Adversarial Training of YOLOv11n Object Detector for Aerial Imagery Object Detection

Luke Unterman
Virginia Commonwealth University
401 West Main Street
Richmond, VA, 23284, USA
untermanlm@vcu.edu

Jannat Lnu
Virginia Commonwealth University
401 West Main Street
Richmond, VA, 23284, USA
lnuj3@vcu.edu

Jimmy Le
Virginia Commonwealth University
401 West Main Street
Richmond, VA, 23284, USA
lejp3@vcu.edu

My Nguyen
Virginia Commonwealth University
401 West Main Street
Richmond, VA, 23284, USA
nguyenm21@vcu.edu

***Abstract*—Deep neural networks (DNNs) have become the de facto standard for various object detection pursuits, including the object detection of aerial imagery. However, DNNs are particularly vulnerable to adversarial attacks, which can imperceptibly alter the images used by object detection models, causing them to make incorrect predictions. This vulnerability to adversarial attacks poses a considerable security threat to the military and intelligence fields, which rely on using imagery collected by unmanned aerial vehicles (UAVs) and satellites to accurately identify objects like military land vehicles. This paper aims to determine how robust the YOLOv11n object detection model is to different adversarial attacks (Fast Gradient Sign Method, Carlini-Wagner Method) when fine-tuned on the aerial imagery in the VisDrone-DET2019 dataset. This paper also aims to create a robust fine-tuned model trained on adversarial images created by the FGSM and CW adversarial methods. Finally, we will compare how the two models differ in mean Average Precision (mAP) to quantify the impact of adversarial training on the robustness of YOLOv11n.**

***Index Terms*— convolutional neural networks, object detection, adversarial training, aerial imagery**

I. INTRODUCTION AND BACKGROUND

An artificial neural network (ANN) is a kind of machine learning model which imitates the form and functionality of biological neural networks in human brains. The architecture of an ANN resembles an interconnected directed network of nodes, primarily consisting of an input layer, one or more hidden layers, and an output layer. Each of the intermediate hidden layers utilize nonlinear activation functions to ensure that a "signal", or the activation function output of the linear combination of the ANN's input features and weights, propagates throughout the network's edges. These weights are trained via backpropagation and gradient descent to minimize an error metric, or loss function, which ensures that loss incrementally decreases with the number of epochs training the ANN's weights.

Deep neural networks are a subset of ANNs which possess multiple levels of nonlinear operations, such as ANNs with many hidden layers [5]. These DNNs, which can progressively learn more and more abstract features as the depth of their network increases, have been found to achieve excellent performance on computer vision, natural language processing, and audio processing tasks [4]. Shallow neural networks can achieve similar performance when addressing these tasks, although shallow architectures often struggle with poor generalization and can require a very large quantity of neurons to adequately represent functions easily represented by DNNs [5].

However, a particularly dangerous property of DNNs trained using backpropagation is that they have intrinsic blind spots which make them susceptible to adversarial attacks [4]. These adversarial attacks work by masking the input images of a model with imperceptible noise optimized to maximize test error, and they can quite effectively dismantle state-of-the-art image classification and object detection-based DNNs. These images produced via adversarial attacks are known as adversarial examples. A famous adversarial example produced by the Fast Gradient Sign Method [3] caused a GoogleNet convolutional neural network (CNN) trained on the ImageNet dataset to classify a clearly identifiable image of a panda as a gibbon with a 99.3% confidence value. This indicates that these image classification models with low test error do not actually learn the underlying concepts behind different classes of images, but they instead extrapolate patterns from bias in their training sets.

The Fast Gradient Sign Method (FGSM) is known as a white-box adversarial attack, as it assumes total knowledge of the GoogLeNet CNN used to classify images from the ImageNet dataset [7]. FGSM is the most ubiquitous of the adversarial attack methods, as it efficiently calculates the optimal perturbation using the gradient of a model's loss function, which is easily calculated using backpropagation [3]. Other popular white-box adversarial attacks include the Projected Gradient Descent Method (PGD) [1] and the Carlini-Wagner (CW) method [11], although at least 78 different adversarial training methods have been proposed [2].

Black-box adversarial training methods, conversely, can be generated without complete knowledge of the underlying

model being used, the data that the model was trained on, or the parameters of the trained data architecture [7]. While white-box adversarial methods can most effectively dismantle image and object detection models due to their complete access to the model and training data distribution, black-box methods are typically more flexible and model-independent than white-box methods as they require less information to generate adversarial examples [2].

The effectiveness of these adversarial examples on DNN-based image and object detectors poses a considerable security threat to the military and intelligence fields, which rely on the accurate detection of different objects from aerial imagery. Other fields that benefit from focusing on adversarial aerial image detection include autonomous driving, face recognition, and malware detection. This is due to aerial images having specific characteristics and applications that require robust detection and current models do not learn the underlying concepts in a robust manner. [1].

