Measuring the Security of Computer Vision Systems to Adversarial Attacks



Giulio Lovisotto

Merton College

University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2022

Acknowledgements

Personal

I would like to begin by thanking my parents for the support they gave me throughout my life and especially during my academic pursuit including this doctorate. You gave me the freedom to follow my instincts and contributed to make me who I am today. I thank Susanna for being with me during this journey, for being supportive of me when I needed and for the good times we have had together. I feel very lucky to have you in my life and I'm looking forward to the future together. Thank you to all of my friends in Oxford, most of whom I also had the chance to play volleyball with. You made this time more enjoyable and I am very grateful to have made such good friendships even now that we have been in different places.

Thanks to all my collaborators and colleagues which managed to make research fun. I also thank Jan Hendrik Metzen and Nicole Finnie for their help and guidance during my internship in BCAI, it was great to work with both of you. Thank to my assessors Nicholas Lane, Kasper Rasmussen, Andrea Vedaldi and Battista Biggio. Over the course of my doctorate, you helped shape my research into this thesis. Finally, I would like to thank my supervisor Ivan, who supported and motivated me for many years me and guided me through the pursuit of my ideas. It is his guidance that made me the researcher I am today.

Institutional

I thank the EPSRC, the Department of Computer Science and DeepMind, Merton College, Mastercard and armasuisse, which have all provided the financial support that allowed me to pursue this work.

Abstract

Thanks to the increase of on-chip computing capability and recent advances in deep neural networks, autonomous systems increasingly rely on statistical learning and on data collected through digital cameras to automate and fullfil tasks. In these systems, the integrity of captured images and the ability of underlying models to correctly interpret image are cornerstones of correct behaviour. Nevertheless, there is still no widespread agreement on what the adversarial evaluation of these systems – the performance of a system with an adversary in-the-loop – should entail.

In this thesis, we take a look at the security of these automated vision systems using on machine learning when an adversary is able to corrupt the integrity of the data the system relies on. While lots of research focused on well-known bounded threat models – where camera-captured images are imperceptibility perturbed according to mathematical constraints – we choose to investigate the security of the systems in view of a realizability constraint, where attacks need to be reproducible in the real-world. By doing so, we are able to identify harms that go beyond model misclassifications which have a more impactful real-world counterpart. We highlight four potential attacks to computer vision systems, these either leverage the adversarial examples phenomenon in neural networks or exploit the low-level dynamics of how scenes are captured by digital cameras.

We hope that in the future, the research in this thesis will help develop evaluation standards for computer vision systems based on cameras. While it is often unlikely that mathematically-strong security guarantees can be provided, we hope that the weaknesses here identified will allow to design comprehensive test-cases for vision models which can provide heuristic guarantees on the correctness of the systems behaviour.

Contents

List of Figures **xi**

List of Abbreviations **xv**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Threat Model . . . 4
1.3 Contribution . . . 4
1.4 Ethical Considerations . . . 7

2 Background **9**
2.1 Introduction . . . 9
2.2 Biometric Recognition . . . 10
2.3 Deep Neural Networks for Computer Vision . . . 11
2.3.1 Image Classification . . . 13
2.3.2 Object Detection . . . 13
2.4 Adversarial Attacks on the Scene . . . 14
2.5 Physical Attacks at the Camera . . . 16
2.6 Adversarial Attacks Evaluation . . . 17

3 PPG Biometrics Using Smartphone Cameras **19**
3.1 Introduction . . . 20
3.2 Background . . . 21
3.2.1 Physiological foundations . . . 21
3.2.2 Measuring PPG . . . 22
3.2.3 PPG for Authentication . . . 23
3.3 Experiment Design . . . 24
3.3.1 System Requirements . . . 25
3.3.2 Collection Apparatus . . . 25
3.3.3 Collection Procedure . . . 26
3.3.4 Research Ethics . . . 26
3.4 Method . . . 27
3.4.1 Signal Extraction . . . 27

3.4.2 Signal Preprocessing . . . 27
3.4.3 Beat Separation . . . 28
3.4.4 Fiducial Points Detection . . . 28
3.4.5 Beat Signal Quality . . . 29
3.4.6 Feature Extraction . . . 30
3.4.7 Feature Selection . . . 31
3.5 Results . . . 32
3.5.1 Multi-class Case . . . 34
3.5.2 One-class Case . . . 34
3.5.3 One-class Cross-Session . . . 36
3.5.4 EER User Distribution . . . 37
3.6 Discussion . . . 37
3.7 Summary . . . 38

4 Poisoning Attacks against Unsupervised Template Updating 41
4.1 Introduction . . . 42
4.2 Background . . . 44
4.2.1 Template Update . . . 44
4.2.2 Poisoning Attacks . . . 44
4.2.3 Differences from Previous Work . . . 46
4.3 System and Threat Model . . . 48
4.3.1 System Model . . . 48
4.3.2 Adversary Model . . . 49
4.4 Attack Concept . . . 51
4.4.1 Overview . . . 51
4.4.2 Challenges . . . 52
4.5 Attack Method . . . 53
4.5.1 Poisoning Sample Generation . . . 53
4.5.2 Poisoning Attack Injection . . . 56
4.6 Evaluation . . . 58
4.6.1 Experiment Design . . . 58
4.6.2 Attack Success and Effect on Error Rates . . . 61
4.6.3 Transferability . . . 64
4.7 Poisoning Countermeasures . . . 65
4.7.1 Detecting Adversarial Samples . . . 65
4.7.2 Template Anomalies . . . 66
4.7.3 Angular Similarity Detection . . . 67
4.7.4 Detection Evaluation . . . 68
4.8 Discussion . . . 71
4.9 Summary . . . 72

