IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Real World Object Detection Dataset For Quadcopter Unmanned Aerial Vehicle Detection

Maciej Pawełczyk, Marek Wojtyra
Institute of Aeronautics and Applied Mechanics, Warsaw University of Technology, Warsaw, Poland

Corresponding author: Maciej Pawełczyk (e-mail: mpawelczyk@meil.pw.edu.pl).

This research was supported by the Institute of Aeronautics and Applied Mechanics funds for scientific research.

ABSTRACT Recent years have shown a noticeable rise in the number of incidents with drones, related to both civilian and military installations. While drone neutralization techniques have become increasingly effective, detection most often relies on professional equipment, which is too expensive to be used for all critical nodes and applications. Therefore, there is a need for drone detection systems that could work on low performance hardware. Its critical component consists of an object detection system. In this article, we introduce a new object detection dataset, built entirely to train computer vision based object detection machine learning algorithms for a task of binary object detection to enable automated, industrial camera based detection of multiple drone objects using camera feed. The dataset expands existing multiclass image classification and object detection datasets (ImageNet, MS-COCO, PASCAL VOC, anti-UAV) with a diversified dataset of drone images. In order to maximize the effectiveness of the model, real world footage was utilized, transformed into images and hand-labelled to create a custom set of 56821 images and 55539 bounding boxes. Additionally, semi-automated labelling was proposed, tested and proved to be very useful for object detection applications. The dataset was divided into train and test subsets for further processing and used to generate 603 easily deployable Haar Cascades as well as 819 high performing Deep Neural Networks based models. They were used to test different object detection methods to determine the long term feasibility of a large scale drone detection system utilizing machine learning algorithms. The study has shown that Haar Cascade can be used as the Minimum Viable Product model for mediocre performance but fails to scale up effectively for a larger dataset compared to the Deep Neural Network model.

INDEX TERMS Aerospace engineering, Aerospace safety, Artificial intelligence, Computer vision, Databases, Image processing, Unmanned aerial vehicles

I. INTRODUCTION

Due to the increasing utilization of civilian Unmanned Aerial Vehicles (UAVs), the need to identify drones in the sky, especially in urban environments, is greater than ever. Recent high publicity events connected to UAVs include the notable Gatwick Airport drone incident [1] and attacks on gas installations in Saudi Arabia [2]. This means both the risk of privacy intrusion and critical infrastructure trespassing, possibly posing a danger of human harm in case of airports, power stations, water treatment plants and other. Furthermore, recent tests have shown that even unintentional drone operation in the vicinity of an airport can result in heavy airplane fuselage damage as analyzed by Dayton University [3], showing the impact of a drone collision with an airplane wing. Typically drones are hard to detect visually due to their relatively small surface area and their limited movement once positioned over a target. This problem is exacerbated by the fact that most industrial camera systems have low frame processing rate, which makes direct utilization of image flow and frame-by-frame comparison techniques difficult. As such it was the authors' motivation to propose Single Shot Detector (SSD) based Haar Cascades and Deep Convolutional Neural Networks machine learning algorithms to determine the detection capability of those technologies in real world application of UAV detection.

The task of detecting UAV requires mid/high resolution images with a relatively small drone (calculated as a percentage of an input image) visible in the background.

Currently, there is no available object detection dataset consisting of a large set of UAVs specifically made for object

detection and not for object tracking. Therefore the authors decided to create an entirely new dataset based on real footage (with all imperfections that only the real world can offer), which would severely limit the number of classes to 1, but at the same time offer superior performance on that single class. Considering that the model needs to learn different object representations given similar foreground, in this situation extracting a high stride sequence of frames from the video footage was an acceptable solution.

Typically UAV detection systems relied on multistage processing systems which include steps like image thresholding, background removal and object tracking. These methods show limited capability in urban areas due to a high false positive rate because of planes, birds, leaves, trash and other objects. The presented machine learning based system utilizes both tested approaches like Haar Cascade based systems along with high performing Deep Learning based approaches.

The article presents an overview of available datasets used for image classification and object detection tasks utilized for machine learning computer vision research and development. Then dataset generation along with the resultant training and testing dataset is shown. Since the dataset was purposefully made from a large set of distributions of real-world footage, object tracking based methods could not be used for image labelling. To reduce to manual burden required the authors decided to begin the tagging process with test set preparation (5375 images, 2625 positives) and just a small fraction of train set (4500 images), which was then in turn used to create ANN-based detection model, working as "drone proposal" network. While this has not completely eliminated human effort required it allowed tagging time reduction from 45 to 22.5 seconds (on average). This novel approach is easily scalable and can be used for labelling various kinds of objects. Hence, the preparation of large domain-specific datasets may be streamlined. In this instance, the method was used to create and propose a novel UAV dataset consisting of 51446 train and 5375 test 640x480 RGB images presenting drones in different types, sizes, scales, positions, environments, times-of-day with corresponding XML label set, prepared for Haar Cascade training. The dataset is aimed to increase the security level in critical civilian and military infrastructure.

This is followed by a brief overview of object detection metrics and methods with a strong emphasis on Haar Cascade and Deep Learning based models, which are used in this research.

In order to create a drone detection system, 603 Haar Cascade and 819 ANN models were evaluated for the purpose of easy to use, low compute applications (Haar Cascade with OpenCV) and more demanding, but higher performing application (security surveillance with ANN MobileNet based models).

Finally, the article presents detection algorithms results, which represent a reference point for the development of a more complicated detection system. The articles ends with a chapter on future work with ideas for future improvements and research.

The primary novelties and importance of the paper may be summarized as follows:

1. We present and share a novel drone detection dataset consisting of 51446 train and 5375 test set images with corresponding annotations (52676 train and 2863 test set drone instances).
2. We propose and successfully use a novel method of obtaining object detection bounding box tags by first defining the test set and a portion of the ultimate train set in order to fine tune pretrained CNNs to create "drone proposal" bounding boxes, which can then be used to accelerate tagging process without relying on object tracking methods.
3. We present developed Haar Cascade based object detection models, which can be easily incorporated into edge devices using existing frameworks such as OpenCV. We also present the results of our research on the amount of positive and negative examples required to maximize the cascade performance.
4. We present fine-tuned MobileNet ANN based models, depicting the results of our research and network optimization along 1 million iterations.
5. We demonstrate a successful application of the proposed methods on the challenging problem of real-world, deployment dataset with a direct side-by-side comparison of Haar Cascade and ANN based solutions, presenting the advantages and disadvantages of both approaches.

II. RELATED WORK

Various approaches to the problem of UAV detection have been proposed [4], usually utilizing motion detection and optical flow detection algorithms combination [5], highly augmented dataset [6], utilizing spatial-temporal information [7] or mainly focusing on the object motion itself [8]. For practical uses, an SSD system — as proposed in this paper — will be used in conjunction with multiple other systems, both software and hardware based. An overview of this multispectral UAV detection was proposed in [9].

Most of the object detection applications for drones consider using a drone mounted camera to detect objects on the ground from a high altitude. It is less common to detect flying objects themselves, specifically drones. While binary differentiation (between drone and non-drone) within one end-to-end neural network is the ultimate goal for models like this, usually the main priority is directed to detecting each of those two classes separately [5].

Typically object detection tasks are addressed using open source image classification and object detection datasets like The Modified National Institute of Standards and Technology (MNIST) dataset [10]. The MNIST dataset contains 60,000 training and 10,000 test images of 28x28 pixel grayscale (1 channel), centered, handwritten digits. Today this dataset represents a typical introduction dataset for image classification problems, due to its small disc space

requirement, multiclass problem statement (10 classes) ease of access and easily available training materials/tutorials.

Since the MNIST dataset does not represent real objects, its relatively modern extension was created in 2009 by the Canadian Institute For Advanced Research (CIFAR) in the form of CIFAR-10 dataset [11], representing 10 classes of real world objects (cats, dogs, horses, birds, deer, frogs, cars, trucks, ships and airplanes) as a research subset of 80 million tiny images dataset [12] for use in machine learning and computer vision challenges. The CIFAR-10 dataset contains 60,000 32x32 pixel color (3 channels) images. The classes are evenly distributed, with 6,000 images for each class. CIFAR-100 is an extension of that dataset containing 100 classes [11] and 600 images for each class, grouped into 20 superclasses (larger classes aggregating smaller subsets).

CIFAR-10 is also often used for general testing of machine learning algorithms but is limited due to its low resolution and lack of bounding box labels/annotations (all objects are simply centered on their corresponding images) needed for object detection tasks. At the same time, it represents an on-going trend to test machine learning algorithms on real world images, rather than on limited datasets.

Therefore, another real world image dataset, called ImageNet, was created in 2009 [13]. As of today, it consists of more than 14 million real world hand-annotated images of more than 20,000 categories, out of which more than 1 million contain bounding box annotations [14]. Since 2010 the dataset has been used to run the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which allows a highly objective comparison between machine learning models. The ILSVRC is not run on the entire data, but rather on its subset consisting of 1000 classes. In 2012, Alex Krizhevsky was awarded the 1st prize in this challenge using Deep Convolutional Neural Networks, leading to the recent resurgence in Artificial Neural Network (ANN) research and development [15]. Since the ImageNet dataset additionally provides bounding box annotations for more than 1 million images it was also used to host the ImageNet Object Localization Challenge, which aims to create a State of The Art (SOTA) model based on 150,000 images and 1,000 categories with bounding box annotations.

While the ImageNet dataset was not inherently designed for object localization/detection tasks, the Pattern Analysis, Statistical Modeling and Computational Learning Visual Object Classes (PASCAL VOC) dataset can be used specifically for these purposes [16]. The dataset has been extended since the challenge origin in 2005 up to its end in 2012, when VOC challenges dedicated to object detection tasks were held [16]. The latest dataset contains 20 classes, 11540 images and a total of 27450 annotations (the testing set does not contain annotations).

