

A study of the robustness of DPatch attack

Marius Larsen

2022/01/18

Abstract

Object detectors has become the state-of-the-art for an increasing number of safety-critical tasks, including autonomous driving. Research has shown that these object detectors are vulnerable to well-crafted perturbations to input images, called adversarial examples. One of the upcoming ways of perturbing these input images is by applying a patch to the scene, which can be deployed in the physical world. DPatch is such an attack and showed initially great attack and cross-model transferability, making it a dangerous attack.

The DPatch attack generates a small, placement independent patch, which attacks the object detectors' RoI, which can lead to both object evasion and fabrication, making it capable of fooling computer vision tasks.

Nevertheless, there has been risen doubt whether the DPatch attack is as robust as first claimed when proposed. Through my studies, I will show examples that the DPatch attack is not robust, and that it fails to evade real objects in the scene, despite being able to produce a patch labeled as the targeted class.

Sammen drag

Objekt detektorer er blitt utbredt i et økende antall sikkerhetskritiske oppgaver, inkludert i selvkjørende biler. Forskning har bevist at disse objekt detektorene er utsatt for nøye beregnede forstyrrelser på input bilder. En av metodene for å genere forstyrrelser er gjennom lapper, som dekker deler av bildet. Disse lappene har egenskapen av å kunne printes ut og bli utplassert i den fysiske verdenen. Et av disse angrepene som utnytter lapper, DPatch, har vist å ha angreps- og overførbarhets-egenskaper da angrepet først ble foreslått. Dette gjør DPatch til et svært farlig angrep om det skulle bli brukt i den virkelige verdenen.

DPatch angrepet genererer en liten, plasseringsuavhengig lapp, som angriper objekt detektorens forslag til regioner av interesse. Dette kan lede til at ekte objekter unngår deteksjon, eller at objekter som ikke eksisterer blir detektert.

Til tross for det store potensialet som først ble beskrevet, har det blitt reist tvil om DPatch angrepet er like robust som først antatt. Gjennom mitt studie vil jeg vise eksempler av at DPatch angrepet ikke er robust, da det feiler på å unngå ekte objekter, til tross for å generere en lapp som blir detektert som mål-klassen.

Contents

Abstract	i
Sammendrag	ii
Contents	iii
Figures	v
Tables	vi
Acronyms	vii
Glossary	viii
1 Introduction	1
2 Background	3
2.1 Convolutional Neural Networks	3
2.2 Image Classification	3
2.3 Bounding boxes	3
2.4 Object Detectors	4
2.4.1 YOLO	5
2.4.2 Faster R-CNN	5
2.5 Adversarial Examples	5
2.6 Performance Metrics	5
2.7 Non-max suppression	7
2.8 Region of Interest Pooling	7
2.9 Attack types	7
2.10 Transferability	8
2.11 Attack Constraints	8
2.11.1 Distance Metrics	8
2.11.2 Patches	9
2.12 DPatch	9
3 Related Work	10
3.1 Adversarial Patch	10
3.2 DPatch	10
3.3 You Cannot Easily Catch Me: A Low-Detectable Adversarial Patch for Object Detectors	11
3.4 RPAAttack: Refined Patch Attack on General Object Detectors	11
3.5 Scale-Adaptive Adversarial Patch Attack for Remote Sensing Image Aircraft Detection	11
3.6 On physical adversarial patches for object detection	11

3.7	Attack analysis	12
4	Method	15
4.1	Motivation	15
4.2	Research Questions	16
4.3	Research Methodology	16
4.4	Implementation	16
5	Results and discussion	18
5.1	Discussion	21
6	Conclusion	22
	Bibliography	23

Figures

2.1	Bounding box example	4
2.2	Intersection over Union illustration	6
4.1	An 80x80 patch for targeted attack on Faster R-CNN	17
5.1	Comparison of patches from this implementation and the original paper	19
5.2	Results of DPatch which confirms some of the problems described in this chapter	20
5.3	Bounding boxes not suppressed, even when the object detector predicts the target class toaster with fairly high confidence	21

Tables

3.1	Analysis of attacks, performed in the pre-study to find attacks with potential research contributions.	12
-----	--	----

Acronyms

- AP** Average Precision. 5–7, 11
- ART** Adversarial Robustness Toolbox. 16
- CNN** Convolutional Neural Network. 3–5, 7, 12
- DNN** Deep Neural Network. 1
- IoU** Intersection over Union. 6, 7
- mAP** mean Average Precision. 4–7, 11, 14, 16
- NMS** Non-max suppression. 7
- NN** Neural Network. 1
- RoI** Region of Interest. i, 7, 9, 10, 13
- RPN** Region Proposal Network. 5, 7, 9
- SSD** Single Shot MultiBox Detector. 4, 16
- YOLO** You Only Look Once. 4, 5, 9–11, 13, 16

Glossary

AAAI Association for the Advancement of Artificial Intelligence, an international AI conference. 22

Idun A high performing computer cluster maintained by NTNU. 17

Chapter 1

Introduction

Deep Neural Networks (DNNs) has achieved an incredible performance in various computer vision tasks. Leading to the rapid development of object detectors and image classifiers. Furthermore, they have been deployed in several safety-critical tasks like autonomous driving [1–3] and healthcare systems [4, 5]. Meanwhile, the rapid development of DNNs proceeded, it was unfortunately discovered that the Neural Networks (NNs) are prone to inputs with non-random imperceptible perturbations [6]. Szegedy et al. managed to make a NN misclassify an image by maximizing the error of the network’s predictions. The first examples of adversarial input attacks included perturbations to the entire image, which could fool image classifiers. These attacks were extended further to target object detectors. Furthermore, patch-based attacks were explored, appose to attacks that perturbed the entire image. Patch-based attacks only target parts of the image and proved to be able to shift the attacks from only being digital to be implemented in the physical world.