Object detection models typically differ from traditional CNN-based image detection models in terms of their outputs; in addition to image classification labels, they also output confidence scores and bounding boxes [6]. The confidence scores are self-explanatory, as they represent the confidence the model has in their object classification, and the bounding boxes are rectangular outlines placed around the object within an image to allow for multi-object classification within an image. Modern CNN-based object detection architectures include R-CNNs, Faster R-CNNs, YOLO, and SSD, which each vary in terms of object detection performance, number of parameters, and runtime performance. Non-CNN based object detection architectures have also been proposed, such as the Orientation Robust Detector (OrtDet) [2], which uses Vision Transformers to detect arbitrarily rotated objects in aerial imagery.

The object detection performance of these models is typically calculated using metrics like Intersection over Union (IOU), Top-N accuracy, and Mean Average Precision (mAP) [6]. IOU is measured as the ratio of the overlap between predicted bounding boxes and ground-truth bounding boxes to their union, Top-N accuracy measures the number of predictions where the correct class is within the top N confidence values, and mAP represents the mean of AP scores across the object classes in the test dataset [6].

Common difficulties associated with object detection in aerial images include insufficient access to large-scale annotated training datasets, training imagery unrepresentative of real-world objects, the overinclusion of small images not containing contextual information about their objects' surroundings, imbalanced ratios between the number of included positive samples over negative samples, and the need for rotation invariant objects [8]. For example, the xView dataset contains 1,413 aerial images, 16 main categories, and 1 million instances, although it uses horizontal bounding boxes in its training dataset instead of oriented bounding boxes [8]. The VisDrone and DOTA-v2.0 aerial imagery datasets have been released in recent years, containing 10,209 and 11,268 aerial images respectively, attempting to remedy the inadequacies of previous aerial imagery datasets [8].

Thankfully, various defense protocols have been proposed to protect these object detection models from adversarial images. For example, Srivastave et al. [10] identified the use of dropout, a technique involving randomly dropping neurons from layers during training to avoid neuron co-adaptation, in DNNs to increase their robustness with respect to adversarial examples. Goodfellow et al. [3] discovered that adversarial training, the process of including accurately labeled adversarial examples in the training set, increased model generalization and reduced the error rate of a CNN trained on MNIST when exposed to adversarial examples from 89.4% to 17.9%. Other approaches include developing detectors capable of detecting objects in arbitrary directions by learning equivariant features, using vision transformers to capture feature similarities, and using group convolutions to extend traditional CNN features to acquire rotation equivalence [2].

II. PROBLEM STATEMENT

Deep neural networks (DNNs) have become the de facto standard for various object detection pursuits, including object detection of aerial imagery. However, DNNs are particularly vulnerable to adversarial attacks, which can imperceptibly alter the images used by object detection models, causing them to make incorrect predictions. This vulnerability to adversarial attacks poses a considerable security threat to the military and intelligence fields, which rely on using imagery collected by unmanned aerial vehicles (UAVs) and satellites to accurately identify objects like military land vehicles.

For this paper, we propose to conduct a study where we evaluate how susceptible a pretrained YOLOv11n object-detection model is to different forms of adversarial attacks and develop a DNN-based object detection model robust to these adversarial attacks via fine-tuning the pretrained model on adversarial images. Our goal in conducting this study is to minimize the negative impact that adversarial images have on object detection models, creating a more robust and generalizable model in the process.

III. SOLUTION

A. ARCHITECTURE

To choose between the different state-of-the-art object detection models for our aerial object detection approach, including the YOLO, Regional CNN (R-CNN), and Single Shot MultiBox Detector (SSD) based architectures, we considered a multitude of different performance surveys related to aerial object detection.

For example, in their paper entitled "Deep Learning Application for Vehicle Detection through Surveillance Drones," Ilyas et al. [19] compared the performance of the YOLOv8, Faster R-CNN, and SSD models across three different datasets. These datasets include the Vehicle Detection Dataset [14], the Pak Vehicles Dataset [15], and the VisDrone Dataset [16].

Although the VisDrone dataset is capable of detecting non-vehicle-based classes, including pedestrians, the performance of each of the datasets was determined by their ability to classify vehicles in different orientations. Furthermore, each model trained for 300 epochs per dataset with a batch size =

16, learning rate = 0.01, weight decay = 0.0005, and SGD optimizer for the SSD and Faster R-CNN models.

Ultimately, the YOLOv8 model handily outperformed the Faster R-CNN and SSD models across all three datasets. For example, the YOLOv8, Faster R-CNN, and SSD models scored mAP(0.5) values of 0.743, 0.131, and 0.067 respectively on the VisDrone dataset and mAP(0.5) values of 0.946, 0.895, and 0.824 respectively on the Vehicle Detection Dataset. Ilyas et al. [19] claims that the YOLOv8 model is generally better for vehicle detection from an aerial imagery than the other models, with higher average model accuracy and less inference time overall. We therefore choose to adopt the YOLO architecture for object detection problem as a result of the sheer dominance it exhibits in the aerial object detection space.

