we found that training a larger input size allows a model to learn a larger set of features, which enables it to accurately detect objects of interest even when performing inference on a smaller resolution. Our strategy delivers high accuracy in detecting objects of various sizes, including small objects, without modifying the model or seeking any other approach during inference. The results showcase our strategy's effectiveness and efficiency, as seen in real-time processing on an embedded device like the Nvidia Jetson Xavier NX.

V. Conclusion and Future Work

In this work, we presented a training strategy for aerial object detection that is capable of detecting vehicles, at various altitudes while maintaining real-time inference time on an embedded device like the Nvidia Jetson Xavier NX. Initially, training was conducted for each altitude dataset, in order to compare the evaluation results of the trained models on all the datasets collected. To further analyze the results, we conducted additional evaluation experiments on a mixed altitude dataset model. The first experiment involved training the mixed-altitude dataset on a larger input size in order to learn a larger set of features, with the intention of performing inference on a smaller resolution. The results showed a significant increase in accuracy. The models were able to detect objects of interest at a variety of altitudes with high accuracy and maintain real-time processing on the Nvidia Jetson Xavier NX. The best-performed model was initially trained on 1088x1088 with the inference size of 640x640 due to achieving above ¿=95% mAP@50%.

The final evaluation test involved testing all possible input sizes on the four models trained on the mixed dataset. The results, shown in Figure 4, revealed that, once again, the model trained with a 1088x1088 input size performed the best. It was able to run at over 21 frames per second for resize resolutions of 640, 512, and 416 while maintaining a precision of 75% in the worst-case scenario and exceeding 90% for 640 and 512 resizing, as seen on 4. This indicates that this model is able to maintain a high level of accuracy and performance even at lower-resolution inputs. Overall, our conclusion is that the best strategy for detecting vehicles in various altitudes while maintaining real-time processing on an embedded device is to have a dataset with various altitudes, train it on a 1088x088 input size and then perform inference with an input size of 640x640. As a final contribution, all the datasets used for this paper are shared publicly on our website[1].

As for future work, we plan to test our approach on more challenging objects, such as people, for low to high-altitude detection. People detection from a high altitude can be difficult due to their features appearing almost invisible in the

[1]https://www.kios.ucy.ac.cy/evai/datasets/

image, resembling a ”dot”. To further challenge our approach, we will also consider tiling techniques as demonstrated in [30].

ACKNOWLEDGEMENTS

The project is co-financed by the European Regional Development Fund and the Republic of Cyprus through the Cyprus Research Innovation Foundation (’RESTART 2016-2020’ Program) (Grant No. INTEGRATED/0918/0056) (RONDA). This work was also supported by the European Unions Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE) and from the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

Christos Kyrkou would like to acknowledge the support of NVIDIA with the donation of GPU platforms.

REFERENCES

[1] Antonis Savva, Angelos Zacharia, Rafael Makrigiorgis, Antreas Anastasiou, Christos Kyrkou, Panayiotis Kolios, Christos Panayiotou, and Theocharis Theocharides. Icarus: automatic autonomous power infrastructure inspection with uavs. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 918–926. IEEE, 2021.

[2] P Petrides, C Kyrkou, P Kolios, Theocharis Theocharides, and C Panayiotou. Towards a holistic performance evaluation framework for drone-based object detection. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1785–1793. IEEE, 2017.

[3] Rafael Makrigiorgis, Nicolas Hadjittoouli, Christos Kyrkou, and Theocharis Theocharides. Aircamrtm: Enhancing vehicle detection for efficient aerial camera-based road traffic monitoring. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2119–2128, 2022.

[4] Christos Kyrkou and Theocharis Theocharides. Emergencynet: Efficient aerial image classification for drone-based emergency monitoring using atrous convolutional feature fusion. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:1687–1699, 2020.

[5] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[6] Falguni Ajmera, Sanidhya Meshram, Sangita Nemade, and Varsha Gaikwad. Survey on object detection in aerial imagery. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 1050–1055, 2021.

[7] Hilal Tayara and Kil To Chong. Object detection in very high-resolution aerial images using one-stage densely connected feature pyramid network. *Sensors*, 18(10):3341, 2018.

[8] Vishnu Chalavadi, Prudviraj Jeripothula, Rajeshreddy Datla, Sobhan Babu Ch, et al. msodanet: A network for multi-scale object detection in aerial images using hierarchical dilated convolutions. *Pattern Recognition*, 126:108548, 2022.

[9] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[10] X. Chen, S. Xiang, C. L. Liu, and C. H. Pan. Vehicle detection in satellite images by parallel deep convolutional neural networks. In *2013 2nd IAPR Asian Conference on Pattern Recognition*, pages 181–185, Nov 2013.

[11] Michael Krump, Martin Ruß, and Peter Stütz. Deep learning algorithms for vehicle detection on uav platforms: First investigations on the effects of synthetic training. In Jan Mazal, Adriano Fagiolini, and Petr Vasik, editors, *Modelling and Simulation for Autonomous Systems*, pages 50–70, Cham, 2020. Springer International Publishing.

[12] A. Mansour, W. M. Hussein, and E. Said. Small objects detection in satellite images using deep learning. In *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 86–91, 2019.

[13] Alexandros Kouris, Christos Kyrkou, and Christos-Savvas Bouganis. Informed region selection for efficient uav-based object detectors: Altitude-aware vehicle detection with cycar dataset. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 51–58, 2019.

[14] G. Plastiras, S. Siddiqui, C. Kyrkou, and T. Theocharides. Efficient embedded deep neural-network-based object detection via joint quantization and tiling. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 6–10, 2020.

[15] Changlin Li, Taojiannan Yang, Sijie Zhu, Chen Chen, and Shanyue Guan. Density map guided object detection in aerial images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[16] Lei Lang, Ke Xu, Qian Zhang, and Dong Wang. Fast and accurate object detection in remote sensing images based on lightweight deep neural network. *Sensors*, 21(16):5460, 2021.

[17] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[18] Xin Luo, Xiaoyue Tian, Huijie Zhang, Weimin Hou, Geng Leng, Wenbo Xu, Haitao Jia, Xixu He, Meng Wang, and Jian Zhang. Fast automatic vehicle detection in uav images using convolutional neural networks. *Remote Sensing*, 12(12):1994, Jun 2020.

[19] C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris, and C. Bouganis. Dronet: Efficient convolutional neural network detector for real-time uav applications. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 967–972, 2018.

[20] A. Womg, M. J. Shafiee, F. Li, and B. Chwyl. Tiny ssd: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 95–101, 2018.

[21] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[22] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[23] Sabir Hossain and Deok-jin Lee. Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with gpu-based embedded devices. *Sensors*, 19(15):3371, 2019.

[24] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[26] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[27] Ye Lyu, George Vosselman, Gui-Song Xia, Alper Yilmaz, and Michael Ying Yang. Uavid: A semantic segmentation dataset for uav imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 165:108–119, 2020.

[28] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 370–386, 2018.

[29] Alexandros Kouris, Christos Kyrkou, and Christos-Savvas Bouganis. Informed region selection for efficient uav-based object detectors: Altitude-aware vehicle detection with cycar dataset. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 51–58, 2019.

[30] George Plastiras, Shahid Siddiqui, Christos Kyrkou, and Theocharis Theocharides. Efficient embedded deep neural-network-based object detection via joint quantization and tiling. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 6–10, 2020.