

# Defending Adversarial Patches: Detect & Blur

Defending Adversarial Patches in the Yolo 11 model through detection and blurring of the image.

Erick Constant  
Student  
Hampton University  
erick.constant@my.hamptonu.edu

Chutima Boonthum-Denecke  
Professor  
Hampton University  
Chutima.boonthum@hamptonu.edu

Idongesit  
Professor  
Florida A&M University  
idongesit.ruffin@famuedu

## ABSTRACT

Computer Vision models has increasingly been embedded into video software to recognize and classify things in the physical world. While this can provide a useful result it also opens the door to vulnerabilities through a physical attack. Using a printed-out generated image, individuals can exploit computer visions models to disguise their true intentions. A possible way to block and mitigate the problems is to detect and blur the entire image to try to allow the AI to inference the said image.

## CCS CONCEPT

Computing methodologies

- Artificial intelligence
- Computer vision
- Computer vision tasks
- Scene anomaly detection

## KEYWORDS

- Computer Vision
- Detection
- Adversarial

## Introduction:

Right now, we live in the impossible, in the past few years things we thought artificial intelligence would never be able to do, currently is being done. We have generative artificial intelligence that can keep a conversation with a human, predictive algorithms that can predict trends in the world that might be a cause of

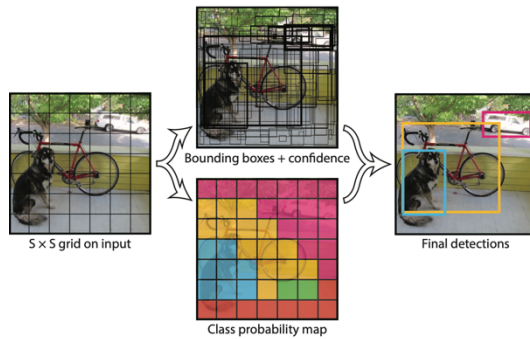
concern, and lastly the one I think has been the most used artificial intelligence, computer vision the system that allows us to protect ourselves, our property and speed up automation.

Computer Vision technology currently exists in many fields like manufacturing, consumer products and even in a government and military capacity. All being trained to recognize certain imagery and do a specific action. Behind this is all math calculations that run behind the scenes to make this all possible.

Currently there are many different computer vision libraries like OpenCV, Tensorflow, PyTorch and SimpleCV that enable developers to be able to produce and train their own computer vision models in a way that might suit their needs better.

Like I mentioned previously computer vision is put into models that allows them to be ran on a device through code. One example and the model I will be demonstrating defending adversarial patches on is *You Only Look Once*.

You Only Look Once or YOLO is a open-source model that was proposed by a few researchers in their work called *You Only Look Once: Unified, Real-Time Object Detection*. This model as described in their abstract “frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.”



The Process described in the You Only Look Once's Artificial Intelligence Paper

Since then, researchers have built iterations to it that has ultimately increased its performability. Currently, the iteration that is the latest is YOLO 11 by Ultralytics. A company dedicated to the use of Artificial Intelligence.

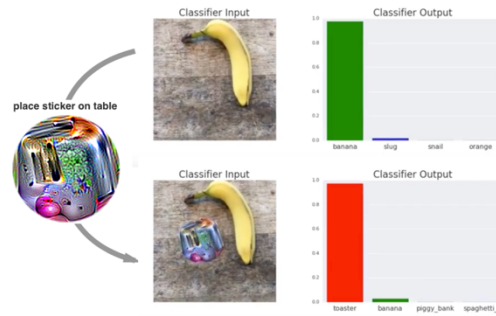
## Computer Vision and Image Processing

### Computer Vision

Computer Vision is the process of using math to teach artificial intelligence to recognize items that are physically present in the world. This enables computers to do tasks that typically require human oversight to now being able to do it automatically. Allowing for in a security sense safer protection while also keeping cost lows for the business.

### What is an adversarial patch and why is its defense important?

In the paper *Adversarial Patch* released in 2018, adversarial patch is mentioned to be a generated image that is completely independent of the original image that causes the neural network behind the computer vision model to focus only on the image rather than the entire image. This ultimately like stated in the paper “allows attackers to create a physical-world attack without prior knowledge of the lighting conditions, camera angle, type of classifier being attacked or even the other items within the scene.” Knowing this information, we can



Example of Adversarial Patch inside of the paper *Adversarial patch in 2018*

suppose that if we had an Artificial Intelligence model that was being used to protect property or life. It essentially allows an attacker to print an adversarial patch already on the enter that would render the entire system useless negating any benefit that the Artificial Intelligence model would have.

### Other works related to the Defending against Adversarial Patch

One paper that is published about defending against adversarial patches is *PatchZero: Defending against Adversarial Patch Attacks by Detecting and Zeroing in the Patch*. Like the name suggests, within the paper it describes a way to defend against adversarial patch by detecting and filling in a patch and then running the entire image through artificial intelligence to get a more accurate result. What makes my work different than the previous suggested article is that once detecting an adversarial patch I am attempting to blur the entire image hoping that the artificial intelligence model will make inference of the image is supposing to be.