5 Short-Lived Adversarial Perturbations for Physical Adversarial Examples 75
5.1 Introduction . . . 76
5.2 Background . . . 78
5.2.1 Projector technology . . . 78
5.2.2 Detection of Physical Adversarial Examples . . . 79
5.3 Threat Model . . . 80
5.4 Method . . . 81
5.4.1 Modelling projectable perturbations . . . 81
5.4.2 AE Generation . . . 85
5.4.3 Training Data Augmentation . . . 87
5.4.4 Remarks . . . 89
5.5 Evaluation . . . 90
5.5.1 Experimental Setup . . . 90
5.5.2 Indoor Results . . . 94
5.5.3 Road Driving Test . . . 96
5.5.4 Defences . . . 97
5.5.5 Attack Transferability . . . 99
5.6 Discussion . . . 101
5.7 Summary . . . 103

6 Vulnerabilities of Cameras using Rolling Shutters 105
6.1 Introduction . . . 106
6.2 Image Sensing Background . . . 108
6.2.1 Photodiode Arrays . . . 109
6.2.2 Scanning Element . . . 109
6.3 Threat Model . . . 111
6.4 Exploiting the Rolling Shutter . . . 113
6.4.1 Controlling the Distortion Size . . . 114
6.4.2 Estimating Parameters . . . 116
6.5 Rolling Shutter Model Evaluation . . . 118
6.5.1 Experimental Setup . . . 118
6.5.2 Accuracy of the Shutter Model . . . 121
6.6 Attack on Object Detection . . . 122
6.6.1 Evaluation on Video Datasets . . . 124
6.6.2 Comparison with Blinding Attack . . . 129
6.7 Countermeasures . . . 131
6.7.1 Hardware Countermeasures . . . 131
6.7.2 Sensor Redundancy . . . 131

6.7.3 Deep Learning Based Detection . . . 132
6.8 Limitations . . . 134
6.8.1 Evaluation Limitations . . . 134
6.8.2 Attack Limitations . . . 135
6.9 Summary . . . 136

7 Dot-Product Attention Vulnerability for Adversarial Patch Robustness 137
7.1 Introduction . . . 137
7.2 Robustness of Vision Transformers . . . 140
7.3 Preliminaries and Threat Model . . . 142
7.4 Robustness of Dot-Product Attention . . . 144
7.4.1 Gradient of Dot-Product Self-Attention . . . 144
7.4.2 Robustness of Dot-Product Attention Weights . . . 146
7.5 Attention-Fool . . . 147
7.6 Evaluation of Attention-Fool . . . 150
7.6.1 Evaluation Setup . . . 150
7.6.2 Attention-Fool on Vision Transformer . . . 150
7.6.3 Attention-Fool on DETR . . . 153
7.7 Limitations . . . 155
7.8 Summary . . . 156

8 Conclusions 157
8.1 Summary of Results . . . 157
8.2 Final Conclusions . . . 158

Appendices

A Supplementary Material for Chapter 5 161
A.1 Attack on Different Objects . . . 161
A.2 Additional Results . . . 162
A.3 SentiNet Description . . . 163

B Supplementary Material for Chapter 6 167
B.1 Experimental Setup and Tested Cameras . . . 167
B.2 Effect on Autonomous Driving . . . 167

C Supplementary Material for Chapter 7 **173**
C.1 Effect of Input Mean on Attention Weight Robustness 173
C.2 Ablation on Attention-Fool on ViT 175
C.2.1 Normalization 175
C.2.2 Patch Location 175
C.3 Ablation on Targeted Attacks on ViT 176
C.4 Adversarial Token Replacement 177
C.5 Additional Visualizations 178

Bibliography **181**

List of Figures

1.1 Realizability Scale. 3
1.2 General threat model considered in this thesis. 4

2.1 Various tasks performed by deep neural networks based on cameras. 12
2.2 Visualization of an adversarial example for a deep neural network. . 15

3.1 PPG authentication system overview 20
3.2 Modes of PPG collection 22
3.3 Signal over the course of the preprocessing steps 28
3.4 Visualization of fiducial point based features for a single beat 30
3.5 EERs in the multi-class case. 34
3.6 EERs in the one-class case. 35
3.7 EERs in the one-class cross-session case. 36
3.8 Distribution of EERs across users in our dataset. 37

4.1 Biometric authentication system and threat model. 48
4.2 Poisoning attack overview. 52
4.3 Poisoning samples for an adversary-victim pair. 61
4.4 IAR, FAR and FRR changes over the course of the poisoning attack. 62
4.5 Detection of poisoning attack with angular similarity. 68
4.6 Poisoning attack success rate with poisoning detection in place. . . 70

5.1 Visualization of an attack using short lived adversarial perturbations. 76
5.2 Depiction of an LCD projector inner workings. 79
5.3 Example of an adversarial patch attack [80] 80
5.4 SLAP attack scenario 81
5.5 Camera light sensor noise visualized. 83
5.6 Plot showing the output space of the learned projection model. . . . 84
5.7 Pixel changes when a certain projection is shown. 85
5.8 Overview of the adversarial samples generation pipeline. 87
5.9 Examples of the projected images computed with the optimization. 92
5.10 Baseline mis-detection rate in absence of the adversary. 93
5.11 SLAP attack success rate. 95

5.12 Detection probability for the stop sign during the road driving test. 96
5.13 Visualization of the SentiNet detection results from the 180 lux setting. 98
5.14 Amount of lux achievable on an object's surface. 101

