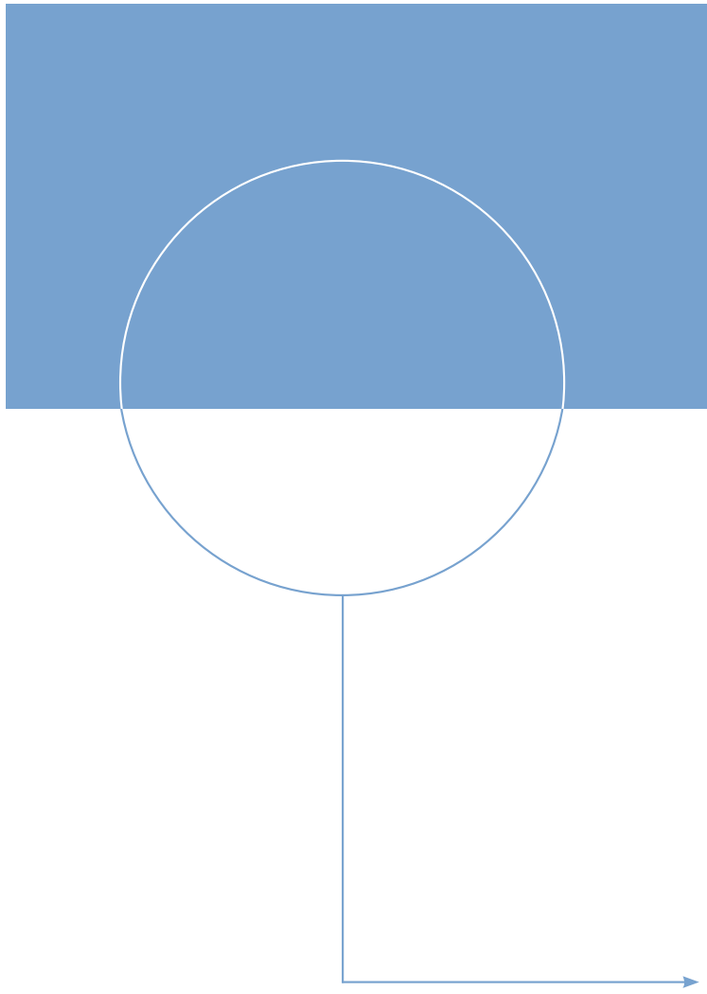


Master's thesis

2021

Torstein Meyer

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Computer Science

Torstein Meyer

Single-pass and Transferrable GAN-Based Black-box Attacks on Object Detectors

June 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Single-pass and Transferrable GAN-Based Black-box Attacks on Object Detectors

Torstein Meyer

MTDT Datateknologi
Submission date: June 2021
Supervisor: Jingyue Li

Norwegian University of Science and Technology
Department of Computer Science

Sammendrag

Objektgjenkjenning er et felt innen datasyn der en objektgjenkjenningsmodell beregner posisjon og type av objekter i bilder. Nylig forskning har vist at dype neurale nettverk er svært godt egnet til dette formålet. Det har blitt vist at neurale nettverk er sårbare mot presist utformede adverserielle angrep, som kan føre til feilklassifisering, objektskjuling, og objektfabrikering. Nylig forskning har klart å utføre vellykkede adverserielle angrep mot objektgjenkjenningsmodeller ved bruk av iterative optimiseringsmetoder. Disse metodene trenger vanligvis å utføre flere kall til objektgjenkjenningsmodellen for å iterativt forbedre angrepet. Mange av angrepene har heller ikke blitt testet mot objektgjenkjenningsmodeller som tar enkle forsvarsmekanismer, som er i stand til å nøytralisere usynlige adverserielle endringer, i bruk. I denne oppgaven formulerer vi et nytt adverserielt angrep mot objektgjenkjenning ved bruk av generative adverserielle nettverk. Angrepet fungerer i ett pass over bildet som skal angripes, og trenger ikke å utføre kall til objektgjenkjenningsmodellen for å fungere. Dette er en viktig fordel, spesielt når man angriper sanntids-objektgjenkjenning. Vi evaluerer angrepet vårt over to forskjellige datasett og mot flere forsvarte svart-boks-modeller. Vi demonstrerer overførbarheten til angrepet, og at angrepet er motstandsdyktig mot en enkel forsvarsmekanisme.

Abstract

Object detection is a computer vision task related to predicting bounding boxes and class labels for one or more objects in an image. State-of-the-art object detection performance has been achieved using deep models. However, deep neural networks have been shown to be vulnerable to carefully crafted adversarial attacks, which can lead to misclassifications, object suppression, and object fabrication. Recent works have successfully generated adversarial examples against object detectors using iterative optimization methods. Many of these attacks need to repeatedly query the target model in order to achieve the attack objectives. Additionally, the attacks are not demonstrated to work against basic defenses that can be used to neutralize imperceptible adversarial examples. In this work, we design an adversarial attack against object detection using generative adversarial networks. The attack works in a single pass over the target image, and does not need to query the target object detector to function, which is a significant advantage when attacking real-time object detection systems. Evaluating on two different datasets and on several different defended black-box detectors, we demonstrate the transferability of our attack, and its resilience towards basic defenses.

Preface

This is the Master Thesis written by Torstein Meyer, carried out in the 2021 Spring semester at NTNU. The project was supervised by Associate Professor Jingyue Li at the Institute of Computer Science and Informatics at NTNU, and assisted by a PhD student under his supervision, Nektaria Kaloudi. I would like to thank them for continuous assistance and motivation throughout the semester.

Torstein Meyer
Trondheim, June 28, 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Research in *artificial intelligence* (AI) has made significant advancements in recent years. AI technology has been applied in a diverse array of fields, such as autonomous vehicles ([39] [10] [5] [16] [9]) recommendation systems, ([46] [18] [29] [51] [19]) and board games ([44]). In many of these fields, the models achieve superhuman performance, e.g. by beating human experts in complicated board games such as chess and go. AI also has shown promise in safety-critical applications, such as autonomous driving.

One important application of AI is to the task of object detection, where a model must output a set of bounding boxes and classifications for an image input. In real-time applications, object detection can be used to track objects that are within the field of view of a camera, which has a variety of applications in e.g. environment sensing and object detection for autonomous driving [15]. This means that object detection has a role in safety-critical applications, where failure can lead to dire consequences.

However, recent research has revealed that AI systems are vulnerable to cyberattacks. In [13], the concept of an *adversarial attack* against image classification networks is introduced, which the authors call the *fast gradient sign method* (FGSM). These finely-targeted attacks can, using human-imperceptible perturbations, significantly impact the performance of various *machine learning* (ML) models. Since then, much research has been done on this quickly-evolving field. Adversarial attacks against image classification is a maturing field that has a high research output. However, attacks against object detection models are significantly less researched, and the state-of-the-art leaves room for improvement. As it is important to understand the threat landscape in order to create sufficiently robust object detection models, it is useful to make contributions to this field.

Adversarial attacks in general are often iterative and necessitate repetitive queries to the

target model in order to continuously improve the adversarial example ([54] [40] [7] [24]). This process might continue until a query limit is reached, or until a certain objective is achieved, e.g. the suppression of all object detections in an image. In some applications, this is a reasonable assumption to make, as the target model might be performant enough to allow for many queries in a short amount of time.

However, this is not the case in object detection, and especially not in real-time object detection. Current state-of-the-art object detection runs at anywhere between 5 FPS [43] and 60 FPS [35]. If an attack against object detection requires a single query, it is immediately weaker, as it would reduce the real-time performance of the object detector by 50%. In the state-of-the-art, the attacks often require significantly more queries than this. Given the current state-of-the-art performance of object detection models, this essentially makes the attacks useless in real-time applications, where a forward pass is made in the object detector each frame. Therefore, single-pass attacks are significantly more viable as attacks against object detection, yet the overwhelming majority of research focuses on iterative attacks.

It is also important that adversarial attacks are *transferable*. It is common to distinguish between three levels of knowledge for an attacker. In *white-box* attacks, the attacker is assumed to have full knowledge of the target system. In *grey-box* attacks, the attacker only has partial knowledge. And finally, in *black-box* attacks, the attacker does not know anything about the internal workings of the target system. If an attack works against several black-box models, we say that the attack is transferable, which is a highly desirable trait. If an attack is shown to be black-box transferable, it represents a much greater threat.

Another important factor to consider is that many attacks can be defended against. Adversarial attacks are usually designed to be as imperceptible as possible by the human eye. This follows from the demonstrated vulnerability shown in [13], where neural networks are shown to be vulnerable to human-imperceptible perturbations. However, much of the existing research does not consider the fact that simple defenses can completely neuter many of the imperceptible attacks, as they are simply designed to cause performance degradation in an undefended network. This same weakness is prevalent in the field of adversarial attacks against object detection.

As we want to consider attacks that do not iteratively generate the adversarial examples, gradient-based optimization attacks are out of the question. A promising alternative to this type of attack is the use of generative models, such as *generative adversarial networks* (GANs). GANs have in recent years been applied to a multitude of image-related generative tasks ([1] [36] [6]). The technique has also been used in several papers on adversarial attacks ([57] [33] [53]). By using a GAN to generate the adversarial example in a single forward pass, the attack would be a queryless attack, which is a highly desirable property when attacking object detection models.

1.2 Research question and contrbutions

In this master thesis, we will explore the possibilities of generating a queryless attack against black-box object detection models. Additionally, we will explore its potential against models with a basic defense applied to them, which is capable of neutralizing imperceptible perturbations.

As such, we will first investigate the state-of-the-art of attacks on object detection in a literature review, and gain an understanding of important properties of these attacks. Then, we formulate a novel approach based on attacking the prior boxes of the SSD object detector [35], which is a state-of-the-art real time object detector. The research question we wish to answer is the following:

Using generative adversarial networks, can we generate state-of-the-art transferrable queryless adversarial attacks against object detectors that have basic defenses applied to them?

In this work, we design four different adversarial attack models which have different properties. The models are trained using a novel method that involves suppressing object detections inside the predefined prior boxes of the SSD [35] object detector. We then carry out an extensive quantitative evaluation on two different commonly used object detection datasets. Of the attack models we evaluate, two achieve state-of-the-art black-box transferability, even in the defended setting.

1.3 Thesis Structure

First, we will go over several theoretical topics that are necessary to understand the contents of the thesis in Chapter 2. We will cover a variety of topics relating to deep learning and object detection. Then, we will review the state-of-the art of adversarial attacks on object detection, in Chapter 3. In Chapter 4, we will propose some novel approaches to enhance adversarial attacks against object detection, focusing on the black-box scenario in which the attacker does not have advanced knowledge about the target object detector. In Chapter 5, we will report experimental results where the proposed attacks are implemented and validated against state-of-the-art object detectors. Then, we will discuss these results, as well as the results obtained in the literature review, in Chapter 6. In Chapter 7, we will propose some directions for future work in the field, before finally concluding the thesis.

Chapter 2

Background Theory

In this chapter, we give an introduction to the theoretical background of the topics covered in the thesis. First, the fundamental theory of object detection will be introduced, starting with a discussion on convolutional neural networks (CNNs) followed by specific examples of object detection algorithms. Then, we will go over the object detection architectures we use in our work, as well as some commonly used classification backbones. Lastly, we will introduce the concept of generative adversarial networks.

2.1 Neural Networks

In this section, we give an introduction to the concept of *neural networks*. The most important unit of computation in a neural network is the *perceptron*. A perceptron takes several numerical values $\mathbf{X} = x1, x2, ..., x_n$ as input, and returns one numerical value as its output. The output, or the *activation*, of the neuron, is computed by a weighted sum of the input values. The weights $\mathbf{w} = w1, w2, ..., w_n$ correspond to the input values in $\mathbf{X}$. As such, the activation $f(\mathbf{X})$ is given by:

$$f(\mathbf{X}) = actv(\mathbf{wX}) \tag{2.1}$$

where *actv* represents an *activation function*. Modern neural networks use a variety of activation functions. Examples are the *sigmoid* function $\sigma(x)$:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

and the *rectified linear activation unit* (ReLU):

$$ReLU(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

When we use several perceptrons, the structure is called a *neural network*. Neural networks may be organized into layers, which we call a *deep neural network* (DNN).

In a *forward pass* through the neural network, some input will be given to the network, which will be represented in the network's first layer. Then, the values propagate through the network, being multiplied by the network's weights in the process. This eventually results in an output in the final layer.

This operation is called a *forward pass*. Mathematically, we write the *activations* $\mathbf{a}_l$ at the l-th layer as:

$$\mathbf{a}_l = actv_l(\mathbf{w}_l \mathbf{a}_{l-1} + \mathbf{b}_l) \tag{2.4}$$

where $\mathbf{w}_l$ and $\mathbf{b}_l$ are the weights and biases of the l-th layer, respectively. The function $actv_l$ is the activation function at the lth layer.

The operation that enables neural networks to learn an approximation to the function we want it to learn is the backward pass. Here we compute the gradients of the weights and biases of all the network neurons with respect to some loss function $\mathcal{L}$. This is done by performing a partial differentiation of the loss with respect to each learnable parameter. The weights are then updated by using an optimization method such as Adam optimization [23]. A *learning rate* hyperparameter is set to ensure that the changes to the weights and biases are not too large, which can result in the training becoming unstable.

2.2 Convolutional Neural Networks

A commonly used deep learning technique is *convolutional neural networks* (CNNs), which is a type of neural network that lends itself particularly well to image-related tasks. This is because the concept of spatial proximity is a part of the network itself, and not something that has to be learned as well. In most image-related tasks, such as image classification and object detection, features in the image that are close together are usually highly useful in determining the correct output.

A CNN is a NN that has *convolutional* layers. In a convolutional layer, the activation of each neuron is based on the neuron's *local receptive field*, which is a section of the neurons in the previous layer. This concept is illustrated in Figure 2.1. In this specific example, the local receptive field of the neuron in the first hidden layer contains the neurons in the 5x5 square marked in the input layer. Each neuron in the hidden layer has its own local receptive field.

One important aspect of the convolutional layer is that all the weights and biases between the neurons in the convolutional layer and the different local receptive fields are shared. For each local receptive field, the corresponding weights and biases are the same. The combination of the shared weights and biases makes up a *kernel*. This is highly beneficial for image-related tasks,

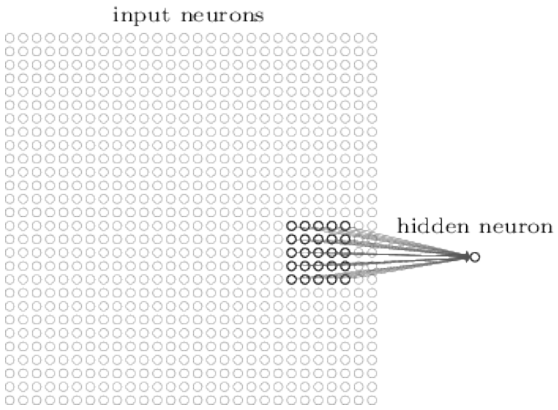


Figure 2.1: Local receptive field. Taken from [41]

because the kernel is essentially trained to recognize certain features at different positions in the image. Because the weights and biases are shared for the receptive fields, the features will be detected everywhere.

2.3 Object Detection

Object detection is the task of detecting and classifying objects in images. The object detector outputs a rectangular bounding box, a class label and a confidence score for each detected object in an image. The confidence score is a probability between 0 and 1, which represents the detector's confidence in a specific prediction. If the confidence score is below some value, typically 0.5, the prediction will be considered negative. In Figure 2.2, we show a typical object detection output for an image.

There are two common types of deep learning-based object detectors: *region-proposal* detectors and *regression-based* detectors. This work focuses on using the *single-shot detector* (SSD) [35], which is a regression-based detector, to generate the attacks. A short description of how SSD works follows, and the reader is encouraged to read the original SSD paper for a more complete understanding.

2.3.1 Single-shot detector (SSD)

The defining property of regression-based object detectors is the way they compute the object predictions in a single forward pass. Instead of separately proposing the object-containing regions and the class predictions, as in region proposal networks, they compute both at the same time. This is done by making use of the concept of *prior boxes*. A prior box is a predefined region proposal of a fixed size. The SSD detector has a configurable amount of prior boxes to be used for the images.

Figure 2.2: Example of object detection.

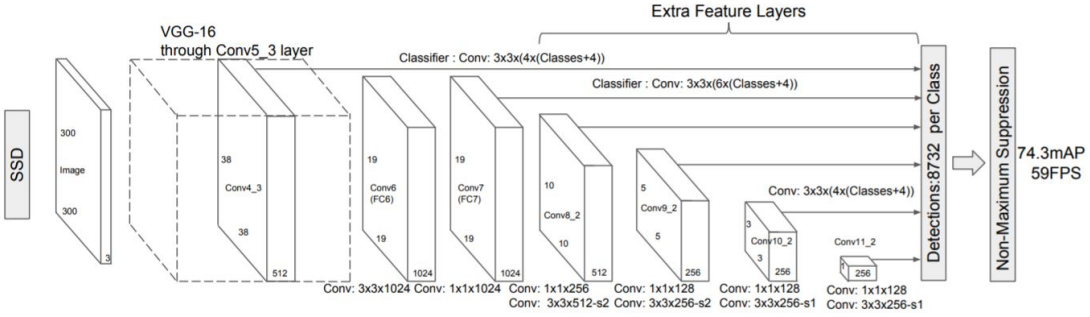


Figure 2.3: Default SSD architecture. Taken directly form [35].

Another important property of the SSD detection architecture is the fact that the backbone network outputs feature maps at different resolution scales. This is done to improve the object detector's ability to detect both small and large obejcts in the image, as the higher-resolution feature maps will be used to detect very small objects at the predefined locations in the image. For each feature map resolution, a convolutional layer is added to the truncated backbone network. The feature map cells progressively decrease in size to allow predictions at multiple scales.

In the forward pass, the regression prediction output of the detector represents a transformation of the prior boxes to fit around the appropriate object. As such, a regression and class prediction is made for each of the prior boxes. The class prediction is made by making use of a traditional image classification network such as ResNet [17] or VGG-16 [34] as the backbone. It is common practice to pretrain the backbone network on image classification before using it for object detection, which is done in both SSD [35] and Faster-RCNN [43]. In training, the predictions made by SSD have to be matched to the ground truth predictions in order to compute the loss. To do this, each ground truth box is matched to the prior box that has the highest *intersection over union* (IoU) score:

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.5}$$

To facilitate learning, prior boxes are matched with ground truth boxes that have an IoU score higher than some threshold, e.g. 0.5.

