TARGETED ADVERSARIAL OBJECTNESS GRADIENT ATTACKS PROJECT- REPORT

submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

TADISETTI HEMA SRI CS21B047

Supervisor(s)

Dr. Chalavadi Vishnu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

MAY 2025

DECLARATION

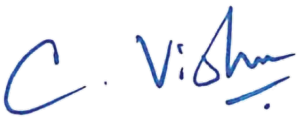
I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/-data/fact/source in my submission to the best of my knowledge. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Tirupati
Date: 12-05-2025

Signature
Tadisetti Hema Sri
CS21B047

BONA FIDE CERTIFICATE

This is to certify that the report titled **TARGETED ADVERSARIAL OBJECTNESS GRADIENT ATTACKS: PROJECT REPORT**, submitted by **Tadisetti Hema Sri**, to the Indian Institute of Technology, Tirupati, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the project work done by him [or her] under our or [my] supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.



Place: Tirupati
Date: 12-05-2025

Dr. Chalavadi Vishnu
Guide
Assistant Professor
Department of Computer Science and Engineering
IIT Tirupati - 517501

ACKNOWLEDGMENTS

I express my sincere gratitude to my guide, Dr. C. Vishnu, for his invaluable guidance, mentorship, and constant encouragement throughout this project. His insights and expertise have been instrumental in shaping the direction and outcome of the work. I also extend my heartfelt thanks to the Department of Computer Science and Engineering, Indian Institute of Technology Tirupati, for providing a supportive academic environment and the necessary resources to carry out this research. Finally, I deeply appreciative the contributions of researchers whose works I have referenced in my study. Their research and knowledge have greatly enriched my study.

ABSTRACT

KEYWORDS: object detection, adversarial attacks, deep neural networks, targeted-object gradient attacks, yolov3.

Deep learning models, while highly effective in tasks like object detection, are vulnerable to adversarial attacks—specially crafted inputs designed to deceive these models. This report focuses on Target-Oriented Gradient (TOG) attacks, a class of adversarial techniques that disrupt object detection by causing effects like object vanishing. Using a YOLOv3 Darknet model trained on the visDrone dataset to detect objects, I applied TOG attacks to generate adversarial images and then passed them through the object detection model and evaluated their impact on detection accuracy. The results demonstrate that these attacks significantly compromise the performance of object detectors, highlighting critical vulnerabilities. This work emphasizes the need to develop robust defense strategies to ensure the reliability and safety of deep learning systems, especially in safety-critical applications like autonomous driving, identifying people from flooded areas etc.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

TOG TARGETED ADVERSARIAL OBJECTNESS GRADIENT ATTACKS

NOTATION

- x'_t: The updated adversarial example at iteration t.
- x'_{t-1}: The adversarial example from the previous iteration $(t-1)$.
- $\prod_{x,\varepsilon}$: Projection operator that ensures the perturbed image remains within an ε-bounded neighborhood of the original input x.
- α_{TOG}: A step size (learning rate) parameter associated with the TOG (Target-Oriented Gradient) method.
- Γ: A transformation or normalization function (e.g., ℓ_2 or ℓ_∞ norm) applied to the gradient.
- $\frac{\partial \mathscr{L}_{\text{obj}}\left(x'_{t-1};\emptyset,\mathscr{W}\right)}{\partial x'_{t-1}}$: The gradient of the object-level loss $\mathscr{L}_{\text{obj}}$ with respect to the adversarial input x'_{t-1}.
- $\frac{\partial \mathscr{L}\left(x'_{t-1};\hat{\mathscr{O}}(x),\mathscr{W}\right)}{\partial x'_{t-1}}$: The gradient of the loss function $\mathscr{L}$ evaluated at the predicted object(s) $\hat{\mathscr{O}}(x)$.
- $\frac{\partial \mathscr{L}\left(x'_{t-1};\mathscr{O}^*(x),\mathscr{W}\right)}{\partial x'_{t-1}}$: The gradient of the loss function $\mathscr{L}$ evaluated at the targeted object(s) $\mathscr{O}^*(x)$.
- $\mathscr{L}$: The general loss function used for training or adversarial optimization.
- $\mathscr{L}_{\text{obj}}$: The object-specific loss function (used to remove or manipulate object detection).
- $\hat{\mathscr{O}}(x)$: The predicted object(s) in the input image x.
- $\mathscr{O}^*(x)$: The target object(s) specified for a targeted attack on x.
- $\mathscr{W}$: The set of model parameters or weights.
- $\emptyset$: Represents the absence of objects (used for object removal attacks).