The main disadvantage of ImageNet and PASCAL VOC datasets is that they usually present given objects on its own, without the context of the surrounding the object usually exists in. In order to counteract that, the Microsoft COCO (Common Object in Context) dataset has been created containing 330,000 images, 1.5 million object instances, 80 categories, pixel-based segmentations and image captions [17]. Neither COCO, PASCAL VOC nor ImageNet dataset contain sequential images extracted from a video. Furthermore, average COCO dataset resolutions are also larger than those of ImageNet and PASCAL VOC datasets.

Recently all three datasets have been used to create general object detection models, which could be used as weight initialization for similar problems and then fine-tuned on the problem dataset allowing quicker training and reducing the need for a labelled dataset. This approach, called transfer learning, allows using a pretrained dataset and model architecture rather than creating an entirely new network based on a limited dataset [18], [19]. In order to ease that process popular deep learning framework called TensorFlow has included a set of pretrained object detection models, typically trained on the datasets mentioned before in the article [20]. COCO, PASCAL VOC and ImageNet datasets represent useful datasets and benchmarks for image classification/localization/detection models and research. Unfortunately, the mentioned datasets combined had little to offer in terms of drone images in different environments, this is why a new, large dataset had to be proposed for extensive research on efficient and high performance UAV detection systems.

Recently two datasets have been proposed and used for 3D flight trajectory reconstruction [21], [22], and specifically for anti-UAV purposes [23], [24]. While those two datasets allow for large size drone-based dataset, they were created in well-defined conditions with chessboard based calibration and special equipment added for UAV tracking and subsequent dataset preparation. While this approach is extensively valuable for research purposes, it does not represent real world occlusion, clutter, illumination, camera movement, and a wide range of urban environments that UAVs need to be tested against. While the anti-UAV dataset allowed the application of transfer learning methods for ANN models such as YOLO [25], an additional dataset was needed to represent UAVs in a broader range of environments.

For this purpose, a high stride sequence of frames was extracted from the video footage. This approach is very similar to ImageNet, PASCAL VOC and COCO, where images used to create the dataset were extracted and processed from third-party image/video URLs, freely available from the general URL search, though the original data are not owned by the authors of this article and can/will be used for research purposes only.

III. DATASET PREPARATION

First, the authors created a custom ANN based model using fully manually labelled images. After obtaining a sufficient number of training and testing data samples, the obtained models were used to accelerate the dataset generation process through a semi-automated process.

Since the resultant training dataset was sufficiently large (up to 51446 images), no overfitting was noticed and as such, no validation dataset was used. Typically, a 70-10-20 training-validation-test split would be used, but in order to facilitate model training, a large training dataset and a diversified testing dataset were used. Subsequent analysis showed that the model did not overfit into the testing dataset. After all the steps mentioned above, a total of 56821 images and 55539 bounding boxes were obtained and divided into the training and the testing dataset. Dataset, its sources and references are publicly available [26].

A. Dataset generation process

The authors decided to utilize publicly available drone footages, both downloaded from the internet and recorded personally by the authors. Authors had analyzed the potential of using synthetically rendered images based on available CAD models [27] but ultimately decided to use only real records, to allow the detection model to capture real-world imperfections. Every 50th frame of the movie (approximately every 2 s) was extracted and saved for later use to create an input database. The first 4500 training images and all 5375 testing datasets were hand labelled by the first author. The hand labelling took approximately 30-60 seconds per item, depending on the image. An overview of the manual labeling pipeline is shown in Fig. 1.

Then the authors used the pretrained COCO *ssd_mobilenet_v1* model [20] to train artificial neural network based drone detectors with 0.50 detection confidence. All model parameters remained default, which included weighted sigmoid classification loss, weighted smooth localization loss and fixed image shape resizer of 300x300. The model training process automatically saved a checkpoint approximately every 10 minutes. All checkpoints created were used for the analysis. The preliminary model was run for 600,000 Stochastic Gradient Descent (SGD) iterations. Then all intermediate checkpoints were saved and finally frozen and tested on the testing dataset.

The best performing model was obtained for iteration 249204 with model accuracy of 67%, F_1 score of 61%, AUC of 0.56, mAP of 52%, precision of 73% and recall of 52% (metric's definitions are presented further in the paper). This model was then run on all frames/images obtained later, thus creating a semi-automated labelling pipeline.

Images labelled by the resultant model were manually corrected to ensure the highest quality of labels and images deemed negative (no drone detected by the ANN model used) were manually checked to ensure that all drones in the images were indeed captured. Since this approach took approximately 15-30 seconds per image, this allowed the ultimate productivity boost of approximately 100%. The process used is shown in Fig. 1. Initial data are acquired and subsequently processed through the labeler (in this instance, the first author) to get labeled data, which were ultimately used to create the detection model tested with the use of a separate validation dataset. The inference model acquired in this process was then used on available footage to acquire more data, in this instance labelled by ANN. In order to maximize the quality of the labeled data, the "human labeler" is also present in the loop to ensure that all bounding box positions are correctly marked.

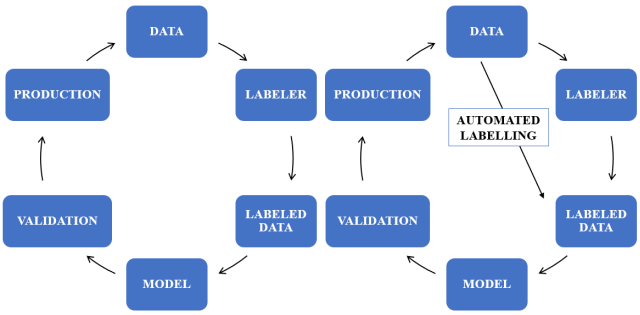

FIGURE 1. Overview of the labelling pipeline with and without implementation of the automated novel labeling process. Its introduction allows approximately 50% tagging time savings.

By introducing a semi-automated process it is estimated that the authors managed to reduce the average image labelling time from 45 to 22.5 seconds. As automated labelling was performed on 46946/51446 training images, the total labelling workload was reduced by approximately 293 hours or 36 eight-hour-long workdays.

B. Training dataset

Initially, the training dataset was created on the basis of readily accessible open source datasets such as ImageNet, Google based graphic search and similar publicly available sources. Unfortunately, this approach was highly inefficient and fewer than 500/51446 training images were created in this way. A vast majority of the training samples came from 578 drone videos obtained from popular video services.

The full training dataset consists of 51446 images scaled down from different resolutions (ranging from 640x480 to 4K) to the resolution of 640x480, out of which 51445 images contain a total of 52676 drone bounding boxes and 1 negative image (not containing any UAVs). For the object detection task, a vast majority of input images are negative examples (do not contain the class in question), therefore adding a negative dataset would not provide any value added to the model. Similarly to the validation dataset, the bounding boxes used for training purposes are presented in bulk in a 2D histogram, as shown in Fig. 2.

In the training dataset, there are more small objects than large objects. Specifically: approximately 40.8% of objects are small (area < 1024), 35.8% are medium (1024 < area < 9216), and 23.4% are large (area > 9216) as specified by the COCO challenge [28]. The area is measured as the number of pixels in each of the bounding boxes. A cumulative histogram representing the bounding box area as a percentage of the entire image area is presented in Fig. 2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3026192, IEEE Access

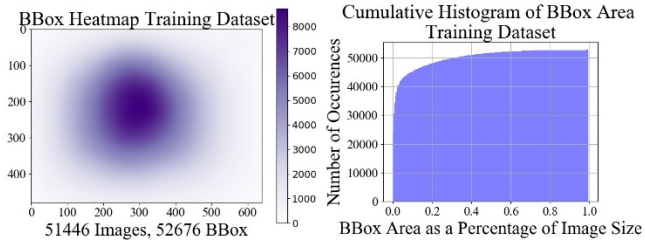

FIGURE 2. Spatial distribution of full training set bounding boxes and normalized cumulative histogram of bounding box areas. The heatmap shows the density of a drone occurrence (with axis representing image width and height) after scaling it to 640x480 pixels. As the histogram shows, the majority of drone sizes are small, which represent the typical expected deployment scenario of the developed model – security applications to prevent drone trespassing.

FIGURE 3. An example from the train set – a drone on the relatively cloudy, blue background.

FIGURE 4. An example from the train set – a drone on the rural background.

Overall, the aim of a large diverse dataset was to present drones of different types, sizes, scales, positions, environments, times-of-day, etc., so as to allow a broad range of representations to train object detection models. This includes small (calculated as a percentage of overall image) drone on blue sky background (as shown in Fig. 3) and drones within an urban environment (as shown in Fig. 4).

C. Testing dataset

The testing dataset was extracted from 21 drone videos obtained from popular video services and 29 videos which do not show drones, which were used to create a negative sample dataset of urban areas, nature, airports and plane footage.

The testing set consists of 5375 images scaled down from different resolutions (ranging from 640x480 to 4K) to the resolution of 640x480, out of which 2750 do not contain any UAVs (negatives) and 2625 images containing drones (positives), a total of 2863 objects. The testing dataset bounding box position is presented in bulk in Fig. 5. Each of the bounding boxes was projected onto a 2D matrix corresponding to 640x480 image resolution, resulting in a density map as shown in Fig. 5. The vast majority of drones were marked in the center of the image, which reflects the training dataset. This is a natural consequence of the chosen data acquisition method, which relied on extracting frames from UAV videos where the drone was a centrally positioned object as the camera operator fixed on it. Ideally, both the training and the testing (validation) dataset should have an even spatial distribution. In practice, models trained on the basis of the proposed dataset are well protected from uneven spatial distribution of the binary class on the given image. Haar Cascade uses a sliding window approach, which is spatially independent (excluding padding) while artificial neural network transfer learning based methods shown later use the region proposal network approach, which significantly reduces dependency on spatial evenness.

