7-27-2023

MACHINE LEARNING-BASED DRONE AND AERIAL THREAT DETECTION FOR INCREASED TURRET GUNNER SURVIVABILITY

Nikolas Koutsoubis
Rowan University

MACHINE LEARNING-BASED DRONE AND AERIAL THREAT DETECTION FOR INCREASED TURRET GUNNER SURVIVABILITY

by
Nikolas Koutsoubis

A Thesis

Submitted to the
Department of Electrical and Computer Engineering
College of Engineering
In partial fulfillment of the requirement
For the degree of
Master of Science
at
Rowan University
June 26, 2023

Thesis Chair: Nidhal C. Bouaynaya, Ph.D., Professor and Associate Dean for Research, Department of Electrical and Computer Engineering, Rowan University

Committee Members:
Gregory Ditzler, Ph.D., Associate Professor, Department of Electrical and Computer Engineering, Rowan University
Thomas Kiel, Senior Design Engineering, US Army DEVCOM Armaments, Picatinny Arsenal

Acknowledgements

First of all, I would like to thank my advisor Dr. Nidhal Bouaynaya for allowing me the opportunity to work in the Rowan Machine Learning & AI and Virtual Reality Center (MAVRC). She was a constant source of encouragement and direction throughout my graduate studies. I want to thank my committee members Dr. Gregory Ditzler and Thomas Kiel. Dr. Ditzler provided invaluable guidance throughout my research and Mr. Kiel provided fantastic direction for the project as a whole. Furthermore, I would like to thank the members of the MAVRC team for their contributions to this work as well as the lessons I was able to learn observing their expertise in their respective fields. A special thanks to AI team member Kyle Naddeo for the assistance and guidance he provided in conducting this research, and the MAVRC VR team for generating the DyViR dataset. Finally, I thank my family for their continued motivation and support.

Abstract

Nikolas Koutsoubis
MACHINE LEARNING-BASED DRONE AND AERIAL THREAT DETECTION FOR INCREASED TURRET GUNNER SURVIVABILITY
2022-2023
Nidhal C. Bouaynaya, Ph.D.
Master of Science

The introduction of aerial drones on the modern battlefield has transformed combat operations, posing a significant threat to ground-based military operations. Detecting drones in safety scenarios is crucial. However, modern machine learning (ML)-based object detectors struggle to detect small objects like drones.

This thesis presents three main contributions: (a) data and algorithmic modifications to improve small object detection in YOLO to aid in drone detection, (b) the development of a benchmark drone detection dataset called DyViR, and (c) the implementation of explainable artificial intelligence (XAI) to ensure transparent and trustworthy decision-making.

To boost the performance of small object detection, we introduce the Normalized Wasserstein distance (NWD) into the loss function of our ML model. By incorporating this distance metric, we can effectively handle small object detection by reducing the penalty assigned to small objects, and appropriately balancing the significance of different object sizes. This allows the model to prioritize the accurate detection of small objects, enhancing overall performance.

To evaluate our algorithm, we developed and tested the DyViR dataset specifically designed for drone detection research. This synthetic dataset provides a benchmark for assessing drone detection performance.

In combat settings, the trustworthiness of ML systems is paramount as their decisions impact user survival. Thus, we implemented explainable AI systems (XAI), specifically Grad-CAM and Eigen-CAM, These techniques provide explanations for the models' decisions, increasing trust in the system for developers and users.

Table of Contents

Table of Contents (Continued)

List of Figures

Figure Page

List of Tables

Table Page

Chapter 1

Introduction

The work in this thesis was conducted as part of the Turret Gunner Survivability Simulation Environment (TGSSE) project by the Machine Learning & Artificial Intelligence and Virtual Reality Center. The objective of this project is to create virtual and field systems to improve the survivability and lethality of turret gunners in a combat environment. This thesis focuses on the artificial intelligence aspect of this project. The primary directive was to develop a system that could be deployed on a combat vehicle to detect aerial drones and other potential aerial threats. In recent years aerial drones have proven to be a significant threat to both combat vehicles and infantry alike. Drones are capable of forward surveillance and targeted attacks [1]. Given their small size and low cost, they have become a powerful tool for terrorist organizations [2]. Creating countermeasures for this emerging threat is critical to the safety of those in a combat environment [3]. This thesis will focus on the creation of a machine learning-based system designed to detect drones. Three main areas of work were conducted: the creation of a deep neural network (DNN) with data and algorithmic level optimizations designed to specifically detect drones and other aerial threats. The next area of work is the creation of a benchmark drone detection dataset. The final area of research was to implement a foundation of explainable artificial intelligence (XAI) in this system to build trust in the decisions made by the network.

1.1 Related Work

1.1.1 Drone Detection Methods

The primary goal of this research was to create a reliable drone detection system. Several modalities for detecting drones have been implemented. One option would be to

utilize acoustic sensors to pick up on the sound created by the rotors of the drone as it flies [4]. Acoustic detection allows for detections to be made outside the line of sight of the sensor. Deep machine learning has transformed the field of computer vision in many different applications [5, 6, 7, 8, 9]. Computer vision-based approaches that take video inputs and use deep neural networks (DNN) to identify drones provide another avenue for detecting drones [10] has also been implemented. These systems take advantage of already established object detectors such as R-CNN [11] and YOLO [12]. Radar-based systems have been used to detect aerial objects for decades and have also been utilized for drone detection [13]. The final notable modality for drone detection involves analyzing radio frequency signals between the drone and its controller and has been successfully implemented for drone detection [14].

1.1.2 Drone Detection Datasets

The availability of real-world drone datasets is limited in the machine-learning community, however, some notable ones exist. The Drone Detection Dataset [15], was captured at three airports in Sweden and comprises four classes (e.g., airplane, bird, drone, and helicopter). This dataset contains 89,000 RGB images with a resolution of 640×512 pixels and 114,000 thermal images with a resolution of 320×256 for a total of 203,328 annotated images. The greatest sensor-to-target distance in this dataset is 200 meters.

The University of Southern California (USC) dataset consists of 30 one-minute video clips of drones in an indoor and outdoor environment [16]. Kaggle has also hosted a few datasets that incorporate drone images taken from other sources such as from drone to drone and from ground to air.

While real-world drone detection datasets exist, the amount of data is sparse and relatively small. Further, collecting, annotating, and verifying a new real-world drone dataset is time-consuming and expensive. With the maturity of real-time rendering engines dedicated to creating virtual reality (VR) simulations, researchers have started generating

drone data within virtual environments. There are many advantages to virtual data that are challenging to collect with real-world data. Specifically, real-world data is limited to the weather conditions at the time of collection. VR provides the opportunity to generate data in many environments. Further, VR data enables us to easily add new classes (e.g., drone types, etc.) without the need to physically acquire such entities in the real world. Finally, these VR data sets can be generated in a fraction of the time, resources, and capital as compared to real-world datasets.

The SimUAV dataset [17] is a simulated UAV small object detection dataset consisting of 29,568 images with eight different background scenes and four types of drones. This dataset was constructed in the Airsim platform [18] which was designed for simulating autonomous driving and has been extended to simulate drones.

1.1.3 Explainable Artificial Intelligence

Eigen-Cam is a Class Activation Map (CAM)-based Explainable AI method that facilitates the visualization of critical regions in an input image that contribute most significantly to the decision made by a model. This approach was first introduced by Zhou et al. [19]. The authors proposed a method to generate class-specific activation maps by weighting the final feature maps of a convolutional neural network (CNN) with its corresponding class label. The CAM technique leverages the spatial information encoded in the final convolutional layer of a CNN. In 2017, Selvaraju et al. proposed a refinement of the CAM method called Grad-CAM [20]. Grad-CAM uses the gradients of the output class score with respect to the final convolutional layer to compute the weights for the feature map, thereby enhancing the accuracy of the generated maps.

Chapter 2

Boosting Drone Detection Performance with Data and Algorithmic Optimizations

Machine learning-based object detection methods have shown superior performance for such tasks, even those that include aerial object detection [21]. As drone technology advances and becomes more prominent in society, there will need to be systems to identify drones (e.g., if a drone loses communication but needs to be identified). For example, drones and other cutting-edge aerial crafts have revolutionized domains related to remote sensing, express shipping, and delivery, agriculture, and law enforcement [6, 7, 8, 9]. Further, they offer the above applications the ability to perform tasks that can limit risks to human health by identifying objects from a safe distance [22]. Drones are also challenging to detect in aerial images due to their small size and distance from the camera, which makes drone detection from the ground a critical need in many applications [22]. It is important to note that in most object detection datasets, the entity in the image is visible to the human eye. Unfortunately, many objects in the sky are more challenging to identify due to the entity's size. Conventional object detection methods, such as YOLO which rely on intersection over union (IoU) losses [23, 24], struggle to detect these tiny aerial objects due to an over-penalization of small errors in prediction [6]. In addition to aerial objects appearing small from the ground, real-world scenes with multiple aerial objects are prone to many false positives. Hence, detecting drones is difficult in practice due to their small size, limited data for training, far distances, and potentially large swarms of drones.

