Impact of Adversarial Patches on Object Detection with YOLOv7

Darrien Hunt[#], Chutima Boonthum-Denecke[#], Idongesit Mkpong-Ruffin[*]

[#]Department of Computer Science, Hampton University, Hampton, VA
[*]Department of Computer and Information Sciences, Florida A&M University, Tallahassee, FL.

Abstract:
With the increased use of machine learning models, there is a need to understand how machine learning models can be maliciously targeted. Understanding how these attacks are 'enacted' helps in being able to 'harden' models so that it is harder for attackers to evade detection. We want to better understand object detection, the underlying algorithms, different perturbation approaches that can be utilized to fool these models. To this end, we document our findings as a review of existing literature and open-source repositories related to Computer Vision and Object Detection. We also look at how Adversarial Patches impact object detection algorithms. Our objective was to replicate existing processes in order to reproduce results to further our research on adversarial patches.

Introduction

Computer Vision is a subset of Artificial Intelligence (AI) that grants computers the ability to "see" and process visualizations, thus extracting valuable information and performing a particular task in response. This technology can be integrated into the features of many existing devices we use today to improve its capabilities. One prominent feature in development under the umbrella of computer vision is **Object Detection**. Object detection allows computers to utilize cameras to track images and perform analysis. Image analysis is generally performed by training deep learning models on a large set of images in order to accurately classify objects. Images are classified by extracting visual features from them, usually by using a sliding window to scale through them and utilizing those features to make distinctions [1].

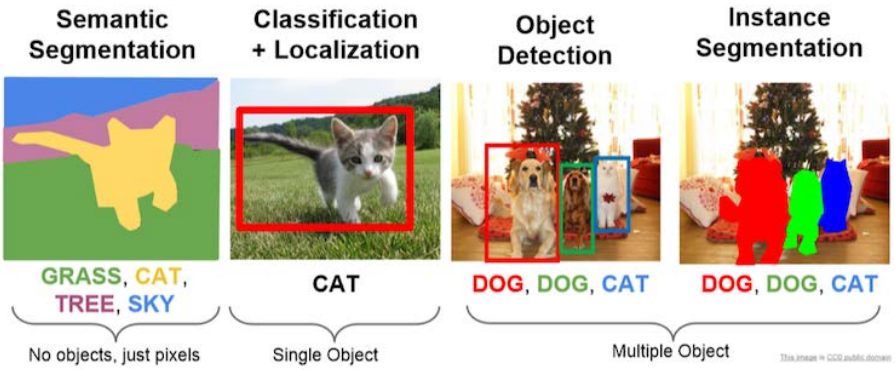

Figure 1: Comparison of semantic segmentation, classification and localization, object detection and instance segmentation. [2]

Throughout the progression of object detection technology, various algorithms such as YOLO (You Only Look Once) [3] have been developed. This has enabled real-time object detection research with a model capable of producing results with high detection speed and accuracy. Compared to previous versions of YOLO, YOLOv7 benefits from an improved network architecture and less of a need for more expensive computation power [4]. YOLOv7 utilizes Convolutional Neural Networks (CNNs) which is a learning algorithm used for image processing. This method allows computers to take images or videos and digitize them by converting them to pixels before taking a window size of the image and extracting features from each window. Its purpose is to find some useful things in each frame and pool them together. Classification can then be performed upon the output of the algorithm to determine what the computer saw. Like other neural networks, this algorithm benefits from a learning process called backpropagation which allows values to be passed back through the algorithm with adjustments in order to train the model more efficiently.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of deep neural network that have been widely used in computer vision tasks, including object detection [5]. CNNs are designed to automatically learn features from raw input data, such as images, by applying a series of convolutional filters that scan the image at different scales and orientations, looking for specific patterns or features.