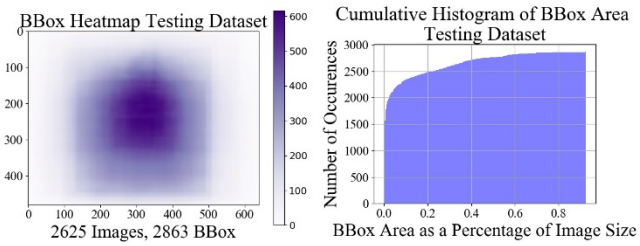

FIGURE 5. Spatial distribution of testing (validation) set bounding boxes and normalized cumulative histogram of bounding box areas. The heatmap shows the density of drone occurrence (with axis representing image width and height) after scaling it to 640x480 pixels. As the histogram shows the majority of drone sizes are small, which represent the typical expected deployment scenario of a developed model – security applications to prevent drone trespassing.

In the testing dataset, there are more small objects than large objects. Specifically: approximately 36.3% of objects are small (area < 1024), 35.3% are medium (1024 < area < 9216), and 28.3% are large (area > 9216) as specified by the COCO challenge [28]. A cumulative histogram representing the bounding box area as a percentage of the entire image is presented in Fig. 5.

Similarly to the training set, the aim of the authors was to create a diverse dataset with even class distribution, highly diversified background, and negative examples with difficult,

typically urban, background. Examples of test dataset frames are presented in Fig. 6 and Fig. 7.

FIGURE 6. An example from the test set – no drone present on the image containing a difficult to classify, urban environment.

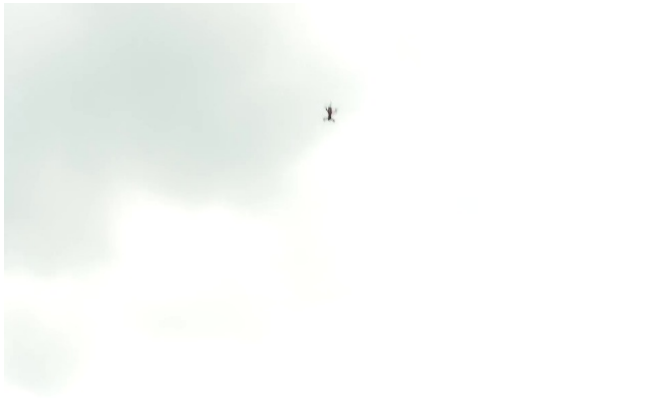

FIGURE 7. An example from the test set – a drone present on the cloudy image.

IV. OBJECT DETECTION OVERVIEW

Object detection was traditionally based on feature extraction based on image statistics (usually after computer vision based preprocessing such as thresholding). Recently this process has been replaced by automated feature extraction with methods such as Haar Cascades and Convolution Neural Networks.

A. Aims and challenges

Object detection is a computer vision technique aimed at classification and bounding box retrieval of one or more classes for the given image assuming more than one instance of a class present on that image. While methods such as convolution neural networks rely on operations directly on the input pixels, typically, machine learning methods relied on task specific features extracted from the image (such as HUE intensity [29], HOG/SIFT features [30], [31], GIST [32], LBP [33], Texton [34], etc.), which were then stored as feature vectors and finally compared with other image-based abstractions in high dimensional hyperspace. Dimensions usually represented statistical descriptions of the feature vectors to be further processed with machine learning methods (Principal Component Analysis, Support Vector Classifier, Linear Discriminant Analysis and others). This process is shown in Fig. 8. The input image is reduced to a specified size (as the common denominator for both training/testing dataset and model deployment), then multiple specified features are extracted in order to create a feature vector (a combination of all extracted features). The feature vector is then processed by a number of Machine Learning routines and finally, one hot encoder result is acquired, which specifies the expected input image class.

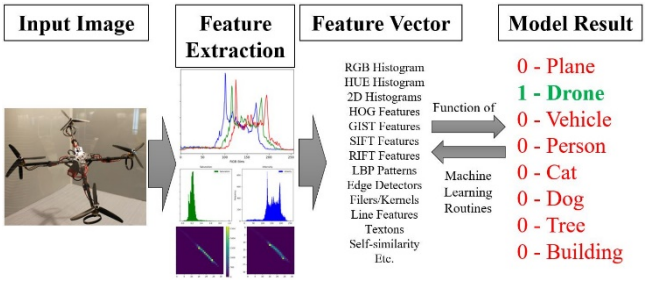

FIGURE 8. Feature-based representation of object detection.

The purpose of the feature vector was to create an abstract representation of input image, which could be then efficiently processed to establish a correct output. Different kernels and filters were used to extract edges and then to determine image features. Usually, a specified, finite set of filters would be tested to determine the best result, sometimes stacked (cascading) on top of each other. This created tree like structures of different processes and filters combined to form a sequential algorithm to maximize the overall performance. This tree like structure, often referred to as hierarchical output, was very time consuming to create and usually showed good performance on one specific application.

A hierarchical representation of the problem is one of the methods allowing successful mitigation of challenges faced by any image processing system. The primary challenges include:

- Viewpoint variation (camera focal properties variation), which include size/scale variation, rotation/inclination variation, position variation, pose variation
- Object variation (intra-class variation), such as object color/shape/size variation and object deformation - different dog breads, different drone models
- Illumination (including shading), reflections, shadows, atmospheric effects (hot gases, natural convection, rain, snow, haze, fog, etc.)
- Occlusion & background clutter

A general approach to building image classifiers is to collect an image dataset (including labels), use a machine learning algorithm to train it, tune the algorithm on the dataset and evaluate the result using a separate dataset called the testing dataset or the test dataset. Assuming that there is no

hyperparameter tuning used, there is no need for a validation set created aside from the testing set. There are multiple approaches to handle computer vision challenges, which typically utilize advanced data processing algorithms such as Support Vector Machine (SVM), Hidden Markov Models (HMM), Principal Component Analysis (PCA). While those techniques are very helpful, they lack an adequate amount of model parameters to sufficiently generalize the given problem, which would allow object detection in any object condition and background. At the same time, they offer industry recognition, acceptable performance on small datasets and the ability to train without expensive equipment. A method that has enjoyed commercial success and industrial application is Haar Cascade based object detection.

B. Haar Cascade algorithm overview

Machine learning Haar Cascade object detection system, proposed in 2001 by Paul Viola and Michael Jones [35] is based on the work by Papageorgiou et al. (1998) [36], although the algorithm itself can be traced back to 1973 [37] or back to the original formulation of the mathematical wavelet function by Alfréd Haar [38], which is the source for the name of the algorithm.

Through the last two decades, the technology has been used preliminarily for face detection [39] in different appliances, such as hand cameras (Fuji camera in 2006 [40]), cell phones, image processing software, social media, etc. This is due to the fact that high model training computation requirements are compensated for by an extremely lightweight and effective solution during model deployment, which is usually illumination invariant [41]. Multiple other cascades exist for everyday items, such as sofas, TVs, vehicles [42] or even agriculture. Haar features have also been used for medical applications [43], industrial applications [44], general purpose detection [45] and for object tracking [46]. The authors did not manage to obtain Drone Haar Cascade for the detection of Unmanned Aerial Vehicle (UAV) of any kind, hence decided to build their own one. Haar features are simple pixelwise relations between adjacent pixels, which form a predefined shape as shown in Fig. 9.

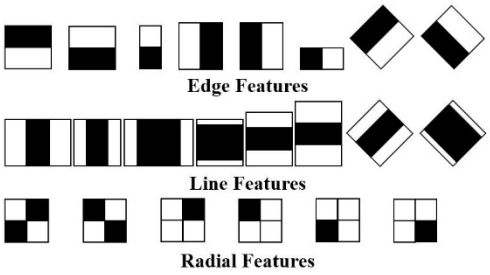

FIGURE 9. Overview of Haar Features. Expanded upon available literature [35], [47], [48].

The Haar detector scans an input image at multiple scales, starting with a base size and incrementally increasing the size of the window. The sliding frame is then converted into integral and "boxlet" image representation, later used to fit within predefined Haar features.

Classifiers are called to solve a sequence of "weak" learning problems, With the final, "strong", classifier representing a weighted combination of weak classifiers followed by a threshold.

The detection phase of the Haar algorithm includes multiple stages, with a weak classifier trained on each stage separately, which allows disregarding the majority of sub-windows as negatives in the first stages. For more complicated detection problems, a higher number of training stages are used (usually more than 20, the system proposed by Viola and Jones used 38 [35]). If a specific area was marked as the object of interest a sufficient number of times, it is considered the object in question.

C. Neural network convolution algorithm overview

In general, the artificial neural network (ANN) works upon the concept of loss minimization, usually defined using Softmax or SVM formulations [49]. and has been recently used for a wide variety of applications from mechanical engineering [50], aerospace engineering [51] to robotics and control [52]. For the purposes of computer vision ANNs typically utilize a combination of Convolutional Neural Networks (CNNs) and Fully Connected Layers (FCs). CNNs have proven to be an effective modeling solution for applications ranging from computer vision (image classification, object detection, neural style transfer) [53], [54], cybersecurity [55], time series processing [56], fluid mechanics [57] and general physics challenges [58].

A simplified CNN based process is shown in Fig. 10. Input image height and width are usually equal. In this example, a 285x255 input image would be scaled to a new size of 255x255 and then the network is trained. As such, any processed image will have to be scaled as well to fit the required input size.

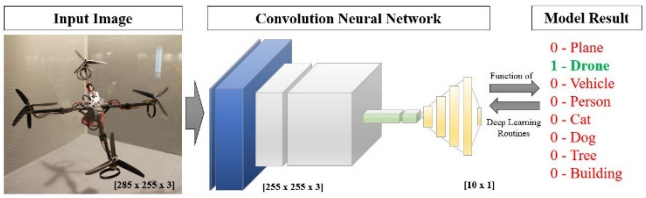

FIGURE 10. Simplified convolutional neural network based object detection.

