KI-1-AI-Sec—Adversarial-Attacks-in-ML-based Remote-Sensing-Applications
CS 4850 - Section 02 – Fall 2023 2023-11-26
Sharon Perry
ki-1-ai-sec.github.io
github.com/KI-1-AI-Sec/adversarial-yolo

 Kyle Bratcher	 Zeke Lipscomb (Team Lead)
 Babrah Koroma	 Matthew Krupczak

Team Members:

Name	Cell Phone	Email
Dr. Kazi Aminul Islam (Client)	470.578.6005 (Office phone)	kislam4@kennesaw.edu
Kyle Bratcher	678.521.9164	kylesbratcher@gmail.com
Zeke Lipscomb (Team Lead)	770.408.8145	Zekeclass@gmail.com
Babrah Koroma	404.952.0637	Babrah.6766@gmail.com
Matthew Krupczak	678.778.7335	matthew@krupczak.org
Sharon Perry (Advisor)	770.329.3895	Sperry46@kennesaw.edu

Table of Contents:

Introduction **3**
Background/Significance **4**
Goals and Objectives **6**
Methodologies **6**
Challenges and Risks **8**
Preliminary Results **10**
Project Management and Team Dynamics **10**
Conclusion and Future Direction **11**
Reference **13**
Supplementary Technical Documentation **14**
Project Onset 14
Initial reproduction of experiment from research paper 15
Modifications for remote sensing applications 16
Results from original adaptation experiment 18
Root cause analysis and discovery 19
Conclusion and technical point of contact information 20

Introduction

Adapting Advanced AI for Enhanced Object Detection Security

In the rapidly evolving domain of artificial intelligence (AI) and machine learning, our Senior Project group, KI-AI-Sec, in CS 4850 S03 for Fall 2023 at Kennesaw State University (KSU), takes on the challenge of enhancing the security of AI-driven object detection systems. Comprising team members Babrah Koroma, Zeke Lipscomb, Kyle Bratcher, and Matthew Krupczak, under the guidance of Dr. Kazi Aminul Islam and with support from Sumaiya Tasneem, our project navigates the complexities of adapting adversarial AI techniques to newer generation object detection frameworks. Mrs. Sharon Perry, as the coordinator of the Senior Project program, oversees our progress and provides invaluable academic support.

The advent of massively parallel computer processors and advanced machine learning algorithms has revolutionized the way AI frameworks understand and interpret diverse information sources. These technologies empower computers to recognize and process non-rigid patterns, such as handwriting, speech, written words, and notably, the identification of objects in images or video feeds. Our daily lives are increasingly intertwined with these AI systems, as they find applications in various sectors ranging from financial markets to physical security systems, automotive driver assistance, and human-computer interfaces.

Our project zeroes in on the realm of object detection for remote sensing applications, a critical component in security-sensitive systems. These systems are pivotal in scenarios ranging from trespasser recognition in security cameras to the analysis of satellite imagery for military and commercial purposes. However, despite their widespread utility, the security vulnerabilities inherent in these systems remain underexplored and pose significant risks.

The crux of our research focuses on addressing these vulnerabilities. We aim to adapt and refine an existing adversarial AI attack pattern generator, initially designed to work with older AI models, for compatibility with a newer generation object detection framework, namely YOLOv8. The goal is to evaluate whether these advanced AI frameworks remain susceptible to adversarial attacks, a phenomenon where malicious entities manipulate AI models to fail in correctly identifying objects.

Our project is not just about identifying vulnerabilities but also about contributing to the development of more secure AI systems. The insights and outcomes from our project are intended to assist Dr. Islam's research group in creating a defense mechanism that can safeguard remote sensing applications from such adversarial attacks. This endeavor places our team at the forefront of research that aims to bolster the security of AI systems against emerging threats, ensuring their reliability and integrity in critical applications.

Background/Significance

The emergence and advancement of Convolutional Neural Networks (CNNs) have marked a significant milestone in the field of computer vision. This type of object classifier is inspired by biological processes in the connectivity pattern between neurons in animals' visual cortex of the brain. Digital CNNs take an image as input, where it is fed through the entire network. These networks, with their data-driven, end-to-end learning process, have shown exceptional results in a wide range of computer vision tasks. A typical CNN learns to identify basic filters at the lower layers and abstract high-level features at the upper layers, containing millions of parameters (which are analogous to individual neurons). While this architecture yields highly accurate models, it also leads to decreased interpretability, making it challenging to understand the rationale behind certain classifications by the network, such as identifying a person in an image.

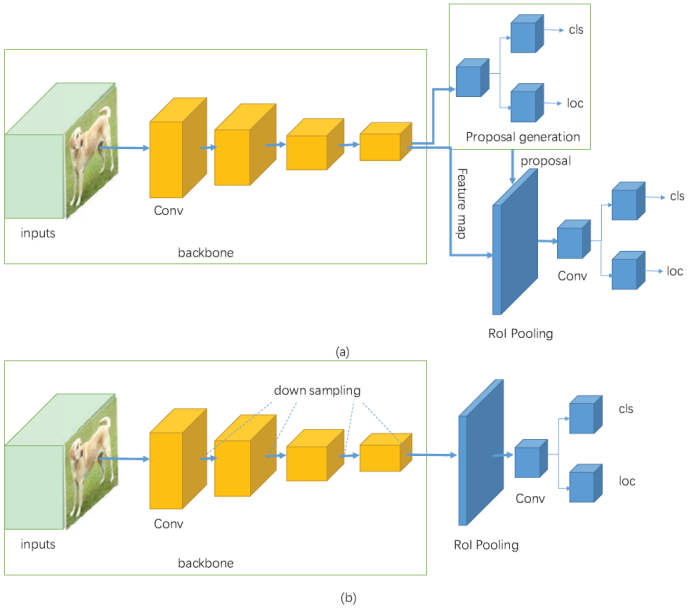


Figure 1: A Convolutional Neural Network contains multiple hidden layers which allow it to model the complexity involved in analyzing visual images. While this technique produces good object detection accuracy, it is difficult to explain how the "black box" arrives at its conclusions.