These challenges related to drone detection correspond to opportunities in the ML community. First, small aerial objects are challenging for humans to detect with high accuracy. There are several factors why the performance of small objects is low. Small objects are challenging to identify from a machine-learning perspective because the human cost of acquiring the data are limited and can be extremely noisy (e.g., large variations in

the bounding boxes). Unfortunately, the nature of collecting enough real-world data for small aerial object detection is a serious challenge in this domain. Therefore, we see that knowledge from other domains can be transferred for small object detection. Second, the intersection over union (IoU) loss function has shown to be quite good for object detection when the object in the image is relatively obvious to see; however, detection methods based on IoU struggle to detect small objects due to an over-penalization of small prediction errors. Therefore, we feel there is a significant opportunity to revisit the formulation of the IoU to make the object detection models robust on small objects. Third, YOLO was transformative to the field of object detection because of the "*only looking once*" aspect of the approach which allowed YOLO to have superior evaluation times compared to the state-of-the-art. We see an opportunity to borrow from the mantra of YOLO's evaluation by exploring the effects that increased label density per image can have on performance. The advantage of the single pass over an image is that it allows us to train on multiple objects simultaneously and perform a single update. Further, we explore this aspect of training with multiple objects using "high-density" images where many objects are available to train in a single image.

In this paper, we introduce a Virtual Reality (VR) data generator that produces aerial objects in an environment that enables the user to evaluate multiple scenarios such as distant and multiple objects in a scene, types of objects and their sizes, etc. Further, data can be generated from different biomes (e.g., deserts, mountains, etc.), weather conditions, etc. These data were shown to improve the performance of aerial object detection, especially coupled with a small amount of real-world data. We also introduce a new cost function for YOLO object detectors that use the normalized Wasserstein distance to learn from smaller objects more reliably than existing approaches based on the IoU loss. Our approach is evaluated on a real-world drone dataset and we demonstrate the value of synthetic data to boost the performance of a machine learning model in data-scarce environments, and we show that the normalized Wasserstein distance provides better performance than the IoU

loss for small object tasks, such as aerial object detection.

2.1 Methods

In this section, we introduce our proposed approach that uses data and algorithmic tools to improve the performance of small aerial objects. Further, we provide details about our extensions to the state-of-the-art and highlight the proposed novelties.

2.1.1 Virtual Reality Data Generation

A VR simulation software environment was developed to generate synthetic digital datasets due to the scarcity of labeled drone data. Such data have broader applicability to the research community for tasks beyond object detection, when data are scarce and challenging to collect. Creating synthetic datasets with VR offers several advantages over traditional real-world pictures and data abstractions. Firstly, in a digital simulation, the visuals can be altered quickly and independently of outside conditions, such as weather, time of day, etc. Additionally, the simulation's conditions can be easily changed to produce data for new types of drones, flight patterns, and landscapes. Thus, rather than needing to purchase multiple physical drones or have them further from the camera for different object sizes, these can be realized in the 3D simulation for no additional cost once the initial assets are created. Finally, the VR simulation contains tools that allow the bounding box data of the drones to be automatically recorded, eliminating the need for manual labeling by a human.

The VR simulation in this work was created in the Unity Game Engine (Unity), a software that allows custom visual and audio assets programmed in a 3D environment. Over time different events can occur such as environmental changes, camera orientations, and drone speeds. These events can be outlined in a timeline to produce training data meeting specified conditions. Additionally, drones or any aircraft can be programmed to fly either along a pre-determined or pseudo-randomized path. As this is a real-time

application, the position, and orientation can be updated every frame. For pre-determined flight paths, several control points can be placed in the 3D environment that the drone will bounce between. For pseudo-randomized flight paths, the drone will update its position and orientation over time using Perlin noise, creating a smooth flight pattern. This behavior is considered pseudo-random because the user can set a "seed" value. The seed is fed into the Perlin noise algorithm, making it possible to replicate the same flight patterns.

Unity has a pipeline for importing 3D assets created by visualization artists, allowing high-quality renderings of image textures and 3D models. This makes the art style of photorealism possible, where the visuals produced from the simulation resemble real-life photography. While possible with real-time technology, it is no easy feat. Achieving photorealism requires extensive work from 3D modelers, texture artists, and technical artists to produce accurate and optimized 3D assets. Additionally, lighting, post-processing, and shader properties need to be configured to match physical phenomena such as reflection, refraction, and diffusion of light. This can be seen in the VR simulation in both the environment and drones. The environment utilizes real-time lighting that produces bounce-lighting shadows on the terrain as the light sources move. The drones use different material types based on the metallicness and smoothness of their components, such as metal, plastic, and composite. These aspects help to upgrade the visual standards to resemble real-life imagery. Attaining photorealism with synthetic datasets is a high priority, as training on realistic imagery will produce better results for object detection on real-world inference data.

2.1.2 Detection of Multiple Small Objects

2.1.2.1 Yolov7-tiny. To detect aerial objects, such as drones, in real-time, the model used must be capable of fast inference times while still maintaining good accuracy. YOLOv7-tiny is the most recent (at the time of submission) implementation of YOLO that achieves state-of-the-art performance [25]. The tiny version was selected for plans

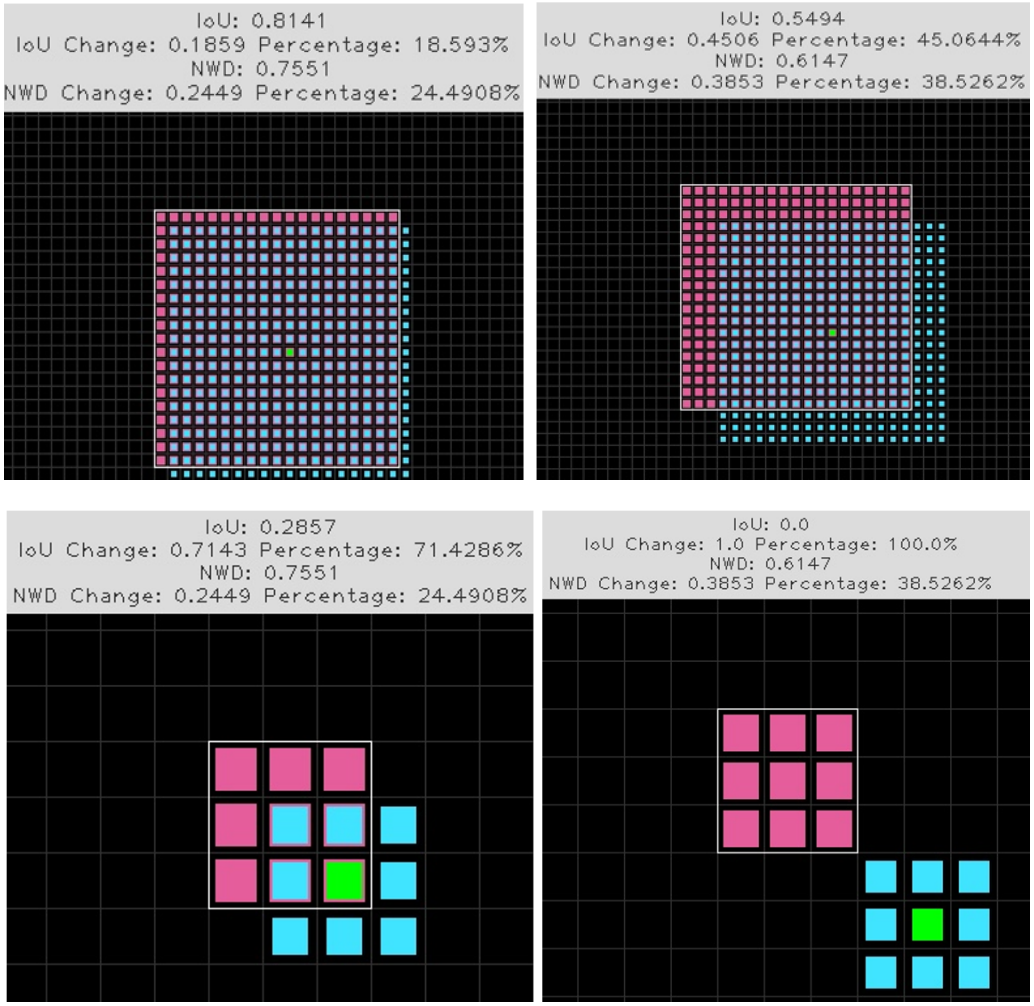
involving model deployment on edge devices [25] model architecture currently surpasses all known object detectors in speed and accuracy in the 5 to 160 FPS range.

2.1.2.2 Multi-Label Training. Many drone datasets have a single ground truth instance per image such as [15]; however, scenarios, such as swarms, exist where multiple drones are controlled in unison. Training data has been generated via the VR module to mimic this scenario better to produce images with multiple objects. There are several noteworthy points to make about these multi-label datasets. First, including multiple classes/objects in an image allows us to extract more information from a single image since it contains multiple classes. Further, it also allows us to train with a fewer number of images in a dataset with more labels. For example, two images with a single object in each allow for two objects to be learned; however, two images with five objects each allow for ten objects to be learned with the same "size" dataset (i.e., two images). This – of course – raises potential concerns about using too densely populated images which could lead to over-training.

