Trustworthy Autonomy in Cyber-Physical Systems

Maichl Hendi and Mostafa Alkousa

***Abstract*—Machine learning models, mainly those utilized in computer vision using convolutional neural networks (CNNs), can currently train on extensive datasets and images. Despite their capabilities, they are still vulnerable to special images called adversarial patches that can manipulate their output. Adversarial patches can penetrate various defense models, such as NutNet and NAPGuard. This thesis aims to attack a representative machine learning vision model, YOLOv2. Such adversarial patch attacks could target YOLOv2, and a defense model, NutNet, that is designed to protect victim models against such attacks. Our study was conducted by creating adversarial patches and exploring their impact based on where the patch was located on the image, the number of patches, real-world execution scenarios, and the size of training dataset, which consists of pictures of people in various environments. Patch creation involved training on different-sized dataset for different numbers of epochs. We also used a patch from other work to launch attacks. The attack methods were also varied, for digital attacks, patches were placed directly on the images. For real-world attacks, the adversarial patches were printed and physically applied on a person. We assumed two situations throughout the evaluations, with and without adversarial patch defenses enabled. This approach led to the effectiveness of adversarial patches being evaluated under various conditions. The result showed that patches could attack the object detector with high effectiveness, reaching 75% when NutNet was deactivated. Without the defense model activated the attack success rate reached up to 49.6%.**

***Sammanfattning*—Maskininlärningsmodeller, som används inom datorseende med konvolutionella neurala nätverk (KNN), tränas på omfattande datamängder och bilder. Trots deras kapacitet är de fortfarande sårbara för fysiska cyber attacker som kallas adversariella patchar. Adversariella patchar kan passera olika försvarsmodeller, såsom NutNet och NAPGuard. Detta arbete syftar till att attackera en representativ maskininlärningsmodell, YOLOv2. En attackerare med patch-attacker skulle kunna rikta sig mot YOLOv2, samt försvarsmodellen NutNet, som är designad för att skydda utsatta modeller mot sådana attacker. Vår studie genomfördes genom att skapa adversariella patchar och undersöka deras påverkan baserat på patchens placering på bilden, antalet patchar, attack scenarier i verkliga världen och storleken på träningsdatan, som består av bilder på människor i olika miljöer. Patchskapandet innebar träning på dataset av olika storlek under olika antal epoker. Vi använde också en patch från annat arbete för att genomföra attacker. Olika attack metoder användes där för digitala attacker placerades patcharna direkt på bilderna. För attacker i verkliga världen skrevs de adversariella patcharna ut och applicerades fysiskt på en person. Vi antog två situationer under utvärderingarna, med och utan aktiverat försvar mot adversariella patchar. Detta ledde till att effektiviteten hos de adversariella patcharna kunde utvärderas under olika förhållanden. Resultatet visade att patcharna kunde attackera objektdetektor med hög effektivitet, och nådde upp till 75% när NutNet var avaktiverad. När NutNet var aktiverad nådde attack säkerheten 49,6%.**

***Index Terms*—Object detection, adversarial patch attacks, deep neural networks, cyber-physical systems.**

Supervisors: Mauricio Byrd Victorica, György Dán and Rolf Stadler.

TRITA number: TRITA-EECS-EX-2025:148

I. Introduction

Video surveillance, automated vehicle control, and person tracking are applications that rely on cyber-physical systems (CPS), which integrate physical processes with embedded intelligence. These applications often use CPS frameworks incorporated with deep neural networks (DNNs) for tasks like object detection and recognition. YOLO (You Only Look Once) is an integrated deep neural network (DNN) designed for fast, accurate, and real-time object detection [1]. Despite significant advancements, DNN-based systems remain vulnerable to security threats, including adversarial attacks, where manipulated input data can lead to misclassification and object detection failure. Therefore, the weaknesses of deep neural networks are also weaknesses of cyber-physical systems [2]. One powerful form of such attacks is the use of adversarial patches. These patches can be placed in the real world, causing models to misidentify or ignore particular objects completely. Regardless of the scene content, lighting, or camera angle, these patches are effective and operate independently of the image characteristics [2], [3].