CHAPTER 1

INTRODUCTION

Deep neural networks (DNNs) have been highly successful in object detection, but are vulnerable to adversarial attacks. These attacks introduce small imperceptible changes in an image to manipulate the prediction of the model. This can have serious consequences, especially in critical applications like self-driving cars and healthcare. My work focuses on understanding and implementing such attacks on object detection models to analyze their weaknesses and potential risks. In this project, I explored adversarial attacks on object detection models, specifically applying TOG attacks such as TOG-object vanishing, fabrication, and mislabeling attacks. The objective is to generate adversarial images (modify input images) such that the object detection model fails to detect them correctly.

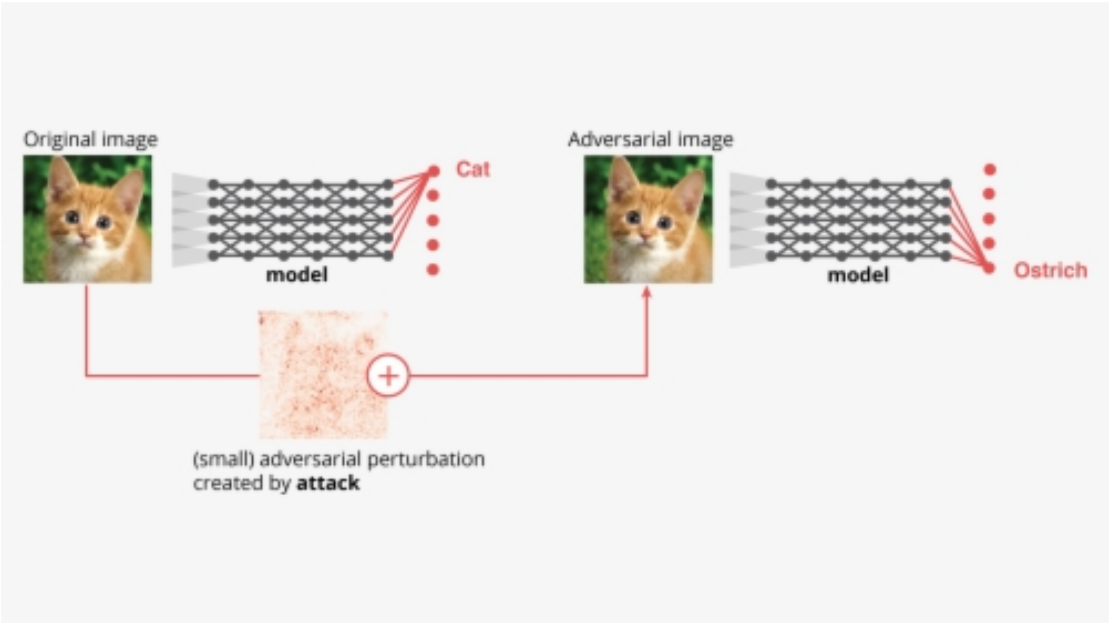


Figure 1.1: Example for Adversarial Attack

CHAPTER 2

Work Done Until Previous Stage

Despite their success, DNNs are vulnerable to adversarial attacks, which exploit model weaknesses to mislead predictions. These attacks raise concerns about the reliability of DNNs in critical systems such as self-driving vehicles, healthcare, and security systems.

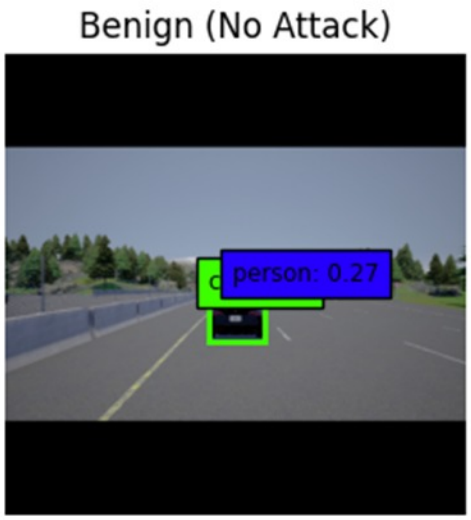
2.0.1 Understanding TOG Adversarial Attacks

I explored the TOG (Targeted Objectness Gradient) attack, which consists of three main attack types:

- **Object-Vanishing Attack:** This makes objects disappear from the detection results, meaning the model fails to recognize them.
- **Object-Fabrication Attack:** This creates false objects in the detection results, making the model detect things that do not actually exist.
- **Object-Mislabeling Attack:** This changes the labels of detected objects, making the model misclassify them.