The training objective of SSD is formulated in the overall objective loss function $L(x, c, l, g)$:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{2.6}$$

where x is the prediction, c is the ground truth class labels, l is the predicted box, g is the ground truth box, and N is the number of matched default boxes. In the case $N = 0$, the loss is set to 0.

L_{conf}, the *confidence loss*, is defined as:

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \text{ where } \hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{2.7}$$

which is a softmax loss. L_{loc}, the *localization loss*, is a smooth L1 loss between the predicted box l and the ground truth box g defined as such:

$$L_{loc}(l, g) = \begin{cases} 0.5(l-g)^2 & if|l-g| < 1 \\ |l-g| - 0.5 & otherwise \end{cases} \tag{2.8}$$

which was originally proposed in the Fast-RCNN detector [12].

In Figure 2.3, the structure of the default SSD configuration is shown. The first part of the network consists of the initial layers of the VGG-16 classification network. Then, six convolutional

layers, representing the six different resolution scales, follow. These layers output the class predictions for each of the prior boxes for that resolution scale. Using *non-maximum suppression*, a method for removing a large amount of negative predictions, the detector achieves real-time performance with state-of-the-art accuracy.

2.3.2 Faster-RCNN

Another commonly used detection strategy are region-proposal networks. In this type of detector, a separate network is responsible for proposing regions of the image that may contain an object. Then, the classification network performs a convolutional feature extraction on the image and classifies each of the regions proposed in the previous network. In this work, we use Faster-RCNN, which is a region-proposal detector, as one of our black-box targets.

The Faster-RCNN detector is comprised of a deep CNN that is responsible for proposing object regions, and a detector module that uses the proposed regions to output object predictions.

The *region proposal network* (RPN) takes the image as input, and outputs a set of rectangular regions, each with an *objectness score* assigned to it. To generate region proposals, a small network slides over convolutional feature map output by the last convolutional layer of the backbone network.

At each sliding window location, the RPN predicts several region proposals, allowing a maximum of k predictions. The RPN outputs $4k$ outputs which represent the coordinates of the boxes, and $2k$ scores estimating the objectness of each region proposal.

To train the RPN, Faster-RCNN uses an objective loss defined as:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2.9}$$

where i is the index of a region proposal in a batch, p_i is the probability that box i is an object, p_i^* is the ground-truth objectness value (1 if it is an object and 0 if not), t_i is the four coordinates of the predicted bounding box, and t_i^* is the matched ground truth location. Furthermore, L_{cls} is a log loss over two classes, representing *object* and *not object*, and L_{reg} is a smooth L_1 loss as in SSD.

To combine the region proposal network with the detection head, the two modules are trained concurrently in an alternating training process. First, the RPN is initialized with pre-trained weights from a trained image classification network. The RPN is trained using the loss specified in Equation 2.9. Then, the detection module is trained using the outputs from the RPN in the previous step.

2.4 Generative adversarial networks

Generative adversarial networks (GANs) is a method for estimating generative models, first proposed in [14]. In this approach, two networks are trained simultaneously: a *generator* $\mathcal{G}$ and a *discriminator* $\mathcal{D}$. These two networks are trained in an adversarial fashion. Essentially, the generator $\mathcal{G}$ is trained to generate new output that should fit with some distribution, while the discriminator $\mathcal{D}$ is trained to recognize if an input sample comes from the distribution or not. In practice, this means that the generator is trained to output samples that fool the discriminator into believing the generated output comes from the distribution.

In the original paper, GAN is formulated as the min-max objective:

$$\min_G \max_D V(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log \mathcal{D}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z})))] \tag{2.10}$$

where p_{data} is the original data distribution. The discriminator is trained to recognize whether an input sample comes from p_{data} or from the generator's distribution $p_{\mathcal{G}}$.

Chapter 3

Related Works

In this chapter, we give an overview of related works that are relevant and related to our work. Adversarial attacks on machine learning-based classifiers and detectors is an emerging field that has been researched extensively in the last few years. We will give an overview of the state-of-the-art on attacks on object detection models by doing a systematic literature review (SLR) in this domain. We follow some of the guidelines specified by Kitchenham and Charters [2], by doing a systematic data extraction and synthesis on the discovered papers.

3.1 Attacks on Object Detection

In this section, we give an overview of the state-of-the-art of attacks against object detection models. We will first introduce the most important papers, before giving an overview of some important traits of all the discovered attacks.

3.1.1 Discovered papers

In [54], the authors propose *Dense Adversary Generation* (DAG). The attack is an iterative optimization attack using gradient descent to minimize the *class loss* defined as:

$$L(\mathbf{X}, \mathcal{T}, \mathcal{L}, \mathcal{L}') = \sum_{n=1}^{N} [f_{l_n}(\mathbf{X}, t_n) - f_{l'_n}(\mathbf{X}, t_n)] \tag{3.1}$$

where $\mathbf{X}$ is an image, $\mathcal{T}$ contains N recognition targets, $\mathcal{L}$ contains N recognition labels, $\mathcal{L}'$ contains the adversarial target labels, and f_c is the classification score for the target label c. The perturbations are generated through using gradient descent. In each iteration, the algorithm finds the set of labels that are still correctly predicted in the image, and computes perturbations from the gradients with respect to the input data. This process continues until there are no

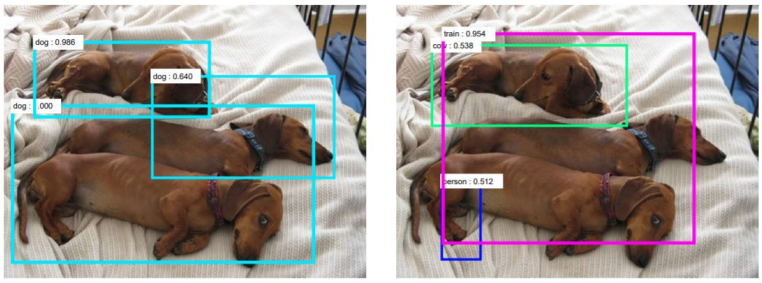


Figure 3.1: DAG perturbations. The figure shows the resulting classification output on the original and perturbed images against a white-box detector. The perturbations are visually imperceptible. Image taken directly from [54].

correctly predicted objects in the image or until a maximum iteration is reached. An example attack with the resulting detection outputs is shown in Figure 3.1.

Figure 3.2: PickObjectAttack. The attack results in an imperceptible targeted misclassification attack that causes the stop sign to be classified as flowers in this example, while the other objects are maintained. Image taken directly from [40].

In [40], the authors propose an adversarial attack against object detectors that allows targeting a specific object type o_{pick} in an image for misclassification, while preserving the classifications for other objects in the image. They target the Faster-RCNN [43] object detector with their attack. The attack works by iteratively computing the gradient of the loss for the bounding boxes with predicted label o_{pick} and applying a perturbation generated from this, until no boxes with an o_{pick}-prediction remain. As it requires gradient computation and prediction outputs from the object detector, this is a white-box attack. An example of pick-object-attack is shown in Figure 3.2.

In [56], an attack is proposed that uses contextual information to increase the effectiveness of the attack. A combination of classification, regression and a novel contextual loss is used to iteratively perturb the input image until the terminal conditions are met. The contextual loss is based on using the detection head to classify *regions of interest* (RoIs), and then perturbing

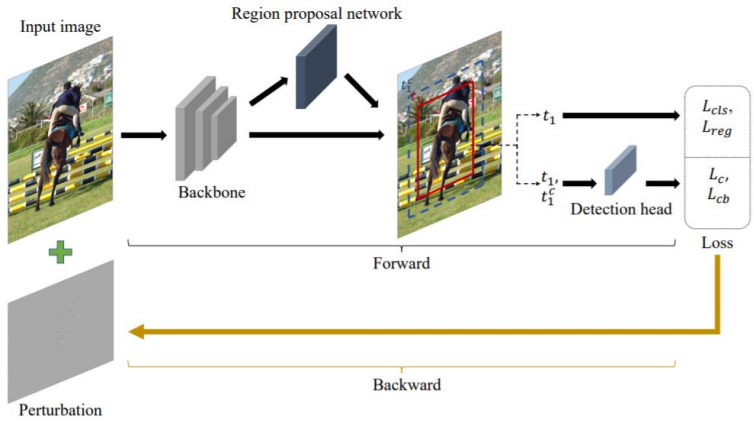


Figure 3.3: Contextual adversarial attack framework. The figure shows the structure of the proposed attack. The dashed blue line represents the contextual region of the object, which the proposed attack aims to suppress. Figure taken directly from [56]

the information present in them. In Figure 3.3, the attack process is shown, showing the usage of the novel contextual loss.

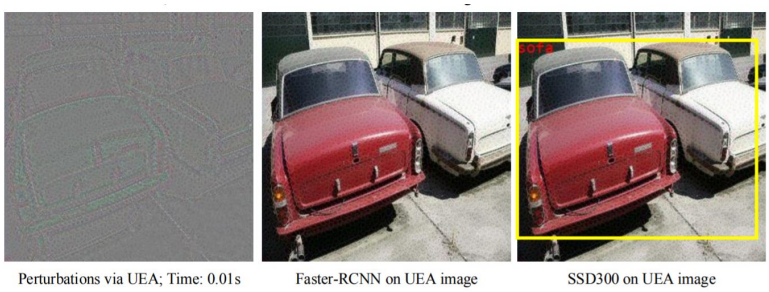


Figure 3.4: UAE classification output. UAE uses a generative model to successfully attack both Faster-RCNN (white box) and SSD300 (black box) in a single pass. Image taken directly from [50].

In [50], a transferable attack, *unified and efficient adversary* (UAE), on video and image object detection is proposed. The authors make use of a *generative adversarial network* (GAN) to generate the perturbations. Novelly, they design a novel feature loss which is combined with the class loss used in DAG from Equation 3.1 to generate transferrable perturbations in real-time. Their *multi-scale attention feature loss* $L_F(G)$ is defined as such:

$$L_F(G) = \mathbb{E}_I[\sum_{m=1}^{M} ||\mathbf{A}_m \circ (\mathbf{X}_m - \mathbf{R_m})||_2] \tag{3.2}$$

where G is the generator, I the image, $\mathbf{X}_m$ the extracted feature map in the m-th layer of

Figure 3.5: UAE perceptibility. The figure shows examples of images perturbed by UAE. Visually, a small anomaly is visible around the objects in the images, but they are difficult to see. Image taken directly from [50].

the feature network of Faster-RCNN, $\mathbf{A}_m$ the *attention weight*, and $\mathbf{R}_m$ a randomly predefined feature map. $\circ$ is the Hadamard product between two matrices. The attention weight $\mathbf{A}_m$ is computed based on the region proposals from the *region proposal network* (RPN) of Faster-RCNN. The purpose of this loss function is to make $\mathbf{X}_m$ as close as possible to $\mathbf{R}_m$, making the feature maps in the object detector into a random permutation.

In [7], the authors propose three *targeted adversarial objectness* (TOG) attacks with different properties. The authors propose four TOG attacks: *object-vanishing*, *object-fabrication*, *object-mislabeling* and *untargeted* attacks. These attacks exploit the gradients of the objectness, regression and classification losses respectively to achieve the specified attack objectives. While the attack is iterative and gradient-based, the authors are able to achieve good results when attacking several different object detectors, showing good transferability.

[48] proposes a black-box iterative object hiding attack, where an initial random noise filter is continuously optimized to hide the objects in the image with the smallest possible perturbation. This is done by querying the target object detector repeatedly until the smallest possible perturbation hiding the objects in the image is generated. The attack does not require any information about the model's parameters and loss functions, relying exclusively on the detector's discrete output. However, it also requires a significant amount of queries per image to achieve imperceptibility.

In [3], the authors propose a sparse patch-based adversarial object hiding attack against object detectors. This attack leverages the observation that the center points of objects in the image are important to the object detector when locating and classifying objects. Their attack is L_0 bounded so that only a certain amount of pixels in the image can be changed. The attack is iteratively optimized using a loss function based on the detection with the highest confidence output by the detector on the adversarial example. When generating the attack, they ensemble a one-stage and a two-stage object detector to obtain black-box transferability. When evaluating the attack on two black-box models, they achieve some promising results.

[21] proposes a physical adversarial attack, where a learned camouflaging patch is painted onto vehicles. The goal of the attack is to deteriorate the performance of object detectors in

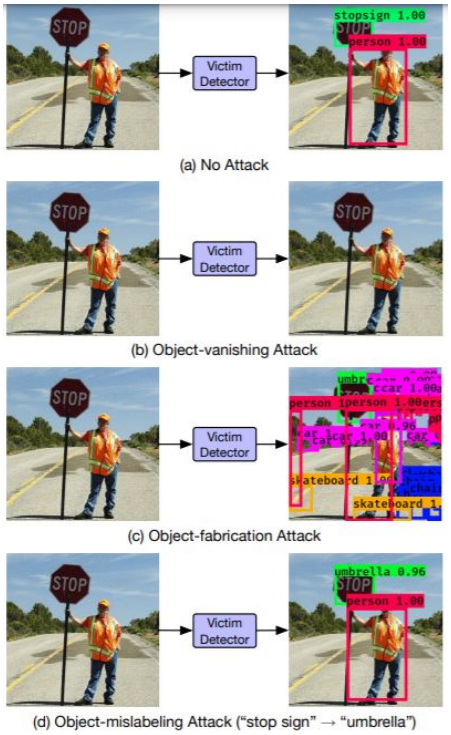


Figure 3.6: Targeted adversarial objectness attack. A configurable attack is proposed that can achieve different objectives. Detector outputs are to the right. Taken from [7]

the automated driving context. The patches are learned using an evolutionary search strategy, iteratively evaluating the performance of the patch and then using the losses of the target object detector to update it. The attacks are validated to be effective against several state-of-the-art object detectors.

In [58], the authors propose a method to generate universal, image-independent perturbations to attack black-box object detectors. Universal perturbations, if generated effectively, have several desirable properties: they are fast to generate, and are essentially black-box attacks. First, a generator is trained using a ResNet backbone to generate the universal perturbation, which is then applied to the input to the object detectors using either resizing or pile-up. The generated universal perturbations are able to reduce the performance of black-box detectors in real-time.

In [4], the authors propose a novel approach to generate attacks on object detectors with high transferability. Instead of using the outputs of the detectors explicitly to generate the attacks, the authors propose suppressing the *relevance maps* of the target object detector. The authors argue that the relevance maps are common properties of all object detectors, and by suppressing it, the attacker does not need additional knowledge about the target model to be successful. In

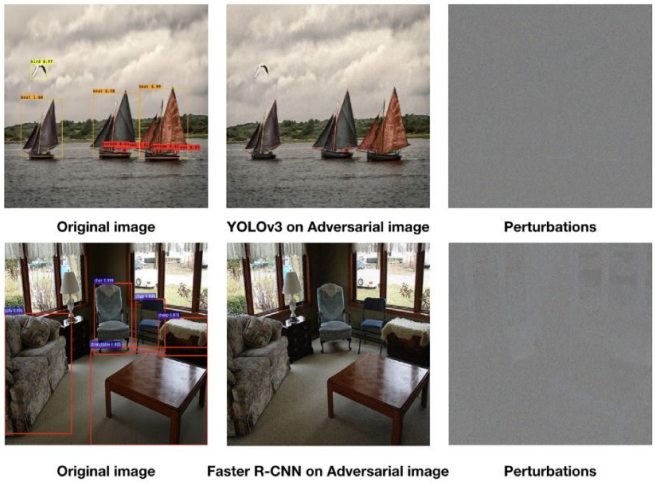


Figure 3.7: Evaporate attack. An iterative attack is proposed that achieves success in the black-box setting, as shown in the figure. The attack is effective against both regression-based (YOLOv3) and region proposal networks (Faster-RCNN). Taken from [48].

this paper, the authors extend existing techniques to generate relevance maps for image classifiers to object detectors, and use these to generate highly transferrable adversarial attacks.

[24] proposes an iterative optimization attack based on the *discrete cosine transform* method. This attack achieves notable results with fewer queries to the object detector than many of the existing state-of-the-art black box attacks. Essentially, the attack uses a *corner alignment* process to position images of the target class over the objects in the image. Then, using discrete cosine transform, the areas of the image corresponding to the objects are gradually perturbed using the positioned images, to achieve a successful misclassification with a low L_2 distance between the clean and perturbed image. An example of this attack is shown in Figure 3.11. We note that the attack is more perceptible than many of the other iterative attacks in other works, which might be a tradeoff against the reduced amount of required queries.

In [30], a category-wise attack is proposed. This attack performs the attack process in a manner that considers each category of objects in the image, instead of each object instance in the image. The attack is carried out in several steps. First, a category-wise target pixel set selection is performed to choose the pixels in the image to attack for each category. Then, the gradients of the heat maps for each category are added together, and using a sign method (like in FGSM), a perturbation is generated. This process is repeated until the attack succeeds. The proposed attack achieves good results on a single-shot detector and is also surprisingly transferrable to the two-shot detector Faster-RCNN.

In [25], a method to generate an adversarial patch that is able to suppress detection of every object in the image is proposed. The attack is even able to hinder detection of objects that are

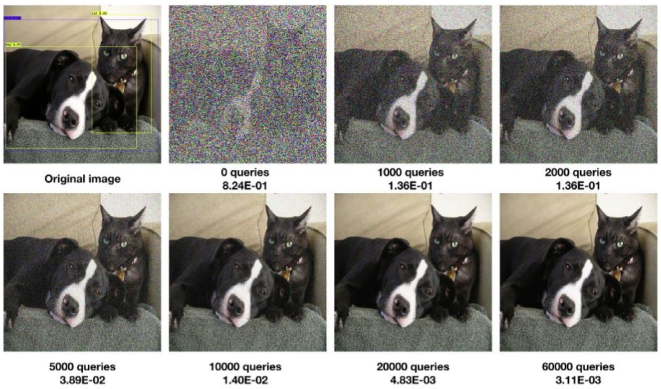


Figure 3.8: Queries needed for evaporate attack. The attack requires a signficant amount of queries to become imperceptible. Taken from [48].

Figure 3.9: Sparse adversarial attack. The patch-based attack works by adding a cross-shaped patch originating from the center of the detected objects. The figure shows examples of perturbed images. Taken from [3].

positioned far away from the patch in the image. The attack is an iterative attack that optimizes the patch based on the gradient of the loss of the target object detector.