2.1.2.3 Small Object Detection. Small object detection is a challenging task for modern object detectors. Further, small drones flying at potentially high altitudes call for the ability to detect distant objects such as drones 100-250 meters away from the camera. To improve performance on small object detection we use the Normalized Wasserstein Distance (NWD) [26] to mitigate the over-penalization seen with conventional object detection methods such as Intersection over Union (IoU). IoU is the most popular metric for object detection benchmarks [5]. The IoU effectively determines the overlap between a label and a prediction on large and medium size objects. Unfortunately, the IoU score drops off harshly with minor errors in the detection when used on small objects. The NWD attempt to mitigate this over-penalization better than the IoU while behaving similarly to IoU on medium to large-size objects.

Figure 1

IoU vs NWD on Small and Large Objects



Note. Comparison of Intersection over Union (IoU) and Normalized Wasserstein (NWD) distance for object detection. The top image shows a "normal" size object, while the bottom image shows a small object. The pink box represents the ground truth label, and the blue box represents the model prediction. The left column displays a 1-pixel diagonal deviation between prediction and label while the right column displays a 4-pixel diagonal deviation. The scores of IoU and NWD for the normal-size object (top row) are 81% and 75% for the 1-pixel deviation, and 55% and 61% for the 4-pixel deviation, respectively. For the small-size object (bottom row), the IoU and NWD scores are 28% and 75% for the 1-pixel deviation, and 0% and 38% for the 4-pixel deviation, respectively.

Let us motivate the NWD with an example. The top row of Figure 1 shows the IoU and NWD scores on a prediction of a normal size object. Here, "normal" size corresponds to an object that is comprised of many pixels. On a normal size object the NWD behaves similarly to IoU in the sense that the penalty assigned for a prediction relative to the label with a high score for a 1-pixel diagonal deviation and a slightly lower score for the 4-pixel deviation that seems appropriate for the level of overlap. Thus, the penalization of the shifts for the large objects is very similar for the IoU and NWD.

The bottom row of Figure 1 shows a small object's IoU and NWD scores. Note that the object in the bottom row of Figure 1 is significantly smaller than that in the top row. It can be seen that the IoU greatly punishes the model for even a minor mismatch in prediction and label. Note that the shift in the number of pixels for the objects in Figure 1 are the same. A single pixel diagonal deviation receives an IoU score of just 0.28 suggesting that the model is inaccurate, but when trying to detect an object that is only three pixels wide, this should be considered a fairly accurate prediction that the NWD provides, with a score of 0.75. A similar scenario can be seen in the four-pixel diagonal deviation where IoU has dropped to zero suggesting this is as bad a prediction as can be made despite being close to the label. In contrast, the NWD gives a more reasonable score of 0.38, telling the model this isn't the best prediction but that it is not way off. The NWD does this by treating the bounding boxes as distributions. The probability density function of an arbitrary 2D Gaussian distribution is presented in (Equation 1):

$$f(\mathbf{x}|\mu,\Sigma) = \frac{\exp(-\frac{1}{2}(\mathbf{x}-\mu)^{\intercal}\Sigma^{-1}(\mathbf{x}-\mu))}{2\pi|\Sigma|^{\frac{1}{2}}} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^2$ is a point given a mean μ and covariance Σ. YOLOv7's bounding box format is of the form $[c_x, c_y, w, h]$ where c_x and c_y represent the center point of the detected object, while w and h represent the width and height, respectively. The values of these bounding

boxes are then used to generate (Equation 2).

$$\mu = \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \quad \Sigma = \frac{1}{4} \begin{bmatrix} w^2 & 0 \\ 0 & h^2 \end{bmatrix} \tag{2}$$

In (Equation 2) the center point of the bounding box is used as the mean of the distribution while the width and height are used to calculate the covariance matrix. The mean and standard deviation can generate a Gaussian distribution for the bounding box. Treating the label and prediction as distributions allows for the usage of (Equation 3):

$$W_2^2(\mathcal{N}_1(\mathbf{m}_1, \Sigma_1), \mathcal{N}_2(\mathbf{m}_2, \Sigma_2)) = \|\mathbf{m}_1 - \mathbf{m}_2\|_2^2 + \left\| \Sigma_1^{\frac{1}{2}} - \Sigma_2^{\frac{1}{2}} \right\|_F^2 \tag{3}$$

where $\mathbf{m}_i$ and Σ_i are the mean and variance of a Gaussian distribution. (Equation 3) displays the second-order Wasserstein distance between two arbitrary two-dimensional Gaussian distributions. With some additional simplifications, we arrive at $W_2^2(\mathcal{N}_a, \mathcal{N}_b)$ being given by Eq. (Equation 4).

$$\left\| \left(\left[c_{x_a}, c_{y_a}, \frac{w_a}{2}, \frac{h_a}{2} \right]^T, \left[c_{x_b}, c_{y_b}, \frac{w_b}{2}, \frac{h_b}{2} \right]^T \right) \right\|_2^2 \tag{4}$$

This equation represents the Wasserstein distances between two distributions created from the ground truth label and bounding box prediction from the model. The value range for (Equation 4) is $(0, \infty]$, to better align with metrics such as intersection over union (IOU) it needs to be normalized between (0,1).

$$\mathrm{NWD}(\mathcal{N}_a, \mathcal{N}_b) = \exp\left(-\frac{\sqrt{W_2^2(\mathcal{N}_a, \mathcal{N}_b)}}{C} \right) \tag{5}$$

(Equation 5) displays the Normalized Wasserstein distance between two Gaussian distributions created from the ground truth and model prediction bounding boxes, where C is a hyper-parameter that can be tuned experimentally, a C value of 3 was used in our experimentation. With this distance, experiments can be conducted to determine its effectiveness in improving the performance of YOLOv7-tiny's ability to detect small objects.

2.2 Experimental Results

In this section, we present the experimental results and provide the implementation details. The source code to reproduce the work presented in this paper will be made publicly available.

2.2.1 Dataset Properties

The VR dataset can be generated by controlling many parameters, such as the biome, weather, etc. The VR drones were randomized across different locations in the datasets. In each image, there is a drone that is d-meters from the camera, where $d \in \{5, 10, 25, 50, 100, 250\}$. Three datasets were generated in the VR environment to determine the benefits of training on VR data with higher ground truth density per image than previous works. The first dataset, named *single label set* (SLS), contains 200,042 480×480 images with 174,866 ground truth labels with two classes: drone, and airplane. Each image in this dataset has at most one ground truth label. That is there are 25,176 images that do not have any objects and serve as a means to capture false positives in the predictions.

Figure 2(a) shows an example image from the single-label dataset. The orange bounding box represents a ground truth label with the associated class label "1" which corresponds to a drone. The single-label dataset contains a wide range of object sizes to increase variability and prepare the model to detect aerial objects both close and far.

Figure 2

Example Images of the SLS, MLS, and HDS

(a) Single Label Set (SLS) (b) Multi Label Set (MLS) (c) High Density Set (HDS)

Note. Example images from the VR dataset for aerial object detection. The figure on the left shows the single label set where each image contains at most one object. The center image shows the multi-label set that has multiple classes per image. Note that multiple classes can be found within the same image. The figure on the right shows the high-density version of the MLS where more objects are allowed within a single image.

The multi-label set (MLS) contains 196,855 480×480 images with 649,004 ground truth labels of the class airplane and drone, where the classes are equally represented in the dataset. The primary difference between the MLS and SLS datasets are that an image in MLS can have up to four ground truth labels per image, two for each class. Due to the methods that generate the data, some images may have fewer than four labels as on occasion one of the objects may be out of frame; however, these images will never have more than four objects.

Figure 2(b) shows a sample image from the MLS. This image has four different objects (e.g., drones and planes), and each object is from a different distance from the camera. One of the key properties to note about MLS and SLS is that they both are both about the same size with the MLS containing 196,855 images and the SLS 200,042 images, however, MLS and SLS have 674,180 and 174,866 objects to learn, respectively.

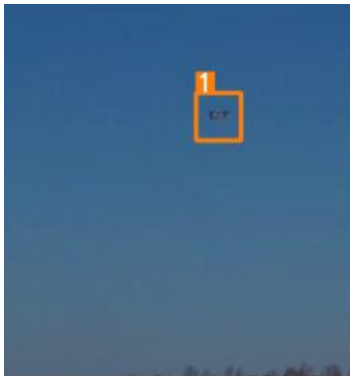
The high-density set (HDS) was motivated by the question: *can a smaller dataset*

with a higher density of ground truth labels within each image produce comparable (or better) results to a larger dataset with a lower ground truth label density while reducing memory size? The HDS dataset contains 100,020 images, which is half the number of images in the prior two datasets. The HDS dataset has 878,015 labels, which is a substantial increase in the number of objects to train on in half the number of images. The HDS contains up to ten ground truth instances per image, five of each class; airplane or drone an over 2x increase in label density over the MLS.

The motivation and goal of the VR datasets is to boost performance on real-world objects. Hence using some VR data with real-world data can lead to better performances than using either dataset alone. Therefore, the Drone Detection Dataset (DDD) [15] was utilized for comparison and fine-tuning. The DDD contains 54,277 images with 18,585 and 35,692 airplanes and drones, respectively. An example of a drone image from this dataset can be seen in Figure 3. Each image in the dataset is similar to SLS because there is only one object per image. A breakdown of the datasets and their properties can be found in Table 1.

Figure 3

Example Image from the Real-world DDD Dataset [15]



Note. Most of the images in this dataset consist of a large sky background with a drone or plane in the image with some having visible ground/horizons.