However, YOLOv8 is not the most recent version of the YOLO architecture; therefore, we also reviewed Jegham et al.'s benchmark study [16] evaluating the performance of YOLOv11 and its predecessors to ensure that we select the best possible object detection model for our problem. We optimized for the following criteria: speed, accuracy, and accessibility.

We define a model to be "accessible" if it is supported by Ultralytics, an artificial intelligence platform that hosts various object detection models. Currently, Ultralytics supports models YOLOv3-11, although Jegham et al. claims that YOLOv4, YOLOv6, and YOLOv7 are only "soft supported." Thus, YOLOv5 and YOLOv8-11 remain the most accessible models.

To find the most performant of the YOLO object detection models, Jegham et al. compared the Precision, Recall, and mAP(IoU=0.5) values across the Traffic Signs, African Wildlife, and Ships/Vessels object detection datasets. To maintain consistency in their experimentation, Jegham et al. used the same number of epochs, AdamW optimizer, batch size, image size, learning rate, and training/validation/test split for each of their trials. Ultimately, they determined that the YOLOv11 models on average were the most consistent performers, striking a balance between high accuracy, low processing time, and efficient disk usage. Thus, we have decided to specifically use the YOLOv11n (smallest, "nano") model for our object detection problem.

B. ADVERSARIAL METHODS

Adversarial attack algorithms have become essential tools for studying the vulnerabilities of pretrained object detection models and for developing more robust models capable of resisting such attacks. These attacks involve crafting perturbations to the input data that deceive the model into misclassifying objects, while remaining imperceptible to human observers. Among the most used adversarial attack methods are the Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), and Carlini-Wagner (C&W) attacks, which are employed to generate adversarial samples against which the model is trained.

The Fast Gradient Sign Method (FGSM), introduced by Goodfellow et al. [3], is a gradient-based white-box attack designed to modify inputs to maximize the model's loss. The optimization can be formulated as:

$$x' = x + \epsilon \cdot \text{sign}(\nabla L(x,y))$$

where x is the original input, x' is the perturbed input, ϵ represents the perturbation budget, $\nabla L(x,y)$ is the gradient of the loss function with respect to the input x and its label y, and sign denotes the element-wise sign function. This method requires only a single gradient computation and perturbation step, making it highly efficient for evaluating a model's sensitivity to small adversarial changes.

Projected Gradient Descent (PGD), as introduced by Madry et al. [1], extends the FGSM by applying multiple iterative steps to refine the adversarial perturbation. In each iteration, the input is adjusted by a step size α, based on the sign of the gradient, followed by a projection operator that ensures the perturbation remains within predefined bounds. The update rule is:

$$x_{t+1}' = \Pi(x_t + \alpha \cdot \text{sign}(\nabla x J(\Theta, x_t, y))$$

where x_t is the input at iteration t, α is the step size, and $\nabla x J(\Theta, xt, y)$ is the gradient of the loss function with respect to the input at iteration t. The projection operator Π ensures that the perturbed input remains within the allowed perturbation bounds.

The Carlini-Wagner (C&W) attack, developed by Carlini and Wagner [11], is an iterative optimization process designed to generate highly effective adversarial examples. The attack optimizes an objective function that balances the perturbation's magnitude and the model's misclassification confidence. The objective function is:

$$J(x') = \alpha \cdot \text{dist}(x,x') + \beta \cdot \text{loss}(f(x'), y_t)$$

where x is the original input, x' is the perturbed input, $\text{dist}(x,x')$ measures the perturbation (typically using the L2 or $L\infty$ norm), and $\text{loss}(f(x'), y_t)$ $\text{loss}(f(x'), y_t)$ represents the misclassification loss of the target model ff on the perturbed input with respect to the target class y_t . The parameters α and β are weighting factors that balance the two objectives. The C&W attack aims to achieve two conflicting goals: minimizing the perturbation to preserve visual similarity with the original input and maximizing misclassification confidence to ensure the perturbed input to be classified incorrectly by the target model.

Selecting either one or multiple of these adversarial methods for use in our project requires examining a tradeoff between the speed of generating adversarial examples versus the effectiveness of the adversarial example in fooling the YOLOv11n object detection model. The Fast Gradient Sign Method is the simplest and fastest, as it only requires calculating the gradient of the model's loss function with respect to its input x, although it can be dispelled through defensive distillation [11].

The Projected Gradient Descent adversarial method is more effective than the Fast Gradient Sign Method, as it uses multiple iterative steps to refine the adversarial perturbation. However, it is slower than FGSM and works poorly on

gradient descent approaches with complicated update steps [11]. Furthermore, the Carlini-Wagner attack is quite effective, as it can bypass the defensive distillation method for making DNNs robust to adversarial examples, although it is slower than iterative gradient sign methods by a factor of 10.

Ultimately, we have decided to test both the Fast Gradient Sign and Carlini-Wagner methods on our YOLOv11n object detection model to represent the most common adversarial example generation method and one of the most effective adversarial methods in our study.