In [49], an iterative gradient-based adversarial attack is proposed against the Faster-RCNN object detector. The authors run experiments where different loss functions are used to compute the gradient, and find that using the total loss (i.e. the sum of class loss and regression loss) to compute the gradients results in the most powerful attacks. The attack is mainly applicable in a white-box setting.

In [22], the authors propose a physical adversarial attack which generates adversarial borders around the target objects. Using this approach, the target object itself remains unchanged, but a significant perturbation is applied around the object which causes the detector to predict a large bounding box where the object is, eventually reducing the object's confidence score. This leads to the object being hidden. The attack is demonstrated to work both physically and digitally on

Figure 3.10: Cross-task universal perturbations. The figure shows an application of a generated universal patch. The left image is clean, and the two rightmost images are perturbed. Taken from [58].

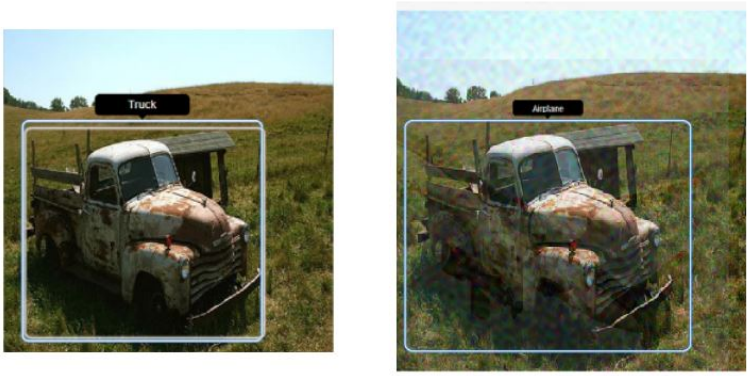


Figure 3.11: Discrete cosine transform-based attack. Shows an example of the proposed attack leading to a misclassification. Taken from [24]

one- and two-shot detectors.

In [20], a physical patch-based attack is proposed that attempts to account for changing light environments and camera angles, which is a significant challenge often encountered by this type of attack. The attack is done by switching between optimized patches dynamically according to the camera's position.

3.1.2 Data Extraction and Synthesis

To obtain a better understanding of the state-of-the-art of object detection attacks, we performed a data extraction procedure on the papers that we discovered in our search. The data points we extracted are shown in Table 3.1. The table shows the data points we extract from each paper, as well as the categories of each data point. If a data point is not easily categorizable, it is represented by a "·". Additionally, a paper can have a combination of one or more categories for each data point.

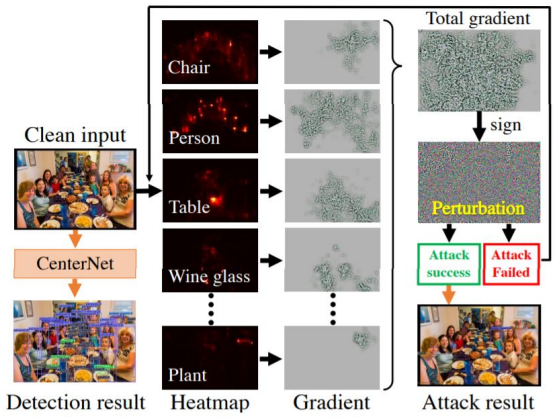


Figure 3.12: category-wise attack. The figure shows the attack process for the proposed attack. The gradients for each category are computed, before being aggregated to form the total gradient. Then, the perturbation is generated using the sign operation. This process is done iteratively until the attack succeeds.

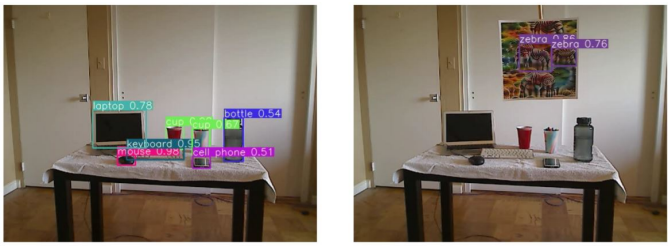


Figure 3.13: Physical adversarial patch. The attack uses a generated patch placed in the real, physical world to suppress objects in the image. The left image is clean, while the right image is attacked. Taken from [25]

3.1.3 Attacker's knowledge

First, we make note of the attacker's knowledge in each of the papers. In the literature, we usually distinguish between *white-* and *black-box* attacks. White-box attacks are attacks that assume complete knowledge of the target model's architecture, training settings, training data, parameters etc. Naturally, such attacks have a significantly higher chance of success against the target model, but these attacks may not be as applicable in real-world scenarios where this information would not be available. Black-box attacks are usually attacks where the attacker can only access the model's input, e.g. an image from a camera, and the resulting model output. We also include the intermediate category of *grey-box* attacks to denote attacks that need some additional knowledge from the model, e.g. prediction losses, but do not require e.g. gradient computation in the target model.

3.1.4 Attack types

Second, we distinguish between a set of *attack types* that occur in the literature. There are some common ways to generate adversarial examples against object detectors, and we will briefly describe each category here. *Optimization-based* attacks are attacks that iteratively optimize the generated perturbation with respect to some loss metric. These attacks may require a significant amount of iterations to achieve the attack objectives (e.g. hiding every object in the image), but usually achieve a very high attack success ratio. Often, these attacks are directly optimized on the gradient of the loss of the object detector with respect to the input image X. This type of optimization was first seen in [13] against image classifiers, and is quite commonly used. However, these attacks are often white-box attacks.

Universal attacks are attacks that aim to generate a single, universal, image-independent perturbation that can be added to any image to reduce the performance of the object detector. The perturbation is usually generated through learning to attack a white-box object detector, with the final objective of generating a transferable perturbation that can be used to attack other black-box detectors as well. This type of attack, if effectively generated, can represent a massive threat to real-time object detection models, as it requires essentially no computation to generate when a model is actively under attack. *Single-pass* attacks are attacks that generate an image-dependent perturbation in a single pass, requiring no iterative optimization. As they are done in a single pass, these attacks can sometimes function without even needing access to the output predictions of the target model. This means that this type of attack is often used in a black-box setting. *Patch* attacks are attacks that perturb only small areas of the image to achieve the attack objective. These perturbations can often be more visually distinguishable than attacks that perturb the whole image. However, this is offset by the fact that the rest of the image is completely untouched by the attacker. These attacks are often used in *physical patch* attacks, where a learned adversarial patch is somehow applied to real-world objects, such as stop signs on a road. These attacks have to account for variables such as changing camera angles and light condition to be successful. However, they may also represent the most likely attack scenario against an object detector, as they do not require having compromised the target system to be able to modify the images. Instead, they simply require changing some real-world object.

3.1.5 Target types

As discussed in Chapter 2, the majority of deep object detectors fall into two categories: region proposal and regression-based detectors. These detectors have some properties that makes it necessary to attack them in different ways. Ideally, a transferable attack should be able to work on both region proposal and regression-based detectors in a black-box manner.

Data point	Possible values
Publication year	·
Attacker knowledge	White-box, Grey-box, Black-box
Attack type	Optimization, Universal, Single-pass, Patch, Physical
Target type	Region proposal, Regression
Queries needed	·

Table 3.1: Extracted data points from the papers

Paper	Year	White-box	Grey-box	Black-box	Optimization	Universal	Single-pass	Patch	Physical patch	Region proposal	Regression	Queries needed
[54]	2017	✓	·	·	✓	·	·	·	·	✓	·	30-50
[50]	2019	·	·	✓	·	·	✓	·	·	✓	✓	0
[22]	2019	·	·	✓	✓	·	·	✓	✓	✓	✓	100-300
[25]	2019	✓	·	·	✓	·	·	✓	✓	·	✓	·
[40]	2020	✓	·	·	✓	·	·	·	·	✓	✓	10-50
[56]	2020	✓	·	·	✓	·	·	·	·	✓	✓	200
[7]	2020	✓	·	✓	✓	✓	·	·	·	✓	✓	10
[48]	2020	·	·	✓	✓	·	·	·	·	✓	✓	1000
[3]	2020	·	✓	·	✓	·	·	✓	·	✓	✓	·
[21]	2020	✓	·	✓	✓	·	·	✓	✓	✓	✓	·
[58]	2020	·	·	✓	✓	✓	·	·	·	✓	✓	·
[4]	2020	·	·	✓	✓	·	·	·	·	✓	✓	30-40
[30]	2020	·	·	✓	✓	·	·	·	·	✓	·	·
[49]	2020	✓	·	✓	✓	·	·	·	·	✓	·	4
[20]	2020	✓	·	·	✓	·	·	✓	✓	·	✓	·
[24]	2021	·	·	✓	✓	·	·	·	·	✓	✓	30-40

Table 3.2: Data extraction.

3.1.6 Queries needed

In attacks that are optimization based, we also note the amount of queries to the target object detector that the attack needs to achieve success. There are some attacks that can achieve significant performance reduction with very small perturbations, but with a query amount that is too high to be practically applicable. Reducing the amount of queries in these attacks can lead to perturbations that are very large. Thus, we make note of the amount of queries that the attacks proposed in the papers need to achieve good results.

In Table 3.2, we show the results of the data extraction step, where a checkmark (✓) means the paper falls into a category. An important observation to be made is that the majority of the related works are iterative optimization-based attacks, which often require several queries to the target object detector. This is a significant weakness of any adversarial attack, especially against a performance-intensive application such as object detection. In the papers where it is shown, we report the amount of queries needed before the attack process finishes. All iterative attacks make at least one query to the detector.

The only single-pass attack in the body of works we analyze here is [50], which uses a generative model to generate the perturbations without requiring any queries. In this thesis, we intend to design an attack that has similiar properties - black-box, single-pass, and transferrable

between object detector architectures. As such, the work that is most closely related to ours is this paper. However, this paper has some drawbacks. First, it only evaluates the proposed attack against two detectors: one white-box and one black-box. Second, it does not evaluate the attack against object detectors that have some defensive measures applied to it. Third, it only evaluates its attack in the scenario where the class labels used by the target detector are known, and does not test its performance on a detector that is trained to use different class labels. Lastly, its performance in the black-box setting is good, but it is not exceptional. In this work, we seek to eliminate these weaknesses.

Chapter 4

Methodology

In this chapter, we will describe the methodology of our research. First, we will formally describe the problem of carrying out adversarial attacks on object detectors. This will be done in the context of our research question specified in Chapter 1. Then, we will describe the research design of the project, where we will specify the methods by which we generate the attacks. Lastly, we will describe the extensive evaluation procedure that we will use to validate the efficacy of our results.

4.1 Research question

In Chapter 1, we specified the research question of our project:

Using generative adversarial networks, can we generate state-of-the-art, transferrable, query-less adversarial attacks against object detectors that have basic defenses applied to them?

To answer this research question, our attack needs to have the following traits:

1. A GAN must be used to generate the adversarial perturbations in a single forward pass
2. The attack must work against white-box and black-box detectors, which will make it transferable
3. The targeted object detectors must have a basic defense applied to them

For the first point, we shall describe an approach to designing and training a GAN that generates adversarial perturbations to deteriorate the performance of the SSD object detector as much as possible. The GAN will also be trained such that the perturbations are as small as possible. This will be done in Section 4.2.

For the last two points, we design a combined loss function for the generative model. The loss is designed so that it will suppress object detection in every part of the image. By training a GAN

this way, the generated perturbations will be effective against white-and black-box detectors. To show that the attack works against a basic defense, we will design an evaluation pipeline that evaluates the different attack configurations on two different datasets, against several different white- and black-box detectors. Our evaluation methodology will be described in Section 4.3.

4.2 Attack design

We want to design an adversarial example $I_p = \Delta I + I$ based on an image I that deteriorates the performance of a target object detector O. Mathematically, we express this as a maximization objective as such:

$$\max L(O(I_p), \mathbf{y}) \tag{4.1}$$

where L is some loss function, $O(I_p)$ is the object detector output and $\mathbf{y}$ is the ground truth detection. Additionally, the perturbation ΔI should be as small as possible, which is formalized with the minimization objective:

$$\min ||\Delta I||_p \tag{4.2}$$

where $p \in \{0, 1, 2, \infty\}$. Combining Equations 4.1 and 4.2, our attack objective is then written as:

$$\max_{\mathrm{L}} \min_{\Delta \mathrm{I}} L(O(I_p), \mathbf{y}) - ||\Delta I||_p \tag{4.3}$$

The specific values of L and p depend on the chosen attack strategy, and we will see that their selection significantly impact the performance of the attacks.

In this section, we attempt to design a black-box, single-pass attack against the SSD300 object detector. The combination of the black-box and single-pass restrictions makes this the most difficult attack scenario possible. As such, it is possible that we can only achieve high attack success rates with very significant perturbations with a high L_2 distance.

To achieve success while restricted to a single pass, we will use a similiar method as in UAE [50], where the authors elected to use a GAN to generate the perturbations. The discriminator serves to reduce the perceptibility of the perturbations. However, in UAE, the attacker and losses are based on attacking the Faster-RCNN detector. We will use GANs in a similiar fashion adapted to the SSD detector architecture.

The generator is trained on the joint objective of generating effective perturbations for the training images, and fooling the discriminator with the perturbed images. The discriminator is trained using the *least-squares GAN* loss function, which has been shown to generate higher-quality images than other methods, as well as being able to stabilize the training process [37]. The least-squares GAN discriminator loss is formulated as follows:

$$\mathcal{L}_D(\mathbf{X}) = (D(\mathbf{X}) - 1)^2 + (D(\mathbf{X} + G(\mathbf{X}))^2 \tag{4.4}$$

Where $\mathbf{X}$ is the original image, $G(x)$ is the generator output, and $D(x)$ is the discriminator output for some image x.

The generator is trained with a combination of several losses. First, we compute a loss based on how well it fools the discriminator with:

$$\mathcal{L}_{G_D} = D(G(\mathbf{X}) - 1)^2 \tag{4.5}$$

again using the least-squares GAN formulation. Additionally, we compute a *performance reduction loss* $\mathcal{L}_{PR}$:

$$\mathcal{L}_{PR} = l(\mathbf{X}, \mathbf{X} + G(\mathbf{X})) \tag{4.6}$$

where l is a configurable loss function, which we vary in our experiments. Lastly, we add a *hinge loss* $\mathcal{L}_H$ in order to control the magnitude of the perturbations early in the training, defined as such:

$$\mathcal{L}_h = max(0, ||G(\mathbf{X})||_2 - c) \tag{4.7}$$

where c, the hinge loss threshold, is a training hyperparameter.

Thus, the complete loss function for the generator $\mathcal{L}_G$ is formulated as:

$$\mathcal{L}_G = \alpha\mathcal{L}_{PR} + \beta\mathcal{L}_{G_D} + \gamma\mathcal{L}_h \tag{4.8}$$

where α, β and γ are hyperparameters.

4.2.1 Choice of loss function l

The choice and design of the loss function $l(\mathbf{X}, \mathbf{X} + G(\mathbf{X}))$ determines the attack objective of the model. Therefore, the choice of this loss function is highly important when attacking the SSD detector. We separate between three different attack objectives:

- Object hiding
- Object fabrication
- General performance reduction

We will describe the choice of l for each attack objective in turn.

Object hiding

When the objective is to hide as many objects in the image as possible, we want to design a loss function that penalizes leaving positive predictions in the image. To do this, we make use of the class prediction output for each of the default SSD prior boxes that is output by the forward pass. Assuming that a negative prediction, i.e. a bounding box contains no objects, is represented by the class prediction label 0, we design the loss function $\mathcal{L}_PR$ as such:

$$\mathcal{L}_{PR} = BCE(\mathbf{p}, \mathbf{y}) \tag{4.9}$$

where $\mathbf{p}$ is an $N_{priors}xN_{classes}$ matrix of class prediction probabilities, and $\mathbf{y}$ is an $N_{classes}$-sized vector, where:

$$y_i = \begin{cases} 1 & if\ i = 0 \\ 0 & otherwise \end{cases} \tag{4.10}$$

BCE is the *binary cross-entropy loss* function, which is defined as:

$$BCE(\mathbf{p}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{n} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)) \tag{4.11}$$

Object fabrication

In an *object fabrication* attack, the goal is to fool the detector to predict objects that are not actually present in the image. Such attacks could be used to corrupt some decision-making process by e.g. fabricating a stop sign in the autonomous driving setting. To do this, we use the same approach as in the object hiding attack, but we modify Equation 4.10 as such:

$$y_i = \begin{cases} 1 & if\ i = \mathcal{T} \\ 0 & otherwise \end{cases} \tag{4.12}$$

where $\mathcal{T}$ is the label of the object class we want to fabricate. Using this method, the network will be trained to generate perturbations that affect all of the target prior boxes, hypothetically resulting in false positives everywhere in the image. Additionally, it could also cause real objects in the image to be misclassified as the class $\mathcal{T}$.

General performance reduction

In a *general performance reduction* attack, we want to degrade the object detector's performance by any means necessary. This could include a combination of object hiding, object misclassification, object moving, and object fabrication. To train this network, we design a loss function based on the regression and classification loss outputs of the SSD detector. The loss function $\mathcal{L}$ is specified as such:

$$\mathcal{L} = \gamma\chi^{-(\Delta\mathcal{L}_{CLS}+\Delta\mathcal{L}_{REG})} \tag{4.13}$$

where γ, χ are hyperparameters and $\Delta\mathcal{L}_{CLS}$, $\Delta\mathcal{L}_{REG}$ are the changes in class and regression loss before and after the perturbation is applied. The exponential operation with χ turns a low degradation of the performance, i.e. a small change in the loss of the object detector, into a high loss for the generator G.

Combined attacks

In our experiments, we will also train models that use a combination of the objective losses specified above.

4.3 Evaluation Procedure

In this section, we describe the evaluation procedure that we use to judge the quality of our results. By using a consistent method, we are able to more effectively compare our results to the current state-of-the-art. Additionally, we make sure that the evaluation procedure is very extensive compared to many of the other works in the field, in order to more thoroughly validate our results.

4.3.1 Metrics

To evaluate our results, we consider the various measurable properties of our attacks. The properties we have to evaluate are as follows:

- MAP reduction
- Queries needed
- Time taken
- Perturbation size

These metrics show the efficacy and efficiency of our results.

