CNN based Drone Detection in Infrared Images

Purbaditya Bhattacharya, Patrick Nowak, Daniel Ahlers, Martin Holters, Udo Zölzer
Department of Signal Processing and Communication
Helmut Schmidt University
Hamburg, Germany
firstname.lastname@hsu-hh.de

***Abstract*—Methods based on convolutional neural networks (CNNs) for drone detection in infrared images are explored in this paper. For the task of drone detection, a dataset of infrared images containing drones is initially built from a publicly available database and drone videos captured with our cameras. The required drone images are extracted and manually labelled with the help of an image labelling tool. Multiple configurations of popular pretrained CNN models including EfficientDet and Yolo are subsequently trained on the combined dataset. The trained models are employed on the test dataset and their performance is evaluated.**

***Index Terms*—drone detection, object detection, CNN, image processing, dataset, EfficientDet, Yolo v5**

I. Introduction

Unmanned aerial vehicles (UAV) or drones have seen a rapid growth in popularity in recent years. While drones were initially adopted and used in military operations, primarily for surveillance, communication, and transportation, they are now widely used in industrial and commercial applications. Some of the use cases include surveillance for law enforcement, monitoring of regions for emergency and rescue operations, transportation, aerial photography, and recreational activities. Given their abundance and gradual increment in usage, public safety or security becomes an area of concern and must be paid an attention to. Hence, reliable and robust methods should be developed in order to detect and track drones to monitor their activities. An automated drone detection system can generate an alert based on any anomalous activity. Such systems can operate based on images captured in the visible or infrared range. In this context, infrared images can provide better visibility under certain conditions where the background is noisy or cluttered and the environmental illumination is low. Recent methods for object detection include the usage of CNNs which usually provide a very high accuracy in such kind of tasks [1]–[3]. In this work, the initial goal is to use a selection of such network models in order to detect drones in infrared images.

This work is part of the dtec.bw project "Digital Sensor-2-Cloud Campus-Platform" (DS2CCP), in which a 5G campus network is set up at Helmut Schmidt University. As can be seen in Figure 1, the 5G campus network is used to connect camera and processing computer. In the uplink, image data is sent from the camera to the processing computer in order to perform the drone detection on a high-performance computer. In this way, image data from multiple cameras can be handled simultaneously on the same computer. Here, the

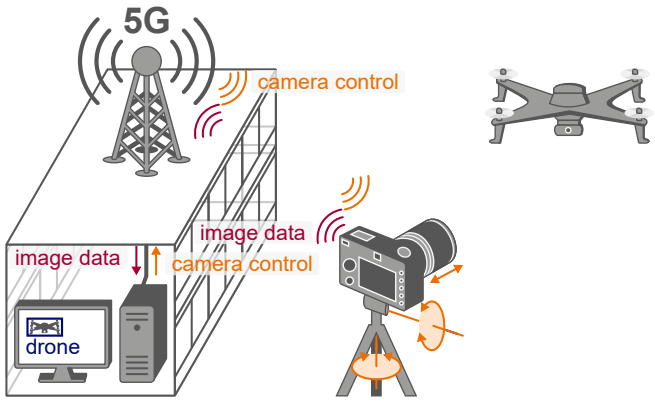

Figure 1. Communication between camera and processing computer via 5G campus network.

5G campus network should offer both high data rates and low latency. Furthermore, the downlink enables the transfer of camera control data, e.g. pan, tilt, and zoom, from the processing computer to the camera allowing flexibility in camera maneuvering and changes in the viewing area of the individual cameras.