A patch can be trained to perform different attacks. The first is the Hiding Attack (HA): This attack makes objects disappear completely from detection. The second is appearing Attack (AA): unlike HA, this type of attack does not try to make the object disappear, but tricks the model into seeing objects that do not exist [4].

Current developments have even created 'Naturalistic Adversarial Patches' (NAPs), which are more advanced because they are visually realistic and challenging to separate from real objects, making the detection task more difficult [5]. Techniques such as Generative Adversarial Networks (GANs) and diffusion models for generating patches have proven effective in deceiving object detection models.

Defensive techniques attempt to overcome these problems. Often, defenses lack accuracy, generalizability, and have difficulty detecting natural-looking patches in different environments [3], [6]. NutNet a defense model that works on detecting Hiding Attacks and Appearing Attacks. The model handles various detection tools such as YOLOv2–YOLOv4, SSD, Faster RCNN, and DETR. It works in real time by dividing images into blocks for better localization of patches, with only 8 percent increased inference time. When patches are detected, a mask is used to eliminate the patch. NutNet has 2.4× better detection results against HA and 4.7× better against AA than earlier existing methods [4].

Ad-YOLO introduces an approach that integrates an extra class to detect patches and real objects. The model uses specialized training datasets such as INRIA-Patch [1].

NAPGuard is another defense model that aims to locate adversarial patches. NAPGuard works by polarizing aggressive and natural features in patches during the training process, leading to improved generalization and overall accuracy in detecting patches [5].

This study focuses on attacking YOLOv2 by placing adversarial patches on images. The object detector was also paired with a defense model which shifted the focus of the thesis to not only attack the object detector but also bypass the defense model. The developers of NutNet claim that it is difficult to create adaptive patches and in this study we will train adaptive patches to test the claim [4]. With the help of the open source code, we trained patches that could attack the detection model YOLOv2 and bypass NutNet [7]. The main results gathered during this study are:

- The adversarial patch had a peak digital attack success rate of 75% on YOLOv2 when NutNet was deactivated. The success rate dropped to 49.6% With NutNet activated, for adaptive patches.
- The attack success rate of the patches was lower in real-world implementations, 28% when NutNet was deactivated and just 11% when it was activated.
- This study further illustrates the significance of the training strategy. Patches trained with more data significantly outperformed those trained with less data. The dataset size, the patch's placement, and the number of patches were important factors when attacking the model.
- Interestingly, applying two patches rather than one increased the attack success rate by approximately 5%.
- Adaptive patches were relatively well protected from detection, despite their weaker attack effectiveness. An example of effective adaptive patches is showcased in Fig. 1.

This document will first proceed to section II, which details the background of the theoretical concepts of adversarial patch attacks, defense models, and the object detection models. Section III outlines the approaches taken within the study related to patch training, digital, and physical attack testing as well as the application of adaptive loss terms. Section IV covers the reports on the experimental results, and Section V follows with a comprehensive discussion on the findings. Finally, Section VI concludes with a summary of the findings and recommendations for future work.

(a) (b)

Fig. 1. Visual results of NutNet defending against an adaptive adversarial patch. In (a), the placed patch is not an adaptive patch and the NutNet locates the patch and masks it leading to a failed attack. In (b), the same image contains an adaptive patch leading to a failure in detection from the defense model and a successful attack on the object detector.