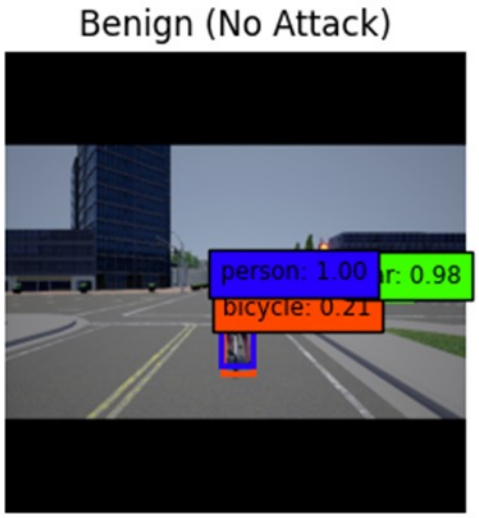
2.0.2 Implementation

I implemented the TOG-vanishing attack on a YOLOv3 object detection model with a MobileNetV1 backbone, using the CARLA dataset. First, I trained the YOLOv3 model to detect cars in images. Then, I applied adversarial perturbations to the input images using the TOG-vanishing attack, which modified the images so that the model failed to detect any objects. This attack works by adjusting the image pixels in a way that minimizes the object detection confidence scores, effectively making objects disappear from the model's predictions. To evaluate the impact of the attack, I compared the detection results before and after applying the adversarial perturbations. The model successfully detected objects in the original images, but after the attack, most objects were no longer visible in the predictions. This confirmed that the TOG-vanishing attack was effective. I also visualized the attack's impact using histograms, showing how the detection confidence scores dropped significantly after the perturbations were applied.

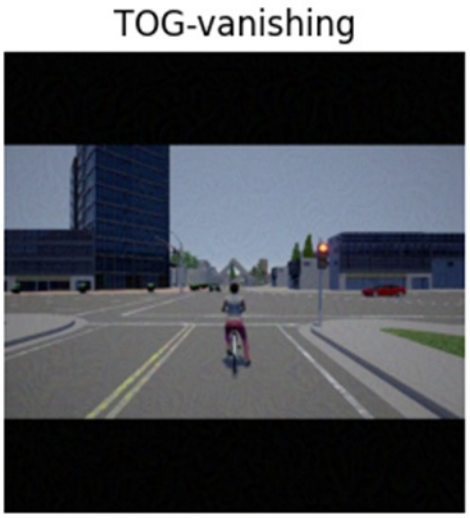


(a) Original 1

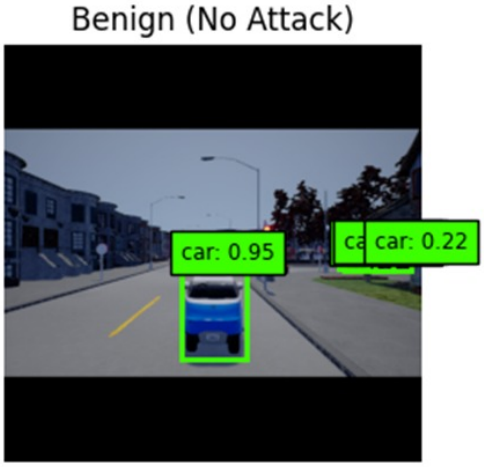
(b) Adversarial 1



(c) Original 2



(d) Adversarial 2



(e) Original 3

(f) Adversarial 3

Figure 2.1: Comparison of original and adversarial images for some examples

CHAPTER 3

Literature Review

Despite their success, DNNs are vulnerable to adversarial attacks, which exploit model weaknesses to mislead predictions. These attacks raise concerns about the reliability of DNNs in critical systems such as self-driving vehicles, healthcare, and security systems.

3.1 Targeted Adversarial Attacks on Object Detection Systems

The **TOG (Targeted Objectness Gradient)** suite of adversarial attacks highlights three major types of vulnerabilities in object detection systems: TOG attacks are effective across different algorithms like YOLOv3, SSD, and Faster R-CNN and datasets such as PASCAL VOC, MS COCO, and INRIA. These attacks work by optimizing perturbations either for specific inputs, universally for any input, or through physical adversarial patches that deceive detectors in real-world scenarios. For example, **TOG-vanishing** reduces the number of detected objects to nearly zero, while **TOG-fabrication** introduces a surplus of false positives.

- Studies reveal significant performance degradation of victim models under adversarial attacks. For instance, YOLOv3's mean Average Precision (mAP) drops from 71.84% to as low as 0.17% under targeted attacks. The transferability of adversarial attacks also depends on the similarity between models. Diverse architectures show greater resistance to black-box attacks.
- Real-world implementation of attacks, such as adversarial patches, demonstrates practical risks in critical systems like autonomous vehicles. However, creating universal defenses remains challenging due to the evolving nature of attack strategies.
- **Emerging Insights and Open Challenges**

While adversarial attacks expose vulnerabilities in DNNs, they also inspire advancements in defense mechanisms:

- **Adversarial training** improves model robustness by incorporating adversarial examples during training.
- **Ensemble learning** leverages model diversity to resist adversarial perturbations.
- **Feature-level defenses** aim to filter adversarial features before classification.

3.2 Object detection Model

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds or images.