Mean average precision

First, we describe *mean average precision* (MAP), which is the most commonly used metric to evaluate the performance of object detection models. Evaluating the reduction in MAP between the benign and adversarial images is therefore a good way to determine the effect the attack has on the object detector's performance. *Average precision* (AP) is a metric that measures the accuracy of the predicted bounding boxes for an object class in the image. This is done by

computing the *intersection over union* (IoU) of the predicted bounding boxes $\mathbf{B}_{pred}$ with the ground truth predictions $\mathbf{B}_{gt}$, as such:

$$IoU = \frac{B_{pred} \cap B_{gt}}{B_{pred} \cup B_{gt}} \tag{4.14}$$

Equation 4.14 is applied to each predicted bounding box $B_{pred} \in \mathbf{B_{pred}}$ and each ground truth bounding box $B_{gt} \in \mathbf{B_{pred}}$. Predicted boxes with a maximum IoU with any ground truth box less than the IoU threshold t are considered false positives.

To calculate the AP for one class label, we consider the concept of a *precision-recall curve*. Each bounding box prediction B_{pred} is sorted by its confidence score. Iterating through each prediction, we calculate the current precision and recall levels at each step. By defining this as a function that maps the current recall to the precision, we get a curve with a zig-zag shape, which we call $p(r)$. The average precision is then defined as:

$$AP = \int_0^1 p(r)dr \tag{4.15}$$

This calculation is done for each of the N class labels in the dataset. Then, the mean of all the APs is computed to give the final MAP score:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{4.16}$$

where AP_i is the AP of the i-th class label.

For our evaluation, we want to measure the performance reduction PR, i.e. the change in the MAP between the original image and the adversarially perturbed image. This is expressed as:

$$PR = mAP_{original} - mAP_{perturbed} \tag{4.17}$$

To evaluate the MAP reduction that the attacks achieve, we calculate the average MAP reduction achieved on the VOC2012 [11] test dataset. We will refer to this dataset as *VOC* from now on.

Other metrics

Additionally, we measure the time taken to complete the attack. This metric is heavily related to the previous one, but it also says something about the complexity of the underlying model used to generate the perturbations. Naturally, the less time required to carry out the attack, the better.

Lastly, we measure the average perturbation size needed to achieve a significant performance reduction. The perturbation size is given by the average L_2-distance, or the Euclidean distance,

Label ID	**Label name**
1	Aeroplane
2	Bicycle
3	Bird
4	Boat
5	Bottle
6	Bus
7	Car
8	Cat
9	Chair
10	Cow
11	Dining table
12	Dog
13	Horse
14	Motorbike
15	Person
16	Potted plant
17	Sheep
18	Sofa
19	Train
20	Tv monitor

Table 4.1: Class labels used in the VOC dataset

Label ID	Label name
1	Person
2	Bicycle
3	Car
4	Motorcycle
5	Airplane
6	Bus
7	Train
8	Truck
9	Boat
10	Traffic light
11	Fire hydrant
12	Stop sign
13	Parking meter
14	Bench
15	Bird
16	Cat
17	Dog
18	Horse
19	Sheep
20	Cow
21	Elephant
22	Bear
23	Zebra
24	Giraffe
25	Backpack
26	Umbrella
27	Handbag
28	Tie
29	Suitcase
30	Frisbee
31	Skis
32	Snowboard
33	Sports ball
34	Kite
35	Baseball bat
36	Baseball glove
37	Skateboard
38	Surfboard
39	Tennis racket
40	Bottle
41	Wine glass
42	Cup
43	Fork
44	Knife
45	Spoon
46	Bowl
47	Banana
48	Apple
49	Sandwich
50	Orange
51	Broccoli
52	Carrot
53	Hot dog
54	Pizza
55	Donut
56	Cake
57	Chair
58	Couch
59	Potted plant
60	Bed
61	Dining table
62	Toilet
63	Tv
64	Laptop
65	Mouse
66	Remote
67	Keyboard
68	Cell phone
69	Microwave
70	Oven
71	Toaster
72	Sink
73	Refrigerator
74	Book
75	Clock
76	Vase
77	Scissors
78	Teddy bear
79	Hair drier
80	Toothbrush

Table 4.2: Class labels used in the COCO dataset

between the perturbed image and the original one. This distance function, between two N-sized vectors X and Y, is defined as:

$$L_2(X,Y) = \frac{1}{N}\sum\sqrt{(y_i - x_i)^2} \tag{4.18}$$

We also report the *mean squared error* (MSE):

$$MSE(X,Y) = \frac{1}{N}\sum(y_i - x_i)^2 \tag{4.19}$$

.

Ideally, the attack should result in a visually imperceptible difference between the two images. While this is not necessarily the same as having a low L_2-distance or MSE, measuring the difference between the original and the perturbed images is likely to be a good estimate of perceptibility.

4.3.2 Target detectors and datasets

In order to properly evaluate the white- and black-box performance of our attack, we will evaluate the different attacks in a variety of scenarios. We are interested in the attacks' performance against different object detectors, against different datasets, and ideally against different types of adversarial defenses.

Datasets

In our training phase, we train the attacker to perturb images to deteriorate the performance of the SSD300 detector on the VOC2012 [11] training dataset. The VOC2012 dataset contains 20 unique class labels which are used in the object detector. These labels are shown in Table 4.1 The evaluation is done on the VOC2012 validation set, which is data that is not used in the training of the attacker.

However, the set of class labels is still the same. In some attack scenarios, it can be a reasonable assumption that the class labels used in the target detector are known by the attacker. In fact, most works in the existing state-of-the-art make this assumption. In our work, we want to investigate the performance of the attacks against detectors using a different set of class labels. To do this, we perform the attack against black-box detectors trained on the COCO 2017 [32] dataset, which uses 80 unique class labels. These labels are shown in Table 4.2.

Object detectors

In our evaluation pipeline, we perform the attack on an extensive variety of object detectors. Naturally, we evaluate the attack in the white-box setting against the SSD300 detector with a VGG16 backbone, which is the detector used to train the generative attack models. Additionally,

we perform a grey-box evaluation on the SSD300 detector using the EfficientNet classification backbone, which will indicate the transferability between classification networks.

In the black-box scenario, we need to evaluate the attacker against a region-proposal object detector, using a different backbone than VGG16. This represents a scenario in which the attacker has no knowledge of the nature of the target detector. In the existing literature on attacks against object detection, the Faster-RCNN detector is commonly used as either a white-box or black-box target. Notably, it is the white-box target in [50], which is the work most closely related to our own. Their black-box target was our white-box model, SSD300. Thus, it will be useful to attack Faster-RCNN in the black-box setting to compare our performance to the existing literature.

We attack Faster-RCNN with a variety of backbone configurations.

Defenses

Ideally, the attacks should also be evaluated facing more robust object detectors, that have a combination of defenses against adversarial attacks available to them. In our work, we only apply a simple adversarial defense, which involves adding generic noise drawn from a normal distribution to the images after the perturbation is applied. In [28], the authors propose a defense for salient object detection models. The defense first uses generic noise to destroy the imperceptible adversarial perturbations, and then learns to predict saliency maps through the noise. This defense is found to be effective against imperceptible optimization-based attacks. Intuitively, this is reasonable, as imperceptible perturbations are precisely optimized to minimize the objective loss, and are therefore highly sensitive to small changes to the image. If our attacks are effective even when this defense is applied, we will have shown that our attacks are resistant to a basic defense against imperceptible attacks. We will use different magnitudes of generic noise in our evaluations.

The noise is applied to the image as an additive term to each channel value of each pixel in the image. The value that the pixel is changed by is drawn from the normal distribution with probability density function $f(x)$:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{4.20}$$

To control the magnitude of the added noise, we vary the standard deviation σ. The mean is set to $\mu = 0$.

4.3.3 Baseline performance

Now, we will show the baseline performance achieved by the different object detectors on the two datasets, with different levels of normally distributed noise applied to the image. When defending the detector using generic noise, the noise should not be so significant that it deteriorates the

Level	σ
σ_1	0
σ_2	5
σ_3	10
σ_4	25
σ_5	50
σ_6	100

Table 4.3: Defense levels used in experiments

ID	**Architecture**	**Backbone**	**Attacker knowledge**
VOC_1	SSD300	VGG16	White-box
VOC_2	SSD300	EfficientNet	Grey-box
VOC_3	Faster-RCNN	ResNet50 C4	Black-box

Table 4.4: Target detectors trained on the VOC2012 dataset

ID	**Architecture**	**Backbone**	**Attacker knowledge**
$COCO_1$	Faster-RCNN	ResNet50 C4	Black-box
$COCO_2$	Faster-RCNN	ResNet101 FPN	Black-box
$COCO_3$	Faster-RCNN	ResNet101 C4	Black-box
$COCO_4$	Faster-RCNN	X101 FPN	Black-box
$COCO_5$	RetinaNet	ResNet50 FPN	Black-box
$COCO_6$	RetinaNet	ResNet101 FPN	Black-box

Table 4.5: Target detectors trained on the COCO dataset

performance of the detector by itself. Therefore, we need to investigate how much noise can be added to the images without signficantly deteriorating the performance of the detectors we use.

First, we specify a set of defense levels, decided by the value of σ. A higher value of σ will result in a stronger noise pattern. The values that we use in our experiments are shown in Table 4.3. For each baseline detector, we will show the performance at each defense level.

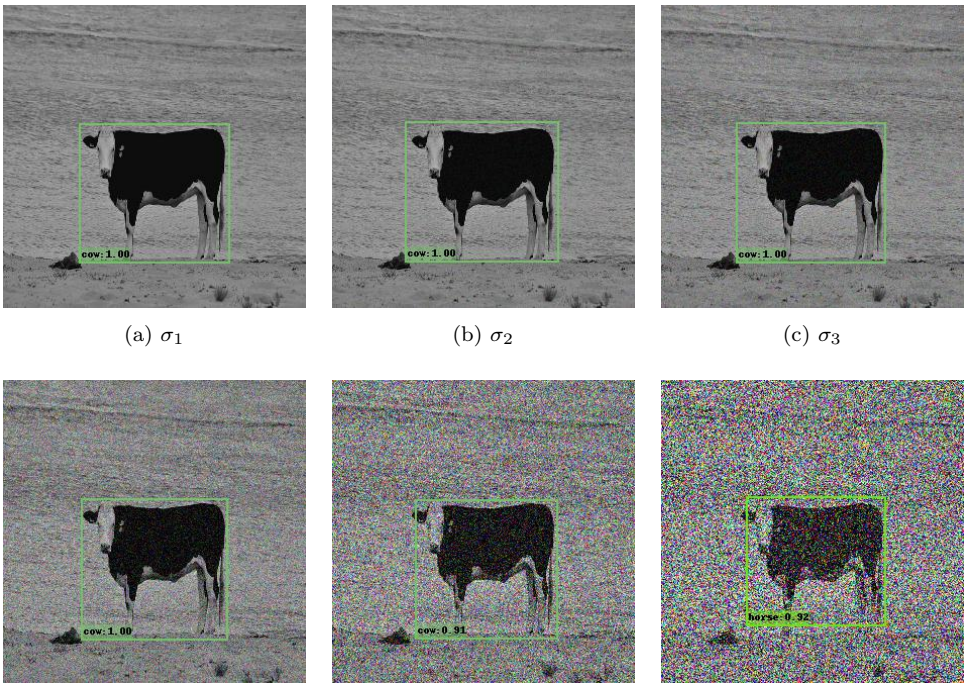
Second, we choose a set of target detectors to use in the evaluations. The detectors we use depend on the dataset the images are being drawn from. For the VOC2012 dataset, which is the dataset the attack models are trained on, we use the detectors shown in Table 4.4. For the COCO dataset, which has an expanded set of class labels compared to VOC2012, we use the detectors shown in Table 4.5. The detectors are evaluated on the COCO2017 validation set.

We use a variety of backbones and architectures in our evaluation. The two black-box architectures are *Faster-RCNN* and *RetinaNet* [31]. As for the backbones, we use VGG16, EfficientNet and ResNet with depth 50 and 101. Additionally, we make use of backbones that use a single resolution scale, marked by *C4*, and backbones that use a feature pyramid structure with several resolutions, marked by *FPN*.

The baseline performances on the VOC2012 test dataset are shown in Table 4.6, and the baseline performances on the COCO dataset are shown in 4.7. The performance in the undefended setting is shown in the σ_1 column, where $\sigma = 0$. It is ideal that these detectors achieve close to state-of-the-art performance on the datasets, to prove that our attacks work against strong detectors. However, we are limited by the fact that we rely on open-source object detection frameworks. As such, the detectors' performance may not achieve state-of-the-art accuracy, but should still be close enough so that our results still show efficacy against strong object detectors. Additionally, when comparing to the state-of-the-art of object detection attacks, we also note that many of the works do not use detectors that achieve state-of-the-art performance either. For example, the SSD300 black-box detector used in [50] achieves a mAP of 0.68 on the VOC test dataset, while our SSD300 white-box detector achieves a mAP of 0.8.

For the COCO dataset, we make use of the pretrained models provided by the Detectron2 library [52] from Facebook reasearch. These pretrained models achieve good performance on the COCO dataset, but state-of-the-art detection has made significant advancements on this dataset. In [8], the proposed detector achieves a mAP of 0.49 on the COCO2017 validation set. This is the detector that has the highest performance on the COCO2017 test-dev set, which is another commonly used object detection benchmark. Therefore, the state-of-the-art on COCO2017 validation is around 0.5 mAP, while our strongest detector ($COCO_4$) has a mAP of 0.41. Therefore, while the detectors we use are strong, they do not achieve state-of-the-art results.

Each column after the Detector ID show the mAP achieved by the detector on the whole dataset with the defense level σ_n applied to the images. As expected, the performance of the detectors deteriorate as the noise becomes more significant. However, we also see that the mAP



(a) σ_1 (b) σ_2 (c) σ_3

(d) σ_4 (e) σ_5 (f) σ_6

Figure 4.1: Levels of defense. Qualitative examples showing the effects of increased noise as a defensive measure. In this example, we see that the defense starts to hide objects in the image and cause misclassifications at defense level σ_6.

decrease in the first three defence levels is not an unacceptable tradeoff if it is an effective defense against adversarial attacks. The fourth level, σ_4, with $\sigma = 25$ in Eq. 4.20, gives a significant mAP drop, but it is still interesting to see what effect this level might have on the performance of the attacks. The last two levels cause an unacceptable mAP drop across the board, and are not reasonable to use in the attack evaluations. To save time, we will only use the first four levels of defense moving forward.

On the datasets we use, we see that some of the detectors have notably better perfomance than others. On the COCO dataset, the $COCO_4$ detector performs best, followed closely by $COCO_3$. Therefore, it is most interesting to see how the attacks affect the performance of these detectors, as they are likely to be the most robust ones in the first place.

We show some quantitative examples of detections under different levels of defense in Figure 4.1. While the defense itself can potentially work against adversarial attacks, there is a drawback to our approach. As we use pretrained weights from the Detectron2 [52] library for our object detectors, we did not have time to train our own object decetion models. Therefore, the detectors we use in the evaluation are not trained with this defense applied to the training images, and as

Detector ID	mAP_{σ_1}	mAP_{σ_2}	mAP_{σ_3}	mAP_{σ_4}	mAP_{σ_5}	mAP_{σ_6}
VOC_1	0.8078	0.8041	0.7822	0.6276	0.3172	0.0715
VOC_2	0.6781	0.6684	0.6341	0.4723	0.2400	0.0702
VOC_3	0.5217	0.5132	0.4877	0.3906	0.2404	0.0182

Table 4.6: Baseline performance on the VOC2012 dataset under different levels of defense

Detector ID	mAP_{σ_1}	mAP_{σ_2}	mAP_{σ_3}	mAP_{σ_4}	mAP_{σ_5}	mAP_{σ_6}
$COCO_1$	0.383	0.374	0.345	0.265	0.160	0.028
$COCO_2$	0.399	0.388	0.363	0.285	0.175	0.037
$COCO_3$	0.408	0.394	0.368	0.287	0.184	0.039
$COCO_4$	0.413	0.403	0.383	0.308	0.189	0.025
$COCO_5$	0.370	0.361	0.338	0.259	0.143	0.024
$COCO_6$	0.386	0.376	0.355	0.275	0.137	0.01

Table 4.7: Baseline performance on the COCO dataset under different levels of defense

such, the performance is more heavily degraded by the noise than it would be if it was trained with noise applied to the training images. Ideally, the detectors should have been trained this way, but this was not possible to do in this thesis due to time constraints. Still, as shown in Tables 4.6 and 4.7, the mAP of the detectors remains almost unchanged until defense level σ_4, when it starts to drop quite a bit. As seen in Figure 4.1, the noise is still quite significant even at defense level σ_3. As such, it is still reasonable to expect this to work against imperceptible attacks.

We note that the performance degradation inflicted by the defense itself becomes high at defense level σ_4. Therefore, to save time, we only evaluate the attacks on defense levels σ_{1-4}.

Chapter 5

Results

In this chapter, we will present the obtained results of the different strategies proposed in Chapter 4. First, we will explain our evaluation procedure. Then, we will provide quantitative and qualitative evaluations of four different attack models that are generated with the approach proposed in Chapter 4.

5.1 Implementation details

5.1.1 Libraries and resources

To experimentally validate the proposed approaches, we need to implement a system to carry out adversarial attacks on object detectors. To implement the required deep networks, we make use of the PyTorch library[42]. For the SSD detector, we make use of an open-source PyTorch implementation [26] which supports the necessary functionality to train and evaluate the object detectors. For Faster-RCNN and RetinaNet, we make use of the Detectron2 library by Facebook Research [52], which supports a wide range of object detection architectures. Both of these libraries provide pre-trained weights for different backbones and configurations that we make use of in our training and validation.

The computations were performed on resources provided by the NTNU IDUN/EPIC computing cluster [45]. The resources provided allowed us to set up a training and evaluation framework to dynamically train and test various attack configurations.

To experimentally validate our approach, we needed the following features in our system: [38]:

- VOC and COCO dataset loaders
- Configurable training of generative adversarial networks

- A configurable evaluation pipeline that calculates the mAP on COCO and VOC
- Detection output visualization

For dataset management, we make use of the Pytorch primitives *Dataset* and *DataLoader*. For the training of GANs, we implemented a training loop which iterates through the VOC2012 train dataset for a set amount of epochs. To evaluate our attacks, we extended the VOC and COCO evaluation pipelines implemented in [26] to support the normally distributed noise defense and the adversarial attack models. For the visualization of outputs, we make use of *vizer* [27].

5.2 Attack configurations

In this section, we give the results of the GAN-based attack proposed in Section 4.2. We will present four different configurations based on the proposed methodology, where the SSD detector's output is used to train the attacking GAN.