C. DATASET

For our study, we have identified two potential candidates to select as our main dataset: VisDrone-DET2019 and DOTA-v2.0.

VisDrone-DET2019

The VisDrone2019 dataset is collected by the AISKYEYE team, at the Lab of Machine Learning and Data Mining, Tianjin University, China. The benchmark dataset consists of 288 video clips formed by 261,908 frames and 10,209 static images, captured by various drone-mounted cameras. The distribution consists of 6,471 images for the training set, 548 images for the validation set, and 3,190 images (from both sets) for the testing set.

This dataset includes annotations for 10 categories including pedestrians, person, car, van, bus, truck, bicycle, tricycle, awning-tricycle, and motor. These categories are particularly relevant for military and surveillance applications, which aligns with our research focus on robust object detection for security applications.

The many advantages of this dataset involve the disperse capturing conditions (different altitudes, weather conditions, camera angles) help models prepare for real-world scenarios, sufficient data volume, detailed annotations that could help understand detection failures, small object detection relevance for military applications, and variation in object densities to support handling both isolated and densely packed objects. For the disadvantages, this dataset consists of only horizontal bounding boxes (which doesn't account for object orientation), class imbalance (heavily representation of cars and pedestrians), geographical focus on Chinese environments, and lower resolution in many images compared to modern UAVs.

DOTA-v2.0

DOTA is a large-scale dataset specifically designed for object detection in aerial images, containing 11,268 images with 1,793,658 object instances across 18 categories including planes, ships, storage tank, bridge, airport, and other categories.

This dataset has the advantages of using only oriented bounding box (OBB) annotations, larger image resolution ranges (800x800 to 20,000x20,000 pixels), 18 object categories to provide broader training context, and multiple views of same locations helpful for understanding adversarial attacks from different angles. The disadvantages, however, are the high-resolution images that may increase computational requirements, a bias towards structured objects rather than vehicles, a lack of environmental diversity, and a more complex processing requirement due to OBB format.

Ultimately, while both datasets have attractive features, including the VisDrone dataset's usage of multiple capturing conditions and the DOTA dataset's usage of OBB annotations, we choose to select the VisDrone dataset as our primary dataset because of processing time constraints. More specifically, we chose the VisDrone dataset because it has approximately 1,000 fewer static images to process, and we highly value processing speed for our experimentation.

D. METRICS

To thoroughly evaluate the effectiveness of our adversarial training approach and quantify the robustness gains in the YOLOv11 object detection model, we will employ a comprehensive set of metrics.

Mean Average Precision (mAP) is the industry standard metric for object detection evaluation, calculated as the mean of Average Precision (AP) scores across all object classes:

$$mAP = (1/N) \Sigma AP_i$$

where N is the number of classes, and AP is defined as the area under a precision-recall curve. A precision-recall curve is a plot detailing the tradeoffs between precision values ((TP)/(TP+FP)) and recall values ((TP)/(TP+FN)) at different thresholds, where all TP, FP, and TN values are determined using the Intersection over Union (IoU) metric. This metric is used to measure how much the object detection model's predicted bounding box overlaps with the ground truth bounding box, and is defined as:

$$IoU = \text{Area of Intersection} / \text{Area of Union}$$

Generally, a predefined IoU threshold (such as 0.5) determines whether a prediction is a true positive (TP) or a false positive. More specifically, if a prediction has an IoU >= 0.5, it is treated as a TP. Generally, a TP means the model correctly detected an object, false positive (FP) incorrectly detected an object, false negative (FN) where the model fails to detect an existing object, and true negative (TN) where the model did not correctly detect a non-existent object.

Additionally, the F1 metric is commonly used for object-detection pursuits, as it tends to be useful for classification and detection tasks involving a class imbalance. This is because it is calculated as the harmonic mean between precision and recall, and it uses the following formula:

$$F1 = (2 * \text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

We will use mAP @ 0.5 (using an IoU threshold of 0.5) as our standard benchmark, and use the F1, precision, and recall scores as secondary metrics for our metrics framework. By employing this metrics framework, we will be able to definitively quantify the effectiveness of our adversarial training approach and provide empirical evidence of improved

robustness against various attack methods relevant to aerial imagery object detection in security applications.

IV. METHODOLOGY

Our project's methodology can be broken down into four discrete steps:

1. Using the Ultralytics Python library, download the pretrained YOLOv11 nano model and fine-tune it on the VisDrone2019-DET (detection) dataset.
2. Using the fine-tuned YOLOv11n model's weights, create adversarial images using the Fast Gradient Sign Method and the Carlini-Wagner method for each of the images within the training, test, and validation splits for the VisDrone2019-DET dataset.
3. Using the generated adversarial images within the validation split, evaluate the fine-tuned model's predictions by calculating the following metrics: Precision, Recall, mAP(0.5), and mAP(0.5:0.95).
4. Fine-tune the VisDrone-YOLOv11n model on the images generated by the FGSM and CW methods and finally measure the adversarial-fine-tuned model's robustness with respect to adversarial images.