Python 3.6+, mainly via Python IDLE and Jupyter framework, was used for dataset generation and training/testing purposes. All models were developed on Windows 10, Intel Core i7-8700K CPU @3.70 GHz with 4x8 GB DDR4-3000 RipJaws V RAM and MSI GeForce GTX 1080 Ti graphics card. Aside from vertical flipping (for deep learning only), no dataset augmentation was used.

TABLE 1.
10 HAAR CASCADE MODELS WITH THE HIGHEST ACCURACY AT 0.5 IOU

Positives Count [#]	5174	4000	1600	4000	5174	3350	1800	2000	3500	1750
Negatives Count [#]	13000	10000	400	12000	10000	6700	3659	4000	7500	3500
Total Images Used [#]	18174	14000	2000	16000	15174	10050	5459	6000	11000	5250
Accuracy [%]	55.4	55.3	55.3	55.2	55.2	55.2	55.1	55.0	55.0	55.0
Precision [%]	79.0	78.3	80.3	81.7	72.9	74.3	76.9	68.2	68.3	78.0
Recall [%]	15.8	15.4	14.8	14.5	16.6	16.6	15.9	18.1	18.0	14.9
Specificity [%]	95.7	95.7	96.3	96.7	93.9	94.2	95.2	91.7	91.7	95.7
F_1 Score [%]	26.3	25.7	25.0	24.7	27.0	27.2	26.3	28.6	28.5	25.1
MCC [%]	19.1	18.6	19.1	19.6	16.4	17.1	18.0	14.4	14.4	18.0
True Positive [#]	451	441	423	416	474	476	454	517	516	428
False Positive [#]	120	122	104	93	176	165	136	241	240	121
True Negative [#]	2697	2703	2718	2715	2687	2665	2673	2647	2646	2696
False Negative [#]	2412	2422	2440	2447	2389	2387	2409	2346	2347	2435
Total Process Time [s]	360	318	76	320	347	279	231	233	302	218
Detection Time [s]	355	313	71	316	342	275	226	229	296	213

V. UAV OBJECT DETECTION RESULTS

This section presents object detection evaluation metrics along with Haar Cascade and CNN based model results.

A. Evaluation metrics

In general, artificial intelligence, machine learning, and—more recently—deep learning aims to solve the regression problem, the classification problem or a combination of both. For the problem of classification, a typical validation metric is derived from the confusion matrix, which represents a comparison between true and predictive conditions. For a single class classification problem it can be represented by a 2x2 solution matrix with supplementary metrics [59]. The final evaluation metrics can be computed as a combination of confusion matrix elements [60]:

$$Accuracy = \frac{\sum True\ Positive + \sum True\ Negative}{\sum Total\ Population} \quad (1)$$

$$Precision = \frac{\sum True\ Positive}{\sum True\ Positive + \sum False\ Positive} \quad (2)$$

$$Recall = \frac{\sum True\ Positive}{\sum True\ Positive + \sum False\ Negative} \quad (3)$$

$$Specificity = \frac{\sum True\ Negative}{\sum True\ Negative + \sum False\ Positive} \quad (4)$$

$$F_1 = \frac{2 \cdot \sum True\ Positive}{2 \cdot \sum True\ Positive + \sum False\ Positive + \sum False\ Negative} \quad (5)$$

$$MCC = \frac{\sum TP \cdot \sum TN - \sum FP \cdot \sum FN}{\sqrt{(\sum TP + \sum FP) \cdot (\sum TP + \sum FN) \cdot (\sum TN + \sum FP) \cdot (\sum TN + \sum FN)}} \quad (6)$$

The metric used in the final evaluation varies depending on the application. In UAV detection, the problem model should be both able to detect drones when they appear (maximize the number of true positives and precision) and be robust enough to rarely detect drones when they do not appear (minimize the nmber of false positives and fallout). If the end-users are interested in one single value combining those two requirements, then the F_1 score is typically the recommended option.

In the case of unbalanced classes, another useful metric is the Matthews Correlation Coefficient (MCC), which is specifically designed for binary classification [61].

To establish a level of overlap for ground truth and proposed boxes, typically the IoU (Intersection over Union) metric is used as shown in Fig. 11. Typically, IoU of 0.5 is used. Higher IoU levels are not recommended for large distance object detection as small pixel changes would result in a large impact on IoU.

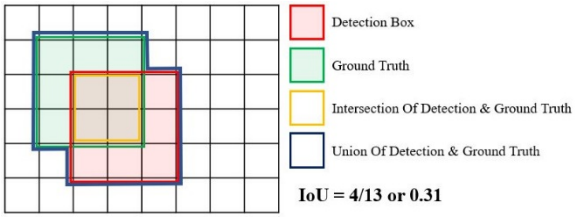

FIGURE 11. IoU metric visualization.

For models providing detection confidence, two additional metrics can be used— the Receiver Operating Characteristic (ROC) curve and the Precision-Recall curve, created on the basis of parameter calculation for the entire confidence range. The area under the ROC curve, mathematically ranging between 0 and 1, is a one value metric called Area Under Curve (or AUC). For multiclass problems, Average Precision is averaged across classes to calculate the mean Average Precision (mAP). In the analyzed problem of single class detection mAP and AP will be equal and will be used interchangeably. The example of these metrics, obtained during the analysis presented further in the article, is shown in Fig. 12. The datapoints were established by 117 individual results aggregations performed for model confidence rising from 0 to 100% (with a step decrease to 0.1% for ranges of 0–10% and 99–100%).

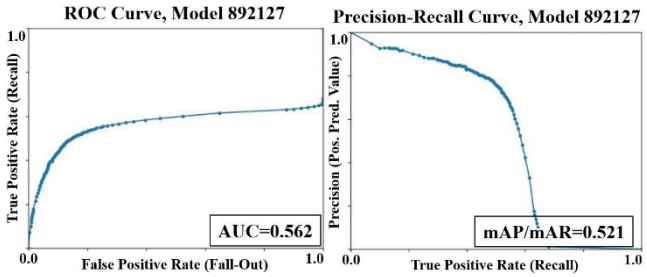

FIGURE 12. ROC curve (left) and Precision-Recall curve (right) as obtained for the ANN (CNN) based model described later in the text.

For the purpose of UAV detection IoU of 0.5 and detection confidence of 0.5 will be used. ROC/AUC/mAP and Precision-Recall curve have been obtained for every single inference model frozen every 10 minutes. MCC, Accuracy, F_1 score, precision, recall, sensitivity and specificity have been calculated for each model.

B. Haar Cascade based object detection results

Tab. 1 shows the results of the best performing Haar Cascades (top 10 as measured by accuracy), while the aggregate results for the remaining models (all 603 of them) are presented later on in this section. As mentioned earlier, all performance metrics are created on the basis of the testing dataset consisting of 2625 positives and 2863 negatives. While a relatively low number of false positive classifications is favorable (resulting in high precision), recall is in fact very low, which is corresponding to overall low model performance. Top accuracy does not exceed 55% and this is mostly due to the fact that Haar Cascade models have optimized themselves to detect only the most obvious drone object examples.

Haar Cascade training process time is growing with the total number of images used. Simple models with up to a few hundred images train in less than 15 minutes. Unfortunately, the largest Haar Cascade based model (18174 images used) took 3 months of constant processing to train. The advantage of this approach is that CPU usage throughout the training process is relatively low, which means that training multiple models/classifiers at the same time is possible, but ultimately this also creates a project bottleneck and is very likely to result in a failure (due to unexpected power outages, forced system/antivirus restarts, company forced restart, internal system errors, malfunctions, etc.). This means that a smaller number of samples should be used. Fig. 13 shows the impact of the number of total images used on the main performance metrics of accuracy and F_1 score.

As shown in Fig. 13, the resultant confusion matrix based accuracy does not exceed 55% and F_1 score does not exceed 0.32. A similar visualization performed on the number of positives and negatives used is shown in Fig. 14 and Fig. 15.

The presented results show the limitations of the Haar Cascade algorithm. The Haar-based classifier does not scale well to a large number of samples and its detection capability is highly limited. This outcome is not dependent on the number of positive/negative images used for the training process. In fact,

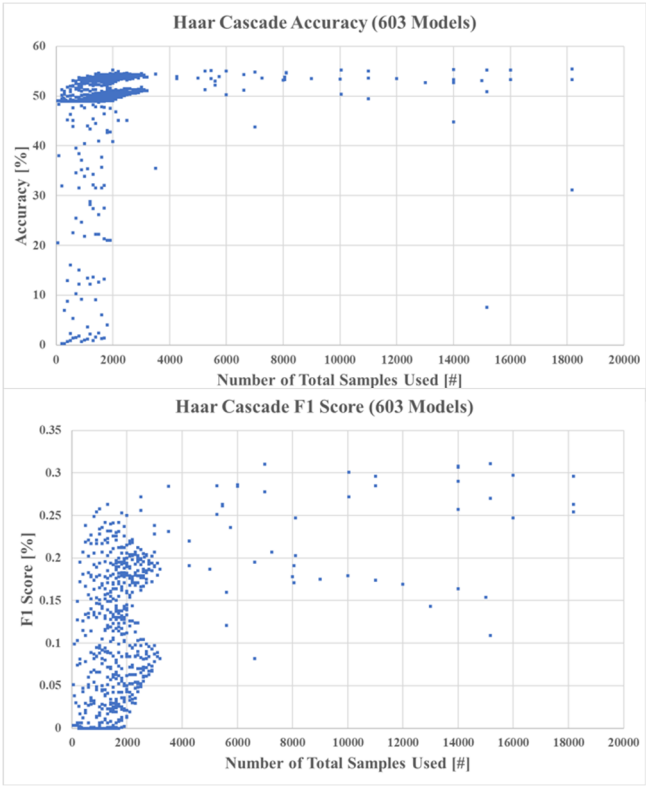

FIGURE 13. Impact of the total number of images used on the main model performance metrics – accuracy and F_1 score.