In the following subsections, we will describe the network structure and the training parameters of the different attack models we train. Then, we show the performance of each attacker model in tables where we report the target model mAP on clean images, the mAP on perturbed images, the average L_1 and L_2 distances per pixel, and the mean squared error. All of the values are calculated by averaging the results from each image in the validation set. We will also show some qualitative examples of perturbed images and the resulting detections with each configuration.

5.2.1 Discriminator

The networks will use a shared discriminator architecture. The discrimination task is essentially a binary image classification task, in which the discriminator must classify an image as 1 (clean) or 0 (perturbed). This is a task that lends itself well to traditional image classification networks. In this work, we chose to use a model similar to ResNet-34 [17] as the discriminator.

The discriminator is trained using the least-squares GAN loss as in Equation 4.4.

5.2.2 GAN_1

In our first network, GAN_1, we used a deep encoder-decoder architecture using residual blocks, as in ResNet [17]. The encoder consists of 5 convolutional layers that outputs a feature map of size $(W//2^5, H//2^5)$, where // represents integer division. The last feature map is then projected onto C channels by using an additional convolutional layer. The decoder first projects the latent space encoding back to the original channel amount. Then, it uses transposed convolutional layers which mirror the convolutional layers from the encoder to reconstruct the image. C is a

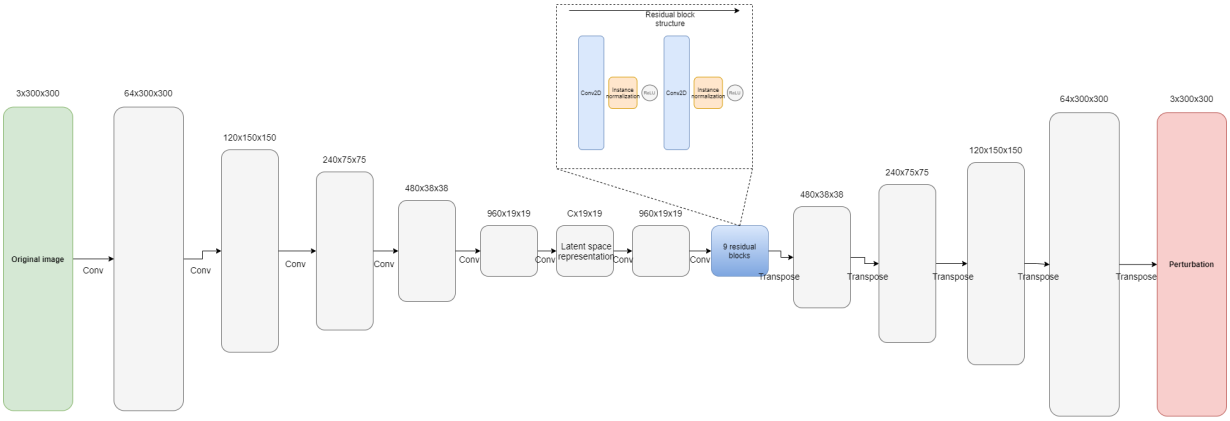


Figure 5.1: Structure of GAN_1. The image is convolutionally processed down to a latent representation of size $Cx19x19$. Then, it is processed with 9 residual blocks, whose structure is shown in the rectangle with dashed borders above. The perturbation is then generated by using transposed convolutions which mirror the convolutions in the encoder.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
VOC_1	0.8078	0.5140	0.8041	0.5139	0.7822	0.4864	0.6276	0.3623
VOC_2	0.6781	0.5725	0.6684	0.5612	0.6341	0.5256	0.4723	0.3800
VOC_3	0.5217	0.4524	0.5132	0.4495	0.4877	0.4288	0.3906	0.3521

Table 5.1: Performance of GAN_1 on the VOC dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

configurable hyperparameter which determines the complexity of the latent space representation of the image. The structure of the encoder-decoder network is shown in Figure 5.1.

This network proved to be quite difficult to get to train properly in conjunction with the discriminator. Quite often, we observed the *mode collapse* phenomenon, in which the generator is able to find a local optimum which consistently fools the discriminator, which is unable to distinguish between this optimum and the unperturbed image. Unfortunately, time constraints stopped us from finding a viable solution to this issue. Thus, the network simply outputs a single, image-independent, universal perturbation, which is learned over several thousand training iterations. However, as the generator is optimized using the loss functions proposed in Chapter 4, the resulting perturbations can still be used successfully in attacks to some capacity. Usually, however, the perturbations are quite perceptible. Nonetheless, we will report these results, both qualitatively and quantitatively. To train the network, we experimented with a variety of optimizer and loss hyperparameter choices.

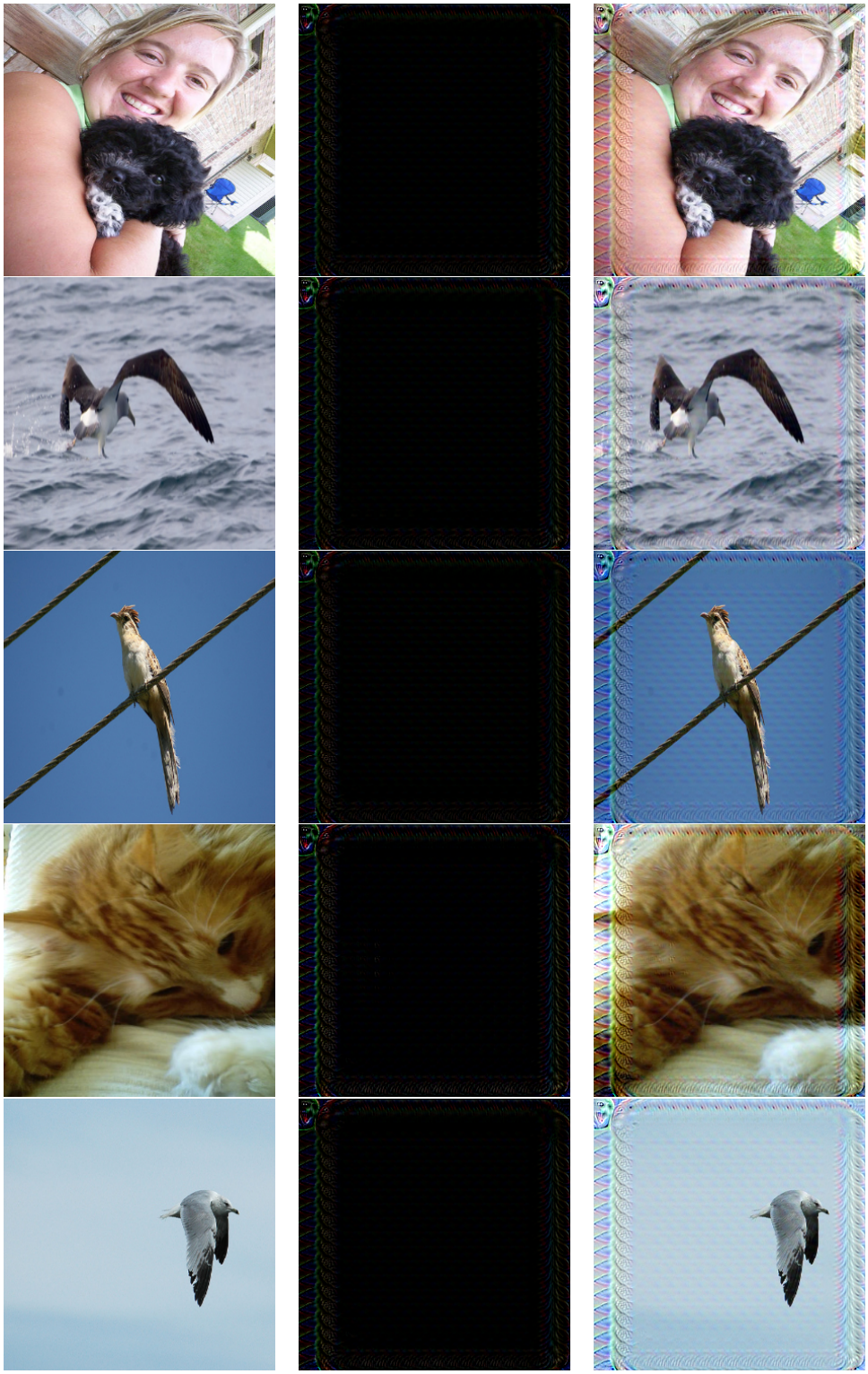


Figure 5.2: Perceptibility of GAN_1. The left image is the clean image, the middle image shows the perturbation, and the right image shows the resulting perturbed image.

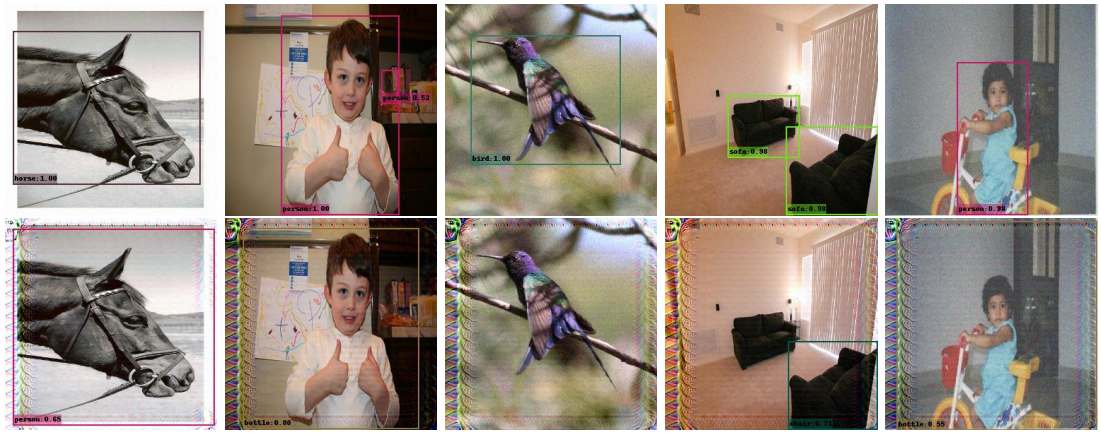


Figure 5.3: Examples of successful GAN_1-attacks against COCO images at defense level σ_1. The detections are done by the VOC_1 detector.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
VOC_1	0.8078	0.3078	0.8041	0.3117	0.7822	0.3124	0.6276	0.2792
VOC_2	0.6781	0.2924	0.6684	0.2948	0.6341	0.3102	0.4723	0.3106
VOC_3	0.5217	0.2330	0.5132	0.2372	0.4877	0.2528	0.3906	0.2825

Table 5.2: Performance of GAN_2 on the VOC dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

In Table 5.1, we show the mAP degradation achieved on the VOC2012 dataset under 4 different levels of defense (σ_{1-4}).

5.2.3 GAN_2

For the second network, GAN_2, we create a generator network that does not encode the image to a latent space representation. It is simply a series of convolutional layers that do not cause any dimensionality reduction at all. We found that models using this network structure were far more easily trained than the encoder-decoder proposed in GAN_1. This was especially useful as we relaxed the imperceptibility constraint of the generated adversarial perturbations, due to the single-pass queryless setting. The network is structured as 9 convolutional layers with 100 channels each. *Instance normalization* [47] is applied after each layer.

This network is trained using the performance degradation loss from Equation 3.1, which seeks to maximize the regression and class losses from the SSD object detector. As we shall see in the qualitative results, this leads to some quite interesting emergent behavior which lead to very powerful object-fabrication attacks in the white box setting. Additionally, by adding the

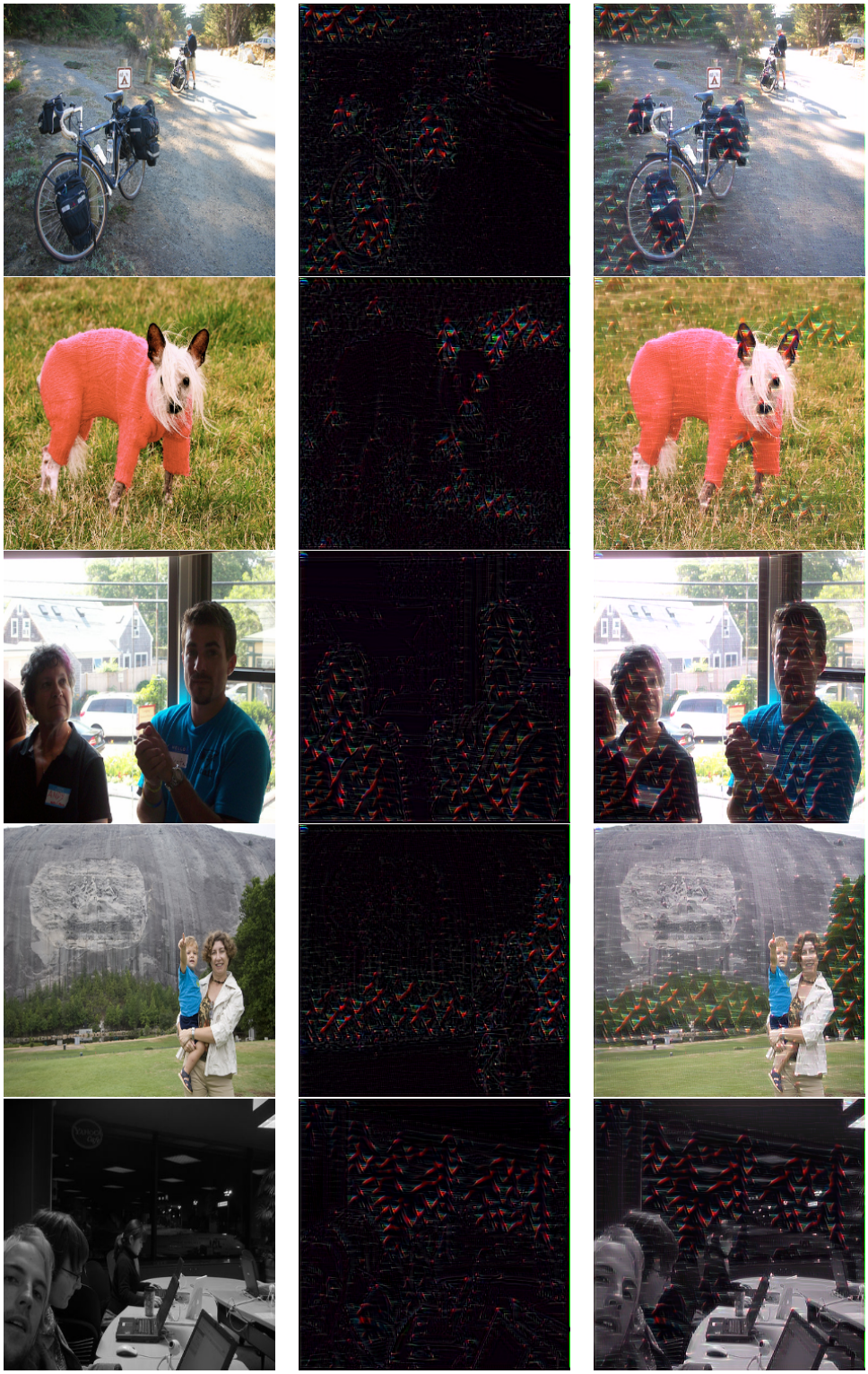


Figure 5.4: Perceptibility of GAN_2. The left image is the clean image, the middle image shows the perturbation, and the right image shows the resulting perturbed image.

Figure 5.5: Examples of successful GAN_2 attacks against the white-box detector VOC_1 at defense level σ_1.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
$COCO_1$	0.383	0.185	0.374	0.195	0.345	0.192	0.265	0.183
$COCO_2$	0.399	0.2	0.388	0.2	0.363	0.202	0.285	0.202
$COCO_3$	0.408	0.193	0.394	0.196	0.368	0.198	0.287	0.196
$COCO_4$	0.413	0.217	0.403	0.214	0.383	0.215	0.308	0.201
$COCO_5$	0.370	0.167	0.361	0.169	0.338	0.177	0.259	0.176
$COCO_6$	0.386	0.196	0.376	0.197	0.355	0.194	0.275	0.179

Table 5.3: GAN_2 performance on the COCO dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

object fabrication loss from Section 4.2.1, we gain additional control over the fabricated objects that are generated in the image.

The perceptibility of the resulting perturbations can be seen in Figure 5.4. As we did for GAN_1, we run the experiment on the VOC2012 validation set to evaluate the model's performance. In Table 5.2, we show the obtained mAP reduction under 4 different levels of defense (σ_{1-4}). We see that the attack reduces the mAP of VOC_1 from 0.8078 to 0.3078, a mAP reduction of 0.5. Additionally, the performance remains more or less constant when more powerful noise is added to the image, suggesting that the attack is resistant to the effects of the defense.

Additionally, the performance on the grey- and black-box detectors VOC_2 and VOC_3 is decent as well, with a performance reduction of over 50% on both networks.

But the mAP reduction is not the only beneficial aspect of this attack. To understand its potential, we have to look at some qualitative detection examples. First, we show some qualitative

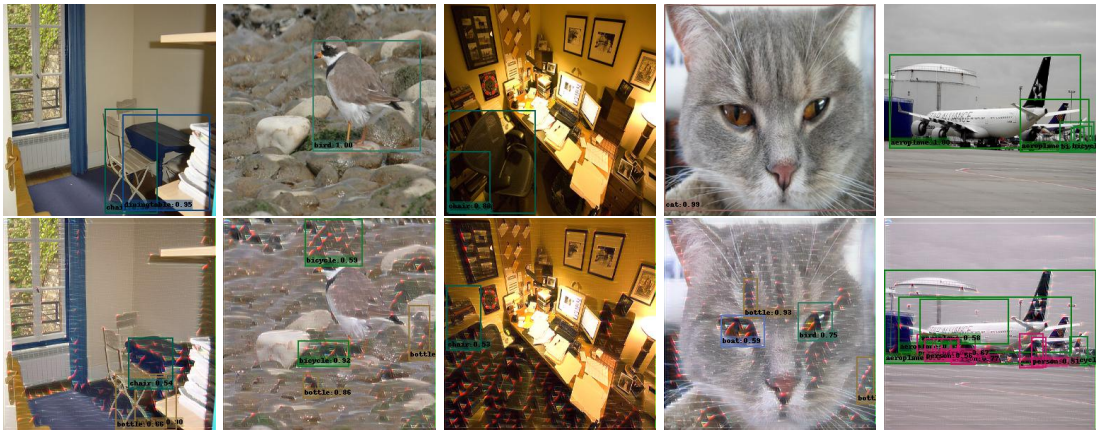


Figure 5.6: Examples of successful GAN_2 attacks against the black-box detector VOC_3 at defense level σ_1.

examples of successful attacks against the white-box detector VOC_1 in Figure 5.5. As the model is trained to maximize the total loss of VOC_1, the attack successfully fabricates false positives in most of the validation images. It also does this while minimizing the L_2-distance between the perturbed and the clean image. As such, we end up with a perturbation that is somewhat perceptible, but also highly effective in the white-box context.