Table 1

Summary of the Dataset Properties (Real-world and VR) Used in this Work

Dataset	**Images**	**Airplane**	**Drone**	**Total**
DDD (train) [15]	45,259	14,207	31,052	45,259
DDD (test) [15]	9,018	4,378	4,640	9,018
SLS (train)	200,042	75,088	99,778	174,866
SLS (test)	20,002	6,880	10,000	16,880
MLS (train)	196,855	256,016	392,988	649,004
MLS (test)	19,899	24,731	39,633	64,364
HDS (train)	97,237	392,923	485,092	878,015
HDS (test)	9,629	36,874	47,888	84,762
Total	597,941	811,097	1,111,071	1,922,168

2.2.2 Experimental Setup

We use the YOLOv7-tiny object detection model [25]. YOLOv7-tiny was trained on SLS, MLS, and HDS with the same hyperparameters (e.g., number of layers, learning

rate, etc.). YOLOv7-tiny was trained for 100 epochs on eight Quadro RTX 8000 GPUs, the total batch size was 1024 images with a batch size of 128 per GPU. The backbone of the YOLOv7 model was pretrained on the COCO object detection dataset [27].

2.2.3 Experimental Results

Our goal in this section is to explore the answer to two questions: (1) *Does using synthetic VR data boost the performance on real-world data?* and (2) *Does the NWD loss provide a performance gain compared to the IoU loss?* Specially, we answer these questions through two experiments. The setup for the experiment to determine the efficacy of increased label density is as follows: the SLS, MLS, and HDS were all used to train YOLOv7-tiny for 100 epochs. We refer to these performances as YOLOv7-tiny+SLS, YOLOv7-tiny+MLS, and YOLOv7-tiny+HDS, respectively. After training was complete, the models were fine-tuned on the real-world DDD dataset for an additional 100 epochs. The DDD dataset was also used to train a model for 100 epochs for comparison with the fine-tuned models (i.e., this model was not trained using any VR data). This model is referred to as YOLOv7-tiny+DDD in the discussion below. After fine-tuning was completed, all models were evaluated on the real-world DDD test set to compare each model's ability to detect real-world objects. Thus, the final performances presented in this work are from the evaluation of the real-world The DDD was trained for 100 epochs using IoU and then NWD. These models were validated on the DDD test set to compare. The primary metric used for determining the accuracy of these experiments was mean average precision (mAP). mAP@0.5:0.95 was tracked and compared between models as this is the standard benchmark used to determine model accuracy in object detection.

Figure 4

mAP Scores of the SLS, MLS, and HDS

(a) YOLOv7-Tiny trained on the SLS, MLS, and DDD

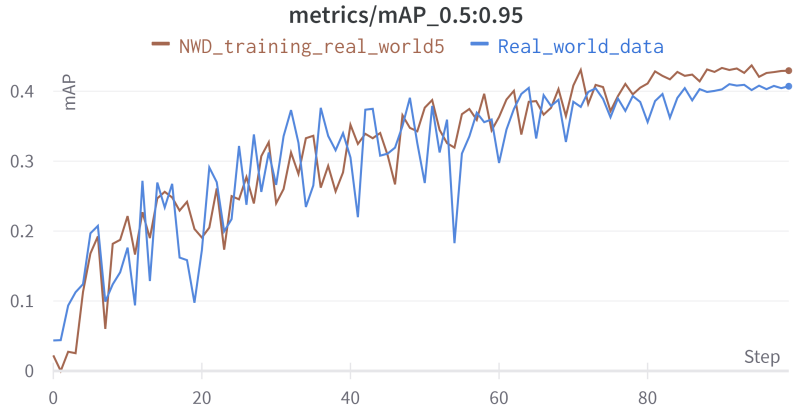
(b) YOLOv7 trained on the SLS, HDS, and DDD

Note. Comparison of mAP@0.5:0.95 for the SLS, HDS, and Real-world datasets. The mAP score is reported on the DDD testing dataset. The top shows the results of the MLS compared with the SLS and DDD while the bottom compares the HDS with the SLS and DDD.

Figure 4(a) shows the real-world test performance of YOLOv7-tiny+SLS (green), YOLOv7-tiny+MLS (red), and YOLOv7-tiny+DDD (blue), where each of the models was trained using the standard IoU loss. Note these curves show the performance of the last 100 epochs when the DDD data was used to tune each model. There are several observations to make from these results. First, the difference in performance between the model trained with DDD, and either model trained with SLS or MLS show a significant improvement in mAP. There is a 0.07 increase in mAP using MLS over DDD. Thus, synthetic VR data (either SLS or MLS) used with real-world data DDD yields better performance mAP scores than using real-world data alone. These results are further supported by the mAP scores shown in Figure 4(b). Figure 4(b) shows the mAP of YOLOv7-tiny+HDS. Again, there is a significant increase in the mAP score by using VR data rather than only the DDD set. Another observation in Figure 4(b) is that the performance for YOLOv7-tiny+SLS and YOLOv7-tiny+HDS is nearly the same; however, it is important to note that YOLOv7-tiny+HDS is trained on half the amount of data compared to YOLOv7-tiny+SLS(MLS)! Hence, the high-density label format produced the same mAP as a larger dataset, but training the HDS model took nearly three hours less! Table 2 shows the training times for each model training using VR data. We also observe that training on the MLS dataset only takes an additional 45 minutes, despite MLS having 474,138 more labels than the SLS.

Figure 5

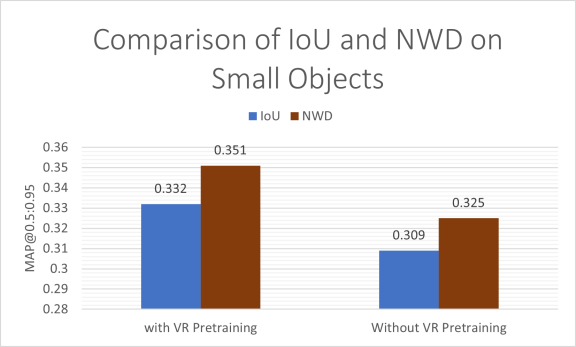
mAP of the DDD Trained Using IoU (Blue) and NWD (Brown)



The experimental results to compare the IoU and NWD are shown in Figure 5. By replacing IoU loss with the NWD, the mAP@0.5:0.95 score increased from 40.7 to 42.93. This result shows that the NWD loss can boost performance on real-world objects. Recall that the NWD is designed to boost performance on small objects. To test this hypothesis, a subset DDD was created containing images less than 16×16 pixels, or 256 square pixels. This subset of the DDD contains the 332 smallest objects in the entire DDD set. Inference was then conducted on these small objects using four models, two models pre-trained on VR data and fine-tuned on the DDD one using IoU and NWD loss, and two models not pre-trained on VR data using NWD and IoU. The mAP@0.5:0.95 of these four models can be observed in Figure 6. the VR data used for pre-training the models was the HDS. As displayed NWD outperforms IoU with or without VR pre-training but when the methods of VR pre-training is combined with the NWD we observe the best performance on small objects.

Figure 6

Results of Data and Algorithmic-level Optimizations on The Smallest Objects in the Dataset



Note. Results on the small object dataset, where small objects are considered those less than 256 square pixels. The performance is reported for the NWD loss, IoU loss, and with and without VR pre-training.

Table 2

YOLOv7-tiny Training Times for the Datasets Evaluated

Model	Train	Fine-tune	Total time
SLS	6.55	2.32	9.45hrs
MLS	7.43	2.1	9.73hrs
HDS	4.26	2.1	6.45hrs

2.3 Conclusion

As drone technology advances and becomes more prominent, there will need to be systems to identify drones. Drone detection introduces new challenges at the data and algorithmic levels. Specifically, curating a real-world drone dataset is an extremely cumbersome task. Further, aerial objects tend to be quite small and existing approaches that use the IoU loss over penalize the small objects, which for drone detection tasks yields suboptimal results. In this work, we introduced the VR data generator to quickly produce aerial object data to help compensate for the lack of real-world drone data. The advantage of the VR tool is that it can simulate environments that would be virtually impossible for a team of researchers to capture. These VR data are used to train a YOLOv7-tiny model on a higher density of labels per image. Including these synthetic data at training means that a smaller dataset can achieve the same mAP score as a YOLOv7-tiny model trained on a larger dataset with lower-density labels. The VR dataset used in these experiments is publicly available. In addition to the VR dataset, this work also introduced the normalized Wasserstein distance (NWD) as a replacement for the IoU loss. The IoU suffers from over-penalization of small prediction errors (i.e., small objects), which limits the utility of models trained with IoU on drone datasets. The NWD loss was shown – experimentally – to perform better on real-world tasks of drone detection.

Our future work will focus on novel modifications to algorithms that improve the integration of VR and real-world data to further increase the performance of the object detector on real-world data. Further, we want to develop adaptive training algorithms to automatically allocate proportions of VR and real-world data in a training set to boost performance.