Furthermore, the computer used to perform each of the calculations had the following specifications:

- OS: Windows 11
- CPU: AMD Ryzen 5600X
- GPU: Nvidia GeForce RTX 3070
- Memory: 32GB DDR4 RAM 3600Mhz CL 14
- Python version 3.11
    - Ultralytics version 8.3.106

A. FINE-TUNING YOLOv11n MODEL

To ensure that the fine-tuned YOLOv11n model's performance on the VisDroneDET-2019 dataset is representative of its true capabilities (e.g. mAP @ 0.5 value of 0.332 in [17]), we used pretrained weights achieving a mAP @ 0.5 value of 0.340 on the VisDrone dataset's validation split. We fine-tuned the pretrained weights of the model for an additional 2 epochs to obtain the performance plots, including F1-Confidence and Precision-Recall curves, automatically generated by the Ultralytics library after training.

The hyperparameters used during training include:

- Downsized image dimensions: (640 x 640)
- Optimizer: auto
- Intersection Over Union threshold for Non-Maximum Suppression: 0.7
- Batch size: 16
- Learning rate: 0.01
- Weight decay: 0.0005

These hyperparameters remained constant for all instances of model training, including adversarial training.

B. CREATING ADVERSARIAL EXAMPLES

For each of the images in the training, test, and validation splits of the VisDrone dataset, we generated adversarial images using the FGSM and Carlini-Wagner methods. As previously mentioned, the FGSM uses the gradient of the model's loss function with respect to the input x and its label y to generate adversarial perturbations/noise. While this process is quite simple to achieve when attacking classification models, it is slightly more complex for adversarial methods. In classification models, each singular image is associated with a singular label. However, for object detection models, there can be numerous objects within an image, where each object is paired with its class label and the coordinates for its bounding box. Thus, object detection models typically use a combination of loss functions to ensure accurate class predictions and precise bounding box selections. These loss functions include cross-entropy loss for classification, Complete Intersection over Union (CIoU) for bounding box loss, and Distribution Focal Loss (DFL) to address class imbalance.
By default, the YOLOv11n model in the Ultralytics library calculates overall loss as a weighted sum of CIoU loss, DFL, and cross-entropy loss such that:

$$L_{overall} = a \cdot L_{DFL} + b \cdot L_{CIoU} + c \cdot L_{CE}$$

For both the FGSM and CW adversarial methods, we chose the above loss function with the values a = 1, b = 1, and c = 2 to slightly prioritize object misclassification over increasing bounding box loss.

Additionally, our implementation of the Carlini-Wagner method is fairly modified in comparison to the original implementation.

Algorithm 1 CW Attack on YOLO Model

Require: Image x, Model f, Training batch B, Constant c, Steps T, Learning rate α

$x \leftarrow \text{clone}(x),\ \text{requires\_grad}(x) \leftarrow \text{True}$
$\delta \leftarrow \text{zeros\_like}(x),\ \text{requires\_grad}(\delta) \leftarrow \text{True}$
$\text{optimizer} \leftarrow \text{Adam}([\delta], \alpha)$
$\text{best\_loss} \leftarrow \infty$
$\text{best\_delta} \leftarrow \delta$
for $t = 1$ to T **do**
  $\hat{x} \leftarrow \text{clip}(x + \delta, 0, 1)$
  $y \leftarrow f.\text{model}(\hat{x})$
  $\ell_{\text{score}} \leftarrow f.\text{model.loss}(B, y)[0].\text{sum}()$
  $\ell_{l2} \leftarrow \text{MSELoss}(\hat{x}, x)$
  $\ell \leftarrow \ell_{l2} + c \cdot \ell_{\text{score}}$
  if $\ell <$ best_loss **then**
    $\text{best\_loss} \leftarrow \ell$
    $\text{best\_delta} \leftarrow \delta$
  end if
  optimizer.zero_grad()
  ℓ.backward()
  optimizer.step()
end for
return $\text{clip}(x + \text{best\_delta}, 0, 1)$

Figure 1. Modified CW algorithm

As shown in Figure 1, the modified CW algorithm still calculates the objective function J as the combination between the L2 norm of the adversarial and input images and the

overall object detection loss of the model. Also, instead of using two tuning parameters α and β, we use a singular tuning parameter *c* as a tradeoff between minimizing image distance and maximizing the loss of the adversarial image. We also use an untargeted approach, while the original Carlini-Wagner method specifically checks if the model misclassifies the image and continuously increases the *c* value until it does.

C. EVALUATING MODEL PERFORMANCE

To determine how each of the adversarial attacks impact the YOLOv11n model's performance, we compare the fine-tuned model's overall mAP@0.5, mAP@0.5:0.95, precision, and recall scores when evaluated on the validation split for the regular, FGSM, and Carlini-Wagner VisDroneDet-2019 datasets. We additionally compare how these scores change for differing epsilon/c values of 0.1, 0.2, 0.3, and 0.4.