II. BACKGROUND

A. Object Detection Models

In order to attack a detection model, we need to understand how the detection model works. Many object detectors' use a two-stage detection model. Faster Region-Based Convolutional Neural Network (Faster R-CNN) uses this architecture, where region proposal and classification are handled in separate steps [8]. This means that the detector will first divide an input image into small regions and propose regional detection probability, then the next stage is to identify the object in the proposed area [4]. There are also other types of object detectors that use for example the visual transformer (ViT) architecture. According to [9], the ViT model works by also dividing the image into small boxes. However, what makes it different is that the model uses a Simple Feature Pyramid that takes the final image and makes different sizes of the image to better detect smaller and bigger objects. The ViT does not perform any classification and instead feeds it's output to an object detection model like Mask R-CNN. This model could also be considered as a two stage detection model. YOLOV2 is also a convolutional neural network that uses a one-stage detection algorithm instead of a two-stage detection algorithm, thus making it faster [4], [8]. The one stage object detection framework formulates detection as a regression problem. The input image is divided into a 32×32 grid, where each grid cell is responsible for predicting a fixed number of bounding boxes and each box consists of a vector. In [7] the components of the vector are given by:

$$[x_{\text{offset}}, y_{\text{offset}}, w, h, p_{\text{obj}}, p_{\text{cls1}}, p_{\text{cls2}}, \ldots, p_{\text{clsn}}] \tag{1}$$

where:

- $x_{\text{offset}}, y_{\text{offset}}$: Offsets of the bounding box center relative to the grid cell position.
- w, h: Width and height of the bounding box, relative to anchor box.
- p_{obj}: Object score, indicating the confidence that the anchor box contains an object.
- $p_{\text{cls1}}, \ldots, p_{\text{clsn}}$: Class scores for n object categories, predicted using cross-entropy loss.

All these predictions are generated in a single forward pass through the convolutional neural network, enabling real-time detection. The YOLOv2 model processes the entire image frame at once [7]. Fig. 2 showcases the process of YOLOv2. YOLOv2 is a neural network trained on visual data focusing on detecting objects. Therefore, an effective adversarial patch must deceive the network by introducing unusual pixel patterns that the object detector can not understand causing it to misinterpret or entirely overlook the input.

B. Adversarial Patch Attack

Adversarial attacks against object detectors generally aim to either hide objects or cause incorrect detections. There are many methods for performing adversarial attacks like creating

adversarial glasses that mislead facial recognition models as shown in [10]. In this project, we focus on patch-based attacks, where a small adversarial patch is added to the image to fool the detector. As a representative method, we follow the approach described in [7], which optimizes a patch using a total loss function designed to mislead YOLOv2. The total loss function is given by:

$$L = \alpha L_{\text{nps}} + \beta L_{\text{tv}} + L_{\text{obj}} \tag{2}$$

where:

- L: total loss used to train the adversarial patch.
- L_{nps}: non-printability score, encourages printable colors.
- L_{tv}: total variation loss, encourages smooth color transitions.
- L_{obj}: objectness loss, hides the object from detection.
- α, β: weight coefficients.

L_{nps} describes the non-printability of the patch. In order to create a patch that could be printed and used in real life attacks, the patch needs to have printable colors. This function for calculating the L_{nps} is described by [7] as follows:

$$L_{\text{nps}} = \sum_{p \in P} \min_{c \in C} \|p - c\| \tag{3}$$

where:

- P: the set of all pixels in the patch.
- C: the set of printable colors.
- p: a pixel color in the patch.
- c: a printable color from C.
- $\|p - c\|$: Euclidean distance between two colors.

In [10] and [7] the L_{tv} term describes the smoothness in transition between pixels to ensure the patch looks normal to the detection model as well as encouraging neighboring pixels to have the same colors. The term penalizes high-frequency variations between neighboring pixels. In both [10] and [7] the total variation loss is given by:

$$L_{\text{tv}} = \sum_{i,j} \sqrt{(p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2} \tag{4}$$

where:

- $p_{i,j}$: pixel value at row i, column j in the patch.

According to [7] the object loss function tries to minimize the object detection score. The YOLOv2 model tries to detect and locate an object but with the help of this term the patch will weaken the object detection score. In [7] the object detection function is given by:

$$L_{\text{obj}} = \max(p_{\text{obj}_1}, p_{\text{obj}_2}, \ldots, p_{\text{obj}_n}) \tag{5}$$

where:

- p_{obj_k}: object probability score for the anchor point k.
- n: number of anchor points in the detection output.