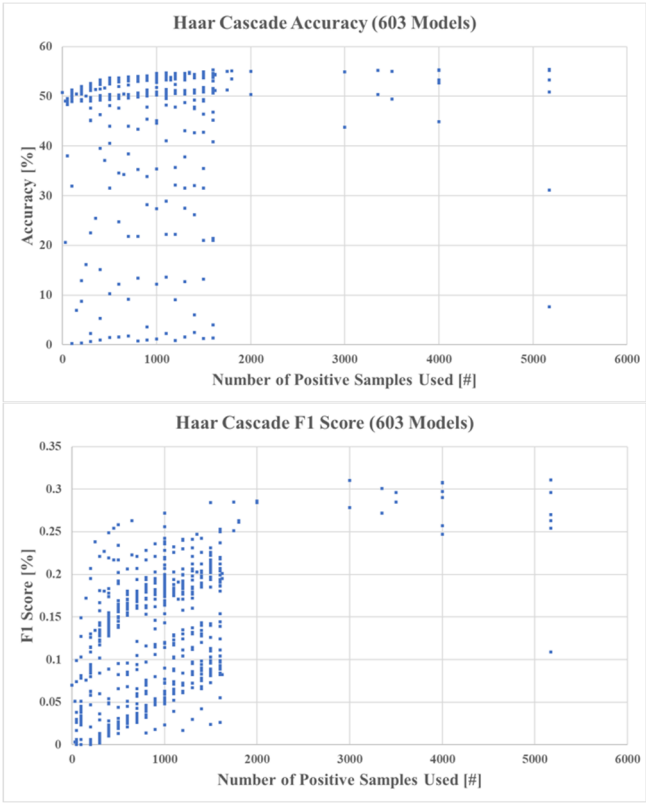

FIGURE 14. Impact of the total number of positives (images showing at least one instance of the object class to be detected) used on the main model performance metrics – accuracy and F_1 score.

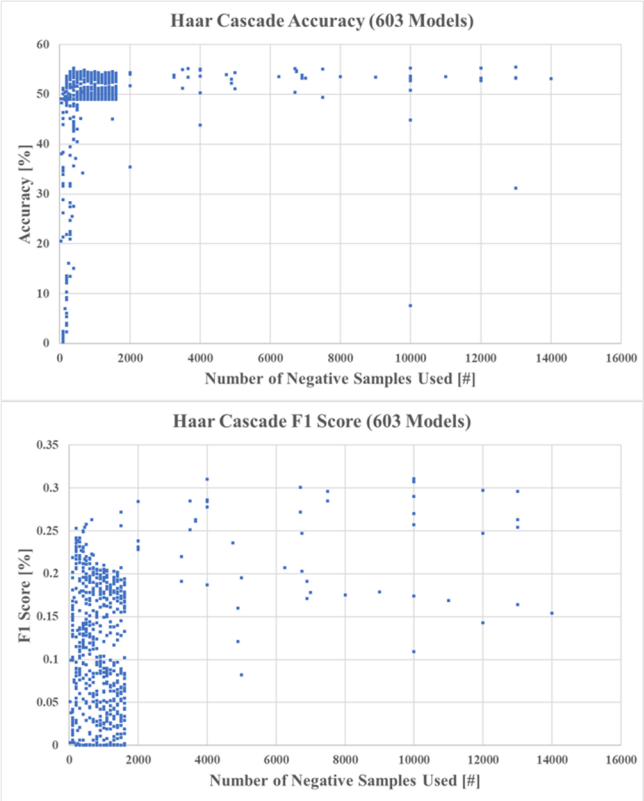

FIGURE 15. Impact of the total number of negatives (images with no instance of object to be detected) used on the main model performance metrics – accuracy and F_1 score.

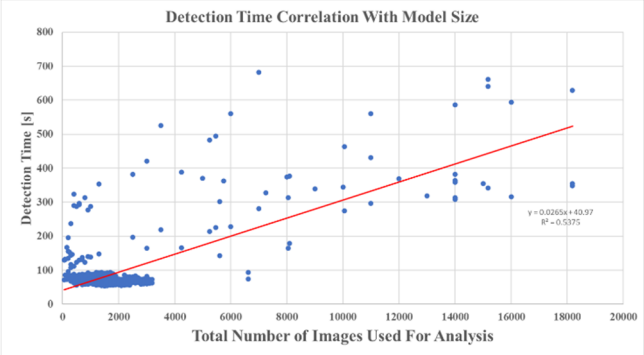

FIGURE 16. 603 Haar Cascade model detection times as a function of the number of images used for the analysis. The linear fit model has been presented (red line).

the third best model record was created on the basis of 2000 images total (1600 positives and 400 negatives), which means that it had no value added from increasing the number of the training dataset beyond that number. At the same time, as shown in Fig. 16, the detection time increases for models with a larger number of images used for training. A linear model fit is a good first approximation of this phenomenon (R^2=0.54).

In order to allow effective insight into the Haar Cascade model performance, the best obtained model (#Positives=5174, #Negatives=13000) has been tested in real world application – drone hoovering over a concert area [62]. The footage depicts a deployment set – none of the frames were used in neither train nor test set. The results show that Haar Cascade is effective in detecting medium size drones on a range of sky backgrounds (clouded, dark, etc.), as shown in Fig. 17.

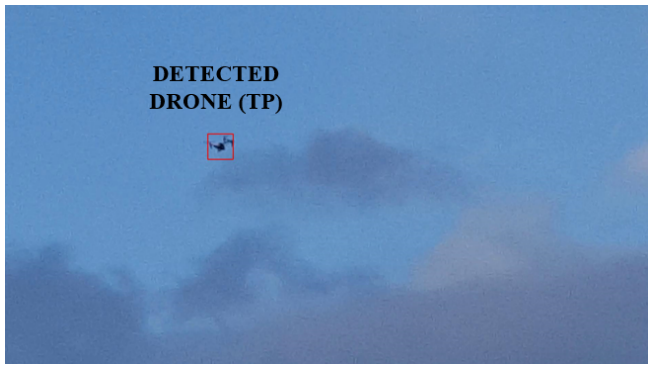

FIGURE 17. OpenCV based deployment of trained Haar Cascade on the real world footage of a drone on the clouded, dark sky background.

The same Haar Cascade fails to detect small-size drones or a drone with an urban background as shown in Fig. 18.

FIGURE 18. Haar Cascade model detection – not only did the model fail to detect the drone (False Negative), but it also detected five nonexistent drones (False Positives).

At the same time, Haar Cascade was able to detect a small percentage of small drones, as shown in Fig. 19. As shown in the Figure, while Haar Cascade was, in fact, able to detect even very small-sized drones (measured as a percentage of overall image), this came at a cost of higher false discovery rate.

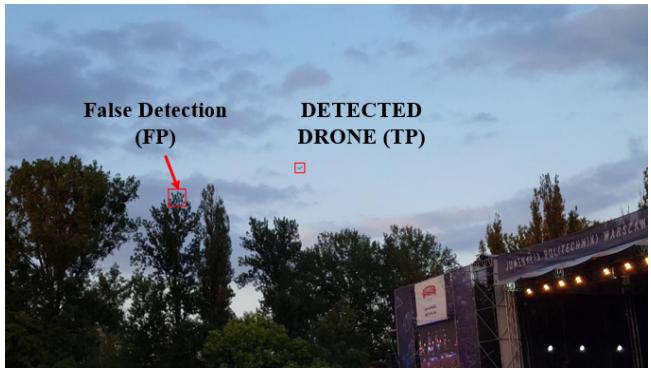

FIGURE 19. Best Haar Cascade model detection with a true positive detection on a small-sized (measured as a percentage of overall image) drone at an incurred cost of false detection.

C. Convolution Neural Network based object detection results

This section presents an analysis of the drone detection model created with Artificial Neural Networks, specifically Convolution Neural Networks. For deep learning implementation, a MobileNet v1 model, pretrained on the COCO dataset, was retrained on the entire training dataset of 51446 images. All model parameters remained default, which included weighted sigmoid classification loss, weighted smooth localization loss and fixed image shape resizer of 300x300. Due to GPU memory size limitation, a batch of 10 images was used. The analysis was run for 1,000,000 iterations (approximately 195 epochs) as a preliminary check of the dataset capacity to generalize the testing dataset. With approximately 0.7 sec/step, the entire analysis took 194 hours or more than 8 days of constant training (although the entire process was paused at multiple instances). Each training model was saved approximately every 10 minutes of training runtime resulting in 819 models, which were then frozen and evaluated separately on the testing dataset. All performance metrics assume the detection confidence threshold of 0.50 and the IoU threshold of 0.5. The results obtained on the basis of the testing dataset are presented in Tab. 2.

TABLE 2.
CONFUSION MATRIX BASED RESULTS FOR THE BEST 10 MODELS TRAINED ON THE 51446 IMAGES TRAINING DATASET RANKED ACCORDING TO MODEL ACCURACY

Iteration Number	True Positives [#]	False Positives [#]	True Negatives [#]	False Negatives [#]	Accuracy [%]	F_1 Score [%]
867503	1283	168	2674	1580	69.4	59.5
892127	1328	348	2658	1535	67.9	58.5
857727	1416	423	2479	1447	67.6	60.2
687375	1148	137	2697	1715	67.5	55.4
819539	1271	306	2654	1592	67.4	57.3
991379	1318	377	2607	1545	67.1	57.8
788738	1291	375	2651	1572	66.9	57.0
401092	1150	216	2686	1713	66.5	54.4
876001	1108	183	2622	1755	65.8	53.3
808454	1159	258	2616	1704	65.8	54.2

As shown in Tab. 2, the true positive detection rate is significantly larger than that of Haar Cascade, however, even the best ANN detection model has not detected more than 54% of true positives in the database (total of 2625). The maximum model accuracy does not exceed 70% and the F_1 score does not exceed 60.2%. Still, ANN results are significantly better than those of Haar Cascades (mind that the Haar Cascades accuracy does not exceed 55% and the F_1 score does not exceed 32%).