- The YOLO machine learning algorithm uses features learned by a Deep Convolutional Neural Network to detect objects located in an image.
- As typical for object detectors, the features learned by the convolutional layers are passed onto a classifier which makes the detection prediction. In YOLO, the prediction is based on a convolutional layer that uses 1×1 convolutions. This means that the size of the prediction map is exactly the size of the feature map before it.
- The YOLOv3 algorithm first separates an image into a grid. Each grid cell predicts some number of bounding boxes (sometimes referred to as anchor boxes) around objects that score highly with the aforementioned predefined classes.
- Each bounding box has a respective confidence score of how accurate it assumes that prediction should be. Only one object is identified per bounding box. The bounding boxes are generated by clustering the dimensions of the ground truth boxes from the original dataset to find the most common shapes and sizes.
- YOLO can perform classification and bounding box regression at the same time. Thus, ensuring efficient and accurate predictions of the predicted bounding boxes.

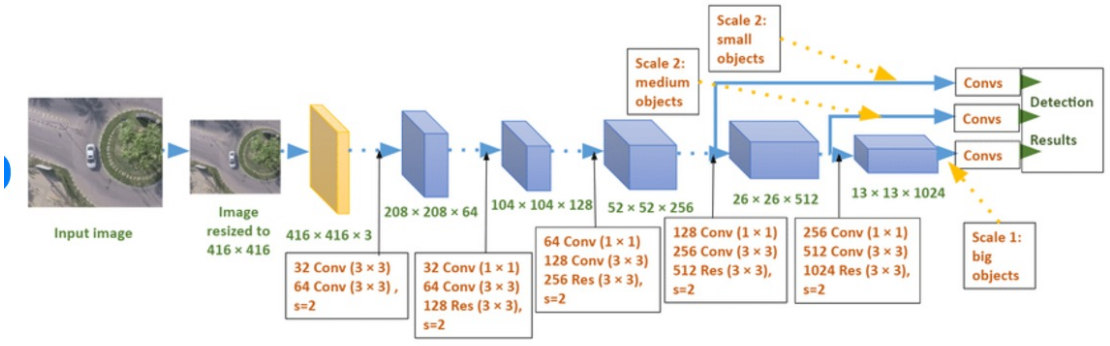


Figure 3.1: YOLOv3 architecture

3.3 Darknet-53

Darknet-53 is a 53-layer convolutional neural network used as the feature extractor for YOLOv3. It is optimized for speed and accuracy, combining residual connections and successive 3x3 and 1x1 convolutional layers. Pretrained on ImageNet, it helps the detector learn robust features from visual data.

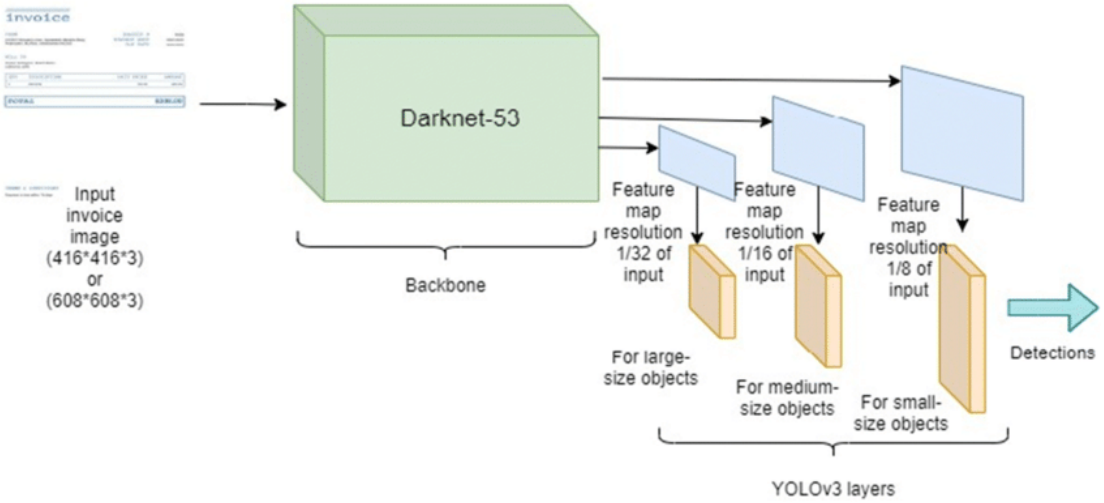


Figure 3.2: Yolo3 architecture with Darknet-53 backbone

3.4 Keras(with TensorFlow backend)

Framework: Keras (with TensorFlow backend) Keras is a high-level neural networks API that allows rapid prototyping and model training. Used with TensorFlow as the backend, it supports efficient GPU computation, multi-GPU training, and offers an intuitive interface for deep learning workflows such as training YOLOv3 models with custom datasets.