All these terms are meant to be minimized during the training process. They are minimized by sending a dataset to a training model and then the training model outputs a patch that is applied on the same training dataset. The patched images are feed to the object detector that gives feedback to the training model. With the help of the feedback the loss function is calculated and then the gradient of the loss function is calculated, which describes how each pixel needs to be changed, and lastly update the patch according to the results of the calculated gradient.

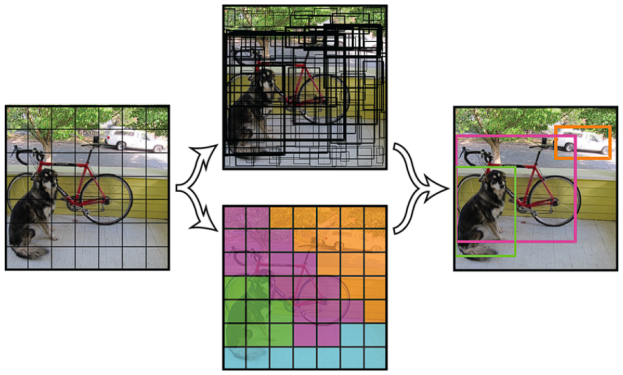


Fig. 2. This figure is taken from [7]. The figure showcases how the YOLOv2 models first takes an image and divides it into small boxes, which the model then processes in one step to classify and locate the object or person. When located and identified, bounding boxes are centered around the objects.

C. Defense models

Object detectors' main task is to locate and identify objects. This leaves a vacuum for adversarial attacks to penetrate models by only focusing on object misclassifications or object detection. If object detectors are integrated with defense models it would seal the vacuum, and adversarial attacks could be detected and handled to defend the object detector. The defense model NutNet was proposed in [4]. The NutNet model works as a protection layer that is based on out-of-distribution detection (OOD). NutNet tries to locate if there is any patch applied to the image and mask it, as shown in Fig. 1, in order to prevent successful attacks [4]. The defense model starts by dividing the image into small grids sized 13x13 (default setting), 26x26 or 52x52 pixels [4]. Then every grid is reconstructed by an autoencoder that has been trained on clean images. This means that the grids containing a part of a patch will be harder to reconstruct due to their unusual texture. The reconstruction error in [4] is given by:

$$\min_{x \sim P_c,\, x' \not\sim P_c} \text{MSE}(x, E(x)) - \text{MSE}(x', E(x')) \tag{6}$$

where:

- $x \sim P_c$: a clean image patch sampled from the clean data distribution.
- $x' \not\sim P_c$: an adversarial or noisy patch that is out-of-distribution.
- $E(\cdot)$: an autoencoder trained to reconstruct clean patches.

- MSE$(\cdot, \cdot)$: the mean squared error.

The autoencoder is trained to minimize the reconstruction error of clean images while maximizing it for patched images [4]. The defense model is later strengthened against stealth attacks that adapt to the reconstruction method. In [4], they claim that if the patch is adaptive to the reconstruction error, then during training of the patch, the gradient loss of the patch will not be optimized to bypass NutNet and attack the object detector model, but could be optimized to only one of the models.

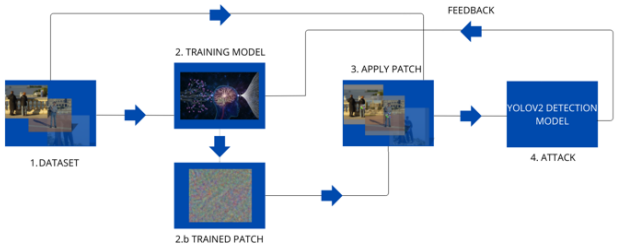


Fig. 3. Overview of the patch training process. The dataset is fed to the training model to train a patch. The patch is later applied on the dataset and used for attacking the YOLOv2 object detector. Once the attack is performed a feedback is sent to the training model and based on the feedback each pixel in the patch is updated.