D. EVALUATING MODEL ADVERSARIAL ROBUSTNESS

After generating adversarial replicas of each of the images in the training, test, and validation splits of the original VisDrone2019-DET dataset, we further fine-tune the pretrained YOLOv11n weights on the adversarial datasets to create two models. The first model, YOLO-FGSM, is exclusively trained on adversarial images generated using FGSM, and the second model is exclusively trained on adversarial images generated using the C&W method. As a benchmark, we evaluate the performance metrics for each of the adversarially-trained methods on the validation splits of the real, FGSM, and C&W datasets.

Finally, to determine how robust the adversarially-trained models are with respect to new adversarial images, we calculate new adversarial images (FGSM and C&W) for the validation split of the VisDrone dataset and compare the difference in how adversarial images negatively impact the original fine-tuned model vs the adversarial fine-tuned models. More specifically, this is quantified as the percentage change for each of the metrics, including mAP, precision, and recall.

V. RESULTS

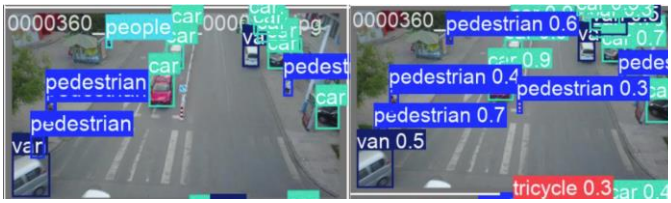

Figure 2. Left: Ground truth aerial image from validation set, annotated with labels. Right: Original fine-tuned model's predictions for bounding box coordinates, class label, and confidence values.

As seen in Figure 2, the original YOLOv11n model fine-tuned on the VisDrone dataset is quite adept at detecting objects and their classes in aerial images. For the most part, each of the objects' classes are predicted correctly, and the model has high confidence in its predictions. Intuitively, the classes associated with the largest bounding boxes (van, car) tend to have higher confidence values, as they are easier to visually differentiate in comparison to smaller objects, like pedestrians and tricycles.

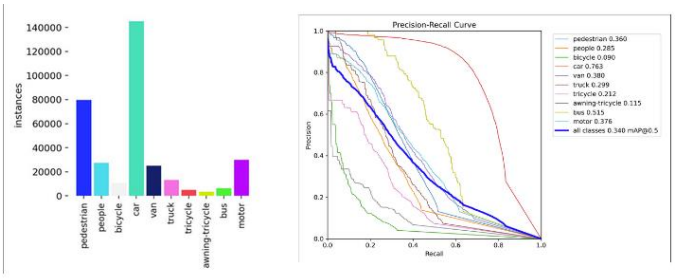

Figure 3. Left: Class label distribution across all the objects in the VisDrone2019-DET training set. Right: Precision-Recall curve from original model's predictions for each of the classes in the VisDrone dataset.

Furthermore, the original YOLOv11n model fine-tuned on the VisDrone dataset has a mAP@0.5 value of 0.34 as shown by the Precision-Recall curve in Figure 3. Generally, the labels with the most representation in the dataset, including cars, pedestrians, and motors, have the highest AP scores of 0.763, 0.360, and 0.376 respectively. However, the "bus" object has the second highest AP score of 0.515 despite its minimal representation in the VisDrone dataset, suggesting that larger objects tend to have higher AP values than their smaller object counterparts.

Attack (eps/c)	mAP@0.5	mAP@0.5:0.95	Precision	Recall
Regular	0.340152	0.196466	0.434741	0.341740
FGSM (eps = 0.1)	0.027831	0.012031	0.033643	0.054278
FGSM (eps = 0.2)	0.011982	0.004566	0.022766	0.005003
FGSM (eps = 0.3)	0.000227	0.000045	0.606554	0.000471
FGSM (eps = 0.4)	2.85×10^{-6}	5.82×10^{-7}	3.00×10^{-1}	3.39×10^{-5}
CW (c = 0.1)	0.114238	0.049800	0.179687	0.128584
CW (c = 0.2)	0.112650	0.047565	0.176507	0.131368
CW (c = 0.3)	0.109986	0.046957	0.201156	0.116052
CW (c = 0.4)	0.110836	0.047280	0.222214	0.112403

Figure 4. Table detailing the mAP@0.5, mAP@0.5:0.95, precision, and recall values for the original fine-tuned YOLO model on the regular, FGSM, and CW validation splits. Each adversarial attack was tested with a range of eps/c values, from [0.1, 0.4] with an increase of 0.1 for each step.