Chapter 3

The DyViR Dataset

3.1 Introduction

Now that a model had been selected and tailored for the detection of drones as explained in Chapter 3, the only thing lacking was a large data pool on which to train this network. A comprehensive literature review was conducted at the start of this project to find what drone data was currently available. The results of this finding are discussed in section 2.2, but the main result found from this literature review was that no major benchmark drone detection datasets were available. While data was available it was on a relatively small scale from a modern machine-learning perspective. This presented an opportunity to create a benchmark dataset for use in the community. The Dynamic Virtual Reality Dataset for Aerial Threat Object Detection (DyViR) dataset was created in collaboration with the MAVRC virtual reality team. Generating real-world data requires immense amounts of labor and can have a high cost, while virtual reality data can be generated effectively at will. Together with the expertise of the MAVRC VR team the DyViR dataset was designed to incorporate many of the data-level optimizations found to be useful in Chapter 3.

3.2 DyViR Specs

Lessons learned from the data level optimizations found in the experimentation with the SLS, MLS, and HDS were taken into account when creating the DyViR dataset. DyViR contains three subsets of data within one large set. the breakdown of these subsets can be seen in Table 3

Table 3

Summary of Synthetic Datasets

Tracked Objects	**Number of Images**
1 Drone	100k
1 Airplane	100k
2 Drones, 2 Airplanes	200k
5 Drones, 5 Airplanes	200k
Total	**600k**

The first subset was inspired by the SLS and follows the convention of most drone data found during the literature review by containing 1 tracked object per image. There is a total of 200k images of this type half drone and half airplane. The next subset follows the format of the MLS containing up to 4 tracked objects per image, 2 drones, and 2 airplanes and the final subset follows from the HDS, containing up to 10 tracked objects per image. In total, this dataset is comprised of approximately 600k labeled images ranging from 1 to 10 labels per image. This variety is designed to obtain the best aspects of all three types of data and provide the variety necessary for the model to generalize to the different scenarios it may encounter in the field. the breakdown of the number of labels per class can be seen in Table 4.

Table 4

DyViR Label Breakdown

Class	**Number of Labels**
Airplane	1,130,174
Drone	1,491,486
Total	**2,621,660**

Despite containing 600k labeled images, DyViR contains over 2.5 million labeled objects for the model to train on. Increasing the label density provides for a higher informational density per image while the SLS-style images provide data closer to the real-world target consistent with the majority of real-world drone data available. The mild class imbalance in favor of drones is due to the nature of the synthetic data generator used to create this data, however, drones are the primary target for this model to identify so this imbalance is acceptable.

3.3 Training and Validation

DyViR is testing the theory that synthetic datasets can be used to supplement and enhance real datasets. To test this theory, synthetic datasets will be generated and tested on an ML model alongside real datasets. The performance with/without the synthetic datasets can then be observed to determine their usefulness.

The goal of using virtual reality data is to compensate for the small quantity of real-world drone data available for training. Ultimately, the object detection model must

be tested on its ability to detect real-world objects and not those synthetically generated. Therefore, a comparison will be made between the performance when using only real-world data to train and when synthetic data is used to enhance performance. The real-world data used to train and test the baseline model is taken from the Drone Detection Dataset [28]. The specifications for this dataset can be seen in Table 5.

Table 5

Drone Detection Dataset Specifications

Dataset	**Images**	**Airplane**	**Drone**	**Total**
Training [28]	45,259	14,207	31,052	45,259
Testing [28]	9,018	4,378	4,640	9,018

Two YOLOv7-tiny models will be trained in this experiment, The real-world model (RWL) will be trained on the Drone Detection Dataset training set for 100 epochs. This model is used as a baseline for the performance that can be achieved using the available real-world data. The DyViR model will first be trained on the DyViR synthetically generated dataset for 100 epochs. This DyViR model is then fine-tuned on the Drone Detection Dataset training set for an additional 100 epochs, allowing the model to learn to detect real-world objects. This model will be used to determine if synthetic data provides any advantages. Both models are tested on the Drone Detection Dataset test set and mAP@0.5:0.95 is tracked over the last 100 epochs of training

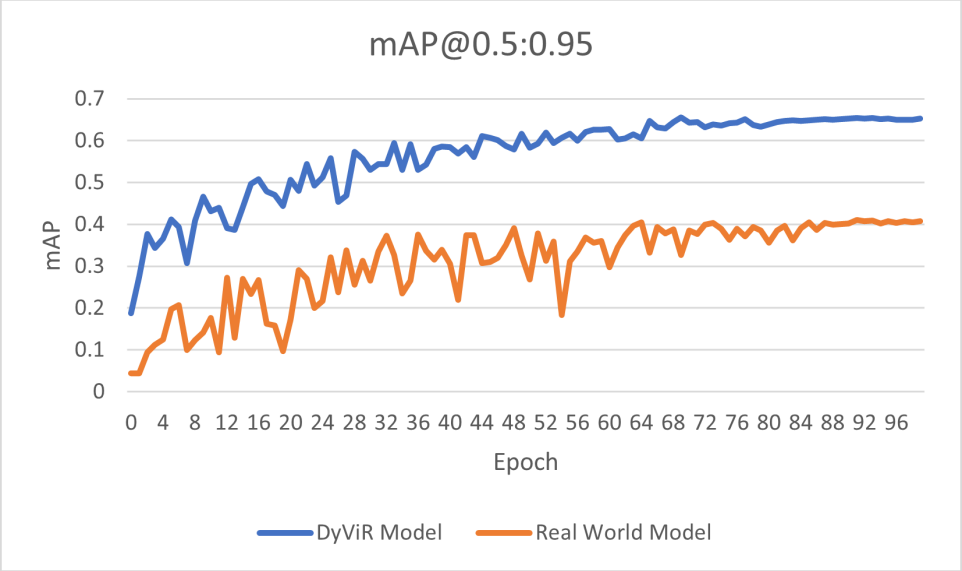
3.4 YOLOv7-tiny Training

Two YOlOv7-tiny models were trained to understand the advantages of using the DyViR dataset in improving mAP. The Real World model was trained only on the Drone

Detection Dataset training set for 100 epochs. The DyViR model was first trained on the DyViR dataset for 100 epochs. Then the DyViR model was fine-tuned on the Drone Detection Dataset training set for an additional 100 epochs to learn real-world features. The mAP@0.5:0.95 was evaluated on the Drone Detection Dataset test set for both models and tracked during the last 100 epochs of training. The mAP scores of each model across 100 epochs are displayed in Figure 7.

Figure 7

Graph of the mAP@0.5:0.95 Scores of Real World and DyViR Models Across 100 Epochs

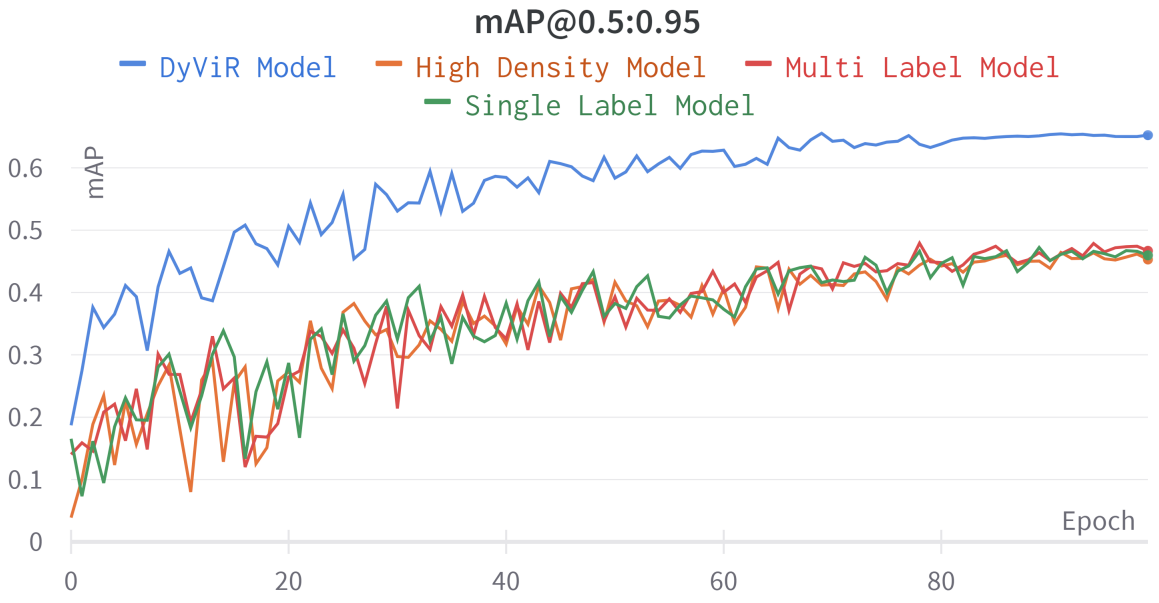


The Real World model achieved an mAP score of 0.407. The DyViR model was able to improve upon this score substantially by pre-training on the DyViR dataset. The mAP score of the DyViR model was 0.653, a 60.4% increase over the Real World model. This suggests that utilizing the DyViR dataset can effectively make up for the lack of real-world data. Generating a comparably sized real-world dataset would require significantly more resources, time, and man-hours to create than synthetic datasets. This result displays the utility of virtual reality data to enhance a machine learning model's performance.

The DyViR model was then compared with the models used for experimentation in Chapter 3, the SLS, MLS, and HDS. All of the models mAP@0.5:0.95s were compared on their performances on the DDD test set. The results of this comparison are displayed in Figure 8.