In object detection, CNNs are typically used in two stages: region proposal and object classification. In the first stage, the CNN is used to generate a set of candidate object regions in the image, which are then passed to the second stage for classification. The candidate regions are typically generated using a technique called selective search, which identifies regions that are likely to contain objects based on their color, texture, and other visual cues. In the second stage, the CNN is used to classify each candidate region as either containing an object or not. This is typically done by applying a sliding window approach, where a small window is moved across each candidate region, and the CNN classifies the contents of the window. If the CNN determines that the window contains an object, the region is considered a positive detection and the object is localized within the region.

CNNs have been shown to be highly effective in object detection, achieving state-of-the-art performance on many benchmark datasets. However, they require large amounts of training data and can be computationally expensive, especially for real-time applications. To address these challenges, researchers are exploring new architectures, such as the YOLO [3] and Faster R-CNN models [6], which aim to balance accuracy and speed.

Adversarial Attacks and Adversarial Patches

Adversarial attacks aim to fool machine learning models. Adversarial attacks can be done by making changes to a physical object so as to fool the machine learning model so that an image or object is mistaken for some other object or image or in some cases, not even detected. This ability to make a change to an object for the purpose of fooling a system is also known as adversarial perturbation. In this work, we investigated the usage of patches to cause the image classifier to misclassify, misidentify or be unable to identify given objects. Adversarial patches are images that once printed, added, or presented to the image classifier, can cause the classifier to ignore the other items or misidentify the items [8, 9].

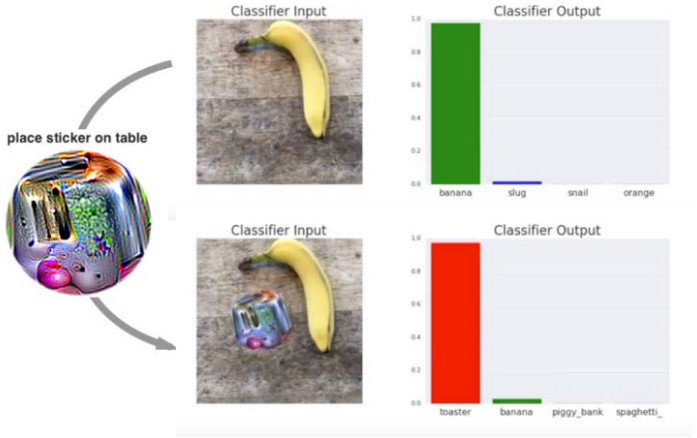

Figure 2: A real-world attack on VGG16, using a physical patch [7]

Experimental Approach

We reviewed various papers detailing generative adversarial patches and their effects on object detection algorithms. These patches seek to prevent the algorithms from accurately detecting objects within the images. Many papers provided access to GitHub repositories, thus allowing us to use pre-trained models on our test datasets to compare the results of the algorithm with varying images and patches.

We worked to experiment with the Pytorch version of the algorithm to observe its effect on a random set of images from ImageNet which consisted of images of people and objects/animals. The set of objects/animals are a bicycle, birds, cars, cell phones, and dogs.

In order to run the algorithm, we had to configure my computing environment by cloning the YOLOv7 repository (https://github.com/wongkinyiu/yolov7) and then installing all the Python libraries listed in the provided text file. We were able to run the detection algorithm on all the images to generate a confidence value and altered images with a border surrounding the classified objects. The confidence value represents how sure the model accurately defines an object. The script we utilized for detection allowed us to customize the parameters in order to set pre-trained model weights, set a confidence value threshold, specify image size, and set a target directory for my images to perform classification successively.

The first step in measuring the effectiveness of the model was to run the detection algorithm on the set of unaltered images to record their confidence values. Following this, an adversarial patch can be applied to an image in an attempt to alter the confidence value output. Two patches were used to apply to images and test the algorithm as shown in Figures 3. They were automatically generated with qualities that may have the ability to fool algorithms. Each patch was applied over both categories of images, and for images of people, they were applied over the face and the body in separate instances to observe the varying effects.