CNNs have allowed machine vision systems to perform much faster and more accurate classification than previous approaches. However, neural networks are vulnerable to adversarial attacks.

Adversarial attacks on machine learning models, particularly classification tasks, have become a significant concern alongside the growth of CNNs. The early stages of these attacks, as noted by Bigio et al. in 2014 and later expanded by Szegedy et al., involved subtle manipulations of image pixel values to mislead the network into incorrect classifications. The development of these attacks evolved rapidly, with methods like the fast gradient sign method by Goodfellow et al. making it more practical to create adversarial images. These methods demonstrated that even minute changes in an image could lead to drastically different outcomes from the AI model.

By modifying the image with a specifically targeted pattern of noise, the object detector can be defeated:

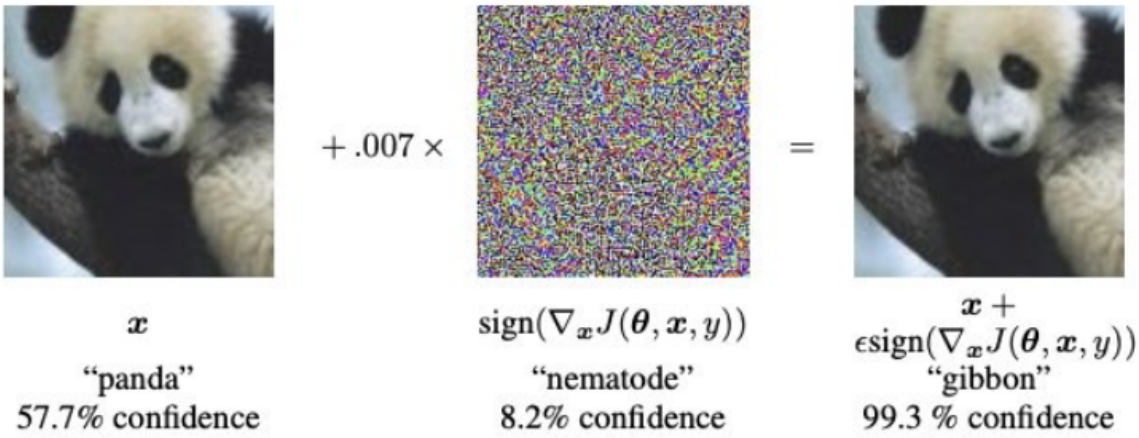


Figure 2: The addition of a specially-crafted noise pattern generated by an adversarial model confuses a detector CNN into thinking an image of a panda is actually a gibbon. The difference to the original image is imperceptible to the human eye.

To a human, the images look identical, however the noise is enough to disrupt an image classifier to recategorize a panda as a gibbon. This approach can also be used to disrupt a CNN's inference that an image contains an object at all. Thys et al. 2019 developed a method to bring this attack into the real world. By placing a "patch" within an image, and carefully tuning the noise within the patch, an object detection framework can be defeated in a real world setting.

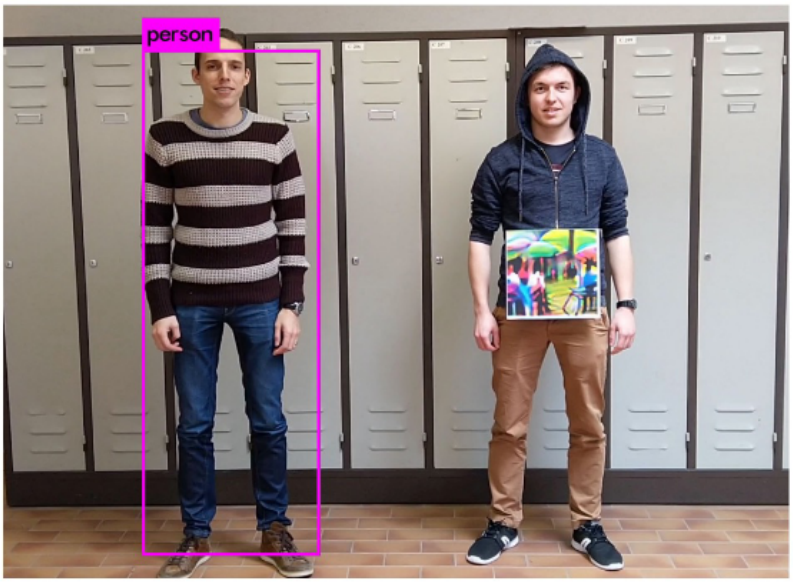


Figure 3: A printout of an adversarial pattern allows an attack to become invisible to a security camera

In the context of object detection, adversarial attacks have been demonstrated in real-world scenarios. For instance, Chen et al. focused on fooling object detectors like Faster R-CNN in detecting stop signs by applying patterns across the entire area of the signs. Similarly, Eykholt et al. used the YOLOv2 detector for a white box attack on stop signs, proving that these attacks could transfer to other detectors. These examples underline that objects with fixed visual patterns, like traffic signs, are vulnerable to such attacks.

The primary goal of our work is to generate adversarial patches that can fool an airplane detection system for satellite imagery in real-world scenarios. This is a more complex challenge compared to previous work on uniform objects like stop signs, due to the highly variable appearance of airplanes in different contexts and orientations. The approach involves optimizing an image to minimize probabilities related to the appearance of an airplane as detected by the model. This optimization process includes various transformations to make the patches robust for real-world applications, such as rotation, scaling, and adjusting brightness and contrast. In addition, a Non-Printability Score (NPS) and Total Variation (TV) score are included in the optimization function. The inclusion of these metrics helps to ensure the end result is capable of being printed using a normal consumer color printer and will be effective under a variety of viewing angles and lighting conditions.