Figure 8

Graph Comparing DyViR to the SLS, MLS, and HDS



The model trained on DyViR not only outperforms when compared to the real-world model but also outperforms the SLS, MLS, and HDS by a wide margin. by increasing the amount of data and the variety of the number of labels within the data performance increased. The DyViR dataset can be used as a benchmark for machine learning models attempting to detect drones.

Chapter 4

Explainable AI for Improved Trustworthiness

4.1 Introduction

At this point in the thesis, an ML-based system for object detection with data and algorithmic optimizations specifically designed for detecting drones had been established. In addition, a benchmark drone detection dataset was created and tested for use internally and in the community to use for drone detection. The drone detection system is functional and capable of detecting drones close and far. This is great news, however, many communities show wariness and distrust for AI/ML systems that stems from concerns regarding their potential to make errors, exhibit biases, and operate without transparency or accountability [29, 30]. Specifically in a scenario where people's safety is at stake, it is critical that the systems in place can be trusted and are transparent. The first step to trusting these systems is by explaining their decision-making process. This is where the field of explainable AI (XAI) has come into play. XAI attempts to open the black box of AI systems and provide transparent explanations for the reasons these systems make their decisions. It is essential that operators and users alike can understand the systems they use. XAI allows developers to gain insight into potential biases and shortcomings of their models and allows its users to have trust in these systems. The final contribution of this thesis is laying a foundation of XAI and implementing it into the software stack to provide a method for understanding how the system is deciding what is and what isn't a drone.

4.2 Class Activation Maps (CAMS)

Class Activation Maps (CAMS) are an XAI technique widely used in computer vision to understand the discriminative regions of an image that contribute to the convolutional

neural network's (CNN) classification decisions. CAMS are obtained by taking the feature maps generated by the last convolutional layer of a CNN. these feature maps contain spatial information and encode different levels of visual features. these feature maps are passed through global average pooling resulting in a feature vector with the number of channels equal to the number of classes in the classification task. Next, a linear transformation is applied to this feature vector to obtain the weight matrix, and the dot product between the feature vector and this weight matrix is calculated to obtain the class activation values. These class activation values are then remapped onto the spatial dimensions of the input image [19, 31]. CAMS provides valuable insight into the way CNNS makes its decisions. This allows researchers to understand the networks reasoning process, serving as a powerful tool for visually explaining model predictions 2 CAMS were implemented into the software stack, Eigen-CAM [32], and Grad-CAM [33]. These CAMS provide a way to explain the model's reasoning process when determining if an object is a drone or another aerial threat.

4.3 Eigen-CAM Implementation

4.3.1 Eigen-CAM

Class Activation Mapping with Eigenvalues (Eigen-CAM) is a class activation mapping technique that utilizes eigenvalues to provide visual explanations for the model's decision-making process. There are numerous techniques for generating class activation maps; however, only a limited number possess real-time performance. Eigen-CAM was implemented into the software stack specifically for this reason, as it eliminates the necessity to perform back-propagation during the generation of the class activation map [34]. The EigenCAM authors stage the question of which features are relevant as the question of which features go through all local linear transformations and stay relevant in the same direction of maximum variation. The direction of maximum variance is simply the features along the eigenvector associated with the highest magnitude eigenvalue of the class-activated output known as the principal component.

4.3.2 Eigen-CAM Examples on the DDD Test Set

Figure 9

Eigen-CAM Examples

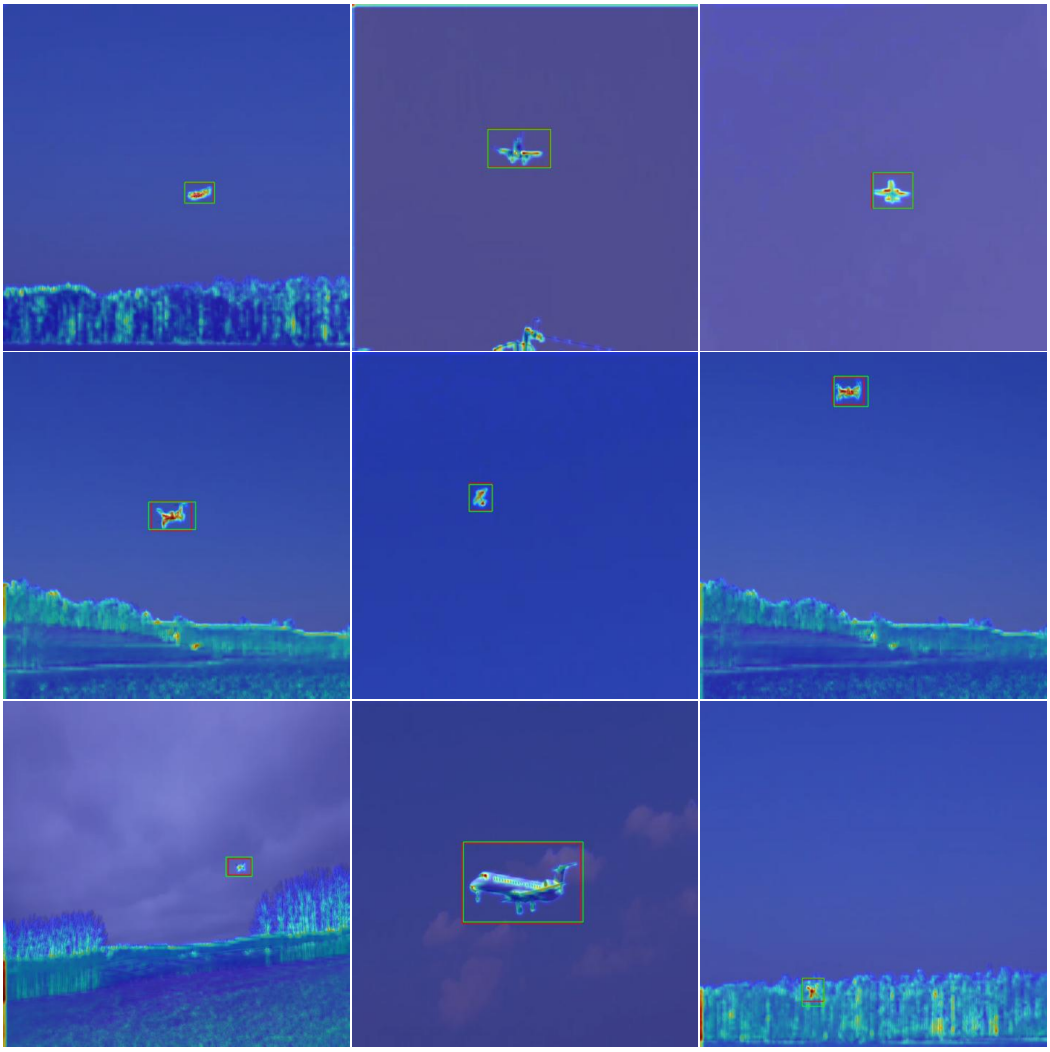


Figure 9 displays the output of running several examples of data taken from the DDD test set through the Eigen-CAM implementation. The heat map displays the areas that the model is paying the most attention to when making a prediction. The model tends to pay attention to the treeline and the object for examples that contain a treeline, suggesting the model is using other areas of the image to make its predictions. The green box displays the

ground truth and the red the model's prediction. Due to the nature of the implementation, these boxes were added in parallel during the computation of the Eigen-CAM. Eigen-CAM provides valuable insights into the decision-making process of the model that can be used by developers and works to gain the trust of its users.

4.4 Grad-CAM Implementation

4.4.1 Grad-Cam

Another more widely known CAM is Gradient-weighted Class Activation Mapping (Grad-CAM). This CAM leverages the backpropagation of the gradients during the backward pass of a network [33]. Since this method utilizes the backward pass of the network it does not possess real-time capabilities, however, it focuses on a different area of the network than Eigen-CAM, potentially providing different insights into the network's decision-making process. While Eigen-CAM exploits the forward pass of the network, Grad-CAM utilizes the backward pass. Providing explanations using 2 different methods could allow developers to gain a more well-rounded understanding of the network.