Zhang and Li [7] describe various sets of safety-critical tasks. These tasks are especially susceptible to adversarial input-attacks, as the NNs are trusted to act without human inference. They describe the reality of the decision-making part of autonomous driving as almost black-box. This black-box characteristic means that the human drivers of the autonomous vehicle have no insight in how and why the NNs makes their decisions. Following the description of the autonomy in self-driven cars, human drivers will rely fully on the decisions made by the car. Thus there is no doubt that there will be fatal consequences if one can, with high certainty, fool cars to misread traffic or signs while driving. This brings concerns when today’s cars can be deployed with auto-pilots, and the topic of adversarial examples is still under investigation with no clear defense to mitigate the effect of all adversarial input attacks. Thus the topic brings a lot of interest for me, as it is critical to explore the realm of adversarial input attacks before they are executed in the real world.

Through this paper, I perform an empiric study of a patch-based attack, DPatch [8], which shows great potential in the original paper. The results of the empiric study imply that the attack, despite being promising, lacks robustness to perform

a targeted attack to make the detector evade real objects. Extensive evaluations show that the attack manages to fabricate fake targeted objects with great certainty.

Chapter 2

Background

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), has become the model architecture of choice for any task including analyzing images [9]. Furthermore, CNN is used to realize most modern object detectors and has become the state of the art in terms of classification and segmentation tasks.

CNNs utilizes features in images to extract feature maps, which are further analyzed to make predictions [10].

2.2 Image Classification

Image classification has become the fundamental problem for computer vision tasks. The task of image classification is given an image, the classification models are to provide a list of the predicted classes of the object dominating the image and their respective confidences.

2.3 Bounding boxes

Taking computer vision a step further from classification alone, certain tasks depends on detecting and localizing multiple objects within an image or a single frame from a video stream.

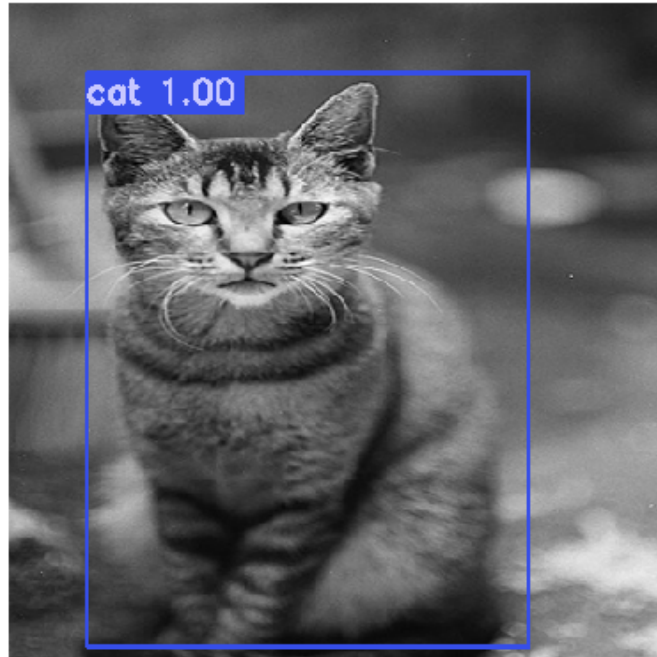


Figure 2.1: A cat with a predicted bounding box detected by a Faster RCNN object detector.

2.4 Object Detectors

Object detection combines the concept of bounding boxes and classification, where we introduce localization of the object to be classified in the image or video. The objects localization can be described through the bounding boxes proposed by the object detectors.

Throughout the years, several object detectors have been developed to perform computer vision tasks. In particular, You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD) [11] and Faster R-CNN are leading object detection models in computer vision tasks [12].

Object detector models can be further divided into one- and two-stage models. Where Faster R-CNN (faster region-based CNN)[13] is an example of a model with two-step architecture. Faster R-CNN extracts possible object regions and performs classifications of the object within the proposed region in two separate steps. While YOLO, a one-stage model, extracts the proposed bounding boxes and performs classification at the same time.

The one-stage models trades off the accuracy of smaller objects for higher speed, giving them the property to achieve real-time object detection. Two-stage models aim for higher mean Average Precision (mAP) scores while reducing the detection speed, making it generally less viable for tasks requiring a high update rate, such as autonomous driving. Faster R-CNN is several times faster than its

predecessor, Fast R-CNN, making it near real-time detector, as described in [13].

2.4.1 YOLO

"You Only Look Once" is the core principle of YOLO, making it an one-stage object detector. YOLO both propose regions for objects and classify the regions in the same step, achieving real-time detection speed with a cost of lower accuracy.

There have been several iterations of the object detectors, where YOLOv1 [14] directly returns the bounding boxes and classification at the output layer. YOLOv2 [15] removes the fully connected layer by adding batch-normalization. This iteration improved the accuracy of the object detector. Detecting small objects was still a problem with YOLO, this was improved in YOLOv3 [16] where multi-scale predictions were used. Lastly, several optimizations were proposed for YOLOv4 [17] to achieve a state-of-the-art mAP and speed.

2.4.2 Faster R-CNN

Faster R-CNN is a two-stage object detector. The first stage of the detector is to propose regions where objects exist. The second stage classifies said regions. To realize the first stage of Faster R-CNN, a RPN is used to output a feature map which again is used as input to a fully connected neural network for object classification.

2.5 Adversarial Examples

CNNs and object detector has been known to be prone to imperceptible non-random perturbations to input images, leading to misclassification [6]. These perturbed examples were termed adversarial examples.

Adversarial examples and why they are effective were further explored in [18]. [18] argues that the reasoning for why adversarial examples are effective is due to the linearity in neural networks. This was a new argument from the focus on nonlinearity and overfitting which was the previous hypothesis. This is further backed up by the generalization of the attacks which makes them transferable across datasets and models.