Goals and Objectives

Thys et al conducted their research using a CNN called You Only Look Once Version 2 (YOLOv2). Our project seeks to use a newer version, YOLOv8, to investigate vulnerabilities within the field of remote sensing. To this end, our project has three major goals. Firstly, a new model trained on a remote-sensing dataset with a newer version of YOLO. This model serves as a target to develop a new adversarial attack. Secondly, the inference pipeline of Thys. et al is modified to target this newer model. Thirdly, the new inference pipeline is used to generate adversarial patches targeting this newer model. This entire process is documented extensively, to build a new pipeline for developing an adversarial attack. Future students under Dr. Islam will be able to use the documentation provided to both advance our current research and develop new adversarial attacks on other frameworks.

Methodologies

Our group started with the adversarial agent code published by Thys et al. 2019 for their experiment which attacks a person detector trained with the INRIA dataset (which contains labeled persons within pictures).

Figure 4: Patch pattern generated by our group in replicating Thys et al.'s experiment.

Our group trained a new object detection model to be the target of our attack for remote sensing. We used the YOLOv8 framework to train a new model on Aircraft Detection from Airbus High-Resolution Satellite Imagery Dataset. This dataset is published by Airbus Defense and Space from imagery captured by their twin Pleiades satellites at 50cm resolution. The dataset contains 103 2560x2560 pixel images covering 1280 meters², each of various airports worldwide.

Airplanes within each image had been hand-labeled in the dataset with a polygon mask. Our group adapted a Jupyter notebook on Kaggle to a Python script which converted each polygon mask for an airplane into a bounding box rectangle in YOLO format.

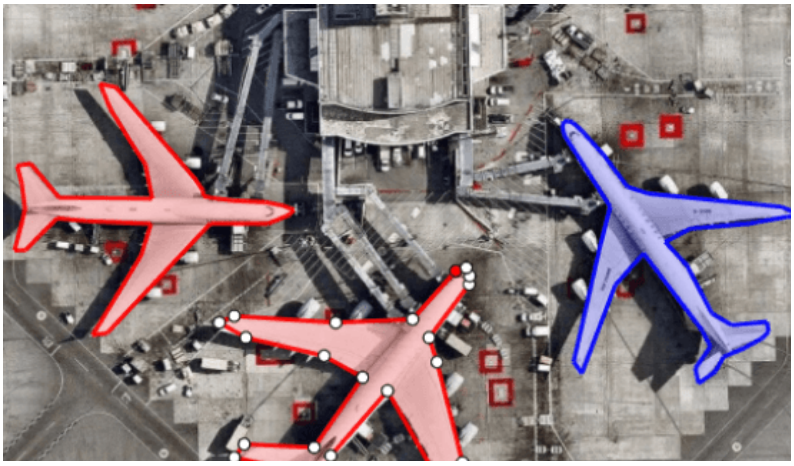


Figure 5: a visualization of polygon masks as appears in the unmodified dataset

The script also broke the images up into 2952 512x512 tiles with 64 pixels overlap. This resultant data was used to train our YOLOv8 airplane object detection model.

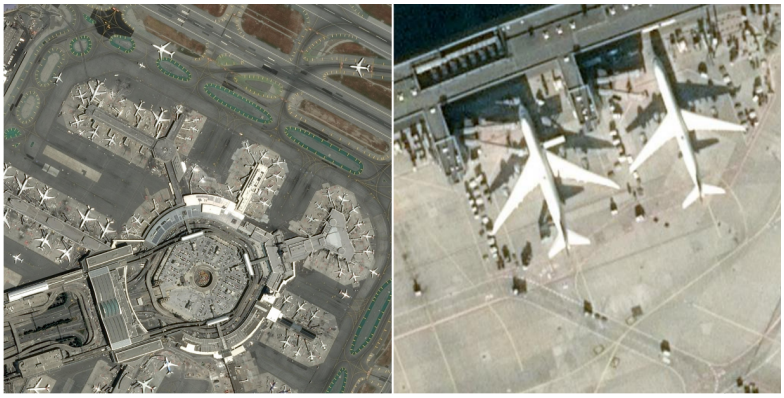


Figure 6: An unmodified 2560x2560 satellite image from the original dataset (left)
One of 2952 512x512 tiles randomly generated by our script (right)
Label data is not shown in either image

Once we finished training this new object detection model, we modified the adversarial model trainer so it could target it in a 'white box' attack scenario. In such a scenario, it is assumed that an attacker has access to a copy of the object detection model they wish to disrupt with the placement of an adversarial patch pattern. Our modifications involved replacing the trainer's YOLOv2 image inference pipeline with the newer YOLOv8 framework and writing our own custom data loader and training configuration. The adversarial model is trained iteratively by the trainer. The adversarial model generates a patch (initially random) and the trainer places it within the bounding box of every target object within an image. Then, the model's loss function is evaluated by the trainer. The weights of this model are modified over subsequent iterations by the adversarial trainer to minimize its loss function using an optimizer function. Training the adversarial model in such a fashion affects the adversarial model's patch output rather than the victim detection model. Object Detection confidence loss, Non-printability score, and a total variation score are included in the trainer's loss function so as to improve the real-world attack capabilities of the adversarial model. The inclusion of these scores incentivizes the trainer to produce an adversarial model that generates patches that prevent AI models from recognizing the target object yet can be printed and deployed in an attack in a real-world setting.