CHAPTER 4

Current Work

I trained a yolov3 model on the VisDrone dataset using keras-yolo3 implementation, which is a Keras-based reimplementation of the original YOLOv3 model with TensorFlow backend, and then applying Targeted Adversarial Objectness Gradient (TOG) attacks to evaluate the model's robustness. The workflow encompasses dataset preparation, victim detector training and then implementation of TOG attacks.

4.1 Adversarial attack on object detection model

4.1.1 Applying TOG (Target-Oriented Gradient) Attacks

After obtaining the trained YOLOv3 weights, the next step is to apply the TOG attacks. The TOG attacks are adversarial attack methods aimed at reducing object detection accuracy by introducing perturbations that specifically fool the detector.

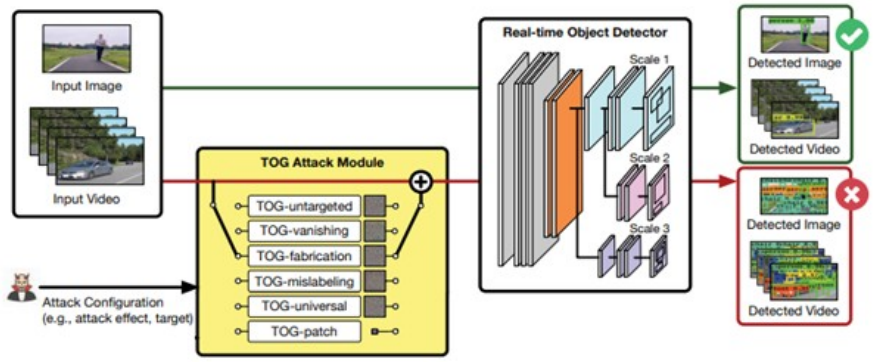


Figure 4.1: An illustration of the adversarial attacks using TOG

The attack implementation will follow these steps:

1. Load the trained YOLOv3 model with the generated weights.
2. Generate adversarial examples by computing perturbations using the TOG framework(apply the perturbations to input images)
3. Generate different adversarial images for different TOG attacks and then pass the images to the victim detector.
4. Observe changes in detection performance.

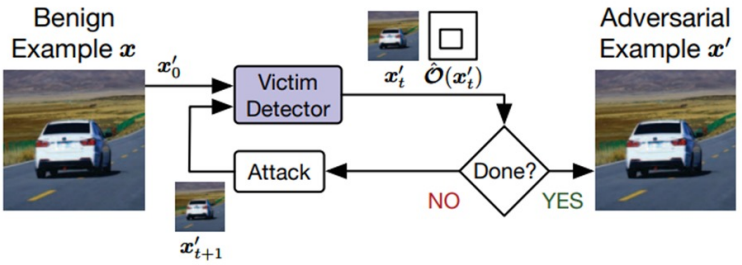


Figure 4.2: The iterative attack strategy adopted by TOG to tailor an adversarial perturbation for each given input.

5. Measure the impact by evaluating the drop in mAP, precision, and recall.

- **TOG-Vanishing Attack:** Make real objects disappear from the detector's view. Used in sensitive applications like medical imaging, where missing a tumor (false negative) is dangerous. This attack focuses only on objectness (Lobj)—the part of the model that decides whether an object exists. The idea is to push the input image so that the detector thinks 'there's nothing here.'

$$x'_t = \prod_{x,\varepsilon}\left[x'_{t-1} - \alpha_{\text{TOG}}\Gamma\left(\frac{\partial \mathscr{L}_{\text{obj}}\left(x'_{t-1};\emptyset,\mathscr{W}\right)}{\partial x'_{t-1}}\right)\right]$$

- **TOG-Fabrication Attack:** Create fake objects that don't actually exist. This attack also uses objectness (Lobj) but pushes it in the opposite direction. It adds false positives—extra boxes that detect things that aren't there. The detector starts hallucinating objects.

$$x'_t = \prod_{x,\varepsilon}\left[x'_{t-1} + \alpha_{\text{TOG}}\Gamma\left(\frac{\partial \mathscr{L}_{\text{obj}}(x'_{t-1};\emptyset,\mathscr{W})}{\partial x'_{t-1}}\right)\right]$$

- **TOG-Mislabeling Attack:** Change the label of detected objects. The detector still sees the correct objects and boxes, but their labels are wrong. For example, a stop sign might be labeled as an umbrella. This is done by tweaking only the classification part of the loss (Lclass), keeping the rest unchanged. You can even force it to pick the most likely wrong label or the least likely one.

$$x'_t = \prod_{x,\varepsilon}\left[x'_{t-1} - \alpha_{\text{TOG}}\Gamma\left(\frac{\partial \mathscr{L}(x'_{t-1};\mathscr{O}^*(x),\mathscr{W})}{\partial x'_{t-1}}\right)\right]$$