In contrast to the Haar Cascade detection model, deep learning provides detection confidences along the detection bounding box itself, which allows more robust results evaluation using metrics like ROC and AUC. These supplementary results are presented in Tab. 3.

TABLE 3.
SUPPLEMENTARY PERFORMANCE METRICS FOR THE BEST 10 MODELS TRAINED ON THE 51446 IMAGES TRAINING DATASET RANKED ACCORDING TO MODEL ACCURACY

Iteration Number	Precision [%]	Recall [%]	Specificity [%]	MCC [%]	AUC	mAP
867503	88.4	44.8	94.1	44.7	0.550	0.533
892127	79.2	46.4	88.4	38.5	0.562	0.521
857727	77.0	49.5	85.4	37.4	0.568	0.547
687375	89.3	40.1	95.2	42.2	0.563	0.547
819539	80.6	44.4	89.7	38.3	0.512	0.485
991379	77.8	46.0	87.4	36.8	0.534	0.505
788738	77.5	45.1	87.6	36.3	0.526	0.489
401092	84.2	40.2	92.6	38.5	0.524	0.495
876001	85.8	38.7	93.5	38.4	0.518	0.500
808454	81.8	40.5	91.0	44.7	0.492	0.469

While the results presented in Tab. 3 allow generation of a short overall summary of the training process, they provide little insight into the sensitivity of the models to confidence level change. This can be only presented on the Precision-Recall and ROC curves, as shown in Fig. 20. These representations offer a detailed insight into model operation with a variable confidence threshold, so ultimately project stakeholders may select the most appropriate model.

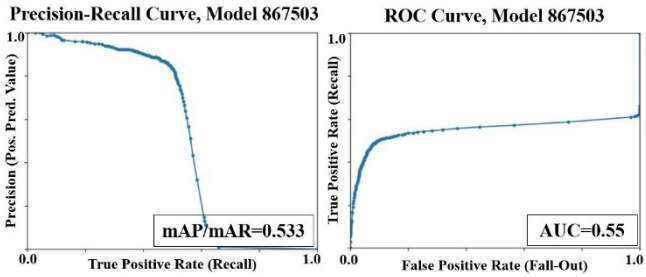

FIGURE 20. ROC curve (left) and Precision-Recall curve (right) as obtained for the best performing ANN (CNN).

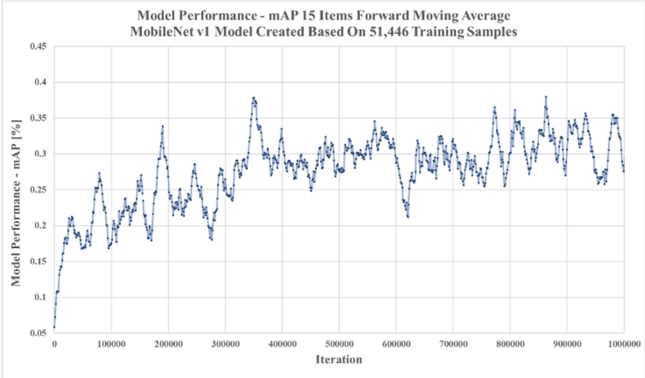

FIGURE 21. Smoothed mAP results for 819 trained models and full 51446 image dataset trained using the pretrained COCO MobileNet v1 model. As shown in the figure, model performance exhibits a globally positive trend. Model performance stabilizes over 750,000 iterations, which is an indicator of the model reaching its detection potential.

As mentioned earlier, the average precision for the Precision-Recall curve, or mAP, can be calculated separately for each model. This way, effective representation of the overall model

performance over different confidence thresholds can be obtained. Unfortunately, reducing 1 million iterations and 819 resultant models into a single table does not give a visualization of the network "learning" process. This is why a representative metric of mAP needs to be visualized over the entire training spectrum (throughout all the iterations performed). The resulting training mAP metric results, smoothed using a moving average of 15 instances, are presented in Fig. 21.

As shown in Fig. 21, the average precision rises from 0 as expected, but then the training process is relatively unstable, peaking at approximately 350,000 iteration with an unexpected local minimum at iteration 625,000. Since the batch size was relatively small, the training process was relatively volatile. Hence, the need for moving average based smoothing (15 instances) emerged for postprocessing. The general trend presented in Fig. 21 is positive, which means that training the same model longer is likely to result in better overall model performance. At the same time, the model seems to be reaching a local plateau at over 750,000 iterations.

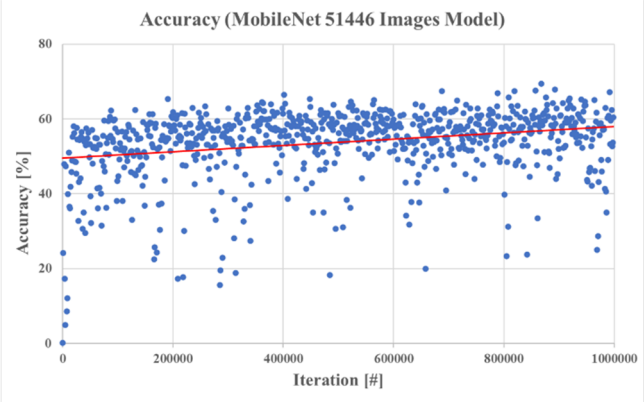

FIGURE 22. Non smoothed accuracy (@0.5 conf. & IoU) results for 819 trained COCO MobileNet v1 models showing steadily increasing performance gains for the models.

Naturally, mAP is only one of the multiple object detection model evaluation metrics. In order to allow intuitive model performance presentation, model accuracy is typically used. The accuracy of the frozen ANN models is presented in Fig. 22 (no smoothing). For the first ~50,000 iterations, the model quickly optimizes for the new problem. Then, the general trend confirms the conclusions from mAP postprocessing that further model optimization would yield better model performance as the overall model accuracy is trending up. However, in contrast to mAP, accuracy shows a steady increase above the 750,000 mark, which is an indicator that further performance gains can be achieved.

The model starts learning at iteration 0 (accuracy equals 0), stabilizes at approximately 50% and then slowly trends upwards. At million iterations the average accuracy checks approximately 58% with large variation and global maxima.

As mentioned earlier, while the accuracy of the model provides valuable information for some users unfamiliar with the confusion matrix, the F_1 score, a harmonic mean between Precision and Recall, is a much better model performance indicator. As shown in Fig. 23, the model is also trending upwards, but in a much more stable fashion than the accuracy plot did and with fewer outliers.

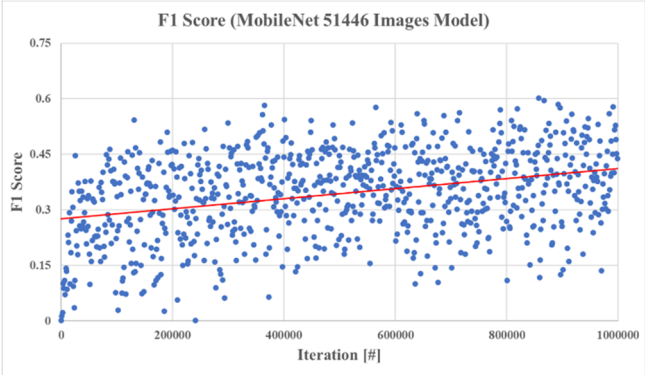

FIGURE 23. Non smoothed model F1 score (@0.5 conf. & IoU) as a function of iteration number showing steady performance gains.

TABLE 4.
ADDITIONAL INFORMATION ON THE BEST 10 MODELS TRAINED ON THE 51446 IMAGES TRAINING DATASET RANKED ACCORDING TO MODEL ACCURACY

Iteration Number	Process Time [s]	Detection Time [s]	Maximum Accuracy [%]	Confidence at Maximum Accuracy	Maximum F_1 Score	Confidence at Maximum F_1 Score
867503	86.4	64.1	70.3	0.33	62.7	0.24
892127	86.7	61.7	68.4	0.40	60.7	0.17
857727	87.5	64.4	67.8	0.41	61.9	0.28
687375	87.8	63.5	70.0	0.15	62.1	0.05
819539	88.4	63.6	67.5	0.48	58.2	0.33
991379	84.6	61.4	67.3	0.57	58.5	0.44
788738	87.3	62.8	67.1	0.48	58.5	0.30
401092	84.4	61.7	67.1	0.29	58.1	0.13
876001	90.5	66.4	66.1	0.39	56.5	0.12
808454	91.4	65.0	65.8	0.50	55.4	0.28

Typically, the 50% confidence level threshold is used to determine if a particular detection is treated by the model as true or not. Naturally, this value may be adjusted so as to maximize accuracy, F_1 score or any other metric. Tab. 4 shows the maximum obtainable accuracy and F_1 score and corresponding confidences needed to acquire such results. As shown in Tab. 4, the confidence threshold of 0.50 is rarely the optimum. Additionally, Tab. 4 presents the detection time metric for the testing dataset. As shown, detection time ranges from 61 to 66 seconds with a low standard deviation. The consistency in the detection time is a direct result of the same neural network architecture, which results in the same number of parameters and computations needed for each model. Note that the worst ANN-based detection time is less than the best time for the Haar Cascade model (71 seconds), which means that—for the same hardware— the GPU-based accelerated ANN approach is faster than ML based Haar Cascades utilizing CPU only.