Challenges and Risks

Adapting to YOLOv8: Architectural Challenges and Hardware Limitations

The process of adapting an adversarial AI attack pattern generator, originally designed for the YOLOv2 framework, to the newer YOLOv8 presented a series of significant challenges and risks, both in terms of technical architecture and hardware requirements.

Architectural Differences between YOLOv2 and YOLOv8:

One of the major challenges we faced was the substantial architectural differences between YOLOv2 and YOLOv8. These differences are not merely incremental updates but represent fundamental shifts in how these models process and interpret visual data. YOLOv8, being a more advanced version, incorporates newer methodologies and algorithms that were not present in YOLOv2. This required our team to deeply understand these architectural changes and develop novel approaches to adapt the adversarial attack model accordingly. The task was akin to retrofitting a new engine into a different make and model of a car, demanding significant modifications and tuning to ensure compatibility and optimal performance.

Hardware Limitations and Computational Challenges:

Our project's success heavily relied on the ability to train the adversarial pattern generation model, a process that is computationally intensive and requires robust hardware support. We encountered several hurdles related to the specific hardware needed for this training. Initially, we considered using Google Colab for its access to NVIDIA GPUs, but the free tier offered limited time, making it impractical for our extensive training requirements. One of our team members had a consumer GPU at home, but its memory capacity was severely constrained, posing a bottleneck for the kind of large-scale data processing we needed.

Kennesaw State University's High-Performance Computer:

In search of a solution, we turned to Kennesaw State University's High-Performance Computer (HPC). This resource proved invaluable, offering the computational power necessary to train our model effectively. However, gaining access to the HPC was not without its challenges. We had to navigate the process of obtaining authorization, which was time-consuming. Once access was granted, we faced technical challenges in setting up and running jobs reliably on the system. The use of TORQUE scripts, essential for managing our tasks on the HPC, presented difficulty because it provided no useful debugging output during development. We spent considerable time debugging and adjusting these scripts to ensure they worked correctly and efficiently.

Moral risks and ethical considerations

The primary motivation for this project is to produce an authentic attack scenario that may be used by researchers to build more robust object detection models for remote sensing that are harder to fool. We needed to consider the ethical implications of developing an attack scenario carefully to ensure that its creation or publication would pose no undue risk. We analyzed whether such an attack scenario would be of greater help to researchers developing defense techniques than the potential harm that could be imposed on system operators and their bystanders by a malicious actor. The reasons for our conclusion that the work should proceed are two-fold: First, the attack scenario, while requiring significant adaptation for the newer framework, was not novel. State actors, or even malicious actors of far lesser resources could perform the same type of work as our project but without making it available to researchers for

the development of defense techniques (akin to a “zero-day” vulnerability, which cybersecurity researchers prefer to discover and disclose responsibly rather than allow to be hoarded privately). Secondly, the fields of adversarial AI attacks and even AI object detection, in general, are relatively new and our work can influence the course of development of object detection technology to be safer and more reliable. This is not a conclusion we came to lightly, as it merited significant thought to ensure our work furthered the public good and was in line with the ethical norms of cybersecurity.

Preliminary Results

We have trained a YOLOv8 object detection model to recognize airplanes using the Airbus Satellite Imagery dataset. This model had a good detection performance, and made a useful target for an adversarial attack:

Box(P	R	mAP50	mAP50-95)
0.954	0.867	0.917	0.689

Figure 7: Validation statistics of the YOLOv8 airbus remote sensing model

With an mAP50 score of 0.917, this model demonstrated not only the ability to detect the aircraft within the dataset, but also define accurate bounding boxes for each aircraft:

Figure 8: An example of the detection output of the YOLOv8 remote sensing model.

Currently, our adversarial model generates a rudimentary patch that targets our YOLOv8 aircraft detection model.

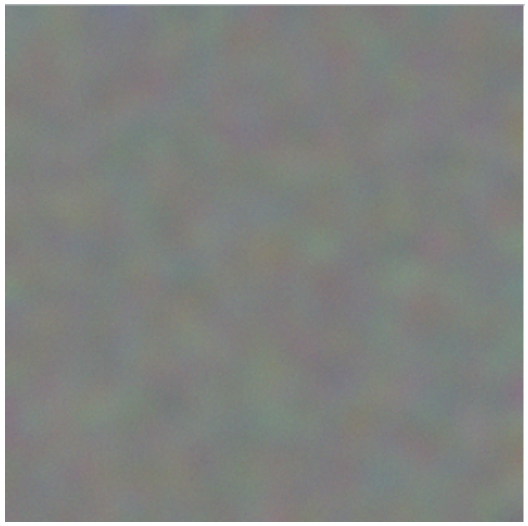


Figure 9: The generated rudimentary patch

This patch lowers the detection capabilities of the model by a measurable amount, reducing both the precision of the bounding boxes as well as lowering the predictive confidence of the model.

Box(P	R	mAP50	mAP50-95)
0.922	0.765	0.811	0.557

Figure 10: The new detection statistics of the YOLOv8 model against patch-applied images.

Of note is that the mAP50 score of the model has decreased from 0.917 to 0.811. However, this number is still high, and reflects that the patch is only capable of reducing the object's detection in specific circumstances. These circumstances are still promising for future changes:

Figure 11: Applied Patch reducing the bounding box precision of YOLOv8

Figure 12: Applied Patch defeating YOLOv8. Definable plane features are visible, yet there is no object detected.

Fundamentally this attack is still only partially successful. The current understanding is that transitioning from a partially successful attack to a meaningful attack will require further access from the target model, and a more in-depth exploration of the model's internal structure. This understanding emerges from the fact that parameter tuning and dataset curation have not meaningfully improved patch performance, and the current approach is set up as a "white-box attack" with direct access to the target model, as opposed to a "black-box attack" where the target model is unknown. Transitioning to a black-box approach before developing and understanding a white-box approach is unlikely to succeed, as white-box attack performance is useful in developing a black-box attack.