- **TOG-Untargeted Attack:** Just confuse the object detector in any way. This attack doesn't care what mistake the detector makes—as long as it makes one. The attack uses gradients from all parts of the detection model: objectness (Lobj), bounding box (Lbbox), and class label (Lclass). It's like throwing the detector off-balance with noise in all directions

$$x'_t = \prod_{x,\varepsilon}\left[x'_{t-1} + \alpha_{\text{TOG}}\Gamma\left(\frac{\partial \mathscr{L}(x'_{t-1};\hat{\mathscr{O}}(x),\mathscr{W})}{\partial x'_{t-1}}\right)\right]$$

4.1.2 Training a YOLOv3 model

- **Dataset Selection and Preprocessing**: For training my YOLOv3 model, I selected the VisDrone dataset, which consists of 10 object classes. The dataset includes aerial images with annotations for different object categories, making it suitable for real-world applications such as surveillance and autonomous navigation. The class labels for this dataset are:
- **Pedestrian-related**: pedestrian, people
- **Vehicle-related**: bicycle, car, van, truck, tricycle, awning-tricycle, bus, motor

Initially, I considered using a pre-trained YOLOv3 model trained on the Pascal VOC dataset. However, Pascal VOC contains 20 classes with different object categories such as animals, vehicles, and indoor objects. Since the classes do not align with VisDrone's object categories, using a pre-trained model directly would not yield optimal results. Hence, it was necessary to train YOLOv3 from scratch on the VisDrone dataset. To prepare the data set for the Darknet YOLOv3 training, I performed the following steps:

- The dataset was downloaded and extracted.
- **Dataset Preparation**: The VisDrone dataset annotations were converted to the format required by YOLOv3: `image_path x_min,y_min,x_max,y_max,class_id ...`
- Each line in the annotation file corresponds to one image, followed by its bounding box labels.
- **Model Conversion**: Pretrained Darknet weights (`yolov3.weights`) were converted to Keras `.h5` format using the provided `convert.py` script:
- `wget https://pjreddie.com/media/files/yolov3.weights`
- `python convert.py yolov3.cfg yolov3.weights model_data/yolo.h5`
- **Training Procedure**: Initial training was done with frozen layers (backbone frozen, head trainable) using the pretrained `yolo.h5` weights for transfer learning. Fine-tuning was then performed by unfreezing all layers for full training. The model was trained using `train.py` with batch size, learning rate, and epoch settings optimized for the dataset.
- **Custom Configuration**: Anchor boxes were kept as default or redefined using k-means clustering (if needed). VisDrone class labels were used to update `classes.txt`.
- Training was monitored using loss metrics and validation mAP on a hold-out set.
- The trained model was evaluated for both image and video inference modes, with the best performing weights saved for further adversarial testing.

Challenges Faced During Training: The size of the VisDrone data set is significantly large, making the training process computationally expensive and time consuming. Due to limited GPU resources, training took an extended period. So, I considered only a subpart of the data, performed training and then implemented attacks, as the main objective to perform adversarial attacks.

CHAPTER 5

Results

When there is no attack the objects are detected correctly. TOG-vanishing makes the objects to not detect by updating the original image in iterative steps with perturbation such that object detection loss is minimized. TOG-fabrication attack makes the model to detect many false objects that are not present in image and TOG-mislabeling makes the model to detect object with wrong labels. TOG untargeted causes detection errors(false positives/ negatives), significant disruption of the original detections.