2.6 Performance Metrics

When building an object detector, there is a need to measure how accurate the predictions of the classification model are. Some basic measures which are widely used are precision, recall and F1. Later, the more advanced metrics AP and mAP have become a more all-around metric heavily used in the field.

Common for these three measures is that they utilize true/false negatives and positives.

Precision is in place to measure whether the model avoids mistakes while classifying a specific class.

$$Precision = \frac{TP}{TP + FP}$$

where TP = True Positive and FP = False Positive

Recall measures how well the model finds all the positives. If the model avoids mistakes of classifying a given class as other classes, the model has a high recall.

$$Recall = \frac{TP}{TP + FN}$$

where TP = True Positive and FN = False Negative

F1 takes in account both precision and recall.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

In some scenarios, a model with high recall at the cost of precision is beneficial. More precisely, when the cost of false negatives are extremely higher than false positives. E.g for autonomous vehicles, false negative on a pedestrian crossing the road can't be allowed to happen. Thus, focusing on high recall would be more beneficial.

For a better comparison of prediction models, a precision-recall curve can be analyzed to evaluate the trade-off between precision and recall. This introduces another performance metric, Average Precision (AP). Which can be further used to calculate mAP. AP and mAP has become the most popular metrics for evaluating prediction models, and have proved to be great at evaluating models against each other on the same datasets. The metrics gained heavy popularity through the usage of the metrics in the challenges COCO¹ and PASCAL VOC².

Firstly, let's look at Intersection over Union (IoU), see Figure 2.2 for visualization. IoU is a metric to evaluate the intersection of the area of the ground truth and the area of the predicted mask. Where IoU will be a number between 0 and 1, and closer to 1 means a more accurate prediction.


$$IoU = \frac{\text{Area of intersection}}{\text{Area of union}}$$


Figure 2.2: Intersection over Union illustration

¹<https://cocodataset.org/>

²<http://host.robots.ox.ac.uk/pascal/VOC/>

By defining a threshold, here α , for the intersection to determine whether to mark the predicted bounding box to be correct or not, we can calculate the IoU.

$$AP@{\alpha} = \int_0^1 p(r)dr, \quad (2.1)$$

where α is the threshold, and $p(r)$ is the Precision-Recall curve

For each class, AP can be calculated. For COCO, this means at one specific α , one can calculate 80 different APs, one for each of the 80 classes. To evaluate a model given all classes in the dataset, all AP values can be averaged to get the resulting mAP.

$$mAP@{\alpha} = \frac{1}{n} \sum_{i=1}^n AP_i, \quad (2.2)$$

sum over all n classes, and α is the IoU-threshold

2.7 Non-max suppression

Non-max suppression (NMS) is an algorithm to solve the problem of multiple overlapping proposals of bounding boxes of the same classes. In short, NMS iterates over all classes and compares IoUs to discard the proposals of lowest confidence when the IoU is above a predefined threshold. This makes the less confident proposals regarded as false positives.

2.8 Region of Interest Pooling

When processing an image, it's first preprocessed in a trained CNN to generate a feature map. This feature map will then be inputted into a Region Proposal Network (RPN) to get proposals of Region of Interest (RoI). These RoIs describes where the RPN predicts there exists object, and are then inputted into the RoI pooling

RoI pooling was needed for object detectors based on fully connected CNNs, since they expect a fixed-sized feature map. RoI pooling handles the output of the RPN which will be of different shapes.

2.9 Attack types

The different attacks can be divided into one of three attack types: Black-box, white-box and gray-box attack. The difference is what kind of information about the target that is available to the attacker. If the attack has full access to the target

model, including weights, dataset, input and output, the attack is classified as a white-box attack.

Furthermore, if only the input and output, optionally with the scores of the output, the attack can be classified as black-box attack. Effective black-box attacks can have severe consequences in safety-critical tasks like autonomous vehicles, as the internal structures of the object detection models will less likely be publicly available. If an attacker can successfully attack given no information about the target, keeping the configuration and output of the object detector private no longer prove as any security measure.

2.10 Transferability

While the most effective attacks often came from white-box attacks, attacks often transfer to other models which the attack was not initially intended to attack [6, 18, 19]. This proves some serious concerns, as the attacker no longer may need to have any information about their target. Transferability can be thought of as the measure of how well an attack against a target can be performed on a separate target with no further configurations.

The transferability can further be specified into cross-model and cross-dataset, where the latter focuses on transferability of an attack meant for model A trained on dataset X can be used for a model similar to A, only trained on dataset Y.

2.11 Attack Constraints

As discussed, the perturbations are meant to be imperceptible for the human eye. To achieve this, attacks can implement certain constraints on how to perturbate the images. These constraints can seek to minimize changes in pixels, or how the images are to be rotated or transformed to generate an adversarial example [20].

2.11.1 Distance Metrics

The most common constraint of perturbations are the ℓ_p -norms ℓ_0 , ℓ_2 and ℓ_∞ [21]. These are distance metrics that can be used to analyze how perceptible the perturbations are. The ℓ_p -norms are a mathematical definition, and not a perfect measurement for how perceptible the perturbation will be for the human eye.

ℓ_p -norms are defined by

$$\|x - x'\|_p \quad (2.3)$$

where the p-norm is defined as

$$\|x\|_p = \left(\sum_{i=1}^I |x_i|^p \right)^{\frac{1}{p}} \quad (2.4)$$

These norms specify how close the adversarial examples are to the benign examples in terms of pixels changed.

- ℓ_0 seeks to minimize the number of pixels perturbed, with no limits on how much they change.
- ℓ_2 minimizes the Euclidean distance between the adversarial and the benign example, as Equation 2.3 becomes the Euclidean distance with $p = 2$. Meaning the ℓ_2 can stay minimized if there are many pixels with small changes.
- Lastly, ℓ_∞ only measures the maximum change of *any* pixel, with no regard to how many pixels changed up to this maximum.