6.1 Computer vision use-cases and impact of rolling shutter attack. . . 107
6.2 Frame acquisition comparison between global and rolling shutter. . 110
6.3 Illustration of the row-wise acquisition of a single frame. 113
6.4 Overview of the parameters that influence the size of the injected rolling shutter distortion. 114
6.5 Distortion size accuracy using the rolling shutter model. 120
6.6 Increment in the expected rolling shutter distortion size with wrong estimates. 122
6.7 Increment in expected distortion size under incorrect exposure time estimate, worst cases. 123
6.8 Rolling shutter attack evaluation on object detection. 124
6.9 Attack success with different frequencies. 126
6.10 Attack success with different exposures and laser on time. 127
6.11 Attack success on objects of different sizes. 128
6.12 Image perturbation metrics for the selected frames in the BDD100K driving video dataset. 129
6.13 Examples of misclassified frames by the rolling shutter attack detection head. 133

7.1 Comparison of clean and adversarially patched input for DETR [35]. 139
7.2 Example of dot-product (self-)attention mechanism for clean (left) and adversarial patch attack (right) settings. 140
7.3 Minimum ℓ_∞ norm perturbation ϵ required for reaching the attacker's goal 145
7.4 Embedded projected key and query tokens for clean and patched input image. 148

A.1 Sample video frame extracted from the outdoor experiment. 162
A.2 SLAP attack on class "Give Way" for Gtsrb-CNN and Lisa-CNN. . 164
A.3 SLAP attack on class "Bottle" for Yolov3 and Mask-RCNN. 166

B.1 Experimental setup used to collect rolling shutter attack patterns. . 168
B.2 Physical evaluation of the rolling shutter attack. 170
B.3 Velocity curve of the agent during normal operation (baseline) and during the rolling shutter attack. 171

C.1 Silhouette score between clusters of projected keys and queries. . . . 174

C.2 Embedded projected key and query tokens for clean and patched input images on each of the 12 DeiT-B layers. 179
C.3 More comparisons of clean and adversarially patched input for DETR. 180

List of Abbreviations

(MS) COCO . . Microsoft “Common Objects in COntext” dataset [112].
1-D, 2-D, 3-D . One-, two- or three-dimensional.
ADAS . . . . . Advanced driver assistance systems.
AE . . . . . Adversarial example.
API . . . . . Application programming interface.
AUC . . . . . Area under the curve.
BDD100K . . . “Berkley Diverse Driving” dataset [236].
CCD . . . . . Charge-coupled device.
CMOS . . . . . Complementary metal–oxide–semiconductor.
CNN . . . . . Convolutional neural network.
CSRT . . . . . Channel and spatial reliability tracker [121].
D.Phil. . . . . . Doctorate of philosophy.
DeiT . . . . . “Data-efficient vision Transformer” network architecture [67].
DETR . . . . . “DEtection TRansformer” object detector [35].
DNN . . . . . Deep neural network.
ECG . . . . . Electrocardiogram.
EER . . . . . Equal error rate.
EOT . . . . . Expectation over transformation.
F(aster-)RCNN Object detection architecture [163].
FaceNet . . . . . Face recognition network architecture [175].
FAR . . . . . False accept rate.
FGSM . . . . . Fast gradient sign method.
FN . . . . . False negative.
FP . . . . . False positive.
FPS . . . . . Frames per second.
FRR . . . . . False reject rate.
FTA . . . . . Failure-to-acquire.
GBT . . . . . Gradient boosted trees.
GPS . . . . . Global positioning system.
GPU . . . . . Graphics processing unit.

Gtsrb-CNN . . Traffic sign recognition network architecture [67].
GTSRB . . . . . "German Traffic Sign Recognition Benchmark" dataset [88].
HD . . . . . High definition.
IAR . . . . . Impostor acceptance rate.
IEEE . . . . . Institute of electrical and electronics engineers.
IoT . . . . . Internet-of-things.
IoU . . . . . Intersection-over-union.
IsFrst . . . . . Isolation forest.
ISO . . . . . When referred to a camera, indicates the camera light sensitivity.
LCD . . . . . Liquid crystal display.
LED . . . . . Light-emitting diode.
LFW . . . . . "Labeled Faces in the Wild" dataset [89].
LiDAR . . . . . Light detection and ranging.
Lisa-CNN . . . . Traffic sign recognition network architecture [67].
LISA . . . . . Traffic signs dataset introduced in [134].
mAP . . . . . Mean average precision.
Mask-RCNN . . Object detection architecture [84].
mRMR . . . . . Minimum redundancy maximum relevance.
MS-SSIM . . . . Multi-scale structural similarity index measure.
NPS . . . . . Non-printability score.
OSVM . . . . . One-class support vector machine.
PCA . . . . . Principal component analysis.
PGD . . . . . Projected gradient descent.
PIN . . . . . Personal identification number.
PPG . . . . . Photopletismography.
PWM . . . . . Pulse-width modulation.
R-CNN . . . . . "Region-based Convolutional Network" object detector [75].
RFCN . . . . . "Region-based Fully Convolutional Networks' object detector [50].
ReLu . . . . . Rectified linear unit.
ResNet . . . . . "Residual Network" network architecture [85].
RGB . . . . . Red-green-blue – the three channels composing color images.
RMI . . . . . Relative mutual information.
ROC . . . . . Receiver operating characteristic curve.
SLAP . . . . . Short-lived adversarial perturbations.
SSD . . . . . "Single-Shot Detector" object detector [114].
SSIM . . . . . Structural similarity index measure.