Project Management and Team Dynamics

Our team is made up of four members, our project client, and one graduate student advisor/project assistant. Project lead is Zeke Lipscomb. Babrah Koroma is the administrative and data science assistant, Kyle Bratcher is YOLO adversarial attack computing researcher and developer, and Matthew Crupczak is also YOLO adversarial attack computing researcher and developer. The project client is Dr. Kazi Aminul Islam, and his graduate assistant Sumaiya Tasneem was added to the project as a technical assistant while using our attack information as research for the defensive side of furthering the science. Our primary lines of communication were Discord, and Microsoft Teams, as well as weekly Thursday meetings with the full team. This provided the team with continuous access to each other through voice and video meetings, when a need arose. Our primary file storage and sharing was through Github, where we stored meeting information, updates, images, and new information. The majority of our class time

would be moved to Microsoft Teams on Tuesdays and Thursdays with a range of times that team members could join to obtain further information from Professor Perry.

In our Meetings, we gave project update presentations to Dr. Islam and received feedback and next-step recommendations. Meetings were recorded if any need for missed or misunderstood came about. Kyle and Matthew gave updates on changes made, as well as any issues that came up with creating the attacks or with the school's system that was being used due to its excellent hardware and resources. Zeke and Babrah gave updates on the classwork and necessary submissions and ran the information by Dr. Islam for feedback. Submissions were often a team collaboration within Discord and at times, Google Docs to make sure that not only administrative information was provided, but also correct technical data.

Project Name:	KI-1-AI-Security																		
Report Date:	9/7/2023																		
						Milestone #1				*Milestone #2*						*Milestone #3*	*Milestone #4*	*C-Day*	
Deliverable	**Tasks**	**Complete%**	**Current Status Memo**	**Assigned To**		09/02	09/09	09/16	09/23	09/30	10/07	10/14	10/21	10/28	11/04	11/11	11/18	11/25	12/02
Plane Obj Detection Model	Obtain access to KSU HPC	100%	Completed	Kyle, Matthew		2	2												
	YOLOv8 model jupyter notebook	50%	Technique is ready	Kyle		1	4												
	YOLOv7 model manual training	10%	Torque scripts available	Matthew			2												
	Model testing and finalization	0%		All				7	10										
Train Adversarial Model	Fork codebase to team GitHub	100%		Kyle, Matthew					1	1									
	Planning for newer YOLO port	100%		Kyle, Matthew, Sumaiya					10	10	2								
	Porting to newer YOLO	100%		Kyle, Matthew, Sumaiya							8	10	10	10					
	Testing and Debugging	100%		All										8	8				
	Prepare training adversarial model	100%		Kyle, Matthew, Sumaiya											8				
Generate Defeat Pattern	Obtain output from trained model	100%		Matthew											2	4			
	Validation testing (mAP)	100%		Babrah												8			
Code and Docs	Code Cleanup	100%		Kyle, Matthew, Sumaiya													8		
	Documentation and reproducability	0%		All													10		
Final report	Presentation preparation	100%		Zeke														10	10
	Poster preparation	100%		Babrah , Kyle, Matthew														12	4
	Final report submission to D2L and project owner	0%		All														8	8
			Total work hours		**198**	**3**	**8**	**7**	**21**	**11**	**10**	**10**	**10**	**18**	**18**	**12**	**18**	**30**	**22**

Legend

Planned	198
Delayed	
Number	Work man hours

Conclusion and Future Direction

As our Senior Project culminates, we reflect on the strides we have made in constructing an adversarial patch attack against a remote sensing YOLOv8 model. Building upon the foundational work of Thys et al., we have ventured into uncharted territories, tackling the complex challenges presented by advanced machine vision systems. While our current outcomes have not reached the level of performance we initially envisioned, the journey has been a rich learning experience, paving the way for future advancements in this field.

Our iterative process, though met with hurdles, has led to a comprehensive understanding of the nuances involved in creating effective adversarial attacks for newer AI frameworks. Dr. Islam, who has been closely involved in guiding and mentoring our efforts, acknowledges the value of our work so far. The insights we have gathered, coupled with the modifications and strategies we have developed, will serve as a robust foundation for his research group. This

foundational work is poised to facilitate further explorations into the vulnerabilities present in remote sensing applications and the development of strategies to mitigate these vulnerabilities.

Looking ahead, the project's trajectory is set to encompass a broader scope beyond YOLOv8 to include a variety of object detection frameworks. This expansion is critical, as it aligns with the evolving nature of AI technologies and the need for comprehensive security measures that keep pace with these advancements. The continuation of this project under Dr. Islam's guidance ensures that future researchers will be well-equipped with the tools and knowledge to fortify object detection frameworks against sophisticated adversarial attacks.

The contribution of our project extends beyond the confines of our university lab. It plays a pivotal role in shaping the future of AI security, particularly in the domain of remote sensing. As AI continues to integrate deeper into critical systems, the importance of ensuring its resilience against adversarial threats becomes increasingly paramount. Our research is a step towards creating a safer, more secure digital world where AI systems can be trusted to perform with integrity and reliability.

Reference

Thys, S., Van Ranst, W., & Goedemé, T. (2019). Fooling automated surveillance cameras: adversarial patches to attack person detection. In CVPRW: Workshop on The Bright and Dark Sides of Computer Vision: Challenges and Opportunities for Privacy and Security.