2.11.2 Patches

Patch-based adversarial examples utilized small patches that are heavily perturbed, but limited to a small area of the image. These types of attacks quickly proved to have great attack effects on image-level classifiers, and they also contained the property to be able to attack physical real-world objects. To achieve a physical attack, the patch could be printed out and attached to the physical world, resulting in the detector failing to detect objects properly [22–24].

2.12 DPatch

[8] introduces DPatch, an adversarial patch-attack with a high attack effect with a limited-sized, location-independent patch. The attack focuses on two state-of-the-art object detectors, YOLOv2 and Faster R-CNN. DPatch has the possibility to perform both targeted and untargeted attacks, leading to devastating attack results. A DPatch trained on YOLO was proved to be effective on a Faster R-CNN based detection model, and vice-versa, meaning the attack contains great transferability potential. Due to the location-independent property, [8] claims the attack is practical for a real-world implementation of the attack. This was further explored in [23, 25].

The targeted attack aims to make the patch the only RoI, with a given label, thus breaking the object detector and making it fail to detect the other objects in the frame. Thus the targeted attack is with the purpose of evading objects in the frame. The untargeted attack on the other hand, seek to break the RPN in such a way that the object detector proposes objects that are not in the frame, simultaneously removing detection of actual objects. Thereby having the untargeted attack both be with the goal of evasion and fabrication.

Chapter 3

Related Work

Throughout the pre-study of this project, several attacks were researched to look for potential research contributions. Thus leading to the Table 3.1, which summarizes the attacks. Some of these attacks relate to this project, as they are state-of-the-art attacks and/or patch-based attacks which share attributes with the DPatch attack. The DPatch attack will also be further elaborated, as well as an attack which has similarities with DPatch and attacks which are compared to DPatch.

3.1 Adversarial Patch

Google’s Brown et. al [22] introduced adversarial patch attacks to fool classifiers to predict a targeted class. This was a new way of perturbing the images, where past attacks against classifiers were based on noise distributed across the entire image. The adversarial patch aims to be robust and universal, being able to attack any scene in a targeted fashion.

3.2 DPatch

[8] presents an adversarial patch attacking both the bounding box regression and the classifiers of YOLO and Faster R-CNN. The attack proves to be very effective with great attack effect in a white-box setting, as well as showing great potential in the black-box settings. One downside of the attack is its need for a large amount of iteration, resulting in a slow attack. The paper describes patches with sizes of 20x20 and 40x40 pixels with more than 180 000 iterations, where the saturation point is found to be 200 000 iterations.

The DPatch is generated by iterating over a dataset and changing the pixels of the patch for each iteration through back-propagation. This attacks in such a way that the targeted model focused on the patch, resulting in fewer RoIs which are mainly located near or at the patch.

DPatch is originally tested for cross-model transferability from Faster R-CNN to YOLOv2 and vice versa. No further examination of cross-models is tested and

is left for future work. The cross-model transferability of the attack was the basis for this project.

3.3 You Cannot Easily Catch Me: A Low-Detectable Adversarial Patch for Object Detectors

[26] discusses the perceptibility and the size of the patch in the DPatch attack. Furthermore, [26] proposes Low-Detectable Adversarial Patch (LDAP) attack, which splits the patch across the targeted image while removing the least important parts of the patch to make it less perceptible. Thus the patch is no longer a rectangular patch as the patch of the DPatch attack. [26] uses an adversarial patch detector, a simple detector network that is trained to detect adversarial patches. The LDAP is not detected by the adversarial patch detector, while DPatch is consistently detected. Furthermore, they show that DPatch needed the largest patch-size of the compared adversarial patch-based attacks. When comparing the detection rate of several patches, DPatch consistently scores the highest, meaning it was the most perceptible patch in the experiments.

3.4 RPAAttack: Refined Patch Attack on General Object Detectors

[27] proposes Refined Patch Attack (RPAAttack) and compares the patch-attack to DPatch. Here, DPatch successfully attacks the images with a large patch that covers 8,32% of the image. In comparison, the proposed RPAAttack perturbrates 0,05% of the same image, while achieving a lower mAP than DPatch.

3.5 Scale-Adaptive Adversarial Patch Attack for Remote Sensing Image Aircraft Detection

[28] proposes an attack, PatchNoobj, and benchmarks against DPatch when attacking YOLOv3. Here, the DPatch performs quite poorly, with almost no decrease in mAP and AP. [28] also discusses how the DPatch attack puts no restrictions on the changes of pixel values, which might put the pixel value out of the range of valid values for the benign image.

3.6 On physical adversarial patches for object detection

While DPatch often manages to fool the detection model to detect the patch as the targeted class with high confidence, it often fails to suppress the other detections. Lee and Kolter present a robust DPatch in [23], where they manage to fool detection models with a physical patch attack without having the patch covering the tar-

geted objects. Their approach is similar to the DPatch, but with a change in the optimization problem. One of the minor faults in the DPatch optimization problems are that for the untargeted attack, a target label is still required ($target_label = 0$ is used to distinguish the untargeted setting), see Equation 3.1.

Equation 3.1 and Equation 3.2 are inspired from [22], where the patch seeks to maximize the loss of a CNN when the patch is applied to an input image. [22] utilizes input images with random locations and transformations, to make the patch effective even with potential transformations when applied in a physical attack. While training the DPatch in an untargeted attack in Equation 3.1, the attack seeks to find a pattern \hat{P}_u such that the loss function, L , is maximized. L takes the input image, x , with a shift s and patch P . An apply-function, A , applies the patch to the image. Furthermore, L takes the true class label \hat{y} and the bounding box label \hat{B} . The targeted attack, Equation 3.2, on the other hand, aims to find the patch \hat{P}_t such that the loss function is minimized when given the target class \hat{y}_t and the bounding box label \hat{B}_t .

$$\hat{P}_u = \arg \max_P \mathbb{E}_{x,s} [L(A(x, s, P); \hat{y}, \hat{B})] \quad (3.1)$$

$$\hat{P}_t = \arg \min_P \mathbb{E}_{x,s} [L(A(x, s, P); \hat{y}_t, \hat{B}_t)] \quad (3.2)$$

3.7 Attack analysis

To find potential research contributions, several attacks were analyzed. Table 3.1 shows some of the attacks which were considered. Here, the white-box attacks would not be researched on whether they could be transferred to other models, which might be left as future work that could be looked into in this project. The black-box attacks which show great attack potential could be further examined if any known defenses would mitigate the attacks.