Figure 3(a). Patch #1

Figure 3(b). Patch #2

	Confidence Value		
Object	Non-people	Non-people Patch #1	Non-people Patch #2
Bicycle	0.947094	0.948981	0.941189
Bird 1	0.785761	0.75917	0.796503
Bird 2	0.687701	0.475139	0.616994
Car 1	0.90718	0.397678	0.907187
Car 2	0.784057	0.540132	0.77098
Dog 1	0.771059	0.916712	0.883784
Dog 2	0.961846	0.944186	0.956071
Average	**0.83496**	**0.71171**	**0.83896**

Table 1. Confidence values of YOLOv7 ran on subset of images representing non-people

Results and Discussion

All of the confidence values retrieved from the output of the algorithms were recorded in **Table 1** and **Table 2**. In order to more easily digest the results and store the values, the parameter of the algorithms that permits saving text files was used, and the script was modified to append the confidence value to the results. Based on observation, the algorithm seemed to be less prone to detecting objects within Patch #2, which proved to be a distraction with Patch #1. Almost all images that utilized Patch #1 were subject to detection of not only the objects in the original image, but also the hidden objects within the patch. This is likely because Patch #1 contains objects that are more easily recognizable based on the images utilized to train YOLOv7 [4, 7].

	Confidence Value				
Object	People	People Patch #1 (Over Body)	People Patch #1 (Over Face)	People Patch #2 (Over Body)	People Patch #2 (Over Face)
Person 1	0.972414	0.965844	0.788071	0.973064	0.967817
Person 2	0.95275	0.949856	0.817727	0.956822	0.956297
Person 3	0.962644	0.960276	0.952369	0.961844	0.950804
Person 4	0.967071	0.953975	0.951217	0.961021	0.966046
Person 5	0.946961	0.934886	0.932216	0.943588	0.898543
Person 6	0.957467	0.949614	0.724247	0.959409	0.949607
Person 7	0.888205	0.877668	0.860247	0.890281	0.867448
Person 8	0.839267	0.9634	0.813454	0.908358	0.824774
Person 9	0.953179	0.950174	0.930229	0.952936	0.951521
Person 10	0.912204	0.894054	0.91012	0.911656	0.917367
Person 11	0.909408	0.907011	0.945225	0.912533	0.884443
Person 12	0.957507	0.949215	0.848984	0.959818	0.940265
Person 13	0.916603	0.874622	0.699182	0.906344	0.920201
Person 14	0.821476	0.740749	0.83696	0.792484	0.85599
Person 15	0.966412	0.964674	0.961278	0.960658	0.969172
Average	**0.928238**	**0.9224**	**0.8647684**	**0.93005**	**0.921353**

Table 2. Confidence values of YOLOv7 ran on subset of images representing people

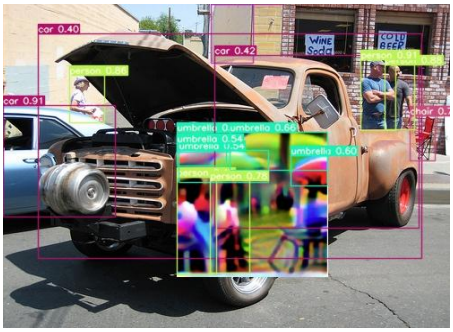

Figure 4. Car 1 with Patch #1

Figure 5. Car 2 with Patch #1

For the set of images representing non-people, the patches had varying effects. While the original images average confidence value was 0.83496, Patch #1 had an overall negative effect on the detection rate of the classifier, producing a result of 0.71171. This drastic decrease was most sharply noticeable with the images of cars which had large patches applied to them and numerous detectable objects in the background of the images. As shown in **Figures 4 & 5**, background objects such as people and other cars, in addition to objects within the patch itself served as potential impediments to the algorithms detection accuracy of the primary car. The size of the patch may also be an overwhelming factor given the algorithm's ability to attempt detection of the patch as well. Patch #2 had a converse effect on the non-people image subject producing slightly better detection rates than even the unaltered images. The patch itself was not generally recognized by the algorithm as a makeup of additional objects.