SVM . . . . . Support vector machine.
TTL . . . . . Transistor-transistor-logic.
UQI . . . . . Universal quality index.
VGG . . . . . "Visual Geometry Group" network architecture [151].
VIRAT . . . . . Video surveillance dataset from [236].
ViT . . . . . "Vision Transformer" network architecture [56].
YOLOv3 . . . . "You Only Look Once - version 3" object detector [162].

1 Introduction

Contents

1.1 Motivation 1
1.2 Threat Model 4
1.3 Contribution 4
1.4 Ethical Considerations 7

1.1 Motivation

In recent years, the use of digital cameras in intelligent systems has become more widespread, with uses spanning from traditional surveillance and monitoring, to more recent user recognition and computer vision. Thanks to the increase of computing capability, deep neural networks (DNN) have now become widely used in order to automatically process these visual data, with convolutional DNNs now setting state-of-the-art performances across a variety of automated tasks. Given the increasing integration of cameras into decision-making processes of intelligent agents, both the integrity of captured images and the ability of underlying models to correctly interpret information have become cornerstones of the system correct behaviour. Nevertheless, most camera-based visual systems are only evaluated in benign settings, often using standardized benchmarks and shared simplified assumptions. Adversarial

evaluation, i.e., evaluating the performance of a system when an adversary is in-the-loop, is oftentimes overlooked as researchers tend to dedicate more efforts to improving (non-adversarial) benchmark performances. Defining what an adversarial evaluation should entail is a complex task, which requires to finely define assumptions around the threat model: the attacker's goal, capabilities and knowledge [25]. Oftentimes, when defining these assumptions, researchers constrain themselves to particular attacks, but leave room for different attack vectors, which fall under distinct assumptions. In fact, in standard evaluation of the resilience of DNNs to adversaries, while generic threat models have been defined [36] and defences are developed accordingly, it has repeatedly being shown that adaptive adversaries (i.e., adversaries aware of the defence) can bypass these defences [38, 37, 203].

Rationale. In this thesis, rather than focusing on the above mentioned generic threat models, I explore novel attack vectors under the constraint of *realizability*. For realizability, I broadly refer to the number of assumptions that have to be relaxed to move from a digital attack to a physical realization. Note that characterizing this definition is a rather complex task which does not have widespread consensus: properties such as attackers' capabilities, knowledge and goal can be factored into a quantification of realizability with different weights [25]. More specifically, this thesis will prioritize attacks bounded by two properties: (i) attackers having white-box knowledge and (ii) the attacks' *physical* realization. The reasoning behind considering white-box knowledge is known in the security field as Kerchoff's principle [155], which values the open design of security mechanisms as a fundamental component for secure systems. In practice, it has also been shown that there is inherent transferability of adversarial attacks between different models [51, 226, 147, 228, 204], which leads to think that assuming black-box adversary knowledge might be optimistic in many scenarios. On the other hand, the thesis puts emphasis on the physical realization of attacks – the ability to re-create attacks in the real-world, and ideally on deployed systems. While most of the research in adversarial machine learning has focused on generic ℓ_p-norm bounded threat models [36] – where malicious input are constrained by a mathematically-defined imperceptibility

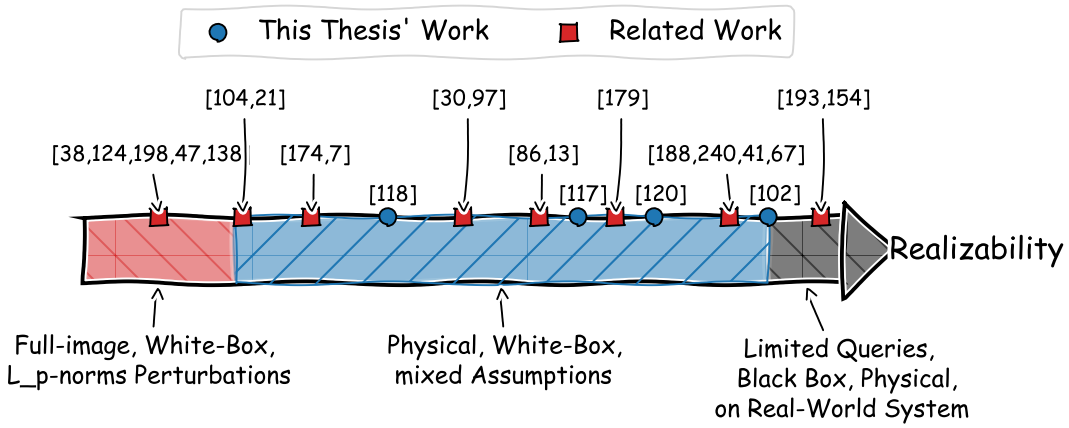


Figure 1.1: How a set of related works on attacks on computer vision systems relates to a "Realizability" dimension which is used as guideline in this thesis. Note that this is my own interpretation of this scale, which I am adding here to explain the reasoning behind the research in this thesis. Different opinions on what constitutes a realizable attack may disagree with this scale. Chapter 4 is based on [117], Chapter 5 on [120], Chapter 6 on [102] and Chapter 7 on [118]. The style of this figure is borrowed from `xkcd` [135].

property – in setting the realizability constraint in this thesis, we can identify more concrete harms that go beyond misclassifications, and have stronger real-world relevance. Figure 1.1 reports a brief overview of work that is either well cited or closely related to the content of this thesis, and of how it fares on a realizability scale which I used to guide my research.