Airbus Defense and Space. (n.d.). Sample Aircraft Detection Dataset from Airbus High-Resolution Satellite Imagery [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/airbusgeo/airbus-aircrafts-sample-dataset

D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," in *The Journal of Physiology*, vol. 195, no. 1, pp. 215-243, Mar. 1968, doi: 10.1113/jphysiol.1968.sp008455.

B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Sˇrndic´, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In Joint European conference on machine learning and knowledge discovery in databases, pages 387–402. Springer, 2013.

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.- Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37. Springer, 2016.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau. Ro- bust physical adversarial attack on faster r-cnn object detec- tor. arXiv preprint arXiv:1804.05810, 2018.

D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, and T. Kohno. Physical adversarial examples for object detectors. In 12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18), 2018.

Supplementary Technical Documentation

The following section is reserved for technical documentation which may be useful for the researchers taking over for this project. It is not intended for a general audience.

Project Onset

The project began on with the goal of adapting the adversarial patch trainer from the paper "Fooling automated surveillance cameras: adversarial patches to attack person detection" to work with YOLOv8. Dr. Islam and our group met the week of August 21st, 2023 to discuss design, deliverables, and goals for the project.

The paper itself:
https://arxiv.org/abs/1904.08653

The original codebase on GitLab:
https://gitlab.com/EAVISE/adversarial-yolo

The original version of the codebase used YOLOv2 for the object detection inference pipeline.

Our group first made a new repository on GitHub and re-uploaded the contents of the codebase to this new repository. It would continue to be used for the remainder of our work on the project:
https://github.com/KI-1-AI-Sec/adversarial-yolo

Some of our first modifications included replacing the large jupyter notebook files with GIT LFS stand-ins, due to GitHub having lower file size limits than GitHub. This has the added benefit of not duplicating these large files across the entire history in the .git folder, saving significant bandwidth and disk space.

Next, we added a .gitignore file so that large dataset files, weights, and Python virtual environment resources would not accidentally be added to the GitHub repo:

```
venv/*
env/*
weights/*
inria.tar
inria/*
```

Next, we created a requirements.txt file to make prerequisite setup easier, since this was not well documented in the original repository. Our requirements.txt file reads as:

```
torch
torchvision
```

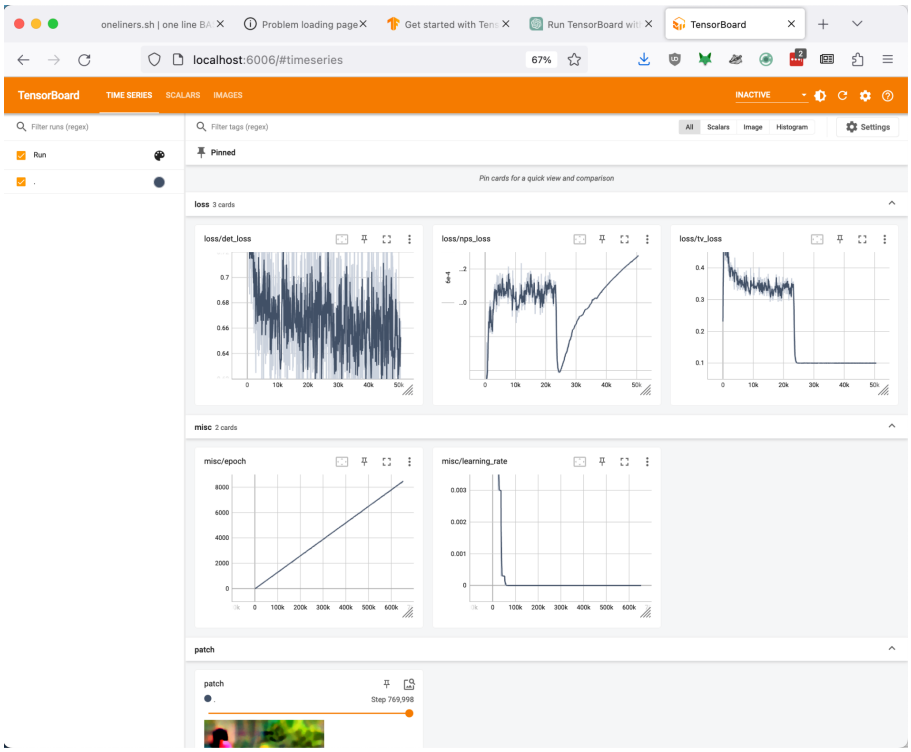
```
pillow
tensorboardX
tensorboard
ultralytics
tqdm
Matplotlib
```

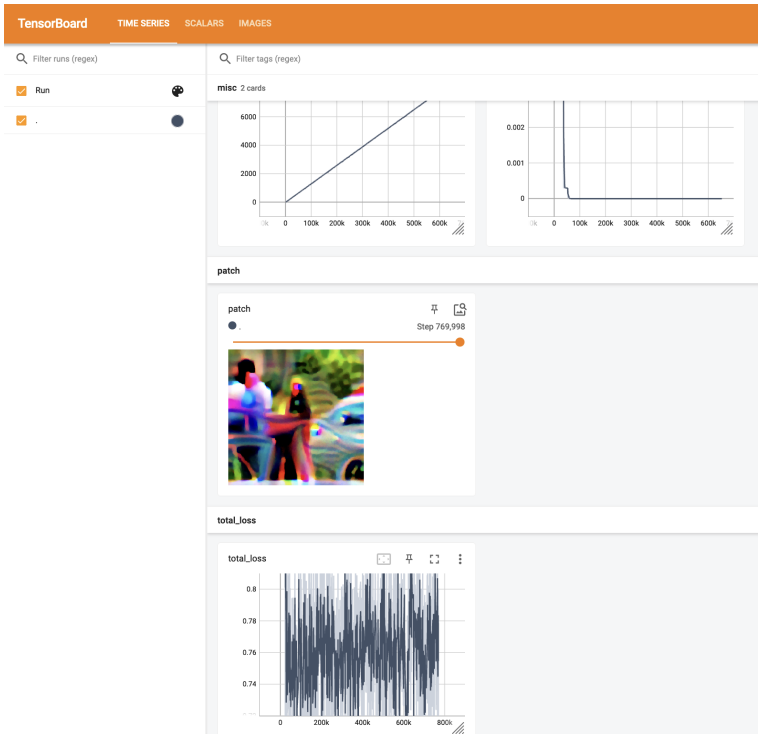
And added instructions to the README.md how these could be installed with the command “pip install -r requirements.txt”.