5.0.1 TOG-Attacks

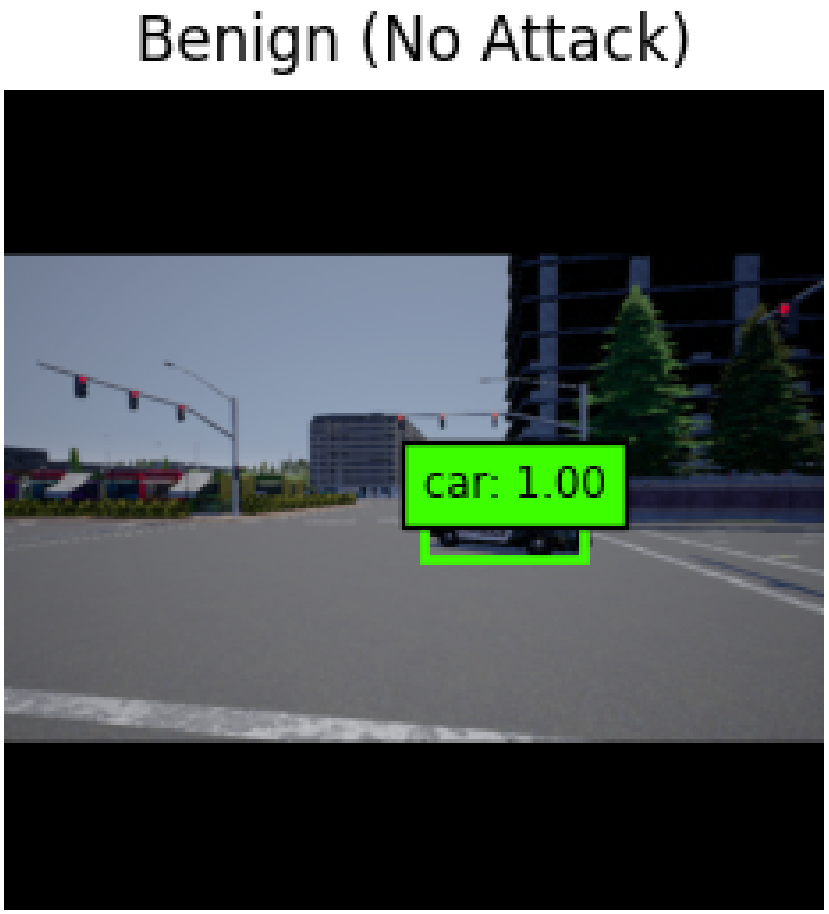


Figure 5.1: Original Detections

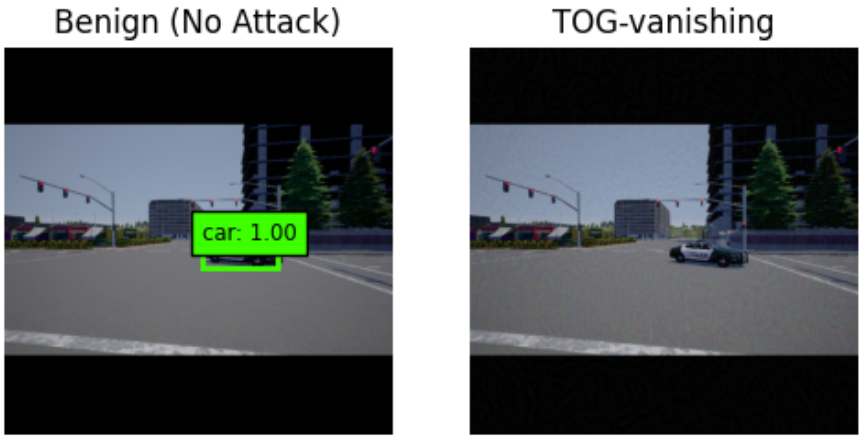


Figure 5.2: TOG Vanishing

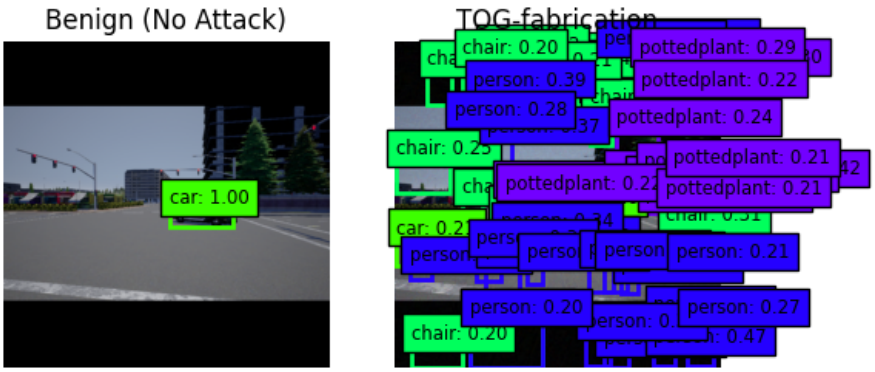


Figure 5.3: TOG-Fabrication

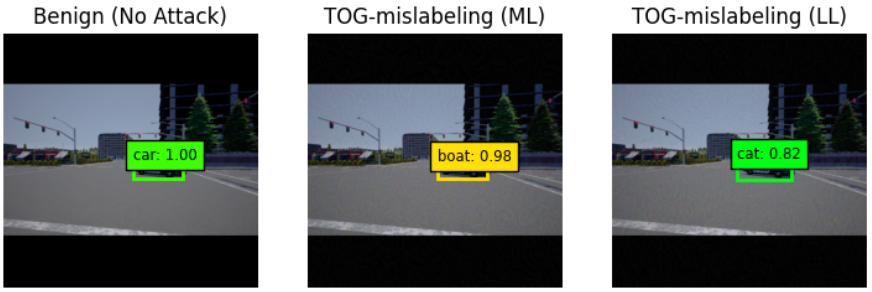


Figure 5.4: TOG-Mislabeling

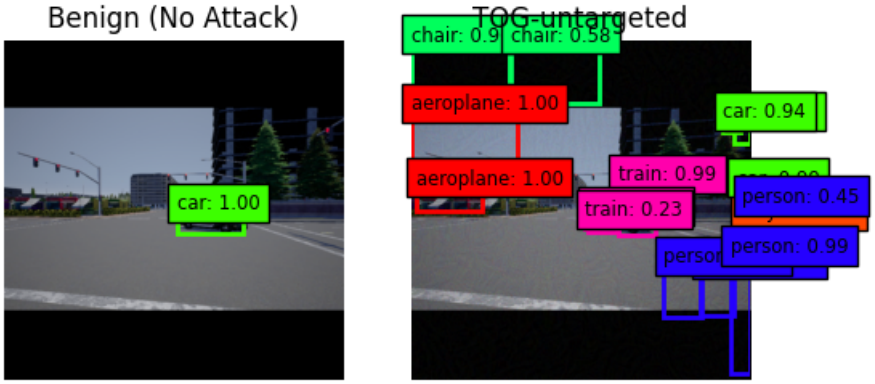


Figure 5.5: TOG-Untargeted

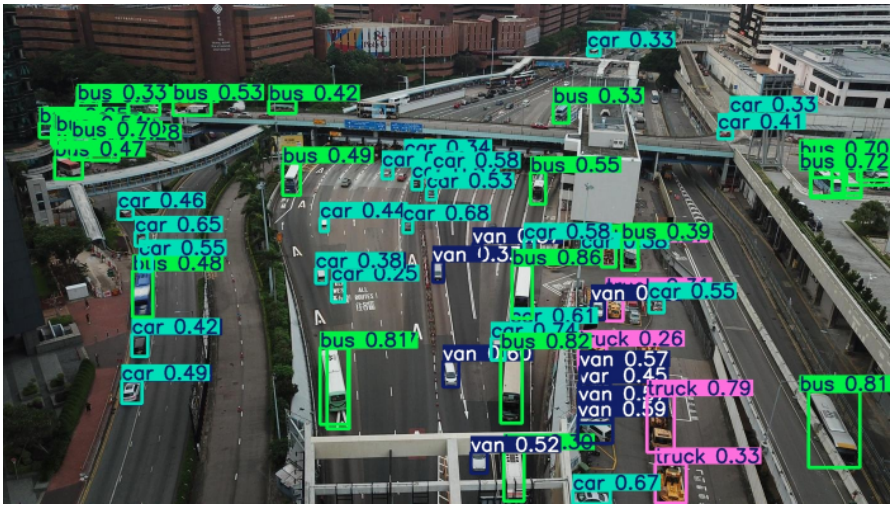


Figure 5.6: Object Detection

Detection of objects using trained model

The attacks were successful as in vanishing the objects are not detected, for fabrication false objects are detected, for mislabeling objects are detected wrong and untarget fools the victim detector to give incorrect results of any form, such as having objects vanished, fabricated, or mislabeled randomly.

CHAPTER 6

SUMMARY AND CONCLUSION

In this project, I successfully implemented and trained a YOLOv3 object detection model using the Keras-YOLO3 framework on the VisDrone dataset comprising 10 classes. The training process included dataset preparation, annotation formatting, model conversion, and iterative fine-tuning using transfer learning. After achieving satisfactory detection accuracy, we evaluated the robustness of the model under adversarial attacks using targeted and untargeted gradient-based perturbations. Our findings highlight both the strength of YOLOv3 in drone-based surveillance applications and its vulnerability to adversarial manipulation, emphasizing the importance of incorporating defense mechanisms for real-world deployment. The findings reveal that even advanced detection