II. Dataset

To train the network models for drone detection a suitable dataset of infrared images containing drones is required. A few infrared image drone datasets are publicly available from which a dataset [4] is selected which is a part of the ICCV drone detection challenge. The dataset contains 70 video files having 1000 frames each, and the drones fly across multiple backgrounds in the videos. Each video has a spatial resolution of 640×480 pixels and features a single drone, located approximately at the center of each frame. Although there are nearly 70000 images with drones, a large number of images have redundant or indiscernible content. The redundancy is primarily due to limited drone movement and uniform background in many videos. Hence, a small subset is selected for training. Apart from the public dataset, a dataset is constructed from drone videos captured using three different cameras. Firstly, a Sony Alpha 6000 camera equipped with a lens with a focal length of 16-50 mm is used to record drone videos in the visible spectrum with a resolution of 1920×1080 pixels. Secondly, a FLIR Scion OTM366 is used to record drone videos in the infrared spectrum with a resolution of 640×480 pixels. Finally, infrared videos with a higher resolution of

https://doi.org/10.24405/14543

FIGURE 2. ANNOTATED IMAGE CAPTURED BY FLIR SCION OTM366.

1024×768 pixels are recorded using an InfraTec VarioCAM HD Z equipped with a zoom lens (25-150 mm). The drone videos were recorded simultaneously with all three cameras on the football field of the Helmut Schmidt University from different perspectives and distances between 100 and 200 meters. In total, three different drones can be seen inside the videos. In most of the videos, the Artcopter Raptor drone is shown. The second drone in some videos is the Holybro X500. A third smaller drone is visible in some videos, which would be the DJI Mavic Pro 2.

TABLE I
TRAINING AND TEST DATASET

Dataset	Public	FLIR	Total
Train	6450	6983	13433
Test	0	1683	1683

In this work, only videos captured via the FLIR Scion OTM366 are used. In order to use these videos for training and evaluation purposes, drones inside the videos must be labeled. For this purpose, the LabelImg [5] graphical annotation tool is used. The annotations are done in the Pascal VOC [6] format which are transformed to YOLO and then to COCO format [7] according to the requirements of training. Presently, three videos containing a total of 21953 frames are being labeled, of which 12608 frames are already labelled and a fraction is used. In the future, videos recorded with the Sony Alpha 6000 and InfraTec VarioCAM HD Z will be labelled as well and a diverse dataset will be created. This dataset will also be expanded in the future. An annotated infrared image captured by FLIR Scion OTM366 containing multiple drones is shown in FIGURE 2.

The dataset prepared with images from FLIR Scion OTM366 is initially divided into easy and difficult images depending on the visibility of drones due to clutter or occlusion. In this work, the relatively easy images are selected where the drones are mostly visible. The final combined dataset contains 13433 images for training and 1683 images for testing, as shown in TABLE I.

III. NETWORK DESCRIPTION

To train a new dataset for detection task, CNNs that are pretrained on large datasets are usually preferred rather than a randomly initialized CNN. A pretrained network is generally able to converge much faster and achieve better performance compared to a network trained from scratch. Hence, the EfficientDet [8] and Yolo v5 [9] network models which are pretrained on a version of the Common Objects in Context (COCO) dataset [10], are lightly modified and used to train the final dataset for drone detection. The selected CNNs are two of the most popular networks for object detection and are therefore selected in this work. A minimal overview of the network architectures is provided in FIGURE 3 with processing blocks. The architectures of both models contain a backbone network, followed by a neck (feature network) and the classification and detection heads.

The backbone network is usually a pyramidal network where the feature maps are downscaled after a certain number of layers. The layers primarily include convolution, batch normalization, rectification, sigmoid linear unit or swish activation, and summation and / or concatenation layers. The EfficientDet architecture modifies the EfficientNet [11] backbone and downscales the feature maps gradually by a factor of 2 until an overall factor of 128. Hence, an input resolution of 640×640 is downscaled to a final resolution of 5×5. The architecture of Yolo v5 is also made of a pyramidal backbone, known as CSPDarknet53, and it downscales the feature maps gradually by a factor of 2 until an overall factor of 32. Therefore, an input resolution of 640×640 is downscaled to a final resolution of 20×20. The backbone of Yolo v5 contains many more layers compared to EfficientDet. Yolo v5 also has an additional spatial pyramid pooling layer (SPP) at the end of the backbone which applies multiresolution parallel pooling kernels and concatenates the generated feature maps.