As a summary, all overall model performance indicators are presented in Tab. 5. The data presented in the table show model accuracy, F_1 score, AuC and mAP metrics for the entire 819 models, aggregated for average and maximum to better generalize the training process.

TABLE 5.
AVERAGE/MAXIMUM AGGREGATION OF KEY PERFORMANCE INDICATORS OF 1 MILLION ITERATION WORTH OR 819 MODELS TRAINED ON THE 51446 IMAGES TRAINING DATASET

Evaluation Metric	Score
Iteration 0-1Mil Avg ACC	61.72
Iteration 0-1Mil Avg F_1	52.42
Iteration 0-1Mil Avg AuC	0.63
Iteration 0-1Mil Avg mAP	0.57
Iteration 0-1Mil MAX ACC	74.77
Iteration 0-1Mil MAX F_1	70.00
Iteration 0-1Mil MAX AuC	0.77
Iteration 0-1Mil MAX mAP	0.74

In order to allow effective insight into ANN model performance, the best obtained model (@Iteration=867503) was tested in real world application [62], identical to the one Haar Cascade was tested upon. The results show that ANN is effective in detecting medium size drones on a range of sky backgrounds (clouded, dark, etc.), as shown in Fig. 24.

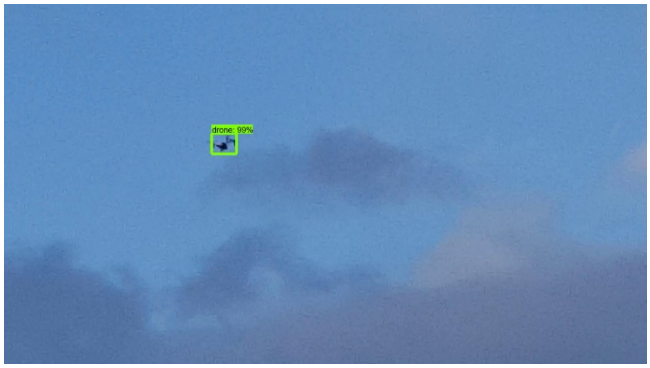

FIGURE 24. ANN detection model (as obtained for iteration=867503) deployed over a drone footage. The model was shown to be very effective in detecting model presented in the figure but often failed to detect drones of a small size (calculated as a percentage of overall image).

Another advantage of ANN model over Haar Cascade is the detection confidence value. Its acceptance threshold can be modified depending on the environmental condition, background type (sky, urban, nature) to maximize overall detection performance.

Furthermore, detection confidence allows performance increase through a two-stage detection process. Firstly, the entire input image would be processed with low detection confidence to provide detection "drone proposals" (similarly to tagging pipeline). The "drone proposals" could be then extracted to prevent resizing distortions and to maintain high image definition and then yet again processed through the ANN. This results in overall higher detection performance and cannot be used for Haar Cascades as their output does not include detection confidence.

Overall, ANN showed better performance than Haar Cascades, mainly due to a significantly lower false detection rate. In fact, ANN failed to detect some of the drones detected by Haar Cascades, as shown in Fig. 25, but rarely suffered from false detection.

FIGURE 25. Best performing ANN detection model (as obtained for iteration=867503) deployed over a drone footage with a correct detection over a small size drone (calculated as a percentage of overall image) with no false detection present.

VI. CONCLUSIONS

Currently available datasets are insufficient to create drone object detection algorithms in versatile environments. Therefore, a novel dataset generation pipeline has been presented with additional methods for labelling process automation and the productivity boost of 50% (dataset tagging time reduction from 45 to 22.5 seconds on average). This novel approach can also be easily scaled and used for any kind of object detection based labelling, which can help to reduce one of the largest challenges of ANN-based computer vision research – lack of large domain-specific datasets and hurdles connected to obtaining them. It also allows the capitalization of model deployment at the early stages of the machine learning model development pipeline.

This approach resulted in a novel dataset of 51446 image training dataset (1 negative, 51445 positives) and a 5375 image testing dataset (2750 negatives, 2625 positives) available for drone detection research purposes.

Haar Cascade is an algorithm created almost two decades ago, although it is still used today in multiple appliances like cell phones or cameras. This is due to the fact that it is easily supported by most operating systems and can be run directly on most CPUs with little software required. Typically, Haar Cascades are used for face detection problems, although they can be trained for any new problem using a predefined dataset. However, since training based on a large number of images can take up to few months, it is recommended to try different combinations of images to maximize test dataset based performance with a smaller number of images. For a drone detection application, the Haar Cascades training algorithm optimizes weak intermediate classifiers to quickly disregard most of the frames, which in practice results in only a few

detections for the entire dataset, including few true positives. However, as 51% of testing dataset images are negatives, the end result is the relatively high accuracy of up to 55% and low F_1 score (not exceeding 32%), for 603 models analyzed. The more images were used for the training, the longer the detection time is. Generally, since Haar Cascade can provide a relatively good result with just a few hundred samples, it can be a starting point for rapid software prototyping and labelling automation. Since Haar Cascade does not provide detection confidences, some of the object detection metrics, like ROC or Precision-Recall curve, cannot be obtained.

While the accuracy of the Haar Cascade model is limited, its low computational cost and ease of deployment (just one line of code required) allow its uses in conjunction with other image processing methods.

Artificial Neural Networks with many hidden layers, also called Deep Learning, have been known for decades, but 2012 ImageNet successes have shown a resurgence in their use with multiple successful industrial applications. For object detection purposes pretrained convolutions based (CNN) models can be used as the starting point to be fine-tuned on a drone dataset in a process called transfer learning. Since model parameters optimized on large inputs take a lot of GPU memory, the mini-batch used at training time has to be relatively small, which results in volatile training results, but with a positive overall trend for both accuracy and F_1 score. For transfer learning purposes, the MobileNet v1 model pretrained on the COCO dataset was run for 1 million iterations (approximately 195 epochs) with a fixed image resizer of 300x300, generating 819 models. The obtained accuracy for 819 obtained models does not exceed 70% and the F_1 score does not exceed 60.2%. CNN based output, in contrast to Haar Cascades, additionally provides detection confidence (50% by default). Assuming that any confidence threshold is available, the maximum obtained accuracy checks 70.3% (at confidence equal to 0.33) and the maximum obtained F_1 score checks 62.7% (at confidence equal to 0.24). Since all models represent the same number of parameters, but iteratively improved over each epoch, the resultant computational requirement and subsequent detection time are relatively constant and superior to Haar Cascade based models. The biggest disadvantage of using CNNs over Haar Cascade is the GPU acceleration required.

In general, ANN based solutions achieve better performance than Haar Cascades but require more computational power, dedicated GPU, and processing time. Its deployment also is more challenging than that of the Haar Cascade alternative. Haar Cascades return more drone detections than ANNs, but with a significantly larger amount of false positives.

VII. FUTURE WORK

Potential improvements and further research directions are discussed in this section. Future work will be entirely devoted to ANN based approaches with the Haar Cascade based model used only as a reference/benchmark.

The top priority will be to run the current model configuration for at least 2.5 million iterations to determine the inflation point of diminishing return, where longer training does not substantially improve model performance. Furthermore, for comparative study purposes only, the training set will be used as a testing set to determine if the current model is able to properly generalize the given input dataset of 51446 images.

Currently, the entire 51446 training image dataset is used for training. While this number seems to be large enough to generalize the drone detection task with sufficient model performance, the exact number of images to be used seems large for a single class task. As such, multiple different training dataset configurations will be run to establish a point in which adding more images to the dataset does not provide sufficient improvement based on the testing dataset performance (the testing set will remain the same for comparison). Every model will be run for 1 million iterations minimum for direct comparison with a model set established during research used for this paper.

REFERENCES

[1] "Gatwick Airport drone incident," [Online]. Available: https://en.wikipedia.org/wiki/Gatwick_Airport_drone_incident. [Accessed 18 April 2020].

[2] "2019 Abqaiq–Khurais attack," [Online]. Available: https://en.wikipedia.org/wiki/2019_Abqaiq%E2%80%93Khurais_attack. [Accessed 18 May 2020].

[3] P. Gregg, "Impact tests prove large aircraft won't always win in collision with small drones," University of Dayton Research Institute, 2018.

[4] V. Tyurin, O. Martyniuk, V. Mirnenko, P. Open'ko and I. Korenivska, "General Approach to Counter Unmanned Aerial Vehicles," in *IEEE 5th International Conference Actual Problems of Unmanned Aerial Vehicles Developments*, Kiev, Ukraine, 2019.

[5] A. Rozantsev, V. Lepetit and P. Fua, "Detecting flying objects using a single moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 5, p. 879–892, 2017.

[6] C. Aker and S. Kalkan, "Using deep networks for drone detection," in *14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Lecce, 2017.

[7] R. LaLonde, Z. Dong and S. Mubarak , "Clusternet: Detecting small objects in large scenes by exploiting spatio-temporal information," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[8] R. Yoshihashi, T. Trinh, R. Kawakami, S. You, M. Iida and T. Naemura, "Differentiating Objects by Motion: Joint Detection and Tracking of Small Flying Objects," arXiv:1709.04666v3.

[9] R. L. Sturdivant and E. K. P. Chong, "Systems engineering baseline concept of a multispectral drone detection solution for airports," *IEEE Access,* vol. 5, pp. 7123-7138, 2017.

[10] Y. LeCun and C. Cortes, "The MNIST database of handwritten digits," 1998.

[11] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, 2009.

[12] A. Torralba, R. Fergus and W. T. Freeman, "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 30, no. 11, pp. 1958-1970, 2008.

[13] J. Deng, W. Dong, R. Socher, L. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE*

Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009.