Another criteria for this project was that the attack which should be researched was meant to attack object detectors, not only classifiers. This was due to object-detection being more used in safety-critical tasks, such as autonomous driving.

Table 3.1: Analysis of attacks, performed in the pre-study to find attacks with potential research contributions.

Attack	Object Detector	Attack Type	Attack Goals
ZOO [29] is a zeroth order optimization based attack which directly estimates the gradients of the targeted Deep Neural Network.	✓	Black-box	Misclassification

DPatch [8] creates a patch which attacks the RoI to fool YOLOv2 and Faster R-CNN object detectors in a white-box setting. Also attacks in black-box settings where patches trained on YOLOv2 can successfully attack Faster-RCNN and vice versa.	✓	White-box Black-box	and	Misclassification, fabrication and evasion.
SpatialAttack [20] focuses on rotations and translations instead of the ℓ_p -norms	✓	Black-box		Misclassification
GenAttack [30] is a zeroth order optimization based attack, which reduces the number of needed queries drastically from the ZOO-attack.	✓	Black-box		Misclassification
Evaporate Attack [31] achieves 84% and 48% fooling rate on YOLOv3 and Faster R-CNN respectively by quering the black-box model and update the perturbations to achieve evaporation of the objects.	✓	Black-box		Evasion
Simple Transparent attack [32] is a simple manual attack which targets Machine Learning as a Service by making important part of the object transparent.	✗	Black-box		Evasion
DeepFool [33] attacks classifiers by producing a <i>minimal</i> perturbations by moving the predictions out of the correct decision boundary.	✗	White-box		Misclassification

Towards cross-task universal perturbation against black-box object detectors in autonomous driving [25] presents an universal adversarial patch which achieves cross-task, model and dataset transferability	✓	White-box	Evasion
Adversarial Patch [22] is an universal adversarial patch, attack classifiers to classify all images with the patch as the targeted label.	✗	White-box	Misclassification
LDAP [26] sparse the patch to have the patch avoid adversarial patch detectors, while reducing the mAP drastically.	✓	White-box	Misclassification and evasion
RPattack [27] determines the pixels with the most importance for attack and uses an ensemble to generate noise on a minimal amount of pixels.	✓	White-box	Evasion

Chapter 4

Method

4.1 Motivation

Adversarial patch-based attacks has proved to have great attack abilities and has become more developed in the later years. This makes it a very interesting field within the domain of adversarial input attacks, with its unturned stones and great potential for future attacks.

By looking into how an attack can be executed in a black-box fashion, a very dangerous part of these attacks can be examined and further researched. The reasoning behind the danger of black-box, or transferable attacks, lays within the fact that the attack needs little to no information of the target. This combined with some well-educated assumptions would make it hard for manufactures of autonomous vehicles to defend against these forms of attacks without rigorous implementing defenses against known attacks.

The DPatch attack is described with huge potential for both being implemented in the physical world, as well as being able to perform black-box attacks. This is one of the more dangerous combinations, as it would mean the attack could be performed against safety-critical tasks, like self-driven cars, without knowing anything about the internal configurations of the models within the vision system of the cars. Thus, we have to look further into the transferability of the DPatch attack.

With autonomous vehicles already on the horizon, and cars on the market already being equipped with autopilot ready to use for the drivers. Making it thus more important to look at these transferable physical attacks and their worst-case scenarios in today's real-world. And as the development of cars has been faster than defenses to adversarial input attacks, today's cars are vulnerable to existing attacks.

4.2 Research Questions

To be concrete of what to research about the DPatch attack, the following research questions were elicited:

- **RQ1:** Is the DPatch attack trained on Faster R-CNN transferable to other models, and newer versions of YOLO(v3 and v4).
- **RQ2a:** If RQ1 is *true*, how can this transferability and attack be mitigated?
- **RQ2b:** If RQ1 is *false*, how can the attack be improved to achieve transferability?

RQ2 was split in two, depending on the outcome of RQ1. RQ2 would most likely be out of scope for this project, but proved there are future work from the results of the project. This would be leading up to a master thesis in the future.

4.3 Research Methodology

To answer RQ1, I chose an empiric study approach. Meaning the DPatch attack had to be performed on a dataset on one of the object detectors used in [8]. Furthermore, the attack were to be performed in a black-box setting against another state-of-the-art object detector. Here, SSD would be sufficient, as it is heavily used in classification tasks [12] and achieves real-time detection speed.

I chose to evaluate the attacks according to the mAP metric, to stay consistent with the original paper and because it's widely used to evaluate attacks and models across multiple classes, as described in section 2.6.

4.4 Implementation

The original code [34] for the attack has not been maintained and uses deprecated libraries. Thus not deemed fit for implementing the attack at current state. The attack has later been implemented in the Python library Adversarial Robustness Toolbox (ART) [35]. ART started as a project from IBM and was later made open source to help developers and researchers to evaluate and defend Machine Learning models.

The attack was implemented through ART, which also provides quick access to pre-trained object detection models through torchvision¹. A pre-trained Faster R-CNN with ResNet-50-FPN backbone was selected.