In the neck, feature maps from multiple levels of the backbone, having multiple resolutions are initially processed by different layers and then combined via summation or concatenation. In order to combine certain feature maps, the lower resolution (LR) feature maps are upsampled by bilinear interpolation while the higher resolution (HR) feature maps are downsampled by convolution with stride. In EfficientDet, the neck contains multiple bidirectional feature pyramid network (BiFPN) structures which upsample the feature maps gradually by an overall factor of 16, and downsample them again while combining the features of similar resolution via summation. In Yolo v5, the processing in the neck is similar to EfficientDet but relatively simpler and is referred to as the path aggregation network (PAN). Additionally, the feature maps of similar resolutions within Yolo v5 are usually combined through concatenation.

The neck is usually followed by several classification and detection heads which are primarily formed by convolution, activation, and batch normalization layers. The heads originate

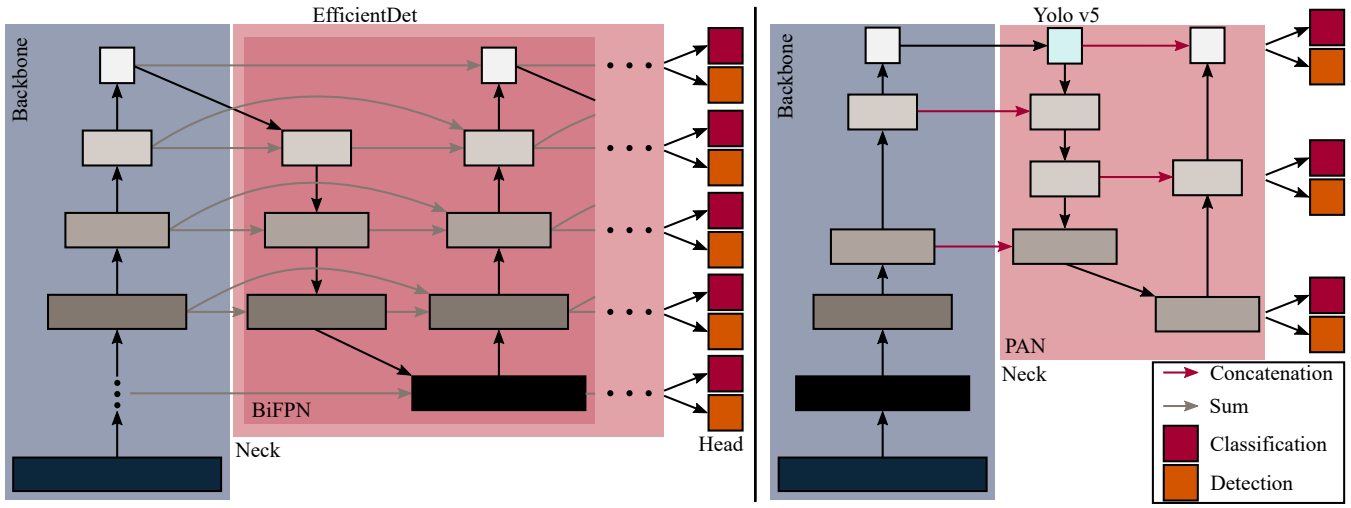

FIGURE 3. A MINIMAL OVERVIEW OF THE EFFIECIENTDET AND YOLO V5 NETWORKS.

from multiple levels of the last pyramid in the neck and have multiple sizes. The classification heads end with softmax layers to transform the values into probabilities. The feature maps at the end of each head are vectorized, reshaped, and concatenated to generate a final vector. The usual shape of this vector is $(4 + numclasses) \times numcandidates$, where numclasses denotes the number of classes, numcandidates denotes the total number of predictions or anchor boxes, and the bounding box refinements are predicted by 4 values in the form of [x, y, height, width], usually normalized. For Yolo v5 the shape of the output vector would be $(5 + numclasses) \times numcandidates$ because of an additional metric computed in the loss layer. Given an input resolution of 640×640, the final vector size of EfficientDet is 5×76725 and the final vector size of Yolo v5 is 6×25200. It is noteworthy to mention that these networks are scalable and can be expanded or shrinked based on the input image dimensions or memory limitations during training. The scalability is normally achieved by adding or removing layer groups to or from the architecture and by varying feature map depths.