Surprisingly, however, this effect also carries over to the black-box detector VOC_3. In Figure 5.6, we show some examples of object detector outputs made by VOC_3, using the same images as in Figure 5.5. We see that various objects are fabricated in the sample images, e.g. resulting in detections of a boat and a bird in the eyes of the cat in the fourth image. This suggests that the object fabrication is transferrable across different architectures and backbones. Additionally, the real objects are also hidden in many cases, which is reflected in the significant mAP reduction. Therefore, judging by the results on the VOC dataset, this attack is a powerful object fabrication attack.

It is also interesting to see how the attack performs on the COCO dataset, which has class labels that are not known to the attacker. In Table 5.3, we show the mAP degradation achieved on the black-box detectors on the COCO dataset under 4 levels of defense (σ_{1-4}). We see that the attack is capable of achieving significant mAP reduction in this setting as well, reducing the mAP of $COCO_4$ from 0.413 to 0.217, a mAP drop of 0.196. We see that the increasing levels of defense reduce the absolute mAP drop slightly, but it is still effective at reducing the detectors' mAP, even when some noise is applied to the images.

Then, we show some qualitative examples of detections made by the $COCO_4$ detector on the COCO dataset in Figure 5.7. Interestingly, the attack is still able to cause object misclassification and object fabrication, even on a dataset with different class labels and against a different classification backbone. However, by manual examination of the detection outputs, we noted

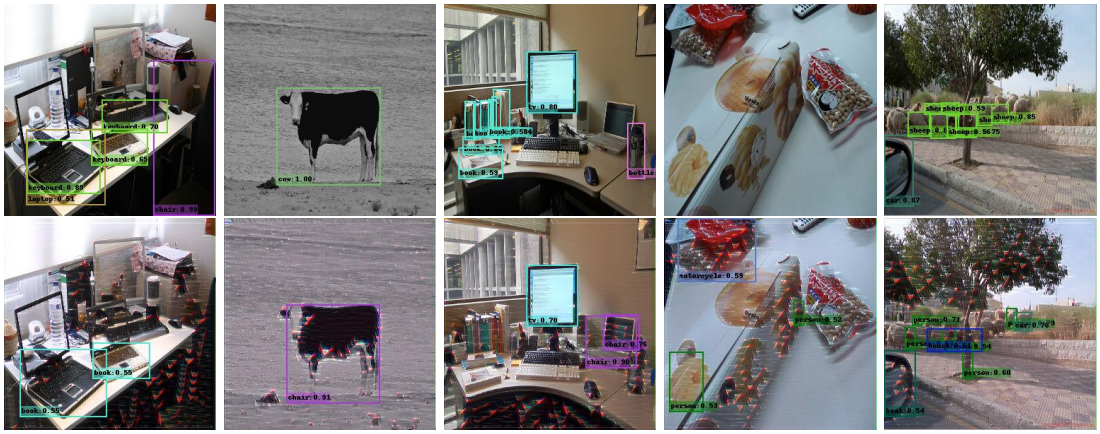


Figure 5.7: Examples of successful GAN_2-attacks against COCO images at defense level σ_1. The detections are done by the $COCO_4$ detector, which is the strongest of the black-box targets on this dataset.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
VOC_1	0.8078	0.6578	0.8041	0.6483	0.7822	0.6242	0.6276	0.4777
VOC_2	0.6781	0.5436	0.6684	0.5359	0.6341	0.5184	0.4723	0.3861
VOC_3	0.5217	0.4219	0.5132	0.4175	0.4877	0.4070	0.3906	0.3383

Table 5.4: Performance of GAN_3 on the VOC dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

that GAN_2 causes significantly less object fabrication on the COCO dataset than it does on the VOC dataset. Therefore, we can only say that this attack is transferrable across black-box object detectors, but perhaps not across different datasets. Still, it is an interesting result.

5.2.4 GAN_3

Our third network, GAN_3, is set up in a similiar way to the GAN_2 network. However, we opted to try a shallower network, with only 5 convolutional layers. Additionally, we make use of a combination of the object-hiding loss in Equation 4.9 and the performance-degradation loss in Equation 3.1 during training. For GAN_3, we wanted to make the attack more imperceptible, and used a hinge loss threshold of $c = 40$.

First, we evaluate the attack on the VOC dataset. The results we obtained are shown in Table 5.4. In the black-box setting, against VOC_3, the attack performs quite poorly, reducing the mAP from 0.5217 to 0.4219, a mAP drop of 0.0998. Compared to the results achieved in GAN_2, this is quite weak. As such, we do not evaluate it on the COCO dataset to save time, as

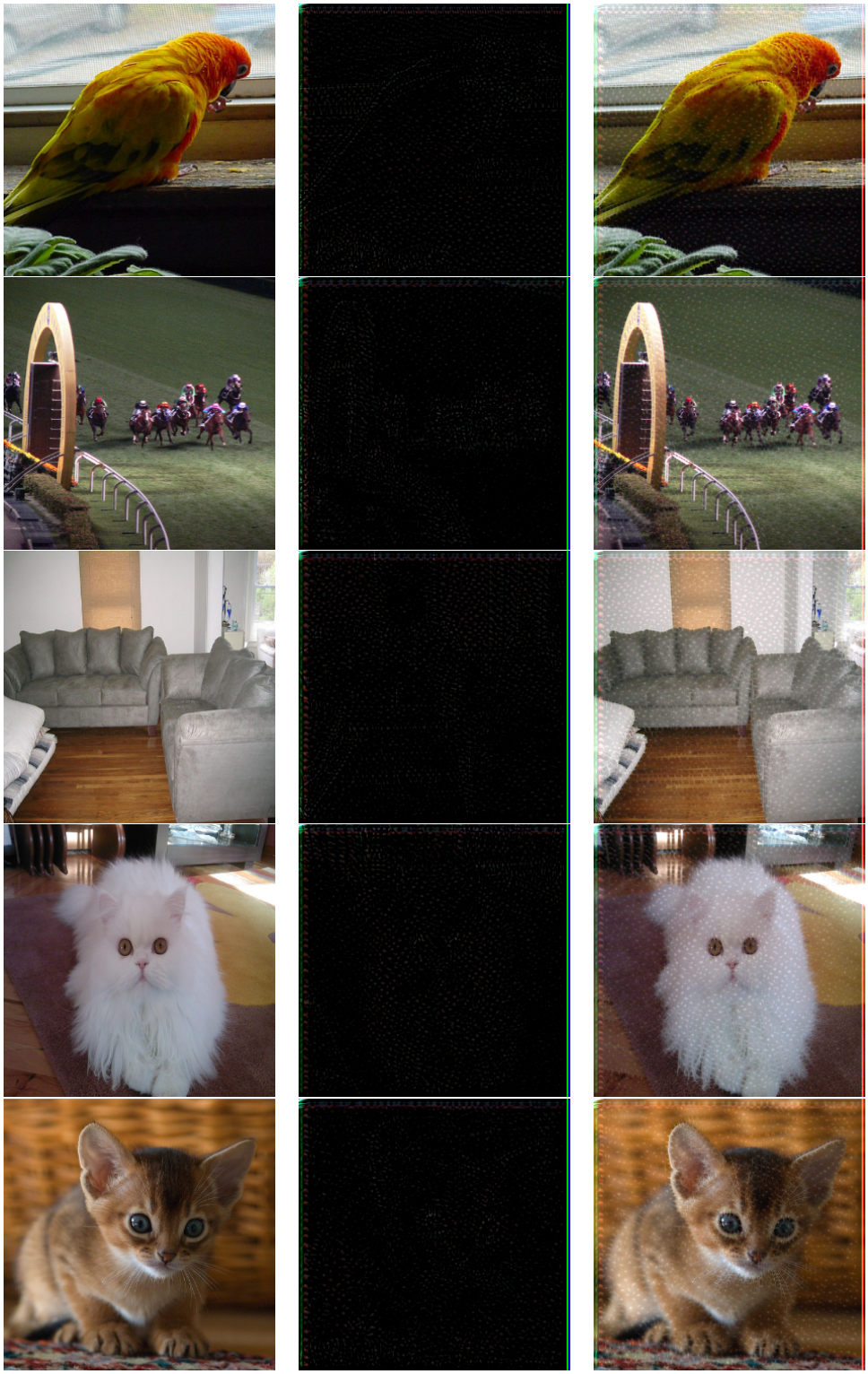


Figure 5.8: Perceptibility of GAN_3. The left image is the clean image, the middle image shows the perturbation, and the right image shows the resulting perturbed image.

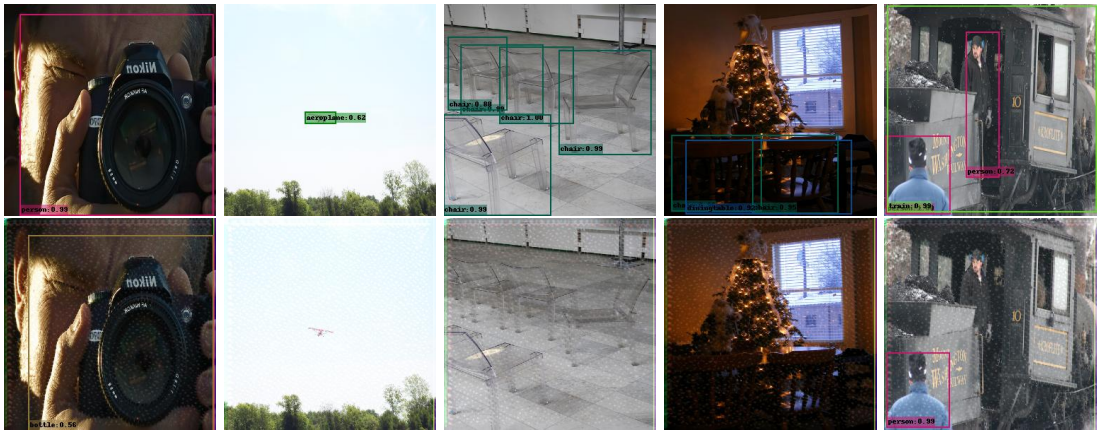


Figure 5.9: Examples of successful GAN_3 attacks against the white-box detector VOC_1 at defense level σ_1.

Model	L_1 distance	L_2 distance	Mean squared error
GAN_1	22.3	0.09	240
GAN_2	32.4	0.11	373
GAN_3	11.1	0.07	156
GAN_4	27.6	0.095	270
GAN_4^{90}	61.2	0.18	915
GAN_4^{110}	91.0	0.27	2108

Table 5.5: Perturbation size measurements of the GAN configurations

the results would likely be similiar there.

We show the perceptibility of the attack in Figure 5.8. Of all our attacks, this one is by far the hardest to see for a human observer. We also show some examples of successful attacks on the white-box detector VOC_1 in 5.9, which shows that the attack can work in some situations. However, given the lackluster mAP degradation achieved on the VOC dataset, this by itself is not sufficient - other works in the state-of-the-art are equally difficult to see, but are more powerful. Therefore, we chose to abandon this network configuration.

5.2.5 GAN_4

Configuration

For GAN_4, we make use of the object-hiding approach specified in Section 4.2.1 as our sole optimization goal, combined with the discriminator loss. We train a GAN to generate perturbations that uses Equation 4.9 in its loss function. Additionally, we use a 9-layer generator architecture

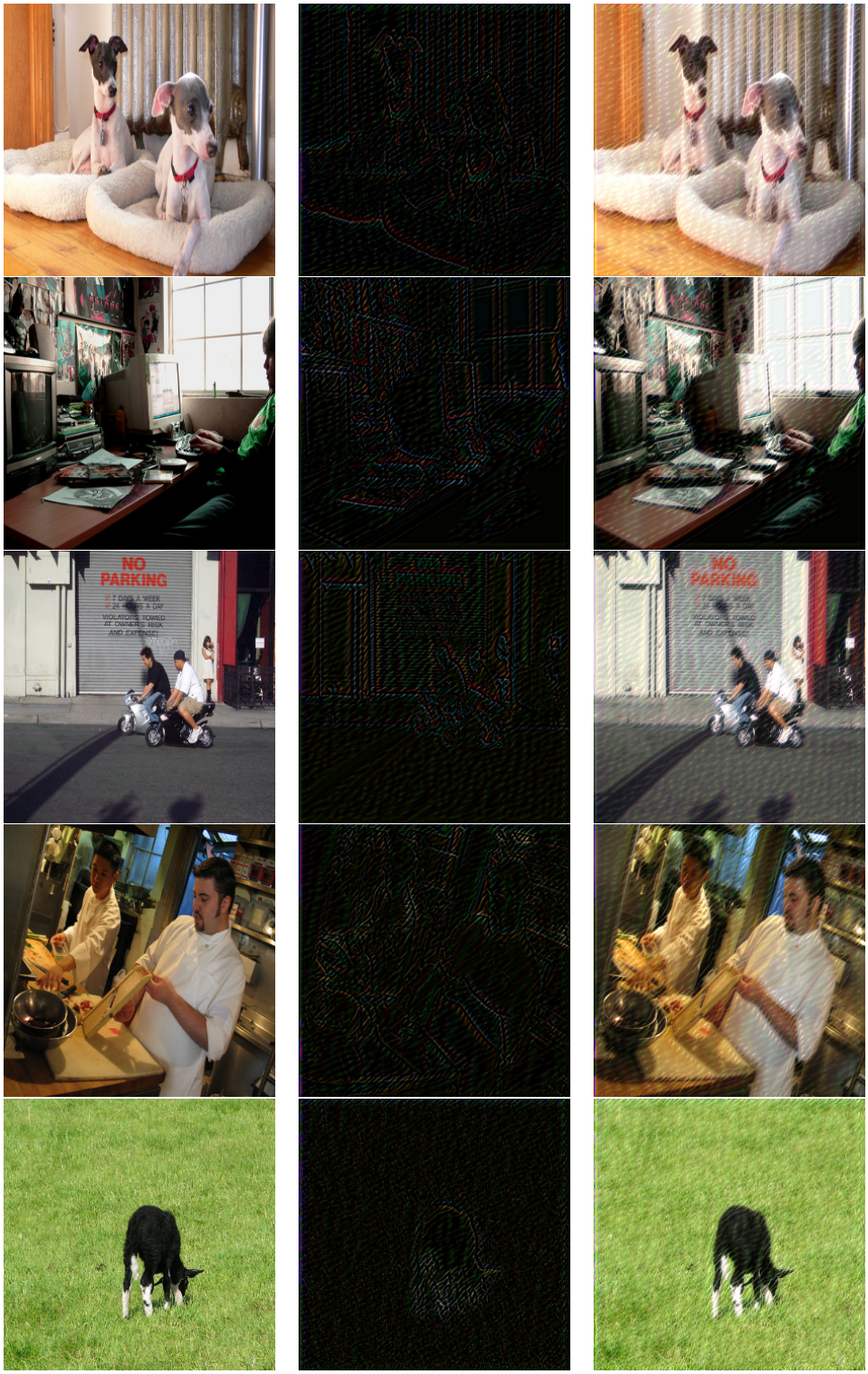


Figure 5.10: Perceptibility of GAN_4. The left image is the clean image, the middle image shows the perturbation, and the right image shows the resulting perturbed image.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
VOC_1	0.807	0.138	0.804	0.139	0.782	0.143	0.627	0.162
VOC_2	0.678	0.281	0.668	0.288	0.634	0.275	0.472	0.207
VOC_3	0.521	0.147	0.513	0.148	0.487	0.152	0.390	0.187

Table 5.6: Performance of GAN_4 on the VOC dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
$COCO_1$	0.383	0.107	0.374	0.102	0.345	0.102	0.265	0.122
$COCO_2$	0.399	0.144	0.388	0.143	0.363	0.142	0.285	0.15
$COCO_3$	0.408	0.148	0.394	0.146	0.368	0.141	0.287	0.152
$COCO_4$	0.413	0.149	0.403	0.147	0.383	0.148	0.308	0.162
$COCO_5$	0.370	0.103	0.361	0.102	0.338	0.103	0.259	0.122
$COCO_6$	0.386	0.125	0.376	0.123	0.355	0.124	0.275	0.13

Table 5.7: GAN_4 performance on the COCO dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

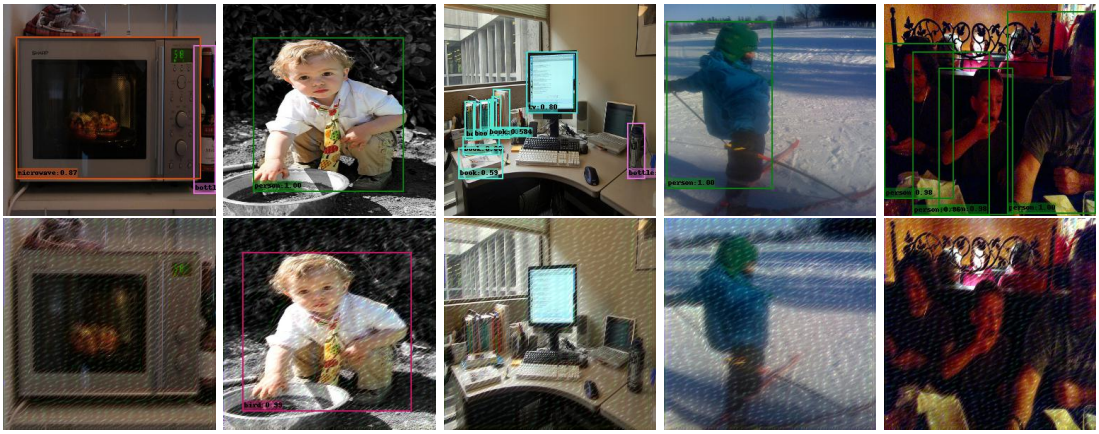


Figure 5.11: Examples of successful GAN_4-attacks against COCO images at defense level σ_1. The detections are done by the $COCO_4$ detector, which is the strongest of the black-box targets on this dataset.

that convolutionally processes the input image to generate the perturbation.