Furthermore, the attack was configured with a patch size of 80x80 pixels. In the original paper, different sizes of patches was described. The smaller, less susceptible patches was focused on. While the larger patches had greater attack effect, with the drawback of being more visible to the human eye, thus weakening the attack. For this project, the focus was transferability, thus a greater attack effect was desirable and a patch of size 80x80 pixels was selected. This was the largest

¹<https://pytorch.org/>

size of the patches described in the original paper. If a 80x80 pixel proved to be transferable, a smaller patch would be desirable to test, to see if the patch size affect transferability. While a bigger patch would be insufficient given it's visibility, which would spoil the attack.

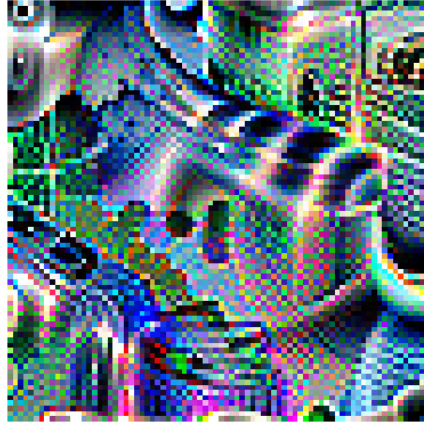


Figure 4.1: An 80x80 patch for targeted attack on Faster R-CNN

The attack was iterated over the PASCAL VOC 2007 dataset containing 5011 training images. Each iteration of the the training would update the patch for each image in the dataset. The attack was trained on Idun [36] for more than 3000 iterations, resulting in ~ 15 million updates to the patch, see Figure 4.1 for the resulting patch.

The resulting patch was a result of a targeted attack, created to be predicted as a "toaster". The patch's effect with the targeted label can be seen in Figure 5.1.

Chapter 5

Results and discussion

As described in section 3.6, Lee and Kolter [23] discussed faults with the DPatch attack which occurred during these experiments. [23] discusses that even though DPatch performs well in creating a patch which is classified as the target label, the attack incur little to no penalty for predicting the other objects in the frame. Thus, the DPatch often fails to suppress detections. Throughout the experiments in this project, the same problems occurred, and the attack seems to fail to evade objects. While training and validating the attack, the DPatch does not seem to suppress any detection of actual objects. Even when the patch is overlapping with objects in the frame. Figure 5.1 shows a few of the attack images, where the attack seems to have no effect other than manufacturing a fake object given by the target label.

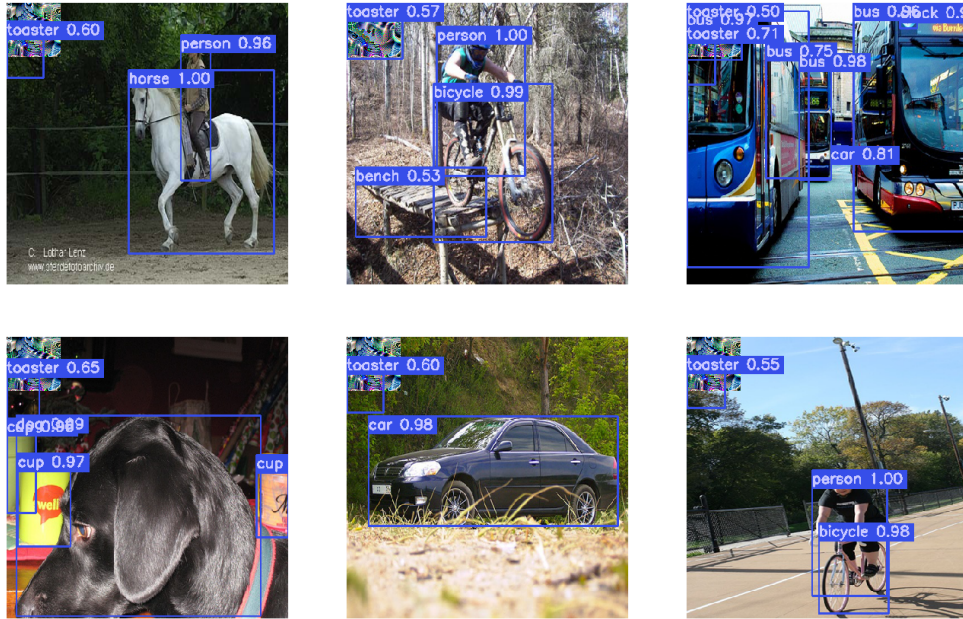


Figure 5.1: Comparison of patches from this implementation and the original paper

While training a patch on a single image, a pattern in the pixels of the patch quickly became clear, see Figure 5.2a. Such a pattern was neither described in [8] nor visible in the patches presented during the paper. Figure 5.2 compares the 40x40 patch from the original paper with one of the patches generated in these experiments. This pattern could be a result of overfitting the patch on a singular image, instead of an entire dataset. Meaning the attack constantly seeks to change the same pixels to reach an optimized solution. This would be expected to result in a patch which performs extremely well on the given image, but can't be transferred to other images. Thus not being a transferable attack, as discussed in the original paper of the DPatch attack.