Both EfficientDet and Yolo v5 define anchor boxes of multiple sizes based on the possible object sizes in the datasets they are pretrained on. The anchor boxes are tiled across an entire image and usually serve as initial bounding box predictions. The positions of the anchor boxes are gradually updated during training. The predictions are processed in the loss layer. In EfficientDet, the layer contains the classification or focal loss, and the box regression loss functions which are given by

$$L_{\text{class}} = -\Sigma \alpha_c \cdot (1 - p_c)^{\gamma} \cdot y_c \log(p_c), \tag{1}$$

$$L_{\text{box}} = \begin{cases} \frac{1}{2\beta} \cdot (p_b - y_b)^2 & \text{if } |p_b - y_b| < \beta \\ |p_b - y_b| - \frac{\beta}{2} & \text{otherwise} \end{cases}, \tag{2}$$

where α_c and γ denote a data balancing and smoothing factor, respectively, β denotes a factor in smoothL1 loss, y_c denotes the ground truth class, p_c denotes the predicted class probability, y_b denotes the bounding box ground truth value, and p_b denotes the corresponding prediction of an object instance. The overall loss is averaged across all examples.

Similarly, Yolo v5 applies the focal loss function but the corresponding box loss function given by

$$L_{\text{box}} = 1 - \text{IOU}(y_b, p_b), \tag{3}$$

and an objectness loss approximately given by

$$L_{\text{obj}} = -\Sigma y_o \log(f(p_o)), \tag{4}$$

where $p_o = \text{IOU}(y_b, p_b)$, $f(\cdot)$ denotes a function, and IOU is the intersection over union between a ground truth and predicted bounding box, given by

$$\text{IOU} = \frac{p_b \cap y_b}{p_b \cup y_b}, \tag{5}$$

where $(p_b \cap y_b)$ is the intersection between the predicted and ground truth bounding boxes and $(p_b \cup y_b)$ denotes the union between them. The final loss is averaged across all examples. For a more detailed description of the network components, network scaling, definition of anchor boxes, model related pre-processing, etc., the original resources should be referred.

IV. TRAINING

The CNN models are trained in Pytorch on a Quadro RTX 8000 machine. In this work, the official implementation of Yolo v5 [9] and an unofficial Pytorch implementation [12] of EfficientDet is used. Pretrained models of the CNNs are finetuned instead of training a new model with randomly initialized parameters to achieve faster better results. The models need to be modified in order to accommodate a 2-class classification task. This results in a new layer or a replacement of weights in the last layer of the pretrained models, initialized randomly. Since the network models require square images, the training images are zero padded instead of resizing, since the natural aspect ratio of the captured objects is preserved. The coordinates of the bounding box labels are also adjusted

accordingly. The labels for Yolo v5 are formatted as text data. Each image has a ground truth text file containing the object class identification number (ID) and the coordinates in the form of $[x_{\text{center}}, y_{\text{center}}, \text{width}, \text{height}]$, normalized by the image width and height, respectively. In case of multiple drones, there are multiple such annotations in the text file. The labels for the EfficientDet network are provided in a format used for COCO data. The label is a single .json file containing the relevant information about all the training or test images including image name, size, image ID, class name, class ID, coordinates, object area, and certain flags. The coordinates are presented in the form of $[x_{\text{corner}}, y_{\text{corner}}, \text{width}, \text{height}]$, normalized by the image width and height, respectively.