4.4.2 Grad-CAM Examples on the DDD Test Set

Figure 10

Grad-CAM Examples

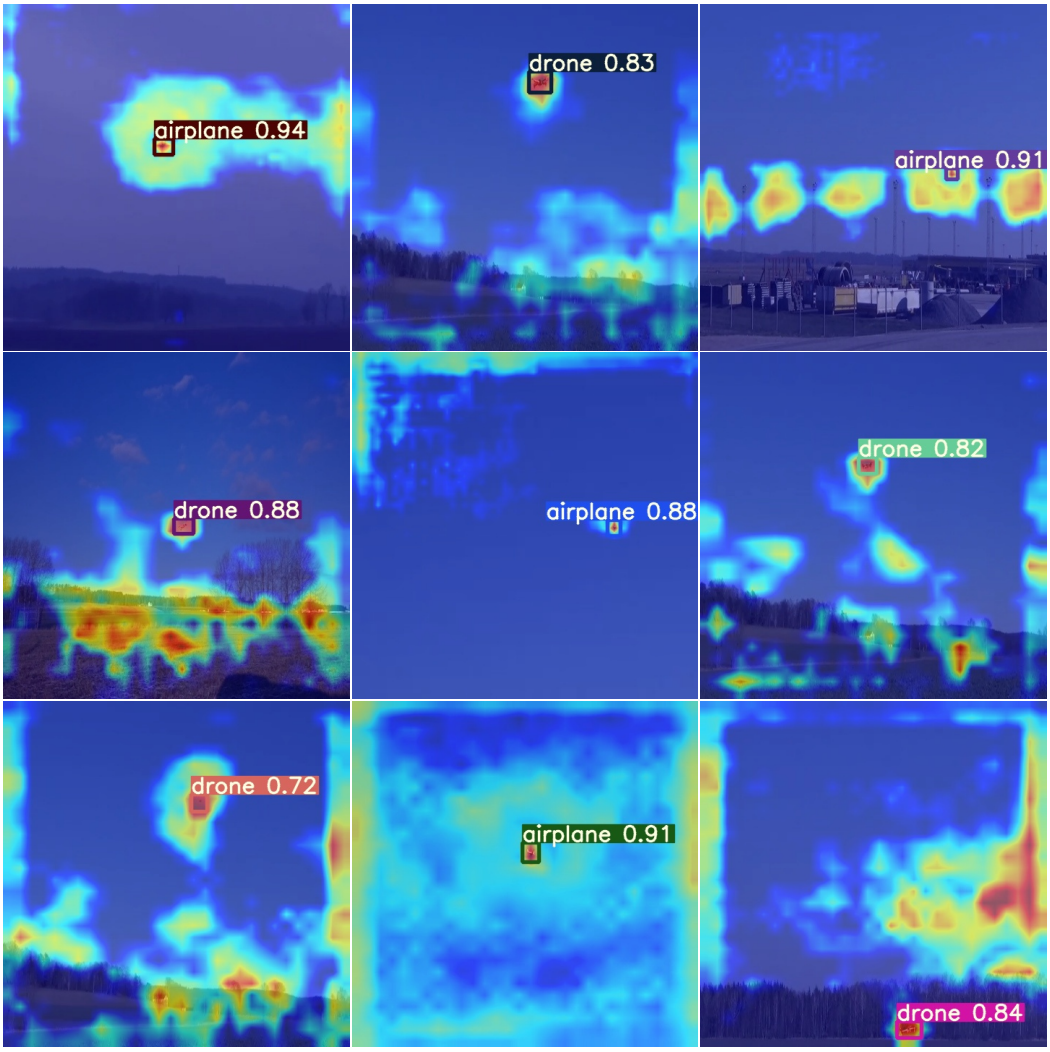


Figure 10 Displays several examples of Grad-CAM on the DDD test set. the heat maps shown display the areas the model is paying the most attention to. In this implementation, Class predictions are displayed along with confidence scores. The networks seem to focus on the areas directly around the object, as well as other pixels throughout the image, suggesting the model is looking at more than just the object it detects when making

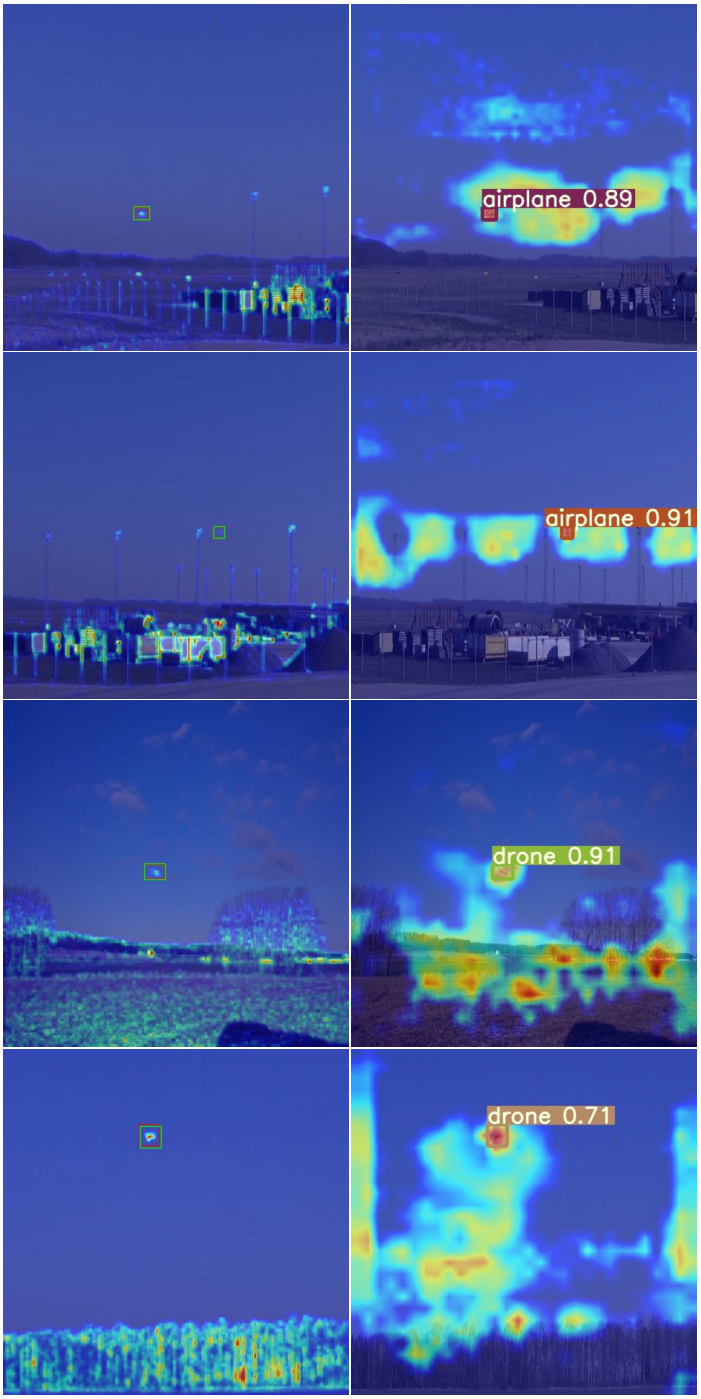
its predictions. Grad-CAM provides important insights into the models' prediction-making process

4.5 Comparison of Eigen-Cam & Grad-CAM

Figure 11 Displays 2 examples from each class of the DDD test wet with Eigen-CAM (left) and Grad-CAM (right) heat maps displaying the models' decision process. The difference in these images suggests that just one CAM does not provide the full picture of how a prediction and different cams can give different insights into the decisions made by the model. The open areas around the object in Eigen-CAM are not being considered in the decision of the model but in Grad-CAM it seems to pay attention to the object and the area around it. Eigen-CAM also utilizes the background object in the image more than Grad-CAM. One reason for Eigen-CAMs' seemingly less noisy heat maps could be due to it being a newer CAM, that utilizes methods to suppress less important pixels in the generation of the heatmap [35]. Both Methods seem to correctly attribute their decisions to the object being detected overall, just in different ways. by using two methods of explanation that focus on different areas of the model, Eigen-CAM being the forward pass, and Grad-CAM leveraging the backward pass, different insights can be gained about how this model is making its decisions. Both of these methods have been added to the software stack, opening the door to future investigations seeking to open the black box of this AI system and understand how the model makes its decisions. By doing so, developers can take these insights and further improve the model, and its users can begin to gain a sense of trust for how the network that is designed to increase their survivability in a combat environment is predicting a potential threat.

Figure 11

Visual Comparison of Eigen-CAM (Left) and Grad-CAM (Right) on the Same Images



Chapter 5

Conclusion

The objective given at the start of this thesis was to detect drones and other aerial threats to increase the survivability of turret gunners on the modern battlefield. Three main contributions were implemented to achieve this objective: An implementation of the YOLO object detector with data and algorithmic-level optimizations to increase performance on drone detection, the creation and testing of a benchmark drone detection dataset for use internally and for the computer vision community, and the addition of XAI systems to the software stack to gain a better understanding of the model, and build trust in its decisions. The YOLO object detector was selected for the task of detecting drones due to its high accuracy and real-time detection capabilities. That begin said, off the shelf this model has some shortcomings when it comes to detecting small objects. For instance, small objects are known to be a difficult area for this model. This thesis explored data-level optimizations such as the addition of multiple labels per image to determine if label density can improve performance. It was found that a higher label density performance could be increased marginally and that a smaller dataset with a higher image density could produce a comparable performance to a larger dataset with a lower label density. This finding suggests a correlation with the density of objects within an image for training, and the overall performance of the model. Algorithmic optimizations were also explored in this thesis. Traditional object detection metrics such as IoU tend to over-penalize small errors when training on small objects. This thesis introduced using the Normalized Wasserstein Distance (NWD) as known as the earth movers distance in place of IoU in the loss function during training. the NWD works by treating the bounding box and ground truth as probability distributions, working to mitigate the over-penalization seen by IoU on small objects. The NWD was able to improve performance on the detection of the smallest real-world

objects available for testing. Finally, the combination of both the data and algorithmic optimizations was proven to boost performance more than when just one of these methods was implemented.

A comprehensive literature review was conducted at the beginning of this research to find publicly available drone datasets. While some data was found including the Drone Detection Dataset [28], the primary conclusion of the literature review was that no major benchmark drone detection dataset was publicly available for the community. the DyViR drone detection dataset was created to serve as a benchmark dataset for drone detection. DyViR is a synthetic dataset designed for training ML models to detect drones incorporating the data level optimizations found in this research. Mean Average Precision improved by over 60% when testing a model trained with DyViR over a model without it on real-world data. This dataset was made publicly available to facilitate further advancements in the field, potentially serving as a benchmark for drone detection.