NOTE: We found we were lucky that everything worked correctly with the latest version of Python (3.11 at the time) and these packages. If you find this project is no longer working with the latest Python and these packages, you may need to adjust the requirements.txt to use specific version codes for each package. Our setup was tested as working on October 10th, 2023.

Initial reproduction of experiment from research paper

We then downloaded the INRIA persons in context dataset, and were able to to train an adversarial model and create a patch according to the default run instructions in the README.md (which under the hood used the ReproducePaperObj() class in patch_config.py):





Modifications for remote sensing applications

Seeing the code worked as expected, we proceeded to make further modifications.

The original codebase was from around 2012, when the VRAM available on consumer GPUs was much smaller and no tensor core hardware was available for ML acceleration. For the default training configuration in patch_config.py, we chose to double the batch size from the default of 8 to the new value 16. We did not increase the learning rate for the default config, though it should have been doubled as well to scale linearly with batch size. Scaling learning rate linearly with batch size is common practice in most ML guidelines.

We then created a custom data loader class in load_data.py called “AirbusDataset”, which would use the Airbus Defense and Space Airplane detection dataset from Kaggle: https://www.kaggle.com/datasets/airbusgeo/airbus-aircrafts-sample-dataset

The original dataset contains 2560x2560 satellite pictures of various airports at 50cm resolution per pixel, and airplanes within them labeled with polygon masks. The jupyter notebook for YOLOv8 available on kaggle included pre-processing instructions that would randomly break these up into a number of 512x512, with the polygon masks converted to rectangular bounding boxes for use with YOLO training. Unfortunately, we could not get this Jupyter Notebook to run on the KSU High Performance Computer, so we converted the preprocessing instructions in the

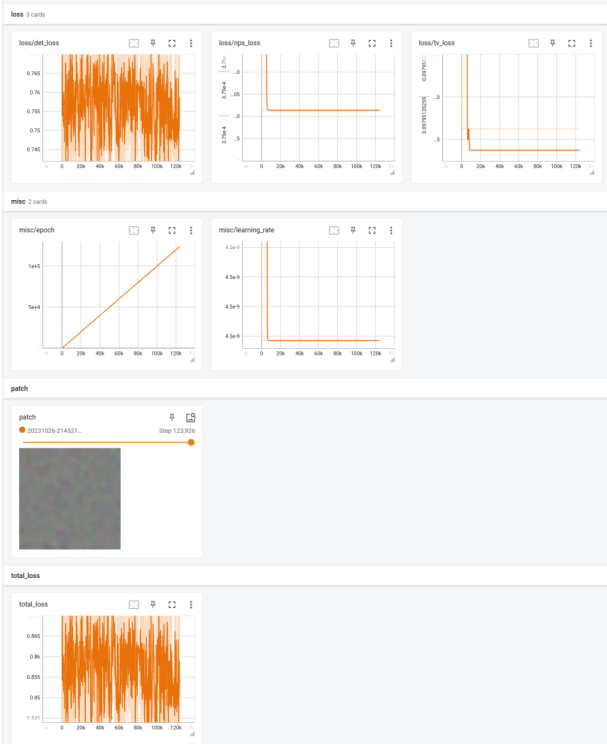
notebook to a standalone Python script, available here:
https://github.com/KI-1-AI-Sec/airbus-aircraft-dataset/blob/main/gen_dataset.py

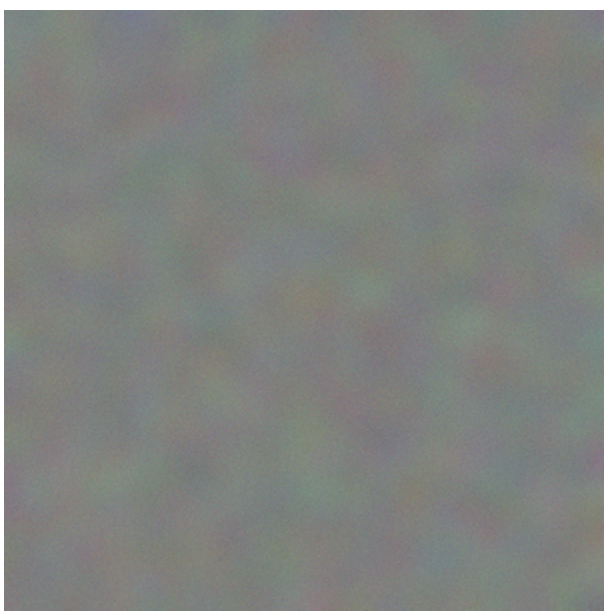
From this dataset, we trained a YOLOv8 model (now available in our adversarial-yolo repo as the file best.pt) to recognize airplanes. In testing, it had 50(mAP50) score of 0.91 with Precision and Recall scores of 0.96 and 0.85, indicating good object detection performance on the aircraft dataset.