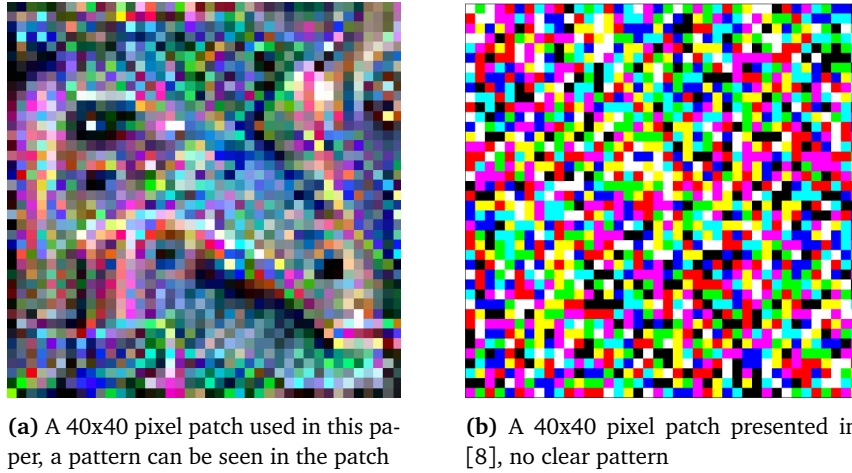


Figure 5.2: Results of DPatch which confirms some of the problems described in this chapter

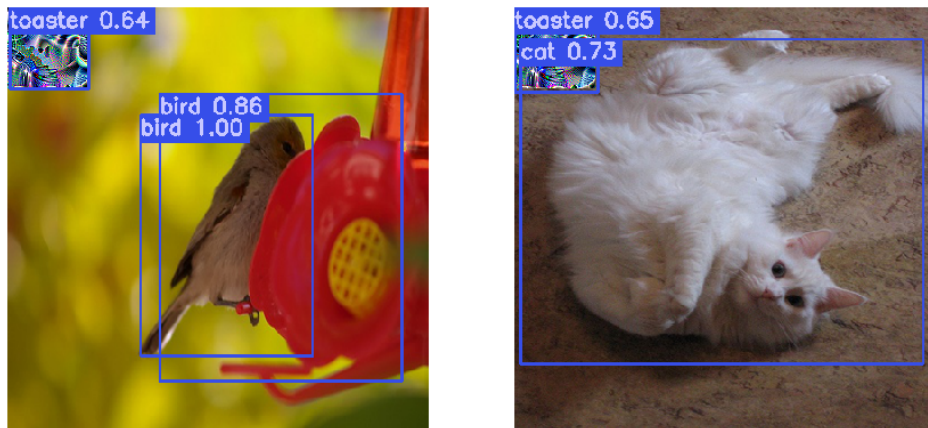
While training the patch on the dataset, containing 5011 images, as described in section 4.4, the same patterns occurred, see Figure 4.1.

As described in section 4.4, a larger patch (80x80 pixels) was selected for the experiments. Bigger patch also lead to the patterns becoming more visible. Throughout the iterations of the attack, one could observe that the pixels in the patches are converging to a pattern. By the first few iterations, the patches looks to be randomly, but after a few hundred iterations, the patterns became stronger. This differs from the original paper, where there was no pattern after 180 000 iterations.

5.1 Discussion

As the DPatch did not produce any successful evasion attacks, the transferability of DPatch was no longer in focus for this project. The observations and remarks about the attacks are further discussed, and is used to bring more confirmation of the concerns discussed in the related papers in chapter 3.

For some of the examples, objects could have multiple predicted boxes of same classes (see Figure 5.3a), meaning the boxes with lowest confidence was accepted as a true positive. As described in section 2.7, the object detector should instead make the less confident proposal a false positive.



(a) Multiple bounding boxes on the same bird

(b) Higher confidence on the patch's targeted class toaster

Figure 5.3: Bounding boxes not suppressed, even when the object detector predicts the target class toaster with fairly high confidence

As seen in some of the examples from Figure 5.1 and in Figure 5.3b, even when the patch overlaps with real objects the attack does not suppress the detection of said object. This brings concerns for the attack, as the attack is no longer effective against objects close to the patch, which is what Mark and Lee describe in [23].

The DPatch attack does not seem to be robust and would need further improvements to impose a real threat being deployed physically. Thus, further studies should be conducted to find strong attacks against object detectors that can be transferred to perform in black-box conditions. Such an attack is still regarded as dangerous to safety-critical tasks and would be essential to research further in the hope to mitigate their effectiveness and consequences.

Chapter 6

Conclusion

Throughout this paper, examples have been provided which doubt the effect of the DPatch attack. The attack seems to achieve manufacturing a fake object with the target label, but suppression of real objects was not found.

An interesting remark is that the original paper [8] was published by AAAI¹ 2019, which brings a certain legitimacy to the paper and the attack. The findings in these experiments, which confirm the problems described in [23], show that the publishing can be misleading and that further validation should be in place to verify the published papers.

As discussed in [28] and [23], the colors of the patch are not clipped to match the targeted images' pixel values. Thus, there is a mismatch between the colors of the patch and the target image, making it less suitable for physical attacks and less robust in digital attacks.

This leaves future work to be done to investigate the attack further, with a goal of pinpointing the problem of the attack and further improve it such that it can perform more reliably. And we can now conclude that the DPatch attack is not robust.

¹<https://www.aaai.org/>

Bibliography

- [1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao and K. Zieba, *End to end learning for self-driving cars*, 2016. arXiv: 1604.07316 [cs.CV].
- [2] F. Codevilla, E. Santana, A. M. López and A. Gaidon, *Exploring the limitations of behavior cloning for autonomous driving*, 2019. arXiv: 1904.08980 [cs.CV].
- [3] S. Grigorescu, B. Trasnea, T. Cocias and G. Macesanu, 'A survey of deep learning techniques for autonomous driving,' *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020. DOI: <https://doi.org/10.1002/rob.21918>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21918>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21918>.
- [4] N. Shahid, T. Rappon and W. Berta, 'Applications of artificial neural networks in health care organizational decision-making: A scoping review,' *PLOS ONE*, vol. 14, no. 2, pp. 1–22, Feb. 2019. DOI: 10.1371/journal.pone.0212356. [Online]. Available: <https://doi.org/10.1371/journal.pone.0212356>.
- [5] H. Greenspan, B. van Ginneken and R. M. Summers, 'Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique,' *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, 2016. DOI: 10.1109/TMI.2016.2553401.
- [6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, *Intriguing properties of neural networks*, 2014. arXiv: 1312.6199 [cs.CV].
- [7] J. Zhang and J. Li, 'Testing and verification of neural-network-based safety-critical control software: A systematic literature review,' *Information and Software Technology*, vol. 123, p. 106296, 2020, ISSN: 0950-5849. DOI: <https://doi.org/10.1016/j.infsof.2020.106296>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950584920300471>.
- [8] X. Liu, H. Yang, Z. Liu, L. Song, H. Li and Y. Chen, *Dpatch: An adversarial patch attack on object detectors*, 2019. arXiv: 1806.02299 [cs.CV].