Research Questions. This work is motivated by the following research questions:

- **RQ1.** Are there unknown attack vectors which can lead to realizable attacks which may hamper the security of camera-based systems?

- **RQ2.** Given an unseen attack vector, can we propose countermeasures or alternative modality of use which do not compromise systems performance but increase their resilience to threats?

Each of the research chapters in this thesis will investigate specific scenarios where an intelligent system uses camera-captured information and where the data's may be corrupted by an adversary.

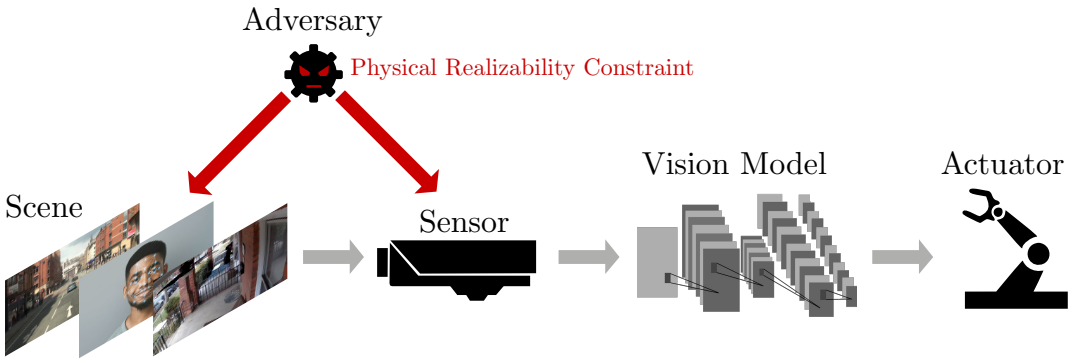


Figure 1.2: General threat model considered in this thesis. An adversary wants to compromise a system which operates based on camera-captured inputs (with the help of a vision model), but the adversary is limited by a physical realizability constraint, so they can either target the scene content or the sensor.

1.2 Threat Model

All the research in this thesis can be unified under a single threat model, shown in Figure 1.2. I will consider adversaries that aim to compromise the operation of a system that functions based on a vision model's output, such system uses camera-perceived images as inputs. The system's task can generally be abstracted from the modelling of the adversary (e.g., it is irrelevant whether the system does face recognition, object detection or home surveillance, as depicted in Figure 1.2). I focus on adversaries in view of a realizability constraint discussed above. In order to compromise the system operation, these adversaries, will either target a certain scene (which is captured by the camera) or will target the camera perception itself. Each chapter which considers a specific attack will detail adversaries' goals, capabilities and knowledge.

1.3 Contribution

In this thesis, each chapter is based on a published or under submission research paper, with the contributions as follows:

- In Chapter 3, we investigate photopletismography as an alternative for face recognition in order to perform biometric authentication. Both photopletismography and face recognition can make use of a camera to capture one's biometric

trait, but photopletismography has the advantage of being an unobservable trait in comparison. These findings have been published in a workshop paper at the Computer Vision and Pattern Recognition 2020 conference [119].

- In Chapter 4, we identify an open attack vector for biometric authentication systems which make use of template updates, i.e., where newly collected user samples are added to the stored biometric reference. We model the way an attacker would exploit this weakness in practice, and we introduce a countermeasure which can stop the attack from happening by preventing malicious samples from entering the biometric template. The results of this study have been published at the IEEE European Symposium on Security and Privacy 2020 [117].

- Chapter 5 outlines a previously unseen attack vector for the realization of adversarial examples in the real-world: by using a projector to shine specifically crafted perturbations on objects which fool deep neural networks into misclassifications. We perform an extensive evaluation of the attack and possible defences and discuss its limitations in practice, while highlighting how such attack vector might be preferred over the commonly known adversarial patches attack. The study underlying this chapter has been published at the USENIX Security Symposium 2021 [120].

- Chapter 6 explores another attack against camera-based systems, highlighting how the rolling shutter used by common image sensors can be exploited in non-trivial ways when the captured frames are used by vision models. We propose a detection mechanism for such attack re-using information from pre-deployed vision models, the findings and results of this paper have been published at the Annual Computer Security Applications Conference 2021 [102].

- Chapter 7 investigates the robustness to adversarial patch attacks on novel Vision Transformer architectures, highlighting how the scaled dot-product attention mechanism introduces layers which harm standard robustness

evaluation techniques due to the inner workings of the self-attention blocks. We perform an extensive evaluation of this vulnerability on both image classification and object detection tasks, finding that one can address this shortfall by adding specific terms to the adversarial patch generation, which lead to tighter robustness estimates. The study underlying this chapter has been published in the Computer Vision and Pattern Recognition conference 2022 [118].

Code and Data Availability. In view of the importance of open sourcing code and datasets to the scientific community, I dedicated efforts to making the code used in this thesis available online: Chapter 3,[1] (including an anonymized dataset of photopletismogram measurements) Chapter 4,[2] Chapter 5,[3] (approved by the USENIX Security Artifacts Review process) and Chapter 6.[4] (approved by the Annual Computer Security Applications Conference Artifacts Review process). I hope that sharing these resources with the community will allow others to build their research upon mine.

Other Contributions. During my doctorate, I had the chance to collaborate to a series of other works which are not used in this thesis, but that constitute part of my research contribution. As second author, I worked on the security and privacy of voice authentication in [207] and [208][5]; these are published in the European Symposium on Research in Computer Security 2019 and in the Computer Speech & Language journal in 2022. I also contributed to research in biometric authentication as second author [59, 60] and as fourth in [73]; these are published in the IEEE Symposium on Security and Privacy 2018, the ACM SIGSAC Conference on Computer and Communications Security 2019 and the Asia Conference on Computer and Communications Security 2022, respectively.