Interestingly, the results of Figure 4 demonstrate that the FGSM adversarial attack outperforms the C&W attack for all mAP, precision, and recall values in terms of percentage decreased from the original model's scores. For example, the original model achieves a mAP@0.5 score of ~0.34 on the validation split, while the FGSM (eps = 0.1) and C&W (c = 0.1) mAP@0.5 scores of ~0.0278 and ~0.11 respectively. In terms of percentage drop, the FGSM attack achieved a ~92% drop in the model's mAP@0.5 score in comparison to the C&W's ~66% drop. While the FGSM attack continues to drastically decrease the YOLOv11n model's overall performance as its epsilon value increases,

the CW method's performance seems to stagnate despite increases in its *c* tuning parameter. This likely indicates that 1) increasing the *c* parameter does not relevantly impact attack performance and 2) our modified C&W algorithm does not perfectly mimic the algorithm described in the original paper.

Figure 5. Left: Adversarial image generated by FGSM with epsilon = 0.1. Right: Adversarial image generated by modified C&W algorithm with c = 0.1, training steps = 0.07, learning rate = 0.05.

Furthermore, as shown in Figure 5, the adversarial image generated by the FGSM attack prevents the YOLOv11n model from generating any predictions on the objects in the image, despite various cars and pedestrians being identifiable by the human eye. The adversarial image generated by our modified C&W attack is visibly slightly noisier, and it instead causes the model to falsely predict part of the red car as a pedestrian, while also correctly predicting the van in the image with a confidence of 0.6. Overall, both models serve to reduce the overall objects detected by the model, while the modified C&W attack causes the model to additionally misclassify an object.

Model (Attack Eval)	mAP@0.5	mAP@0.5:0.95	Precision	Recall
FGSM_ADV (Regular)	0.252497	0.132417	0.362583	0.266214
FGSM_ADV (FGSM, eps=0.1)	0.105810	0.055632	0.169709	0.144581
FGSM_ADV (CW, c=0.1)	0.246956	0.127041	0.452329	0.239991
CW_ADV (Regular)	0.247207	0.127419	0.364752	0.261190
CW_ADV (FGSM, eps=0.1)	0.102117	0.055769	0.156865	0.140300
CW_ADV (CW, c=0.1)	0.541405	0.282877	0.645073	0.503025

Figure 5. Table comparing the performance of two fine-tuned models, one trained on FGSM examples and the other trained on C&W examples, over three different validation splits. The first validation split contains regular images from the VisDrone dataset, the second contains FGSM images with epsilon = 0.1, and the third contains modified C&W images with c = 0.1.

According to Figure 5, both the FGSM_ADV and CW_ADV models, which were fine-tuned on the pretrained weights [18] of the YOLOv11n model with adversarial examples, experience a drop in the mAP@0.5 value of approximately 0.09 in comparison to the original model shown in Figure 4. However, when exposed to both FGSM and modified C&W images, the performance of each of the adversarially-trained models substantially improves in comparison to the values displayed in Figure 4. For example, the unprotected model achieves a mAP@0.5 score of ~0.028 when exposed to FGSM-generated images with epsilon = 0.1, while both protected models achieve a mAP@0.5 score of ~0.10 on the same images. Interestingly, the models fine-tuned with adversarial examples also are robust with respect to the adversarial methods they weren't exposed, indicating that the adversarial protection is transferable when exposed to other adversarial methods. Finally, the CW_ADV's exceedingly high metrics when exposed to a CW validation-split suggests potential overfitting of the model.

Model (Attack Eval)	mAP@0.5	mAP@0.5:0.95	Precision	Recall
FGSM_ADV (Clean)	0.252497	0.132417	0.362583	0.266214
FGSM_ADV (FGSM, eps=0.1)	0.065888	0.033161	0.108802	0.094636
FGSM_ADV (CW, c=0.1)	0.412218	0.217973	0.570834	0.379980
CW_ADV (Clean)	0.247207	0.127419	0.364752	0.261190
CW_ADV (FGSM, eps=0.1)	0.060067	0.031424	0.107990	0.083493
CW_ADV (CW, c=0.1)	0.680371	0.375808	0.791908	0.606575

Figure 6. Performance of adversarially-trained models on adversarial images generated using adversarial pretrained weights. This table differs from Figure 5, as the adversarial examples in Figure 5 were generated using the original, unprotected model's pretrained weights.

Finally, as a sanity check, we also calculated new adversarial examples using the adversarially-trained models' weights, instead of the adversarial examples generated using the original model's weights. Almost all of the results remain consistent across Figures 5 and 6, suggesting the adversarially-trained models are genuinely more robust to adversarial examples than the unprotected model. Note that the CW_ADV model's metrics when exposed to CW-generated images continue to be unnaturally high, suggesting further overfitting.

VI. CONCLUSION

For our project, we proposed the method of fine-tuning the pretrained weights of a YOLOv11n-based VisDrone object detector on adversarial images to evaluate the model's overall robustness with respect to adversarial attacks. Using the FGSM attack and a modified version of the C&W attack, we found that fine-tuning the YOLOv11n model on adversarial attacks did in fact improve its robustness. When the original, unprotected model was exposed to FGSM and C&W attacks with eps/c values of 0.1, it respectively achieved dismal mAP@0.5 scores of ~0.0278 and ~0.11 in comparison to its base mAP@0.5 score of ~0.34. However, after fine-tuning two different models on exclusively FGSM and C&W images, both models achieved an approximate mAP@0.5 score of ~0.25 on a validation split consisting of unprocessed images, demonstrating a massive increase in robustness. However, our modified C&W implementation did not accurately represent the power of the original implementation, as it failed to defeat the FGSM method's percent drop in mAP score. Finally, we suspect that the inflated results of the fine-tuned C&W model's predictions when evaluating C&W generated images is a result of some sort of model parameter overfitting during the validation phase of training.