Next, we wrote a script for a tool called TORQUE, which is used to define and submit jobs to be run on the KSU High Performance computer:
https://github.com/KI-1-AI-Sec/ksu-hpc-torque

An earlier version of this script was not working and it was difficult to diagnose why. Eventually, the root cause of the error was determined to be a one character syntax error, a semicolon ';' where there should have been a regular colon ':'. Torque did not provide any output to suggest this was causing an error, but it still ignored all content on the same line after the error. This line contained the part of the script that defines the job's required GPU allocation, its omission caused the script to queue on regular compute nodes without any GPU's allocated for the training jobs.

Finally, we added a new training configuration object AirbusFull() to train_patch.py. This configuration had the batch size of 12 (a 50% increase from the default 8) and learning_rate of 0.045 (a 50% increase from the default 0.03). Because we did not have the HPC script working at the time, one of our members needed to run this on their personal computer with an NVIDIA RTX 4070 with 12 GB of VRAM, which was not enough for any higher batch size than 12. This was the configuration that we used for our initial experiments on the Airbus satellite imagery remote sensing dataset. Unfortunately, the results of our experiment did not turn out as expected:





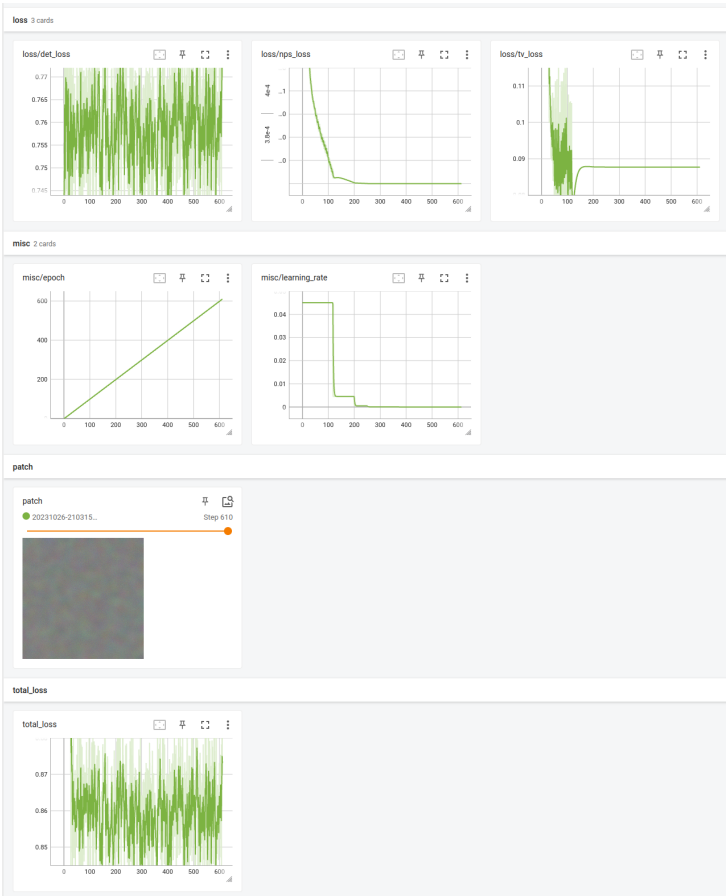
Results from original adaptation experiment

The output patch pattern had a negligible effect on the detection score for the airplane detection model, rendering it ineffective as an adversarial attack example.

We discussed with Dr. Islam what might be some causes of this type of output. He suggested we investigate the number of images used for training the patch itself. We believed that the large number of images in the dataset may contain too much variation between batches to allow a gradient to inform the training process. He suggested we set the number of training images to be equal to batch size; we scaled both back to 8.

We cherry picked 8 examples of clear training images with aircraft present, and placed them in the folder [airbus-subset-8] of our adversarial-yolo GitHub repo. We made the config object AirbusSubsetEight in patch_config.py.

Next, we did a training run with the new configuration, to see if the reduced dataset size would improve training outcomes. We found it did not:



Root cause analysis and discovery

We met with Dr. Islam to discuss these results (or lack thereof). We discussed multiple possible explanations and more tests we could perform to diagnose the issue. Dr. Islam suggested we could try adjusting the hyperparameters for the training run and reducing the effect of the non-printability score.

Just before Thanksgiving break, Kyle had a breakthrough, and discovered that the reason the training was not working as expected is that the original YOLOv2 training process used access to the detection model's internal tensors to calculate the gradient for the adversarial trainer, whereas our modified implementation does not. The original is literally a 'white box' attack technique, where the internal state of the model is used by the adversarial trainer while it optimizes its attack pattern. YOLOv8 has built in functionality that manipulates the output tensor for easier manipulation. However this newer output does not connect to the actual model. This means that while the training code can easily optimize both printability and variation scores, it cannot effectively compute a gradient for detection loss. Effectively, instead of a whitebox attack, it is performing a black-box attack, without any optimizations necessary for an effective black box attack. To perfrom an effective white box attack with the newer model, the internals need to be accessed directly so a gradient can be computed. Unfortunately, the process for correcting this error would be very involved, and we were running out of time to implement the changes needed before submitting our application for C-Day and producing our final report for the project. We discussed this time constraint with Dr. Islam, who agreed that it may be best if we focus our efforts on documenting these experiments so far so his research group may be better prepared to take over the project.

Towards the end of the project, we noticed that the university web page hosting the original INRIA persons in context dataset went down. Because it was not readily accessible elsewhere on the web, we added this data directly to the folder 'inria' within our adversarial-yolo GitHub repo. This will make it easier for others to run the ReproducePaperObj training regimen if they need for testing after implementing other changes.

Conclusion and technical point of contact information

This concludes the Supplementary Technical Documentation of this report. Our two programming contributors: Matthew's email is matthew@krupczak.org, Kyle's is kylesbratcher@gmail.com. Please contact us if you need any more information to continue progress on this project after our semester Senior Project at KSU concludes.