III. METHODOLOGY

A. Patch Training

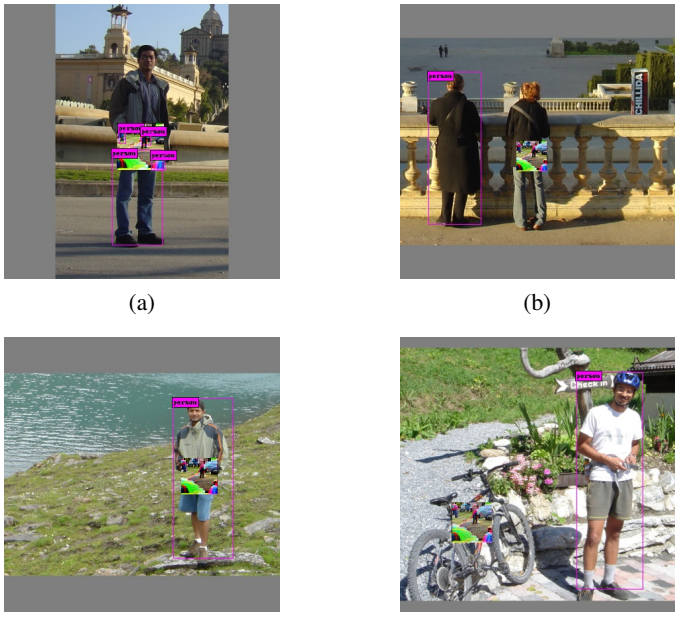
For all patch training the INRIA dataset was used. The data consists of pictures of people that are located in different environments, like inner cities, forests, and open field areas. In order to create a robust patch that can work in different environments with distinct lighting, background, and objects, the training data must contain many scenarios to increase robustness. First, the patch training module used a pre-selected patch consisting of 300x300 gray pixels. The gray patch is not trained and does not perform any successful attacks, it is solely used as an initial starting point for the optimization. The methods used for patch training went as follows:

- Train the gray patch on 30 INRIA images based on eq.(2).
- Train the gray patch on 298 INRIA images based eq.(2).
- Train a patch taken from [7] on the 298 INRIA images based on eq.(2).
- Train the three patches mentioned above on eq.(8) separately.
- Train the patch from the second point and the third point based on eq.(8) with data collected by us in different KTH environments.

An overview of the training process is showcased in Fig. 3. All patches were trained for different numbers of epochs: 100, 300 and 500.

B. Digital Patch Attack

There are many ways of attacking YOLOv2. In order to be consistent, if more than one patch is applied then the patch pixel size should vary, but the total amount of attacked pixels needs to remain the same, whether it is one patch or two. The attack process went as follows:



(a) (b) (c) (d)

Fig. 4. These are visual results given by YOLOv2 after it is attacked by the patch trained on only 30 pictures from the INRIA data set. As seen in (a) the person is visible but detected as 4 people. In (b) the person with the patch is not visible but the person without the patch is detected. In (c) and (d) the person is detected, where in (d) the bike is not detected.

- Create a patch trained on a set of data (INRIA or collected data)
- Apply the trained patch on the same data. The patch had to be resized from 300x300 pixels to 20% of the object size.
- Attack the YOLOv2 detection model, when NutNet is activated/deactivated, by sending the data with patches applied.

After the attack, visual defense results as well as numerical results were showcased for documentation. An example of the visual output is shown in Fig. 4. This method is used to only attack YOLOv2 without the use of any defense mechanisms. In [4], they introduce a protection layer before the detection model is fed with the attacked images. This makes it harder to trick YOLOv2, and the patch needs to be adjusted to bypass the protection layer as well as attack YOLOv2 successfully. The procedure for the attack is adjusted where the images are preprocessed by NutNet and then the output of NutNet will be fed to YOLOv2. The detection model was attacked with and without the protection of NutNet, and the results were compared.