models can be misled by subtle, malicious perturbations, rendering objects undetectable and compromising the system's reliability. The results underline the need to develop and implement robust defensive mechanisms. Approaches such as adversarial training, gradient masking, and ensemble-based methods could mitigate these vulnerabilities. Furthermore, combining multiple defensive strategies and testing under real-world conditions will enhance the resilience of object detection systems against increasingly sophisticated adversarial threats. This research highlights the importance of understanding and addressing adversarial vulnerabilities in deep learning models. By fostering a deeper exploration into attack and defensive strategies, this work lays the foundation for creating more secure and reliable object detection systems, ensuring their safe deployment in mission critical applications.

REFERENCES

[1] Ka-Ho Chow, Ling Liu, Margaret Loper, Juhyun Bae, Mehmet Emre Gursoy, Stacey Truex, Wenqi Wei, Yanzhao Wu, *Adversarial Objectness Gradient Attacks in Real-time Object Detection Systems*, 2020. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9325397

[2] Disl Lab, *TOG - Towards Adversarial Robustness*, 2020. Available at: https://github.com/git-disl/TOG/tree/master

[3] Adamdad, *Keras-YOLOv3-Mobilenet*, 2020. Available at: https://github.com/Adamdad/keras-YOLOv3-mobilenet/tree/master

[4] qqwweee, *Keras-yolo3*, 2018. Available at: https://github.com/qqwweee/keras-yolo3

[5] adityatandon, *VisDrone2YOLO*, 2023. Available at: https://github.com/adityatandon/VisDrone2YOLO/tree/main