- [9] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez and J. Garcia-Rodriguez, *A review on deep learning techniques applied to semantic segmentation*, 2017. arXiv: 1704.06857 [cs.CV].
- [10] Z. Qin, F. Yu, C. Liu and X. Chen, *How convolutional neural network see the world - a survey of convolutional neural network visualization methods*, 2018. arXiv: 1804.11191 [cs.CV].
- [11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, 'Ssd: Single shot multibox detector,' *Lecture Notes in Computer Science*, pp. 21–37, 2016, ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2.
- [12] D. Wang, C. Li, S. Wen, Q.-L. Han, S. Nepal, X. Zhang and Y. Xiang, *Daedalus: Breaking non-maximum suppression in object detection via adversarial examples*, 2020. arXiv: 1902.02067 [cs.CV].
- [13] S. Ren, K. He, R. Girshick and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2016. arXiv: 1506.01497 [cs.CV].
- [14] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: 1506.02640 [cs.CV].
- [15] J. Redmon and A. Farhadi, *Yolo9000: Better, faster, stronger*, 2016. arXiv: 1612.08242 [cs.CV].
- [16] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: 1804.02767 [cs.CV].
- [17] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV].
- [18] I. J. Goodfellow, J. Shlens and C. Szegedy, *Explaining and harnessing adversarial examples*, 2015. arXiv: 1412.6572 [stat.ML].
- [19] N. Papernot, P. McDaniel and I. Goodfellow, *Transferability in machine learning: From phenomena to black-box attacks using adversarial samples*, 2016. arXiv: 1605.07277 [cs.CR].
- [20] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt and A. Madry, *Exploring the landscape of spatial robustness*, 2019. arXiv: 1712.02779 [cs.LG].
- [21] N. Carlini and D. Wagner, *Towards evaluating the robustness of neural networks*, 2017. arXiv: 1608.04644 [cs.CR].
- [22] T. B. Brown, D. Mané, A. Roy, M. Abadi and J. Gilmer, *Adversarial patch*, 2018. arXiv: 1712.09665 [cs.CV].
- [23] M. Lee and Z. Kolter, *On physical adversarial patches for object detection*, 2019. arXiv: 1906.11897 [cs.CV].

- [24] Y. Wang, H. Lv, X. Kuang, G. Zhao, Y.-a. Tan, Q. Zhang and J. Hu, ‘Towards a physical-world adversarial patch for blinding object detection models,’ *Information Sciences*, vol. 556, pp. 459–471, 2021, ISSN: 0020-0255. DOI: <https://doi.org/10.1016/j.ins.2020.08.087>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025520308586>.
- [25] Q. Zhang, Y. Zhao, Y. Wang, T. Baker, J. Zhang and H. Jingjing, ‘Towards cross-task universal perturbation against black-box object detectors in autonomous driving,’ *Computer Networks*, vol. 180, p. 107388, Jul. 2020. DOI: 10.1016/j.comnet.2020.107388.
- [26] Z. Zhu, H. Su, C. Liu, W. Xiang and S. Zheng, *You cannot easily catch me: A low-detectable adversarial patch for object detectors*, 2021. arXiv: 2109.15177 [cs.CV].
- [27] H. Huang, Y. Wang, Z. Chen, Z. Tang, W. Zhang and K.-K. Ma, ‘Rpattack: Refined patch attack on general object detectors,’ in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021, pp. 1–6. DOI: 10.1109/ICME51207.2021.9428443.
- [28] M. Lu, Q. Li, L. Chen and H. Li, ‘Scale-adaptive adversarial patch attack for remote sensing image aircraft detection,’ *Remote Sensing*, vol. 13, no. 20, 2021, ISSN: 2072-4292. DOI: 10.3390/rs13204078. [Online]. Available: <https://www.mdpi.com/2072-4292/13/20/4078>.
- [29] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi and C.-J. Hsieh, ‘Zoo,’ *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, Nov. 2017. DOI: 10.1145/3128572.3140448. [Online]. Available: <http://dx.doi.org/10.1145/3128572.3140448>.
- [30] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh and M. Srivastava, *Genattack: Practical black-box attacks with gradient-free optimization*, 2019. arXiv: 1805.11090 [cs.LG].
- [31] Y. Wang, Y.-a. Tan, W. Zhang, Y. Zhao and X. Kuang, ‘An adversarial attack on dnn-based black-box object detectors,’ eng, *Journal of network and computer applications*, vol. 161, p. 102634, 2020, ISSN: 1084-8045.
- [32] J. Borkar and P.-Y. Chen, *Simple transparent adversarial examples*, 2021. arXiv: 2105.09685 [cs.CV].
- [33] S.-M. Moosavi-Dezfooli, A. Fawzi and P. Frossard, *Deepfool: A simple and accurate method to fool deep neural networks*, 2016. arXiv: 1511.04599 [cs.LG].
- [34] X. Liu, <https://github.com/veralauee/DPatch>, 2019.
- [35] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy and B. Edwards, ‘Adversarial robustness toolbox v1.2.0,’ *CoRR*, vol. 1807.01069, 2018. [Online]. Available: <https://arxiv.org/pdf/1807.01069>.

- [36] M. Sjölander, M. Jahre, G. Tufte and N. Reissmann, ‘EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure,’ *arXiv:1912.05848 [cs]*, Dec. 2019. arXiv: 1912.05848 [cs].