Inspired by the work in [4], we wanted to test the capability of the NutNet defense model by implementing an extra term for the adaptation. This term is L_{recon}, the reconstruction loss. In their work, they claim it would be difficult to trick both models thus we wanted to test whether this method will work in practice. This means that the patch need to be adapted in order to both bypass the defense model and also fool the object detector. L_{recon} is a function trying to minimize the mean squared error between the patch and the reconstruction

of the patch. The mathematical equation for the reconstruction loss is given by:

$$L_{\text{recon}} = (p - E(p))^2 \tag{7}$$

where:

- p: adversarial patch image.
- $E(p)$: reconstruction of the patch by NutNet's autoencoder.

With the implementation of the L_{recon} term and λ, the weighting factor that controls the influence of the reconstruction loss, the modified total loss function is given by:

$$L = \alpha L_{\text{nps}} + \beta L_{\text{tv}} + L_{\text{obj}} + \lambda L_{\text{recon}} \tag{8}$$

(a) (b)

Fig. 5. The pictures in this figure showcase how we performed some of the real world attacks. In (a) we used a printed patch in an outdoor environment and in (b) we used a tablet screen in an indoor environment. Both images are taken at KTH.

C. Real-World Patch Attack

Inspired by the work in [7], we tested our trained patch in real life in order to understand the performance of the patch outside the virtual world. Attacks performed digitally are usually simulations of real world attacks. Most detection models are used in the real world and creating the best possible attack is mostly aimed for real world scenarios, thus it is important to attack both digitally and in real world scenarios. For the real world attacks, the same patches used for the digital attacks were printed by a normal printer in 2 different sizes, 19.5x19.5 cm and the normal patch size, 9.75x9.75 cm. In [7], they used a bigger patch printed on a cardboard panel, but in this thesis, we are using normal A4 paper sheets in order to study the accessibility of these patches. Once the patches were printed, numerous tests were conducted.

- Each person held a different patch, where one had the 9.75x9.75 cm. patch, and the other had the 19.5x19.5 cm patch. Both people were visible in the frame.
- Only one person holding a single patch, but one or more people may be in the frame.
- To test the robustness, different background and lighting conditions were used during testing.

To test the non-simulated performance of the patch, a video was recorded with one of the scenarios mentioned above, and later the video was converted to frames by an online MP4 to PNG converter. Once the data is obtained, the frames are fed to the YOLOv2 and NutNet. With the help of this method, we could test the performance of the patch in real-life conditions, examples of some attacks are illustrated in Fig. 5.

IV. Results

The results of the attacks on YOLOv2 varied depending on the different methods that were used. The study also involved real-world application of the patches, as well as downloading patches from other works and retraining them. The attack performance is measured by percentage points. In order to know if we have a successful attack we send both the patched image and the clean image to the object detector and compare the size of the bounding box for the object. With the help of the bounding boxes we calculated the Intersection of union (IoU) which is given by $\frac{\text{area of overlap}}{\text{area of union}}$. If IoU < 0.5 then that counts as a successful attack because we have manipulated the output of the object detector.

Firstly, we attacked only digitally with a patch trained on only 30 INRIA images. The second row of Table I describes the attack success rate of this method. At 500 epochs we obtained an optimal patch that resulted in an attack success rate of 21% when NutNet was not activated.

In order to improve the performance we used the full INRIA dataset containing 298 images, the results are shown in Table I row three where we obtained an attack success rate of 67%. For further improvement of the attack success rate we took an already trained patch from [7] and retrained the patch on 30 and 298 INRIA images. This method boosted the attack success rate up to 75% when NutNet was deactivated.

TABLE I. Success rate of patch attacks on YOLOv2 when NutNet is deactivated. The ...% represents the success rate of the attack. The training of the patches in this table are based on eq.(2).

Patch Type	100 Epochs	300 Epochs	500 Epochs
Trained on 30 images	13.0%	18.0%	21.0%
Trained on 298 images	37.0%	52.0%	67.0%
Printed patch (30 images)	7.0%	10.5%	12%
Printed patch (298 images)	18.0%	26.0%	28.0%
Pretrained patch (30 images)	45.0%	55.0%	72.0%
Pretrained patch (298 images)	40.0%	62.0%	75.0%

(a) (b)

Fig. 6. The patches represented in this figure are trained on different conditions and on 298 INRIA images. (a) is patch (b) but retrained based on eq.(8), while (b) started as a gray patch and trained based on eq.(2). These patches were trained for 500 epochs.