Results on the VOC dataset

In Table 5.6, we show the performance of GAN_4 on the VOC2012 dataset. First, we consider the white-box setting, which is the attack against the VOC_1 detector. Here we expect a high performance. We see that the mAP of the detector against perturbed images at defense level σ_1 (i.e. no noise is applied to the image) is significant, going from 0.8078 to 0.1380, which is a mAP drop of 0.6698. We see that as the level of defense increases, the performance of the detector on the perturbed images increases slightly. This is an expected result in the white-box setting, as the attack model will be strongly adapted to this detector. As such, the random changes to the image make the perturbations slightly less effective. However, the mAP drop under the initial levels of defense is still very significant, and the attack is sufficiently powerful in this regard.

Against the grey-box detector VOC_2, which uses the SSD architecture with a new backbone, we see that the attack reduces the mAP significantly, from 0.6781 to 0.2815, resulting in a mAP drop of 0.3966. However, the mAP drop is far less significant than in the white-box setting. We also see that the defense levels do not significantly affect the mAP drop the attack achieves, until σ_4, at which point the detector's performance is also significantly degraded by the defense itself.

Finally, against the black-box Faster-RCNN-based detector, VOC_3, the attack also performs quite well. It reduces the mAP from 0.5217 to 0.1476 at defense level σ_1. Interestingly, we see that the defense levels up to σ_4 result in a higher mAP on the perturbed dataset, indicating that the defense has the potential to robustify the Faster-RCNN detector to some extent.

We show some examples of perturbations and perturbed images in Figure 5.10. Qualitatively,

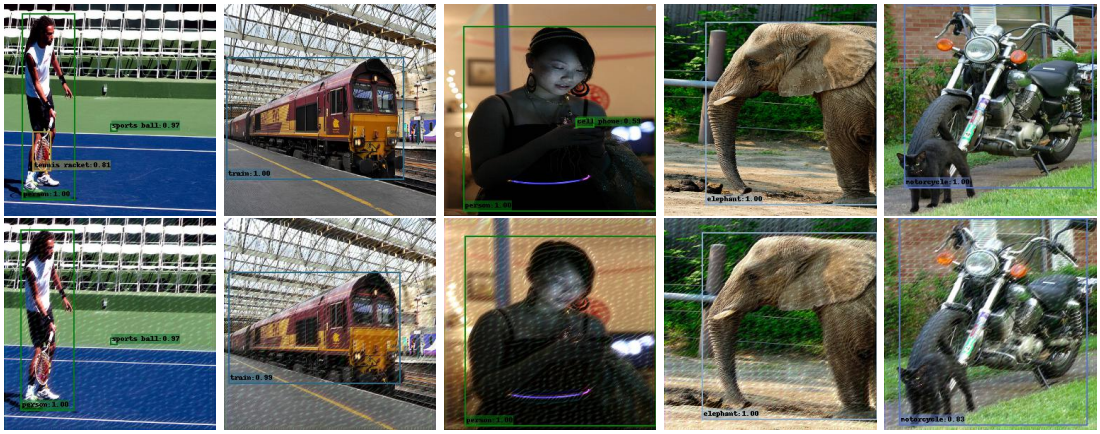


Figure 5.12: Examples of failed and partially failed GAN_4-attacks against COCO images at defense level σ_1. The detections are done by the $COCO_4$ detector, which is the strongest of the black-box targets on this dataset.

we can note the relatively innocuous appearance of the generated perturbations. While it is evident that the image has been modified, the appearance of the perturbed images could also be attributed to some other innocuous factor, such as weather conditions like rain. Additionally, given the impressive black-box performance reduction against faster-RCNN, it is possible that the increased perceptiblity of the perturbations is a sacrifice worth making in this case.

Results on the COCO dataset

Then, we consider the results achieved on the COCO dataset, which are given in Table 5.7. Some examples of detections made by $COCO_4$ on the clean and perturbed images are shown in Figure 5.11. Initially, in the first level of defense, we see a similiar performance on all the detectors. On $COCO_4$, the strongest detector, the mAP drops from 0.413 to 0.149 on the undefended images, giving a mAP drop of 0.264. This is a quite solid mAP drop on the COCO dataset, as state-of-the-art mAP by object detectors on this dataset is generally much lower than on e.g. VOC. We also see that the attack works on RetinaNet-based detectors ($COCO_{5-6}$) as well as on Faster-RCNN-based detectors ($COCO_{1-4}$), as well as with all the backbones we evaluate on, indicating significant black-box transferability. We also see that the attack is robust against the levels of noise defense that do not significantly reduce the performance of the detector in the first place (σ_{1-3}), suggesting that more complex defenses would be required to successfully combat GAN_4 with its initial configuration.

However, the performance of the detectors is not degraded completely - a significant amount of detections are still being made correctly. We show some qualitative examples of failed attacks in Figure 5.12. Some of the attack failures are partial, which means that some of the objects are

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
VOC_1	0.807	0.087	0.804	0.088	0.782	0.085	0.627	0.088
VOC_2	0.678	0.139	0.668	0.139	0.634	0.138	0.472	0.114
VOC_3	0.521	0.039	0.513	0.059	0.487	0.062	0.390	0.064

Table 5.8: Performance of GAN_4^{90} on the VOC dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
$COCO_1$	0.383	0.051	0.374	0.054	0.345	0.055	0.265	0.065
$COCO_2$	0.399	0.081	0.388	0.077	0.363	0.077	0.285	0.078
$COCO_3$	0.408	0.081	0.394	0.08	0.368	0.079	0.287	0.081
$COCO_4$	0.413	0.081	0.403	0.079	0.383	0.076	0.308	0.08
$COCO_5$	0.370	0.049	0.361	0.05	0.338	0.051	0.259	0.059
$COCO_6$	0.386	0.065	0.376	0.065	0.355	0.067	0.275	0.063

Table 5.9: GAN_4^{90} with hinge loss threshold $c = 90$ performance on the COCO dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

suppressed, but not all.

5.2.6 Varying the perceptibility of the perturbations

We are interested in investigating the relationship between the perceptibility of the perturbations and the resulting performance drop of the object detectors. Given that the initial configuration of GAN_4 results in a model that can generate perturbations that are somewhat difficult to see, we can see what happens when we vary the constraints on the magnitude of the generated perturbations. Intuitively, a more perceptible perturbation will result in a more effective attack.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
VOC_1	0.807	0.027	0.804	0.026	0.782	0.023	0.627	0.034
VOC_2	0.678	0.070	0.668	0.069	0.634	0.076	0.472	0.075
VOC_3	0.521	0.039	0.513	0.059	0.487	0.062	0.390	0.064

Table 5.10: Performance of GAN_4^{110} on the VOC dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

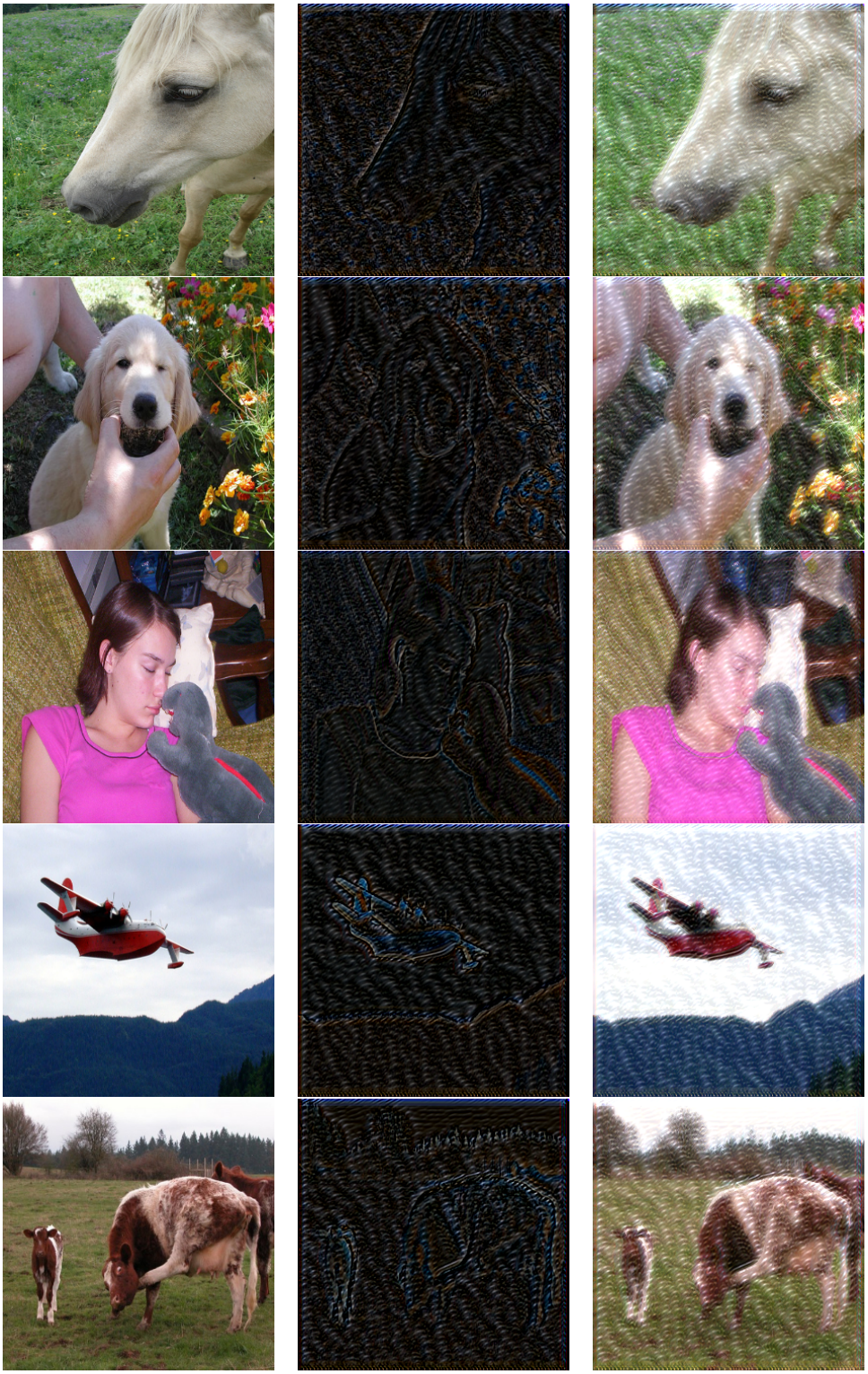


Figure 5.13: Perceptibility of GAN_4^{90}. The left image is the clean image, the middle image shows the perturbation, and the right image shows the resulting perturbed image.

Figure 5.14: Examples of successful GAN_4^{90}-attacks against COCO images at defense level σ_1. The detections are done by the $COCO_4$ detector, which is the strongest of the black-box targets on this dataset.

Detector ID	mAP_{σ_1}		mAP_{σ_2}		mAP_{σ_3}		mAP_{σ_4}	
	C	P	C	P	C	P	C	P
$COCO_1$	0.383	0.015	0.374	0.015	0.345	0.014	0.265	0.016
$COCO_2$	0.399	0.03	0.388	0.03	0.363	0.03	0.285	0.027
$COCO_3$	0.408	0.026	0.394	0.024	0.368	0.024	0.287	0.022
$COCO_4$	0.413	0.014	0.403	0.013	0.383	0.014	0.308	0.014
$COCO_5$	0.370	0.024	0.361	0.011	0.338	0.011	0.259	0.011
$COCO_6$	0.386	0.016	0.376	0.016	0.355	0.016	0.275	0.015

Table 5.11: GAN_4^{110} with hinge loss threshold $c = 110$ performance on the COCO dataset under different levels of defense. The mAP on clean images (C) and perturbed images (P) are shown.

However, the objects in the image should still be visually recognizable by a human - naturally, a perturbation that renders the image visually incomprehensible will lead to a very high mAP drop.

A simple way to change the perceptibility constraint during training is to change the hinge loss $\mathcal{L}_H$ from Equation 4.7, which we do by changing the hinge loss threshold c. In GAN_4, the initial hinge loss threshold is set to $c = 70$. Thus, we train two additional networks with higher hinge losses and carry out the evaluation procedure on these as well. The additional networks we train are denoted GAN_4^{90} and GAN_4^{110}, with hinge loss thresholds set to 90 and 110, respectively.

First, we present the results of GAN_4^{90}. In Figure 5.13, we show the perceptibility of the attack. As this model was allowed a stronger perturbation than the original GAN_4, it is clear that the resulting perturbations are more obvious to the eye than those of GAN_4. White patches

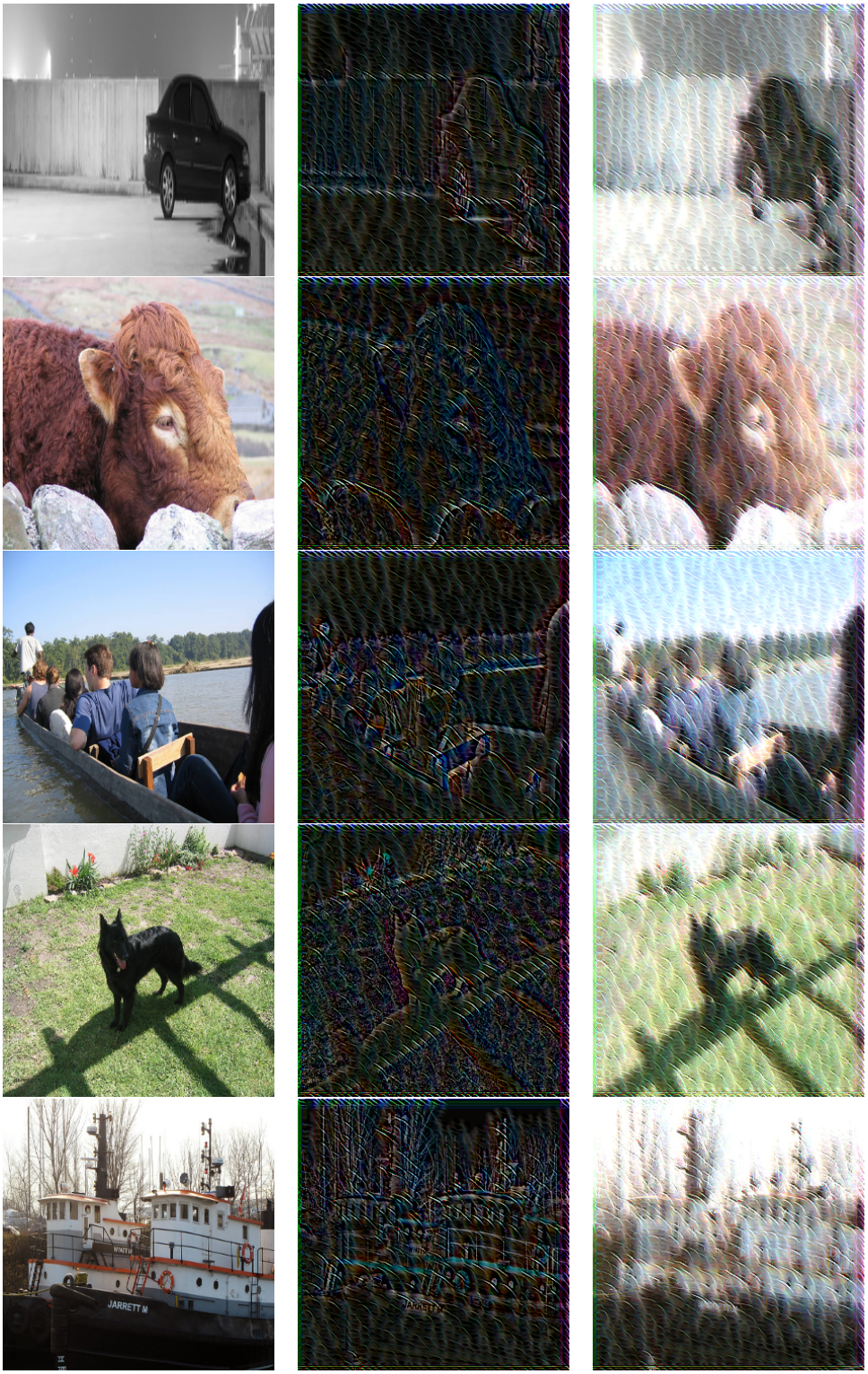


Figure 5.15: Perceptibility of GAN_4^{110}. The left image is the clean image, the middle image shows the perturbation, and the right image shows the resulting perturbed image.

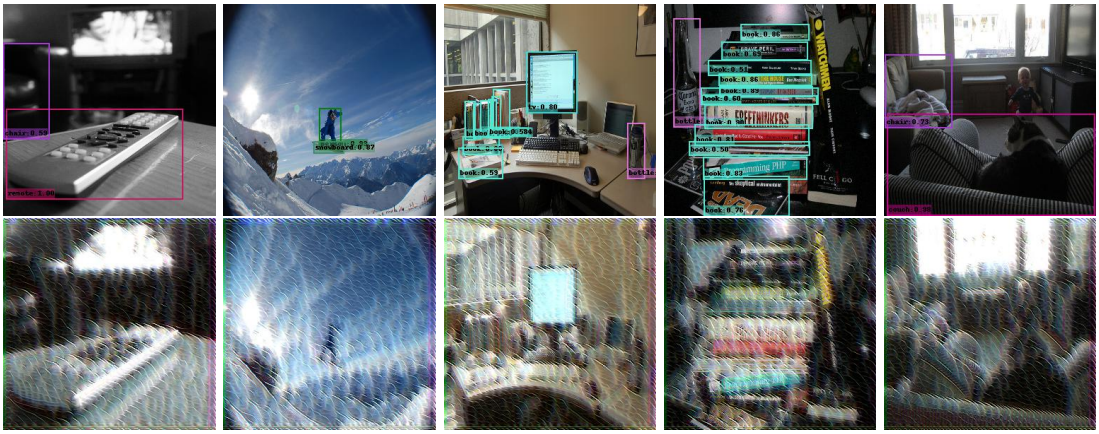


Figure 5.16: Examples of successful GAN_4^{110}-attacks against COCO images at defense level σ_1. The detections are done by the $COCO_4$ detector, which is the strongest of the black-box targets on this dataset.

are visible throughout the image. However, the objects are still visually distinguishable in most cases, which an object detector should be able to classify.

In Table 5.8 we show the results of the attack on the VOC dataset. We see that the attack has a significantly higher effect on the performance of the detectors, achieving a mAP drop of $TODO$ in the undefended case in the black-box detector. The white- and grey-box performances are also similiarly impressive compared to that of GAN_4.