It is crucial that the systems built in this research can be trusted as their intended purpose is to increase the survivability of turret gunners in a combat environment. That being said, the next logical step after building the drone detection system was to implement XAI system into the software stack to gain a better understanding of the model's decision process. Eigen-CAM and Grad-CAM, are two state-of-the-art methods for explaining how an object detection model is making its decisions about the location and class of an object. Adding these methods for explainability work to allow the developers to understand any shortcomings or biases in the network, and to gain trust in the eyes of the user that these systems are unbiased and reliable.

The TGSSE project was in its beginnings at the start of this thesis, starting from ground zero an objective was given, to detect drones. This thesis has contributed a reliable system to detect drones and other aerial threats such as airplanes, created a benchmark

drone detection dataset leveraging data level optimizations found during research, and implemented Eigen-CAM and Grad-CAM, laying the foundations of XAI in this project to begin understanding the inner workings of the model and gain the trust of both developers and user alike. The findings and methodologies presented in this thesis lay a foundation for further research and advancements in drone detection technologies to contribute to the TGSSE project, ultimately working to increase the survivability of boots on the ground in today's modern combat environment.

"In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Rowan University's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from Rights Link. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.[36, 37]"

References

[1] D. Gregory, "From a view to a kill: Drones and late modern war," *Theory, culture & society*, vol. 28, no. 7-8, pp. 188–215, 2011.

[2] B. A. Card, "Terror from above: How the commercial unmanned aerial vehicle revolution threatens the us threshold," *Air & Space Power Journal*, vol. 32, no. 1, pp. 80–96, 2018.

[3] R. J. Wallace and J. M. Loffi, "Examining unmanned aerial system threats & defenses: A conceptual analysis," *International Journal of Aviation, Aeronautics, and Aerospace*, vol. 2, no. 4, p. 1, 2015.

[4] P. Casabianca and Y. Zhang, "Acoustic-based uav detection using late fusion of deep neural networks," *Drones*, vol. 5, no. 3, 2021, ISSN: 2504-446X. DOI: 10.3390/drones5030054. [Online]. Available: https://www.mdpi.com/2504-446X/5/3/54.

[5] S. H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. D. Reid, and S. Savarese, "Generalized intersection over union: A metric and A loss for bounding box regression," *CoRR*, vol. abs/1902.09630, 2019. arXiv: 1902.09630. [Online]. Available: http://arxiv.org/abs/1902.09630.

[6] J. Han, J. Ding, N. Xue, and G.-S. Xia, "ReDet: A Rotation-equivariant Detector for Aerial Object Detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2785–2794, ISBN: 9781665445092. DOI: 10.1109/CVPR46437.2021.00281. [Online]. Available: https://ieeexplore.ieee.org/document/9577528/ (visited on 01/25/2023).

[7] G.-J. Horng, M.-X. Liu, and C.-C. Chen, "The Smart Image Recognition Mechanism for Crop Harvesting System in Intelligent Agriculture," *IEEE Sensors Journal*, vol. 20, no. 5, pp. 2766–2781, 2020, ISSN: 1530-437X, 1558-1748, 2379-9153. DOI: 10.1109/JSEN.2019.2954287. [Online]. Available: https://ieeexplore.ieee.org/document/8906085/ (visited on 01/25/2023).

[8] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," en, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 117, pp. 11–28, Jul. 2016, ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2016.03.014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0924271616300144 (visited on 01/25/2023).

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The {KITTI} Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[10] B. Taha and A. Shoufan, "Machine learning-based drone detection and classification: State-of-the-art in research," *IEEE access*, vol. 7, pp. 138 669–138 682, 2019.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016. DOI: 10.1109/CVPR.2016.91.

[13] B.-K. Kim *et al.*, "Drone detection with chirp-pulse radar based on target fluctuation models," *ETRI Journal*, vol. 40, no. 2, pp. 188–196, 2018.

[14] S. Al-Emadi and F. Al-Senaid, "Drone detection approach based on radio-frequency using convolutional neural network," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, IEEE, 2020, pp. 29–34.

[15] F. Svanstrom, C. Englund, and F. Alonso-Fernandez, "Real-time drone detection and tracking with visible, thermal and acoustic sensors," *International Conference on Pattern Recognition (ICPR)*, 2021. DOI: 10.48550/ARXIV.2007.07396. [Online]. Available: https://arxiv.org/abs/2007.07396.

[16] Y. Wang, Y. Chen, J. Choi, and C.-C. J. Kuo, "Towards visible and thermal drone monitoring with convolutional neural networks," *APSIPA Transactions on Signal and Information Processing*, vol. 8, e5, 2019. DOI: 10.1017/ATSIP.2018.30.

[17] C. Rui, G. Youwei, Z. Huafei, and J. Hongyu, "A comprehensive approach for uav small object detection with simulation-based transfer learning and adaptive fusion," *arXiv:2109.01800*, 2021. DOI: 10.48550/ARXIV.2109.01800. [Online]. Available: https://arxiv.org/abs/2109.01800.

[18] S. Shah, D. Dey, C. Lovett, and A. Kapoor, *Airsim: High-fidelity visual and physical simulation for autonomous vehicles*, 2017. DOI: 10.48550/ARXIV.1705.05065. [Online]. Available: https://arxiv.org/abs/1705.05065.

[19] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929, 2016.

[20] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

[21] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," *arxiv:1905.05055*, 2023. [Online]. Available: http://arxiv.org/abs/1905.05055 (visited on 01/23/2023).

[22] P. Zhu, L. Wen, X. Bian, H. Ling, and Q. Hu, "Vision Meets Drones: A Challenge," *arxiv:1804.07437*, 2018. [Online]. Available: http://arxiv.org/abs/1804.07437 (visited on 01/25/2023).

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788, ISBN: 9781467388511. DOI: 10.1109/CVPR.2016.91. [Online]. Available: http://ieeexplore.ieee.org/document/7780460/ (visited on 01/25/2023).

[24] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, 2017, pp. 6517–6525, ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.690. [Online]. Available: http://ieeexplore.ieee.org/document/8100173/ (visited on 01/25/2023).

[25] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv:2207.02696*, 2022. DOI: 10.48550/ARXIV.2207.02696. [Online]. Available: https://arxiv.org/abs/2207.02696.

[26] C. Xu, J. Wang, W. Yang, H. Yu, L. Yu, and G.-S. Xia, "Detecting tiny objects in aerial images: A normalized wasserstein distance and a new benchmark," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 190, pp. 79–93, Aug. 2022. DOI: 10.1016/j.isprsjprs.2022.06.002. [Online]. Available: https://doi.org/10.1016%2Fj.isprsjprs.2022.06.002.

[27] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8693, Cham: Springer International Publishing, 2014, pp. 740–755. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10602-1_48.

[28] F. Svanstrom, C. Englund, and F. Alonso-Fernandez, "Real-Time Drone Detection and Tracking With Visible, Thermal and Acoustic Sensors," 2020, Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2007.07396. [Online]. Available: https://arxiv.org/abs/2007.07396 (visited on 02/28/2023).

[29] S. Jackson and N. Panteli, "Trust or mistrust in algorithmic grading? an embedded agency perspective," *International Journal of Information Management*, vol. 69, p. 102 555, 2023.

[30] C. Dwork and M. Minow, "Distrust of Artificial Intelligence: Responses & Sources from Computer Science and Law," *Daedalus*, vol. 151, no. 2, pp. 309–321, May 2022, ISSN: 0011-5266. DOI: 10.1162/daed_a_01918. eprint: https://direct.mit.edu/daed/article-pdf/151/2/309/2060588/daed_a_01918.pdf. [Online]. Available: https://doi.org/10.1162/daed%5C_a%5C_01918.

[31] H. Jung and Y. Oh, "Towards better explanations of class activation mapping," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1336–1344.

[32] M. B. Muhammad and M. Yeasin, "Eigen-CAM: Class activation map using principal components," in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2020. DOI: 10.1109/ijcnn48605.2020.9206626. [Online]. Available: https://doi.org/10.1109%2Fijcnn48605.2020.9206626.

[33] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[34] M. B. Muhammad and M. Yeasin, "Eigen-cam: Class activation map using principal components," *CoRR*, vol. abs/2008.00299, 2020. arXiv: 2008.00299. [Online]. Available: https://arxiv.org/abs/2008.00299.

[35] M. B. Muhammad and M. Yeasin, "Eigen-cam: Class activation maps with eigen-consistency regularization," *arXiv preprint arXiv:2104.10245*, 2021.

[36] N. Koutsoubis *et al.*, "© 2023 ieee. reprinted, with permission, from: DyViR: Dynamic Virtual Reality Dataset for Aerial Threat Object Detection," in *SPIE: Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications*, 2023.

[37] N. Koutsoubis, K. Naddeo, G. Williams, G. Lecakes, G. Ditzler, and N. Bouaynaya, "© 2023 ieee. reprinted, with permission, from: Boosting Aerial Object Detection Performance via Virtual Reality Data and Multi-Object Training," in *submitted to the IEEE/INNS International Joint Conference ion Neural Networks*, 2023.