After training patch (b) from Fig. 6 based on eq.(2), the same patch was trained based on eq.(8) to make an adaptive

patch that could attack both YOLOv2 and NutNet. When comparing the result from Table I row seven column four with Table II row seven column two we see the attack success rate drops by 25 percentage points. This means that training the adaptive patch based on eq.(8) leads to performance drop. The real world attacks also prove to be less effective against the YOLOv2 when NutNet was activated. From Table I row five column four with Table II row five column two we see a performance drop of 17 percentage points. In Table II the seventh row is the most interesting result. Despite NutNet being helpful if two patches were applied instead of one, the attack success rate improved by five%. To clarify the area covered by two patches is still the same as one patch but lead improvement in attack.

TABLE II. Success rate of patch attacks on YOLOv2 when NutNet was activated. The ...% represents the success rate of the attack. The training of the patches in this table are based on equation 2 and later retrained based on eq.(8) for 500 epochs.

Patch Type	Success Rate
Trained patch (30 images)	0.0%
Trained on 298 images	37.0%
Printed patch (30 images)	0.0%
Printed patch (298 images)	11.0%
Pretrained patch (30 images)	43.33%
Pretrained patch (298 images)	49.6%
Trained on 298 images, 2 patches applied	42.0%

In Fig. 6 we see the difference between the adaptive patch (a) and a normal patch (b). (a) is adapted to by pass NutNet and still be effective against YOLOv2 while (b) is only effective against YOLOv2. The most noticeable differences are the pixel variations in the patches, where in (a) the patch is has less vibrant colors while in (b) the patch is more colorful.

The real world attacks gave underwhelming performance. The main focus in the project was to use printable patches. In Table I row four and five we see that the printed patches performed worse than digitally applying the patch and it further decreased when NutNet was activated which is seen in Table II row four and five. In Fig. 5 we can see the different types of methods used during attacks, the laptop screen, printed patch, and tablet screen. The best performance was achieved using the tablet screen, with an attack success rate of 20%, the laptop screen gave an attack success rate of 0% and the printed patch gave an attack success rate of 11%, as seen in Table II row five, column two.

V. Discussion

As expected, the patch attacks had meaningful variations in performance. We found that patches trained on the large dataset achieved significantly higher success rates than those trained on the smaller dataset. This demonstrates that the diversity and size of the training data contribute to the effectiveness of adversarial patches. However, the success rates decreased substantially with the introduction of NutNet's defense model. As demonstrated in the results, there are many cases in which NutNet covers a large portion of the image in its attempt to mask the patches, leading to image corruption and, in some cases, failure to detect specific objects in the image due to their lack of clarity after the modifications. This demonstrates NutNet's ability to detect and prevent patch attacks. However, it struggles to handle such situations effectively and consistently. While we note that success rates were high in digital environments, real-world attacks were unsurprisingly less successful. These results are likely related to changes in lighting, background, and the inaccuracy and clarity of the patches. The results of patch attacks are consistent with previous work [2], [4], [5]. These patches significantly reduce the effectiveness of object detection model, but their implementation in the real world is much more difficult. The effectiveness of the adaptive patches remained high thus weakening the claim stated in the introduction about creating adaptive patches that could both bypass the defense model and fool the object detector [4].

This study faced several challenges and limitations. It used printed patches on paper instead of stronger materials, and it was not easy to implement real-world testing for several reasons, most notably lighting. Additionally, there was limited availability of physical locations for testing. The real-world attacks were also difficult to perform because the attacks had to be filmed and later cut into frames instead of continuously feeding the object detector a video. Relying on a single dataset (INRIA) also had a negative impact and limited the generalization of the results. Further work should integrate scaling training to multiple datasets, evaluate various object detection algorithms, and create more realistic adversarial patches using GAN techniques. Investigating defenses other than NutNet could also deepen the understanding of the general vulnerabilities of defense models.