TABLE II provides a summary of the important information related to the training of both networks including the number of epochs, learning rate and related hyperparameters, batch size, optimization, and data augmentation methods used. The default augmentation method refers to translation, rotation, flipping, etc. Various model configurations having different sizes are also trained to review their performance during evaluation. Two variants of EfficientDet, EfficientDet-D0 and EfficientDet-D1, are trained where the earlier model is less complex than the latter in terms of parameters. Similarly, three versions of Yolo v5 denoted by nano (n), small (s), and medium (m) are trained and the nomenclature suggests an increasing order of complexity.

TABLE II
TRAINING DETAILS

	EfficientDet	**Yolo v5**
Number of epochs	100	100
Initial learning rate	1e-4	1e-2
Momentum	0.9, 0.999 (betas)	0.94
Weight decay	1e-3	5e-4
Batch size	16	16
Optimization	Adam, SGD	SGDM
Data augmentation	Mean subtraction, default augmentation	Default augmentation, mosaic input augmentation
Models trained	EfficientDet-D0, D1	Yolo v5-n,s,m

V. EVALUATION

To evaluate an object detection task various metrics related to a successful classification of an object and a successful detection of that object are used and combined in various ways to create more intuitive metrices. In the case of classification, an object is assigned with predicted confidence scores which denote its probabilities of belonging to multiple classes. When its highest score is above a pre-defined threshold the object is deemed to belong to the corresponding class. This prediction is compared with a label to determine its success or failure and thus generate a binary metric. In the case of detection, the intersection over union (IOU) metric is calculated which is given by Eq. (5). An IOU threshold provides a binary outcome in terms of success or failure.

A successful classification and detection results in a true positive (TP) example while an unsuccessful outcome results in a false positive (FP) or a false negative (FN) example. FP refers to detections that have no ground truth and FN refers to correct objects that are not detected. Based on these examples the precision (P) and recall (R) metrics are defined as

$$\text{P} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{6}$$

$$\text{R} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{7}$$

which can be further used to plot a precision-recall (PR) curve based on the outcome of all test examples. Finally, the average precision (AP) and the mean average precision (mAP) metrics are calculated from the PR curve.

FIGURE 4. EXAMPLE RESULT FROM EFFICIENTDET-D0.

FIGURE 5. EXAMPLE RESULT FROM YOLO V5M.

In this work the evaluation is performed based on AP as defined by the COCO evaluation standard. One of the AP metrics is calculated based on an IOU threshold of 0.5. A second AP metric is also calculated based on increasing IOU thresholds from 0.5 to 0.95 with a step size of 0.05 and then calculating the average value. The AP value lies between 0 and 1 and a higher value indicates a better performance.

TABLE III provides the evaluation results of the five trained models on the test dataset in terms of the AP values, the number of model parameters, and model complexity in FLOPS. As

TABLE III
EVALUATION OF MODELS

Network Models	Number of parameters	Complexity (GFLOPS)	$AP_{IOU=0.5}$	$AP_{IOU=0.5:0.95}$
EfficientDet-D0	3828k	3.8	0.870	0.626
EfficientDet-D1	6555k	5.8	0.852	0.600
Yolo v5n	1761k	4.1	0.955	0.732
Yolo v5s	7013k	15.8	0.963	0.734
Yolo v5m	20853k	47.9	0.968	0.740

shown in the table, the models perform well on the test dataset. The Yolo v5 models perform particularly well with Yolo v5m producing the best AP values. The AP scores of EfficientDet-D0 and D1 are also quite good but lower than Yolo v5. This is primarily due to the failure of the EfficientDet models to detect smaller drones with correct bounding boxes in some images. In some images, multiple bounding boxes with same confidence scores might occur based on the choice of IOU threshold for non-maximum suppression (NMS), and the wrong bounding box is selected if the threshold is lowered. This might occur due to imprecise anchor box definition for the task or a lack of proper weighting in the loss function, and thus requires further investigation. FIGURE 4 shows an example detection result from EfficientDet-D1, where the smaller drone is not detected.

The Yolo v5 models perform admirably on the test images and can detect very small or distant drones as well. FIGURE 5 shows an example detection result from Yolo v5m, where the bounding box around the smaller drone is quite tight and has a high confidence score. Occasionally, the models are also able to detect drones from a relatively difficult background although they are not trained with any difficult examples.