For future work, we suggest testing our adversarially-trained models on different attack methods to determine if they are truly more robust with respect to adversarial attacks than the original model. We also encourage evaluating our methods on additional datasets besides the VisDrone dataset, as different datasets may incur different results.

VII. ALLOCATION OF WORK

- Luke Unterman:
  - Conducted research on adversarial examples, object detection architectures, aerial imagery datasets, and adversarial defense strategies
  - Synthesized research conducted by other team members
  - Analyzed performance of adversarial methods on fine-tuned YOLOv11n dataset
- Jimmy Le
  - Conducted research on making DNNs robust with respect to adversarial attacks and Projected Gradient Method
  - Synthesized research via contribution to literature review
  - Created class paper presentation slides
- Jannat Lnu
  - Conducted research on understanding the metrics to use to quantify performance
  - Wrote code for FGSM attack on YOLOv11n model
- My Nguyen
  - Conducted research on adversarial attack and defense methodologies and orientation robust object detectors
  - Wrote code for modified C&W attack on YOLOv11n model

VIII. REFERENCES

(1) Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv.org*. Published online 2019.

(2) Zhao, L.; Liu, T.; Xie, S.; Huang, H.; Qi, J. OrtDet: An Orientation Robust Detector via Transformer for Object Detection in Aerial Images. Remote Sens. 2022, 14, 6329. https://doi.org/10.3390/ rs14246329

(3) Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples (Version 3). arXiv. https://doi.org/10.48550/ARXIV.1412.6572

(4) Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks (Version 4). arXiv. https://doi.org/10.48550/ARXIV.1312.6199

(5) Bengio, Y. (2009). Learning Deep Architectures for AI. In Foundations and Trends® in Machine Learning (Vol. 2, Issue 1, pp. 1–127). Now Publishers. https://doi.org/10.1561/2200000006

(6) R. Padilla, S. L. Netto and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niteroi, Brazil, 2020, pp. 237-242, doi: 10.1109/IWSSIP48289.2020.9145130.

(7) Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2021). A survey on adversarial attacks and defences. In CAAI Transactions on Intelligence Technology (Vol. 6, Issue 1, pp. 25–45). Institution of Engineering and Technology (IET). https://doi.org/10.1049/cit2.12028

(8) Ding, J., Xue, N., Xia, G.-S., Bai, X., Yang, W., Yang, M. Y., Belongie, S., Luo, J., Datcu, M., Pelillo, M., & Zhang, L. (2021). Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges. arXiv. https://doi.org/10.48550/ARXIV.2102.12219

(9) Zhu, P., Wen, L., Bian, X., Ling, H., & Hu, Q. (2018). Vision Meets Drones: A Challenge (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1804.07437

(10) Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014

(11) Carlini, N., & Wagner, D. (2016). Towards Evaluating the Robustness of Neural Networks (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1608.04644

(12) D. Du et al., "The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking," 2018, arXiv. doi: 10.48550/ARXIV.1804.00518.

(13) G.-S. Xia et al., "DOTA: A Large-scale Dataset for Object Detection in Aerial Images," arXiv, 2017, doi: 10.48550/ARXIV.1711.10398.

(14) CVproject. Vehicle detection dataset. https://universe.roboflow.com/cvproject-y6bf4/vehicle-detection-gr77r , dec 2022. visited on 2024-02-07.

(15) Pak Vehicles. Pak vehicles dataset. https://universe.roboflow.com/pak-vehicles/pak-vehicles , may 2024. visited on 2024-05-07.

(16) N. Jegham, C. Y. Koh, M. Abdelatti, and A. Hendawi, "YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions," 2024, arXiv. doi: 10.48550/ARXIV.2411.00201.

(17) G. Ren, J. Wu, and W. Wang, "Research on UAV Target Detection Based on Improved YOLOv11," Journal of Computer and Communications, vol. 13, no. 03. Scientific Research Publishing, Inc., pp. 74–85, 2025. doi: 10.4236/jcc.2025.133006.

(18) Erbayat, "yolov11n-visdrone," erbayat/yolov11n-visdrone at main, https://huggingface.co/erbayat/yolov11n-visdrone/tree/main

(19) A. Ilyas, I. Rahmani, S. Imran, T. S. S. Hashmi, and M. N. Yousaf, "Deep Learning Application for Vehicle Detection through Surveillance Drones," 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET. IEEE, pp. 1–6, Jul. 25, 2024. doi: 10.1109/icecet61485.2024.10698500.