The people image subset allowed us to view the effects that a patch might have when applied on certain parts of a person, namely the face and the body. It was initially suspected that applying the patch over the face of a person may serve as more of a threat than anywhere else because the face possesses many identifiable characteristics of a person. Overall, the unaltered images possessed an average confidence value of 0.928238. When Patch #1 was applied over the body and the face, the confidence values generated were 0.9224 and 0.8647684 respectively. While the average confidence value for Patch #1 applied over the body of the person was on par with that of the unaltered images, the same patch applied over the face generated a lower average value. Patch #2 resulted in values of 0.93005 and 0.921353 when applied over the body and the face respectively, citing an increase in detection accuracy over the body and a miniscule decrease over the face. Overall, most images were reflective of that, but in one instance, the application of the patch over the face seemed to have a noticeable effect (**Figures 6 & 7**). While more testing and further analysis should be conducted to determine if the application of a patch over the face or body has drastic effects, it remained consistent that the algorithm conducted using Patch #2, no matter where on the object it was applied, produced better results than Patch #1.

Figure 6. Person 8 with Patch #2 applied over the body

Figure 7. Person 8 with Patch #2 applied over the face

Future Work

In the future, there's a variety of methods that can be executed to further analyze the effectiveness of YOLOv7. Throughout the experiment, it was shown the varying effect that different kinds of patches can produce, so a more in depth analysis of why one patch that contained objects was more easily detectable over another is warranted. The patch size is another potential factor to consider, especially when utilizing patches with identifiable objects within them, so calling patches over images is another avenue of interest. Patch location seemed to add another dimension to the detection rate in some instances, so in order to determine its effects, the same patch should be applied multiple times on the same image in separate locations. There is also room to utilize image datasets that are more expansive and diverse. While this experiment was generally limited to specific objects that the YOLOv7 model was trained on, perhaps more images containing multiple recognizable objects can be used to make determinations on whether or not this will have a positive or negative effect, especially if those objects are overlapping. The test dataset used for this experiment was also minimal, so testing on a larger dataset would also be of great benefit to evaluate more results and average confidence values. A comparison between previous versions of YOLO on the same image set would also help provide greater context.

Conclusion

Overall, this project helped develop my understanding of computer vision and object detection. Exposure to rapidly improving object detection models such as YOLOv7 has heightened my interest in the field and strengthened my knowledge. Reproducibility of this project was the first step in understanding how object detection works and to opening the door to working on improving the model itself. We were able to determine the effects that adversarial patches had on detection accuracy and analyze the results to create more hypotheses.

References

[1] Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems, 30(11), 3212-3232.

[2] Jaikumar, P., Vandaele, R., & Ojha, Varun. (2022). Transfer Learning for Instance Segmentation of Waste Bottles using Mask R-CNN Algorithm. In: Abraham, A., Piuri, V., Gandhi, N., Siarry, P., Kaklauskas, A., Madureira, A. (eds) Intelligent Systems Design and Applications. ISDA 2020. Advances in Intelligent Systems and Computing, vol 1351. Springer, Cham. https://doi.org/10.1007/978-3-030-71187-0_13

[3] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[4] Boesch, G. (2022, August 11). YOLOv7: The Most Powerful Object Detection Algorithm (2022 Guide). Viso.ai. https://viso.ai/deep-learning/yolov7-guide/

[5] Yamashita, R., Nishio, M., Kinh, R. & Togashi, K. (2018) Insights into Imaging (2018) 9:611–629, https://doi.org/10.1007/s13244-018-0639-9

[6] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[7] Thys, S., Van Ranst, W., & Goedemé, T. (2019). Fooling automated surveillance cameras: adversarial patches to attack person detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops (pp. 0-0).

[8] Brown, T. B., Mané, D., Roy, A., Abadi, M. & Gilmer, J. (2017) Adversarial patch, 2017, [online] Available: https://arxiv.org/abs/1712.09665

[9] Nemcovsky, Y., Jacoby, M., Bronstein, A. M., and Baskin, Chaim (2022) Physical Passive Patch Adversarial Attacks on Visual Odometry Systems, Retrieved February 2023: https://openaccess.the cvf.com