[1] `https://github.com/ssloxford/seeing-red/`
[2] `https://github.com/ssloxford/biometric-backdoors/`
[3] `https://github.com/ssloxford/short-lived-adversarial-perturbations/`
[4] `https://github.com/ssloxford/they-see-me-rollin/`
[5] implementation at `https://github.com/ssloxford/DistributionPreservingXVectors`

1.4 Ethical Considerations

All the research in this thesis was carried out according to the University of Oxford ethical guidelines, clearance was obtained when required, details are in each chapter. Adversarial attack such as those highlighted in this thesis can be used for benign purposes, such as reliably evaluating the safety and robustness of machine learning systems to adversarial inputs. Nevertheless, malicious usages are also a possibility, where an adversary may use these attacks (or build on them) to exploit vulnerabilities in real-world systems. We are confident that this research provides valuable guidelines for the evaluation of these systems, which will lead to safer machine learning system by mitigating the risks here highlighted and the resulting potential for negative social impact.

2
Background

Contents

2.1 Introduction 9
2.2 Biometric Recognition 10
2.3 Deep Neural Networks for Computer Vision 11
2.3.1 Image Classification 13
2.3.2 Object Detection 13
2.4 Adversarial Attacks on the Scene 14
2.5 Physical Attacks at the Camera 16
2.6 Adversarial Attacks Evaluation 17

2.1 Introduction

Even though all research in this thesis shares the common trait of analysing security of systems based on camera inputs, the thesis touches a variety of topics. In this chapter, I give an overview of the relevant background and related works. Each following chapter will contain additional related work which better contextualizes their research content. In the following, I first describe the topics of biometric recognition and automated vision in with deep neural network, specifically object detection; these two of the main camera-based computer vision tasks whose security is evaluated in this thesis (in Chapter 4, 5, 6 and 7). Later, I give an overview

of existing attacks which fall under the threat model described in the previous section: adversarial machine learning – which roughly envelops attacks on the scene (see Figure 1.2) – and physical attacks at the camera.

2.2 Biometric Recognition

Both the research community and industry have shown significant interest in biometric recognition in recent years. Often times, recognition of biometric traits is proposed as an approach to mitigate the shortcomings of passwords (such as bad memorability, password reuse and easy to guess passwords [28]). Biometric systems work in two phases: an initial *enrolment* phase, few samples from users are collected and stored into a biometric template or reference. In the recognition phases, when users want to authenticate, they provide their claimed identity and allow the system to collect their biometric samples, which are compared with the stored template associated with the identity; when the samples match the template, users authenticate successfully. This concept of biometric template is important to contextualize Chapter 4, which explores how the template's security can be compromised. In the following we will give a brief overview of face recognition systems, as these are investigated in Chapter 4, and of other biometric traits, which give context for the new camera-based authentication system proposed in Chapter 3.

Modern face recognition systems are based on deep convolutional neural networks, where the input image undergoes transformation over several convolutional layers (e.g., FaceNet [175], VGG-Face [151], ResNet [32]). In recent years, state-of-the-art models have shown to outperform humans in recognition accuracy. Most face recognition DNN can also be used as feature extractors, allowing them to work with faces unseen in the training data. In this case, another simpler classifier is added onto the recognition pipeline that uses the output of the neural network as its inputs.

While face recognition has a long history of being used as a biometric trait, many other more exotic traits have been proposed. Many of these new traits can also be used in continuous authentication scenarios, where samples are being collected continuously and users can be authenticated over time. Notable examples include

eye's movements [61], gait [142], mouse movements [164], touch dynamics [69] or electrocardiography [70], which I have also worked with during my D.Phil. in [59, 60]. Many of these traits have the benefit of being unobservable when compared to face recognition or other physiological traits: these traits cannot easily be captured by an adversary. In practice this means that when an adversary wants to impersonate a certain victim with the biometric system, the bar to obtain the victim's biometric trait is higher, making the attack harder (i.e., in comparison to face recognition, when a simple photo of the victim is sufficient [21]). Unfortunately many of these modalities require additional hardware in order to capture one's trait, which may not always be available in practice. Chapter 3 will investigate how we can re-use a simple camera (e.g., such as a smartphone one) in order to capture a photopletismogram, which can be used in order to perform authentication.

2.3 Deep Neural Networks for Computer Vision

Deep Neural Networks Advances. The improvements in computing power enabled the use of deep neural networks in the autonomous driving scenario, to perform automated vision. Since these advances, convolutional neural networks [106, 103] (CNN) have remained the predominant and best performing way to extract information from images [126]. CNNs are deep neural networks which make use of convolutions, where many layers composed of sets of learned convolutional filters which slide along the input features and respond to the image content, synthesising it into lower-dimensional spaces. More recently, Dosovitskiy et al. [56], inspired by the successes of Transformer-based architectures and the self-attention mechanism in Natural Language Processing [212], developed a new architecture known as Vision Transformer (ViT). In ViTs, images are split into subpatches and a Transformer encoder model is provided with a sequence of linear embeddings of these patches as an input. ViTs have been shown to outperform CNN-based models on a variety of benchmarks [56, 202, 116], while at the same time requiring less parameters overall – possibly leading the way for a paradigm shift in the architectures used in computer vision. ViTs and Transformer-based architectures will be the focus of Chapter 7.