As mentioned, the study could have used other defense models to test and train patches that could bypass more than one defense model and still manipulate the output of the object detector. For future work, the study could work on the NAPGuard defense model presented by [5]. This defense method focuses on identifying high-frequency components to locate the patch. An adaptive patch could be trained where the loss function (2) would have an added term, L_{freq}. This loss is computed by applying a Fourier transform to the patch and applying a filter that masks high-frequency components, which correspond to rapid variations in contrast between neighboring pixels. By minimizing this loss function, the patch would be adapted to the NAPGuard model and consist mostly of low-frequency magnitudes. This method might allow the patch to go undetected and bypass the defense model, thus attacking the object detector. This assumption is made considering the constraints of NutNet that we found in our results, about cases where some adaptive patches evaded detection successfully. It may seem that this loss term is not different from the L_{tv} term. However, the L_{tv} term handles color variations in the patch while L_{freq} handles the frequency components with the help of fourier transforms.

VI. Conclusion

This thesis demonstrated weaknesses in modern object detection models and defense models, specifically YOLOv2 and NutNet, against adversarial patch attacks. Experiments

showed that adversarial patches can achieve high success rates, sometimes reaching 50%. Among the most remarkable findings was the success of adaptive attacks, where patches retrained with a reconstruction loss term bypassed NutNet and attacked YOLOv2 successfully. Advanced defense mechanisms such as NutNet have proven successful and effective in detecting attacks, but also shown weaknesses in countering them without disrupting or compromising the image content. A larger and more diverse training dataset effectively trained better patches, while real-world experiments revealed complex challenges such as varying lighting conditions and environmental changes. These results confirm that despite improvements in defense methods, they are still not perfect and effective enough against adaptive adversarial patch attacks. Future research should focus on studying other defense models and creating adaptive patches that could train on more datasets to enhance robustness.

ACKNOWLEDGMENT

The authors would like to thank our supervisor, Mauricio Byrd Victorica, for his support, helpful advice, and guidance during this project. His knowledge and feedback helped us improve our work and solve challenges along the way. We appreciate the contribution of the meetings and discussions towards improving the quality of this project.

REFERENCES

[1] N. Ji, Y. Feng, H. Xie, X. Xiang, and N. Liu, "Adversarial yolo: Defense human detection patch attacks via detecting adversarial patches," 2021. [Online]. Available: https://arxiv.org/abs/2103.08860

[2] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," 2018. [Online]. Available: https://arxiv.org/abs/1712.09665

[3] X. Liu, H. Yang, Z. Liu, L. Song, H. Li, and Y. Chen, "Dpatch: An adversarial patch attack on object detectors," 2019. [Online]. Available: https://arxiv.org/abs/1806.02299

[4] Z. Lin, Y. Zhao, K. Chen, and J. He, "I don't know you, but i can catch you: Real-time defense against diverse adversarial patches for object detectors," 2024. [Online]. Available: https://arxiv.org/abs/2406.10285

[5] S. Wu, J. Wang, J. Zhao, Y. Wang, and X. Liu, "Napguard: Towards detecting naturalistic adversarial patches," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 24 367–24 376.

[6] X. Wang and K. Wang, "Generating visually realistic adversarial patch," 2023. [Online]. Available: https://arxiv.org/abs/2312.03030

[7] S. Thys, W. Van Ranst, and T. Goedemé, "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*, 2019.

[8] W. Tan, Y. Li, C. Zhao, Z. Liu, and Q. Pan, "Doepatch: Dynamically optimized ensemble model for adversarial patches generation," 2024. [Online]. Available: https://arxiv.org/abs/2312.16907

[9] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection," 2022. [Online]. Available: https://arxiv.org/abs/2203.16527

[10] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1528–1540. [Online]. Available: https://doi.org/10.1145/2976749.2978392