In terms of computational complexity or parameters, the EfficientDet models are relatively small along with Yolo v5n, while the remaining models have a higher complexity. Based on the results on the current test dataset it can be concluded that Yolo v5n is a good model considering complexity and performance. A test run on the FLIR Scion OTM366 videos shows that it can achieve an average speed of 80 frames per second (FPS) on GPU.

VI. CONCLUSION

In this work, drone detection is performed on infrared images based on EfficientDet and Yolo v5 CNN models. The image dataset is created by combining a public and a custom dataset. Videos of flying drones are captured with multiple cameras and the images are extracted for annotation. The LabelImg tool is used to annotate the images captured by FLIR Scion OTM366 and create the custom dataset. A big fraction of the annotated images along with images from the public dataset are used for training and testing the models. The evaluation of the finetuned models on the test images indicates a good performance from the EfficientDet models and a very good performance from the Yolo v5 models.

In the future, the goal is to annotate the videos captured by all cameras, while continuing to capture more videos of drones in different surroundings. This should be followed by a statistical analysis and overview of the final dataset. While CNNs tend to improve quite rapidly, more recent and better models can be experimented with. Such models can be modified with additional CNN modules that usually improve object detection tasks. Recurrent models can be created or existing CNNs can be modified to be recurrent in nature and retrained on sequential images for improved detection and tracking. The comparatively difficult images where the drone is partially occluded or obscured due to background should also be integrated in the dataset to improve the performance and robustness of the models.

ACKNOWLEDGMENT

This project is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr which we gratefully acknowledge [project "Digital Sensor-2-Cloud Campus-Platform" (DS2CCP)]. We would also like to thank the department of Electrical Measurement Engineering at Helmut Schmidt University for the opportunity of recording their drones during test flights on the football field.

REFERENCES

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

[2] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81 .

[3] W. Liu et al., "SSD: Single Shot MultiBox Detector," In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) *Computer Vision – ECCV 2016. ECCV, 2016. Lecture Notes in Computer Science*, vol. 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0 _2.

[4] N. Jiang et al., "Anti-UAV: A Large Multi-Modal Benchmark for UAV Tracking," in *ArXiv*, abs/2101.08466, 2021, [Online]. Available: https://github.com/ZhaoJ9014/Anti-UAV [Visited on 26. August 2022].

[5] Tzutalin, "LabelImg," Free Software: MIT License, 2015, [Online], Available: https://github.com/heartexlabs/labelImg. [Visited on 26. August 2022].

[6] M. Everingham, L. Van Gool, C.K.I. Williams et al., "The Pascal Visual Object Classes (VOC) Challenge," in *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2010, https://doi.org/10.1007/s11263-009-0275-4, [Online], Available: http://host.robots.ox.ac.uk/pascal/VOC/ [Visited on 26. August 2022].

[7] T.Y. Kim, "Yolo-to-COCO format converter," [Online], Available: https://github.com/Taeyoung96/Yolo-to-COCO-format-converter [Visited on 26. August 2022].

[8] M. Tan, R. Pang and Q.V. Le, “EfficientDet: Scalable and Efficient Object Detection,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10778-10787, doi: 10.1109/CVPR42600.2020.01079 .

[9] G. Jocker et. al., “ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements,” Zenodo, Oct 2010, https://doi.org/10.5281/zenodo . 4154370, [Online], Available: https://github.com/ultralytics/yolov5 [Visited on 26. August 2022].

[10] T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) *Computer Vision – ECCV 2014. ECCV, 2014. Lecture Notes in Computer Science*, vol. 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1 _48, [Online], Available: https://cocodataset.org/#home [Visited on 26. August 2022].

[11] M. Tan and Q.V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” in *ArXiv*, abs/1905.11946, 2019.

[12] zylo117, “Yet-another-EfficientDet-Pytorch,” [Online], Available: https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch [Visited on 26. August 2022].