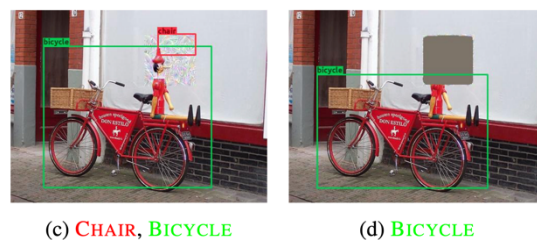


Image retrieved from the PatchZero paper, demonstrating its effectiveness.

## Methodology

First step in creating the experiment I gathered as many pictures as possible of adversarial patches that I can find that is available on the open internet. This led me to a dataset published on Roboflow called “*Adversarial Patch on Imagenet Dataset*” I was then able to fork the project of the dataset to my local device.

calculated the percentage of change between the original and the adversarial patch.

In the table, it is reflected that in most cases after the blurring and adversarial patch the image drops the ability to detect items with the same accuracy. This is especially the case when it comes *Image 2*, when it comes to cars.

Using the dataset described above, I set up my

	Objects	Original	Adversarial	Change Percentage
Image 1	Oven	0.52	0.25	-51.92%
	Refrigerator	0.94	0.91	-3.19%
	Person 1	0.79	0.83	5.06%
	Person 2	0.86	0.85	-1.16%
	Clock	0.88	0.62	-29.55%
Image 2	Train	0.73	0.88	20.55%
	Cars	0.34	0	-100.00%

computer to use my installed graphic card to perform the calculations that is required to train the model using the dataset. Afterwards I tested to make sure that the model was able to run independently, before integrating it into the main python.

However, in some cases the blurring of the image was able to improve the accuracy of some images like in the case of person 1 in Image 1 and the train in Image 2.

After testing to make sure the train model was able to run from another file. I then started to flush out writing the other file which iterated a directory that held the images and was automatically attempting to detect the patch, based on the result blurring the image until it is no longer detectable and then running it through another model.

Attached below are the images that the results were gathered from.

The other model was another YOLO11 model that was pre-trained off the COCO8. It was then tasked to predict and export the results of the imagery.

## Results

After running the code, I find that in most cases, the model was still able to detect the subject of the image with a limited accuracy. In the table below I've selected 2 distinct images and have



Figure 1

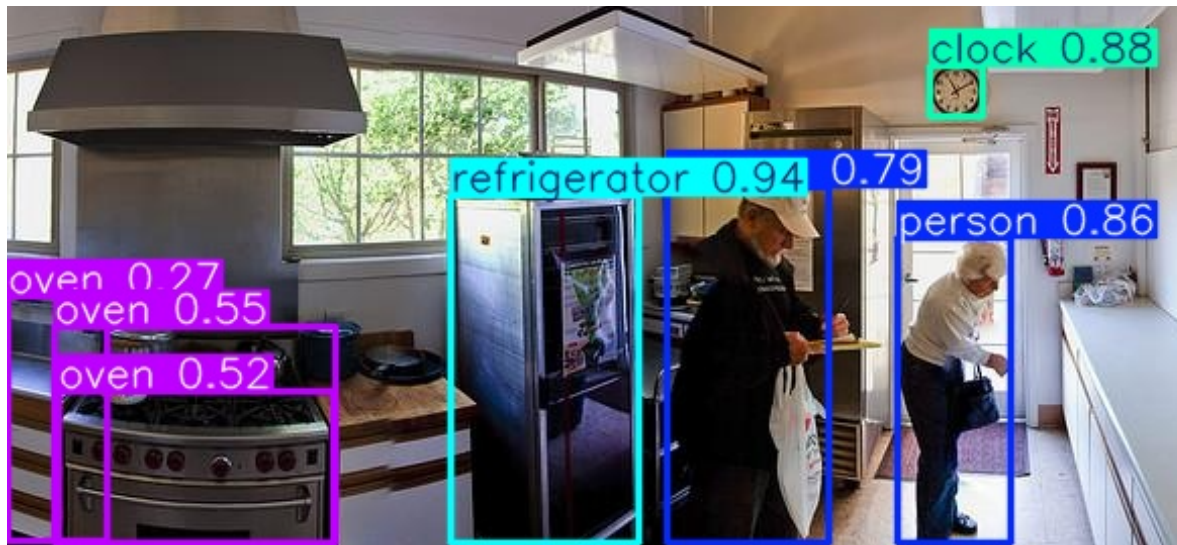


Figure 2



## REFERENCES

blusonik. (2024, May). Adversarial Patch on Imagenet Dataset. Roboflow Universe. Roboflow. Retrieved February 3, 2025, from [<https://universe.roboflow.com/blusonik/adversarial-patch-on-imagenet>](<https://universe.roboflow.com/blusonik/adversarial-patch-on-imagenet>)

Brown, T., Mane, D., Roy, A., Abadi, M., & Gilmer, J. (2017). Adversarial Patch. arXiv preprint arXiv:1712.09665.

Jocher, G., & Qiu, J. (2024). Ultralytics YOLO11 (Version 11.0.0) [Software]. Available from [<https://github.com/ultralytics/ultralytics>](<https://github.com/ultralytics/ultralytics>)

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). Microsoft COCO: Common Objects in Context. arXiv preprint arXiv:1405.0312.

Xu, K., Xiao, Y., Zheng, Z., Cai, K., & Nevatia, R. (2023). Patchzero: Defending against adversarial patch attacks by detecting and zeroing the Patch. *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. <https://doi.org/10.1109/wacv56688.2023.00461>