[14] "ImageNet," [Online]. Available: http://image-net.org/about-stats. [Accessed 23 March 2020].

[15] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097-1105.

[16] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman , "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision volume,* vol. 111, p. 98–136, 2015.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and L. C. Zitnick, "Microsoft COCO: Common Objects in Context," in *13th European Conference on Computer Vision – ECCV 2014*, Zurich, 2014.

[18] H. Yun, C. Zhang, C. Hou and Z. Liu, "n Adaptive Approach for Ice Detection in Wind Turbine With Inductive Transfer Learning," *IEEE Access,* vol. 7, pp. 122205 - 122213, 03 July 2019.

[19] X. Liu, Z. Liu, G. Wang, Z. Cai and H. Zhan, "Transfer Learning references: Ensemble Transfer Learning Algorithm," *IEEE Acces,* vol. 6, pp. 2389 - 2396, 2017.

[20] "GitHub — Tensorflow detection model zoo," 2017. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md. [Accessed 6 May 2020].

[21] J. Li, J. Murray, D. Ismaili, K. Schindler and C. Albl, "Reconstruction of 3D flight trajectories from ad-hoc camera networks," arXiv:2003.04784v2, 2020.

[22] "GitHub — CenekAlbl / drone-tracking-datasets," [Online]. Available: https://github.com/CenekAlbl/drone-tracking-datasets. [Accessed 25 August 2020].

[23] "GitHub — ZhaoJ9014 / Anti-UAV," [Online]. Available: https://github.com/ZhaoJ9014/Anti-UAV. [Accessed 25 August 2020].

[24] "test-dev: 100 videos for testing your tracker," [Online]. Available: https://anti-uav.github.io/dataset/. [Accessed 25 August 2020].

[25] A. Koksal, K. G. Ince and A. A. Alatan, "Effect of Annotation Errors on Drone Detection with YOLOv3," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, 2020.

[26] "GitHub — Maciullo / DroneDetectionDataset," [Online]. Available: https://github.com/Maciullo/DroneDetectionDataset. [Accessed 26 August 2020].

[27] M. Pawełczyk and P. Bibik, "Usage of modern engineering software in the design of unmanned rotorcraft," in *Materiały IX Krajowego Forum Wiropłatowego. Instytut Lotnictwa*, Warsaw, 2013.

[28] "COCO Common Objects in Context," [Online]. Available: http://cocodataset.org/#detection-eval. [Accessed 28 April 2020].

[29] Y. Jiang and J. Ma, "Combination Features and Models for Human Detection," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* p. 240–248, 2015.

[30] N. Dalal, B. Triggs and C. Schmid, "Human detection using oriented histograms of flow and appearance," *In Proc. ECCV,* p. 428–441, 2006.

[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60(2), p. 91–110, 2004.

[32] Z. Li and L. Itti, "Saliency and gist features for target detection in satellite images," *IEEE Trans. Image Process,* vol. 20, pp. 2017-2029, 2011.

[33] X. Wang, T. X. Han and S. Yan, "An hog-lbp human detector with partial occlusion handling," in *IEEE 12th International Conference on Computer Vision*, Kyoto, 2009.

[34] J. Shotton, M. Johnson and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, 2008.

[35] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, 2001.

[36] C. P. Papageorgiou, M. Oren and T. Poggio, "A general framework for object detection," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Bombay, India, 1998.

[37] R. M. Haralick, K. Shanmugam and I. Dinstein, "Textural Features for Image Classificatio," Vols. SMC-3, no. 6, pp. 610-621, 1973.

[38] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme," *Mathematische Annalen volume,* vol. 71, pp. 38-53, 1911.

[39] "GitHub — opencv/data/haarcascades/," [Online]. Available: https://github.com/opencv/opencv/tree/master/data/haarcascades. [Accessed 27 April 2020].

[40] A. Bigdeli, C. Sim, M. Biglari-Abhari and B. C. Lovell, "Face Detection on Embedded Systems," in *Third International Conference, ICESS 2007*, Daegu, Korea, 2007.

[41] K.-Y. Park and S.-Y. Hwang, "An improved Haar-like feature for efficient object detection," *Pattern Recognition Letters,* vol. 42, pp. 148-153, 2014.

[42] M. Pawełczyk, "Data Augmentation for Haar Cascade Based Automobile Detection," *The Archives of Automotive Engineering – Archiwum Motoryzacji,* vol. 82(4), p. 117–129, 2018.

[43] H. Gong, L. Chen, C. Li, J. Zeng, X. Tao and Y. Wang, "Online Tracking and Relocation Based on a New Rotation-Invariant Haar-Like Statistical Descriptor in Endoscopic Examination," *IEEE Access,* vol. 8, pp. 101867 - 101883, 2020.

[44] Y. Yu, H. Ai, X. He, S. Yu, X. Zhong and M. Lu, "Ship Detection in Optical Satellite Images Using Haar-like Features and Periphery-Cropped Neural Networks," *IEEE Access,* vol. 6, pp. 71122-71131, 2018.

[45] A. Luo, F. An, X. Zhang and H. J. Mattausch, "A Hardware-Efficient Recognition Accelerator Using Haar-Like Feature and SVM Classifier," *IEEE Access,* vol. 7, pp. 14472-14487, 2019.

[46] E. Liu, Y. Chu and L. Zheng, "Object Tracking Based on Compressive Features and Extreme Learning Machine," *IEEE Access,* vol. 7, pp. 45994-46003, 2019.

[47] "OpenCV — Face Detection using Haar Cascades," [Online]. Available: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html. [Accessed 28 April 2020].

[48] G. Bradski and A. Kaehler, Learning OpenCV, Sebastopol, CA: O'Reilly Media, Inc., 2008.

[49] "CS230 Deep Learning," Stanford, [Online]. Available: https://cs230.stanford.edu/. [Accessed 18 May 2020].

[50] D. Machalica and M. Matyjewski, "CAD models clustering with machine learning," *Archive of Mechanical Engineering,* vol. 66(2), pp. 133-152, 2019.

[51] M. Pawełczyk, S. Fulara, M. Sepe, A. De Luca and M. Badora, "Industrial Gas Turbine Operating Parameters Monitoring and Data-Driven Prediction," *Eksploatacja i Niezawodnosc – Maintenance and Reliability,* vol. 22(3), p. 391–399, 2020.

[52] Ł. Woliński, "Comparison of the adaptive and neural network control for LWR 4+ manipulators: simulation study," *Archive of Mechanical Engineering,* vol. 67(1), pp. 111-121, 2020.

[53] J. Ker, L. Wang, J. Rao and T. Lim, "Deep Learning Applications in Medical Image Analysis," *IEEE Access,* vol. 6, pp. 9375-9389, 2017.

[54] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho and S. W. Baik,

"Convolutional Neural Networks Based Fire Detection in Surveillance Videos," *Khan Muhammad ; Jamil Ahmad ; Irfan Mehmood ; Seungmin Rho ; Sung Wook Baik,* vol. 6, pp. 18174-18183, 2018.

[55] R. Vinayakumar, M. Alazab, K. P. Soman and P. Poornachandran, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access,* vol. 7, pp. 41525-41550, 2019.

[56] B. Zhao, H. Lu, S. Chen, J. Liu and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics,* vol. 1, pp. 162-169, 2017.

[57] S. Ye, Z. Zhang, X. Song, Y. Chen and C. Huan, "A flow feature detection method for modeling pressure distribution around a cylinder in non-uniform flows by using a convolutional neural network," vol. 10, no. 4459, 2020.

[58] Z. Geng and Y. Wang, "Automated design of a convolutional neural network with multi-scale filters for cost-efficient seismic data classification," *Nature Communications,* vol. 11, no. 3311, 2020.

[59] "Confusion matrix," [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix. [Accessed 28 April 2020].

[60] K. M. Ting, "Confusion Matrix," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA, Springer Science+Business Media, 2010, pp. 209-209.

[61] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics,* vol. 21, no. 6, 2020.

[62] "YouTube — Drone Detection - Haar Cascade vs Convolutional Neural Networks," [Online]. Available: https://www.youtube.com/watch?v=o6mxjR6GdA4. [Accessed 26 August 2020].

MACIEJ Ł. PAWEŁCZYK received the B.S. and M.S degree in mechanical engineering from Warsaw University of Technology, Warsaw, Poland, in 2012 and 2014 respectively. He also completed postgraduate studies in Warsaw School of Economics in statistical analysis and datamining in business. He is currently pursuing the Ph.D. degree in mechanical engineering at Warsaw University of Technology, Warsaw, Poland.

From 2012 to 2014 he worked as automotive design engineer for Rücker Automotive and, since 2014, as aerospace engineer in Warsaw Institute of Aviation/General Electric cooperative in Warsaw, currently in position of Advanced Lead Engineer. His research interest include applications of deep learning in industry for robotic process automation and predictive maintenance.

Mr. Pawełczyk's awards include GE Engineering Recognition Day 2017 award in "Fielded Product Value" category, 2018 GE Engineering Leadership Meeting award for "Lean & Operational Excellence". He is also a recipient of GE Challenge 2018 contest.

MAREK WOJTYRA received the Ph.D. degree in biomechanics in 2000 and the D.Sc. degree in automation and robotics in 2013 from Warsaw University of Technology, Warsaw, Poland.

He is currently an Associate Professor with the Institute of Aeronautics and Applied Mechanics at WUT, Warsaw, Poland; he is the Head of the Division of Theory of Machines and Robots. He published a book and over 80 journal and conference articles. His research interests include multibody systems and robotics with its applications.

Dr. Wojtyra serves as a reviewer for several journals; in 2017 he received from Elsevier a Certificate of Outstanding Contribution in Reviewing. He also received the Andrzej Komor Foundation Award for his Ph.D. theses and several awards for conference presentations.