(a) Object detection. **(b)** Instance segmentation. **(c)** Semantic segmentation.

Figure 2.1: Various tasks performed by deep neural networks based on camera-inputs in autonomous driving. Three use cases are reported: object detection in Figure 2.1a, instance segmentation in Figure 2.1b and semantic segmentation in Figure 2.1c. All images are taken from the BDD100k dataset [236].

Vision and Self-Driving Vehicles. Self-driving vehicles may rely on a variety of input sensors, but cameras are amongst the most important ones, as they provide visual information which is paramount for several driving tasks. Such information is so paramount, in fact, that Tesla self-driving cars only make use of RGB cameras for navigation [213]. Well known camera-based tasks in autonomous driving include automated lane detection [220, 156], object detection [162, 114, 163], semantic segmentation [237, 144, 216], examples are reported in Figure 2.1. In this thesis, I focused mainly on the object detection task, as at the time of research it was sufficiently established and with clear baselines and evaluation methodologies; object detection is used throughout Chapter 5, 6 and 7. In Chapter 5 and 7, in order to investigate fundamental properties of the attack proposed in the same chapters, I also consider the simpler task of image classification, i.e. determining which semantic class an image belongs to given its content; Chapter 5 specifically looks at this scenario where the images contain traffic signs. While the traffic sign or image classification task might not be in used directly in modern autonomous vehicles, it allowed me to build up and compare my own work to previous work which established common baselines (specifically, [88, 134, 67, 2, 42]). In the following I describe in more detail how object detection works and give a brief overview of the relevance of image classification in security settings.

2.3.1 Image Classification

Since the advent of ImageNet [103] in 2012, the task of classifying images into semantic classes based on the images' content has become increasingly popular. ImageNet has become the best known dataset for this task, as it contains a thousand classes and is often used both as baseline to test new methods (e.g., new network architectures) and also often used in transfer learning, where a network pre-trained on ImageNet classification is further re-purposed into a different task. It has been shown that this type of pre-training is beneficial to various applications, as the network learns general filters which can be more easily adapted to specific tasks [54, 181].

The task of traffic sign recognition is just an instance of a general multi-class classification scenario. Differently from object detection (described in the next section), in traffic sign recognition, networks typically require a cut-out of the sign as input, rather than the full scene. Several datasets of videos from car dash cameras are available online, such as LISA [134] or GTSRB [88], in which a region of interest that identifies the ground-truth position of the traffic sign in each video frame is generally manually annotated. In Chapter 5, for continuity, we consider two different models for traffic sign recognition, Lisa-CNN and Gtsrb-CNN, both introduced in [67], one of the earliest works in real-world robust adversarial examples.

2.3.2 Object Detection

Object detection refers to the task of identifying and locating instances of semantic objects in an image. The output of object detectors is generally a set of coordinates of bounding boxes in the input image that contain specific objects. While object detection is useful in a variety of use-cases, in this thesis we mostly focus on the autonomous driving scenario, where object detectors are a component of a self-driving agent which uses the detected objects information (both location and classes) to inform its behaviour. In the following we detail four object detectors, Yolov3 [162], SSD [114], Faster-RCNN [163] and DETR [35] which at the time of the research were generally accepted as well-known state-of-the-art baselines to evaluate performance; these are used in Chapter 5, 6 and 7.

Yolov3 and SSD are single-shot detectors which run inputs through a single convolutional neural network. The CNN integrates a backbone network that computes features maps for the input, these feature maps are used by the rest of the network to predict bounding boxes offsets from anchor boxes and bounding boxes classes. This is done differently in SSD and Yolov3, with the first using different aspect ratio with multiple boxes, while the second one having a fixed grid cell aspect ratio (with three different scales, 13x13, 26x26, 52x52). Yolov3 is used in many real-time processing systems [8, 26, 211].

Faster-RCNN is the result of a series of improvements on the initial region-based CNN (R-CNN) object detector network [75]. Faster-RCNN uses a two-stage detection method, where an initial network generates region proposals and a second network predicts labels for proposals (this is somewhat similar to using an attention mechanism [212]). More recently, Mask-RCNN [84] extended Faster-RCNN in order to effortlessly add object segmentation (i.e., computing fine-grained object masks) to object detection, which is achieved by adding an additional network head. Both Yolov3, SSD and Mask-RCNN use non-maximum suppression in post-processing to remove redundant boxes with high overlap.

DETR – short for DEtection TRansformer – is a novel approach to object detection, where input features extracted by a backbone CNN are fed through a Transformer (with both encoder and decoder) to be refined and in order to obtain box predictions [35]. DETR has the benefit of simplifying the detection pipeline by replacing the need for hand-designed components that encode prior knowledge, including spatial anchors and non-maximal suppression – both used in Yolov3, SSD and Faster-RCNN.

2.4 Adversarial Attacks on the Scene

In recent years, many publications have highlighted weaknesses of deep (convolutional) neural networks to adversarial examples (AE): samples which may seem legitimate, but actually have subtle changes that lead to potentially dangerous misclassifications [198]. An example of such adversarial sample is reported in

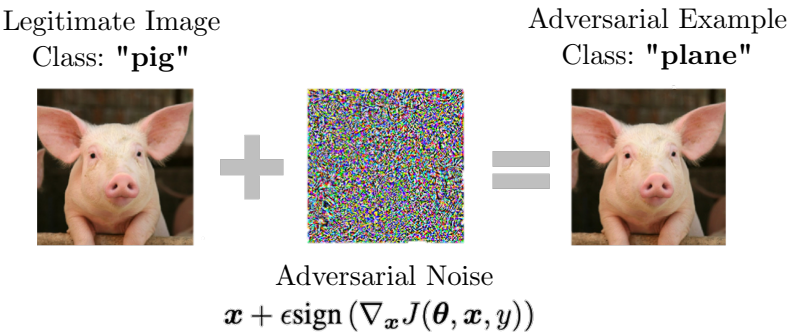


Figure 2.2: Visualization of an adversarial example for a deep neural network. An image of a pig is classified correctly in a legitimate case, but when a worst case inconspicuous adversarial noise is added to the image, the network outputs a wrong classification.