In Table 5.10 we show the results of the attack on the COCO dataset. A similiar improvement in the attack impact is seen here, obtaining a mAP drop from 0.413 to 0.081 on the $COCO_4$ detector, a mAP drop of 0.332. As before, we see that the attack is not significantly affected by the application of the normally distributed noise until σ_4, at which point the performance of the target detectors are deteriorated on the clean images as well. As such, this attack has a very impressive performance on all the black-box detectors.

In Figure 5.15 we show the perceptibility of GAN_4^{110}. With a hinge loss threshold as high as 110 we see that the resulting perturbations significantly change the image. While the objects originally present in the image are usually visually distinguishable, it is still a very perceptible perturbation, and is pushing the limits of what is acceptable in this regard. Nonetheless, with such a significant perturbation, one would expect the resulting performance reduction to be significant.

In Table 5.10 we show the results of GAN_4^{110} on the VOC dataset. We see that on each of the detectors, the attack achieves a significant performance reduction. Notably, the black-box detector VOC_3 has its mAP reduced to 0.04 on the undefended images. The performance reduction on the white- and grey-box detectors VOC_1 and VOC_2 is also significant. Interestingly, the grey-box detector retains the most of its performance out of the three detectors on this

dataset.

As expected, the attack works incredibly well on the black-box detectors using COCO labels, achieving an almost complete performance degradation and reducing the mAP to practically zero. Clearly, this model has crossed a threshold where the perturbations are so significant that the detectors can no longer function at all. However, as can be seen in Figure 5.15, we see that the perceptibility is close to rendering the underlying objects visually unrecognizable. In the last image, for example, it is difficult to see if the object in the image is a boat or something else. However, in some of the other images, the underlying objects are still recognizable.

In Table 5.5, we show the average L_1 distance, L_2 distance, and the MSE between the clean and perturbed images in the VOC dataset. As expected, the attacks that were most successful in reducing the detectors' mAP (GAN_4^{90} and GAN_4^{110}) have very high MSEs. Even though a high MSE does not necessarily mean that an attack is more perceptible by humans, it is still desirable to keep the MSE as low as possible. Therefore, the attacks that strike the best balance between perturbation magnitude and performance degradation are GAN_2 and GAN_4 respectively. GAN_4 in particular achieves state-of-the-art black-box performance degradation while being somewhat difficult to see.

Chapter 6

Discussion

In this chapter, we will give a discussion on the results we obtained in Chapter 5. We shall outline the way in which the results answer the research question, and reflect over the position of our work in the current state-of-the-art on attacks against object detection models.

6.1 Comparison to state-of-the-art

In this work, we wanted to develop an attack against black-box, defended object detectors, without querying the target model. This attack setting is perhaps the most restrictive of all, disregarding system limitations such as restrictions on available memory. In the related works that we outlined in Chapter 3, the vast majority of the attacks did not operate in the single-pass queryless setting, requiring iterative optimization to work. The work that is most closely related to our own is [50], which also uses a generative model to attack black-box detectors. In their evaluation on the VOC dataset, they achieve a white-box mAP degradation from 0.7 to 0.05 against the Faster-RCNN detector, and a black-box mAP drop from 0.68 to 0.20 on the SSD300 detector. Additionally, their attacks are quite difficult to perceive by the human eye. While their results are good, the evaluation they carry out is far less extensive than ours. First, we evaluate our attacks against 1 grey-box and 7 black-box detectors, using 3 different detector architectures, while they only evaluate against one white-box and one black-box detector. Second, we apply a basic generic noise defense to the evaluation images, while their evaluation is done without any defenses applied. Third, we evaluate our attacks against two different datasets, one with an expanded set of class labels. Lastly, we report the average perturbation sizes of our perturbations, which is not done in [50]. Therefore, we argue that the results we achieve have a stronger foundation than those achieved in [50].

We can also directly compare our results with theirs. We highlight GAN_4 as an effective attack that balances peceptiblity with mAP degradation. As seen in Tables 5.6 and 5.7, the

gan_4 attack successfully degrades the performance of the detectors. However, there is room for improvement. One weakness of the attack is that it does not perform well in the grey-box setting. Because of this, we also showed that by allowing a stronger perturbation by changing the hinge loss threshold, the resulting attacks quickly became much stronger. Depending on the attack scenario, the perceptibility of GAN_4^{90} could be an acceptable tradeoff. GAN_4^{110} is likely too perceptible.

With GAN_2, we were also able to generate a powerful transferable object-fabrication attack, which also degraded the mAP of the detectors we evaluated the attacks on. This attack was also resistant to the noise defense. While the attack that achieved the best tradeoff between transferability and perceptibility was GAN_4, we also believe that the results achieved in GAN_2 are significant, and they show the potential threat that a GAN-based attack represents to object detection models.

6.2 Threats to validity

Our work has some threats to validity that are important to point out, and that can be improved in future works. One important aspect of our work was evaluating the attack against object detectors that have a basic defense applied to them. We elected to use a normally distributed noise pattern, which has been used as part of defenses to combat finely-crafted imperceptible perturbations. However, the target object detectors should ideally be trained with this defense, so that the defense itself does not play a part in degrading the performance of the detector. While we show that our attacks are effective against the defense levels that do not notably degrade the performance of the detectors on clean images (σ_{1-3}), the detectors could likely be made more robust to the effects of the noise defense by applying this noise to the images used during training. Perhaps the detectors could handle the higher defense levels gracefully in that case. Unfortunately, we did not have the time and resources available to retrain the detectors in this way, and as such, the defense is not as strong as it could have been.

Second, it would be ideal to use object detectors that achieve state-of-the-art performance on the datasets we use. However, the detectors we use do not achieve results as good as some cutting-edge state-of-the-art models. This is a limitation, as object detection models will continue to improve with time, and showing that the attacks work against the best state-of-the-art models would lend more credibility to our results. It must be noted, however that many of the other works proposing attacks on object detection suffer from this same flaw, including [50]. We compensate for this flaw by evaluating on a diverse array of architectures and backbones that achieve good baseline performances.

6.3 Academic implications

In this work, we have made contributions to the domain of adversarial attacks against object detection. Notably, our attack is queryless, resistant to basic defense, and transferrable across datasets and black-box detectors. We observed in Chapter 3 that the majority of existing adversarial attacks on object detection models are not single-pass, and are therefore not viable when attacking real-time object detection models. Additionally, many works are not demonstrated to be effective against even basic defenses, including [50], the closest work to our own. Therefore, it would be good for the field if the evaluations of attacks in the future are more extensive.

We have also used the SSD300 detector as the white-box base model for generating black-box attacks in a novel way. In [50], Faster-RCNN was used as the white-box detector. They used the region proposals of the RPN to train the attacker, while we used the default prior boxes of SSD300. In general, it seems as though using inherent design features of object detectors to generate attacks is a good way of generating attacks. For example, one could use the attention maps in [8], the highest-performing detector on COCO (at the moment of writing), to train a generative model.

6.3.1 Physical adversarial attacks

In our literature review, we found that physical adversarial attacks were underrepresented, compared to the threat they represent. After all, a digital adversarial attack assumes that the attacker is able to modify the digital input to the object detector. However, it is difficult to compromise a system such that this is possible. Therefore, an adversarial attack that can deteriorate the performance of object detectors by simply placing a physical patch within the field of view of a camera is a more likely threat to encounter. Naturally, successfully designing such an attack is a much more difficult task, as one must consider various external factors, such as outside lighting conditions.

Additionally, finding defenses against this type of attack is equally important. This is the type of attack that one is most likely to encounter in the field, and must be defended against in safety-critical systems.

6.3.2 Single-pass attacks

Another field of research that should be pursued is designing new single-pass attacks. These attacks are significantly more viable against object detection models than iterative optimization attacks, due to the fact that they can be done without querying the detector. In this thesis, we have designed such an attack, but there are still significant improvements that can be made in this domain of attacks.

6.3.3 Reducing perceptibility of existing attacks

One potential approach could be to reduce the perceptibility of a generated attack. Essentially, one would have to maintain the efficacy of the generated perturbations, while reducing the L_2-distance between the perturbed and clean images. For example, one approach could be using RL to control the intensity of the perturbations that are applied to the images. In [55], an adversarial attack on image classifiers is proposed that leverages RL to position adversarial patches in the image. A similiar strategy for adversarial attacks on object detection could be attempted.

6.3.4 Mitigations

Naturally, a very important subfield of adversarial attacks is developing effective mitigation strategies against the attacks. Many techniques have already been applied to robustify various deep learning-based models against such attacks. This is especially the case for imperceptible, gradient-based optimization attacks. One of the most common mitigation strategies is adversarial retraining, where adversarial examples are used as part of the training process of a deep model. However, this method is weakened by the fact that the adversarial examples generated by a malicious attacker will in all likelihood differ from those used in the training process.

Another mitigation approach that has been tested is various denoising methods, the purpose of which is to remove the dangerous part of the adversarial examples, while maintaining classification accuracy. In this work, we used a normally distributed noise pattern as a defense, but there are other works with more complex defenses. Denoising defenses, that can be used in an architecture-independent fashion, is therefore an interesting research direction.

Another approach, which could be more effective against the attack presented in this work, is developing a method to detect if the image has been modified by an attacker. In essence, this is a classification task with 2 classes, so perhaps typical image classification methods could be applied to this task. When an anomaly is detected, the outputs from the object detector will at least no longer be considered trustworthy.

6.4 Implications for industry

There are some takeways to be made from this work with regards to industry as well. We have demonstrated the potential of defense-resistant real-time attacks using GANs. This represents a significant threat against industry object detection applications. However, one significant drawback of the attacks proposed in this work is that they are more perceptible than other works in the state-of-the-art. Therefore, while it might be more difficult to develop object detection models that are robust against our attack, they are easier to automatically detect. If the attack can be detected, the object detector can fall back to some secondary system, or flag its outputs as not to be trusted. Either way, the damage of the attack would be mitigated until the compromised

system is secured. Therefore, to protect object detection systems against adversarial attacks in general, we propose that the system needs to both be robust against smaller, less perceptible attacks, and also be able to detect slightly more perceptible attacks such as GAN_4. However, detecting our attacks is not trivial, as avoiding false positives would be crucial.

Another interesting aspect of our attacks is the transferability across datasets. We see that GAN_4 achieves significant performance degradation on the COCO validation dataset, even though it was trained on a VOC dataset split. In industry applications, this can also represent a significant threat, because an attacker does not need knowledge of the class labels the target detector uses to generate effective attacks. We also note that GAN_2 is even able to successfully fabricate objects with class labels that are not in the VOC dataset. Therefore, both GAN_4 and GAN_2 have significant implications for object detection security in industry.

Chapter 7

Conclusion and future work

In this chapter, we conclude our thesis and summarize our contributions. Then, we provide a discussion on where future work in the field of object detection security should go, based on the results we achieved in this thesis.

7.1 Conclusion

In this work, we have proposed a novel method to generate adversarial attacks using generative adversarial networks. The method involves suppressing the detection of objects inside the prior boxes of the SSD300 object detector. By using this technique, we proposed four different attack configurations, GAN_{1-4}. The GAN_2 configuration resulted in a powerful black-box object fabrication attack with relatively low perceptibility. GAN_4 resulted in a black-box attack that caused a significant mAP degradation across multiple black-box detectors. We also showed that by increasing the threshold of allowed perturbation magnitude, we caused an even more significant mAP degradation with GAN_4^{90} GAN_4^{110}. The resulting attacks achieve state-of-the-art performance in the black-box, queryless, single pass, defended setting. The attack works on both different datasets and target detectors. Additionally, we have evaluated our attack on detectors that use a basic denoising defense, which has been shown to neuter imperceptible, gradient-based attacks, and showed that the attacks maintain their performance.

7.2 Future work

In future work, the focus will be on reducing the perceptibility of the proposed attacks, while maintaining, or even improving, the attacks' efficacy. One interesting approach could be to train the GAN to imitate certain real-world conditions, such as rain or dust, that could occur naturally in an outside environment. In this way, an adversarial attack could be disguised as

a benign feature of the image. We see in the qualitative examples of GAN_4 that this could be a possible approach to generating defense-resistant and imperceptible adversarial attacks. Additionally, future works should propose defenses against this attack, which will likely involve detecting the presence of the attack rather than making models robust to it.

Bibliography

[1] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool. Generative adversarial networks for extreme learned image compression. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 221–231, 2019.

[2] K. BA and S. Charters. Guidelines for performing systematic literature reviews in software engineering. 2, 01 2007.

[3] J. Bao. Sparse adversarial attack to object detection, 2020.

[4] S. Chen, F. He, X. Huang, and K. Zhang. Relevance attack on detectors, 2021.

[5] Z. Chen and X. Huang. Accurate and reliable detection of traffic lights using multiclass learning and multiobject tracking. *IEEE Intelligent Transportation Systems Magazine*, 8(4):28–42, 2016.

[6] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. 11 2017.

[7] K.-H. Chow, L. Liu, M. Loper, J. Bae, M. E. Gursoy, S. Truex, W. Wei, and Y. Wu. Adversarial objectness gradient attacks in real-time object detection systems. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 263–272, 2020.

[8] X. Dai, Y. Chen, B. Xiao, D. Chen, M. Liu, L. Yuan, and L. Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7373–7382, June 2021.

[9] F. Ebadi and M. Norouzi. Road terrain detection and classification algorithm based on the color feature extraction. In *2017 Artificial Intelligence and Robotics (IRANOPEN)*, pages 139–146, 2017.

[10] B. M. Elbagoury, A.-B. M. Salem, and L. Vladareanu. Intelligent adaptive precrash control for autonmous vehicle agents (cbr engine hybrid a path planner). In *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 429–436, 2016.

[11] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[12] R. B. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[13] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[15] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.

[16] S. Hamdi, H. Faiedh, C. Souani, and K. Besbes. Road signs classification by ann for real-time implementation. In *2017 International Conference on Control, Automation and Diagnosis (ICCAD)*, pages 328–332, 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[18] X. He, Z. He, X. Du, and T.-S. Chua. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research amp; Development in Information Retrieval*, SIGIR '18, page 355–364, New York, NY, USA, 2018. Association for Computing Machinery.

[19] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks, 2016.

[20] S. Hoory, T. Shapira, A. Shabtai, and Y. Elovici. Dynamic adversarial patch for evading object detection models, 2020.

[21] S. Hu, Y. Zhang, S. Laha, A. Sharma, and H. Foroosh. Cca: Exploring the possibility of contextual camouflage attack on object detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7647–7654, 2021.

[22] Y. Huang, A. W.-K. Kong, and K.-Y. Lam. Attacking object detectors without changing the target object. In A. C. Nayak and A. Sharma, editors, *PRICAI 2019: Trends in Artificial Intelligence*, pages 3–15, Cham, 2019. Springer International Publishing.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[24] X. Kuang, X. Gao, L. Wang, G. Zhao, L. Ke, and Q. Zhang. A discrete cosine transform-based query efficient attack on black-box object detectors. *Information Sciences*, 546:596–607, 2021.

[25] M. Lee and Z. Kolter. On physical adversarial patches for object detection, 2019.

[26] C. Li. High quality, fast, modular reference implementation of SSD in PyTorch. `https://github.com/lufficc/SSD`, 2018.

[27] C. Li. Vizer. `https://github.com/lufficc/Vizer`, 2019.

[28] H. Li, G. Li, and Y. Yu. Rosa: Robust salient object detection against adversarial attacks. *IEEE Transactions on Cybernetics*, 50(11):4835–4847, 2020.

[29] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. CIKM '17, page 1419–1428, New York, NY, USA, 2017. Association for Computing Machinery.

[30] Q. Liao, X. Wang, B. Kong, S. Lyu, Y. Yin, Q. Song, and X. Wu. Category-wise attack: Transferable adversarial examples for anchor free object detection, 2020.

[31] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 42(02):318–327, feb 2020.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[33] Z. Lin, Y. Shi, and Z. Xue. Idsgan: Generative adversarial networks for attack generation against intrusion detection, 2021.

[34] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015.

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[36] D. Mahapatra, B. Bozorgtabar, S. Hewavitharanage, and R. Garnavi. Image super resolution using generative adversarial networks and local saliency maps for retinal image analysis. In M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins, and S. Duchesne, editors, *Medical Image Computing and Computer Assisted Intervention MICCAI 2017*, pages 382–390, Cham, 2017. Springer International Publishing.

[37] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, 2017.

[38] T. Meyer. Implementation of Single-pass and Transferrable GAN-Based Black-box Attacks on Object Detectors. `https://github.com/tormey97/MasterProject`, 2021.

[39] H. A. Najada and I. Mahgoub. Autonomous vehicles safe-optimal trajectory selection based on big data analysis and predefined user preferences. In *2016 IEEE 7th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 1–6, 2016.

[40] O. M. Nezami, A. Chaturvedi, M. Dras, and U. Garain. Pick-object-attack: Type-specific adversarial attack for object detection, 2020.

[41] M. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[43] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[44] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[45] M. Själander, M. Jahre, G. Tufte, and N. Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

[46] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, and T.-S. Chua. Adversarial training towards robust multimedia recommender system. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):855–867, 2020.

[47] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017.

[48] Y. Wang, Y. an Tan, W. Zhang, Y. Zhao, and X. Kuang. An adversarial attack on dnn-based black-box object detectors. *Journal of Network and Computer Applications*, 161:102634, 2020.

[49] Y. Wang, K. Wang, Z. Zhu, and F.-Y. Wang. Adversarial attacks on faster r-cnn object detector. *Neurocomputing*, 382:87–95, 2020.

[50] X. Wei, S. Liang, N. Chen, and X. Cao. Transferable adversarial attacks for image and video object detection. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 954–960. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[51] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu. Personal recommendation using deep recurrent neural networks in netease. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1218–1229, 2016.

[52] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[53] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. *ArXiv*, abs/1801.02610, 2018.

[54] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1378–1387, 2017.

[55] C. Yang, A. Kortylewski, C. Xie, Y. Cao, and A. Yuille. Patchattack: A black-box texture-based attack with reinforcement learning, 2020.

[56] H. Zhang, W. Zhou, and H. Li. Contextual adversarial attacks for object detection. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2020.

[57] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu. Poisoning attack in federated learning using generative adversarial nets. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 374–380, 2019.

[58] Q. Zhang, Y. Zhao, Y. Wang, T. Baker, J. Zhang, and J. Hu. Towards cross-task universal perturbation against black-box object detectors in autonomous driving. *Computer Networks*, 180:107388, 2020.