Figure 2.2. The adversarial perturbations can be small, often *imperceptible*, yet have a profound effect on the vision-based system. There as been large amount of work on defending against adversarial examples in a digital domain [149, 38, 124, 47] (i.e., as in Figure 2.2, where adversaries can manipulate individual pixels in the image perfectly). Recently, with the introduction of Vision Transformers, many have investigated the properties of these architectures against adversarial examples [138, 20, 177], without reaching a clear consensus on whether ViTs can be classified as more or less robust than classical CNNs (see Section 7.2 for further discussion). Nevertheless, the notion of imperceptibility of AE has been criticised as it does not often correspond to a practical attack constraint [178, 205]. On the other hand, it has also been shown that adversarial examples can be extended in the real-world.

Real-world adversarial examples started with Kurakin et al. [104] showing that perturbations computed with the fast gradient descent [198] method can survive printing and re-capture with a camera. However, these perturbations would not be realizable on a real (3D) input, therefore other works on physical attacks against neural networks have focused on adversarial patches [13, 30, 97, 86, 67]. Eykholt et al. [67] showed how to craft robust physical patch perturbations for stop signs, that survive changes when reproduced in the physical world (e.g., distance and viewing angle). The perturbation is in the form of a poster overlaid on the stop sign itself or a sticker patch that the authors apply to the sign. Sharif et al. [179]

showed that physical adversarial examples for face recognition can be realized by using coloured eye-glass frames, further strengthening the realizability of the perturbation in the presence of input noise (e.g., different user poses, limited colour gamut of printers). More recent works have focused on AE for object detection, using patches [188, 240, 41, 188], including the attack presented in Chapter 7 [118]. Another possibility is that of by projecting light [184, 129]; the attack presented in Chapter 5 falls under this latter category [120].

2.5 Physical Attacks at the Camera

It has been shown that attacks which alter the camera-perceived scene are possible without directly leveraging deep neural network sensitivity to adversarial examples by simply by exploiting physical phenomenons [150, 193, 185]. Petit et al. [154], highlighted how corrupting camera inputs by shining additional light with a laser, either in a blinding attack or in an attempt to trigger auto-exposure adjustments, can lead to reduced safety. [140] demonstrated that cameras recognize objects projected onto the road as genuine objects or obstacles leading to unintended manoeuvrers. Instead of projecting objects onto the road, [129] highlighted that it is possible to project objects directly onto the camera lens.

Recently, [108] demonstrated an attack that exploits the rolling shutter mechanism in complementary metal–oxide–semiconductor (CMOS) image sensors. The researchers showed that the rolling shutter can be abused to create a backdoor in the training dataset for face recognition systems. Meanwhile, work in [174] demonstrated that the rolling shutter can be exploited when the attacker controls the ambient lighting in an indoor environment. Our research in Chapter 6 happened at the same time as [108, 174], but we reduce the assumptions on the adversary's side and focus on the inner workings of physically exploiting a rolling shutter in dynamic environments [102].

2.6 Adversarial Attacks Evaluation

Since adversarial examples for DNN were initially highlighted, the field has moved towards standardizing what is commonly referred to as "adversarial robustness" evaluation, i.e., the definition of common practices and guidelines for the evaluation of the weakness to adversarial examples [36]. At the time of writing, many software toolboxes containing sets of adversarial attacks (i.e., methods to generate adversarial examples) have been created and are currently maintained by the community with the prompt addition of newer, more recent attacks. Examples include RobustBench [45], AutoAttack [47], Foolbox [161], the Adversarial Robustness Toolbox [143] and secML [132]. These toolboxes contain large set of common adversarial attacks typically defined in the digital domain and with classical norm-bounded constraints. While these tools can (and should) be used as part of a comprehensive adversarial attacks evaluation, in this thesis we will also focus on attacks that span over the digital domain, which are tied to diverse sets of assumptions which help bridge the digital versus physical world gap. These attacks could be integrated into these toolboxes in the future.

3

PPG Biometrics Using Smartphone Cameras

Contents

3.1 Introduction 20
3.2 Background 21
3.2.1 Physiological foundations 21
3.2.2 Measuring PPG 22
3.2.3 PPG for Authentication 23
3.3 Experiment Design 24
3.3.1 System Requirements 25
3.3.2 Collection Apparatus 25
3.3.3 Collection Procedure 26
3.3.4 Research Ethics 26
3.4 Method 27
3.4.1 Signal Extraction 27
3.4.2 Signal Preprocessing 27
3.4.3 Beat Separation 28
3.4.4 Fiducial Points Detection 28
3.4.5 Beat Signal Quality 29
3.4.6 Feature Extraction 30
3.4.7 Feature Selection 31
3.5 Results 32
3.5.1 Multi-class Case 34
3.5.2 One-class Case 34
3.5.3 One-class Cross-Session 36
3.5.4 EER User Distribution 37
3.6 Discussion 37
3.7 Summary 38

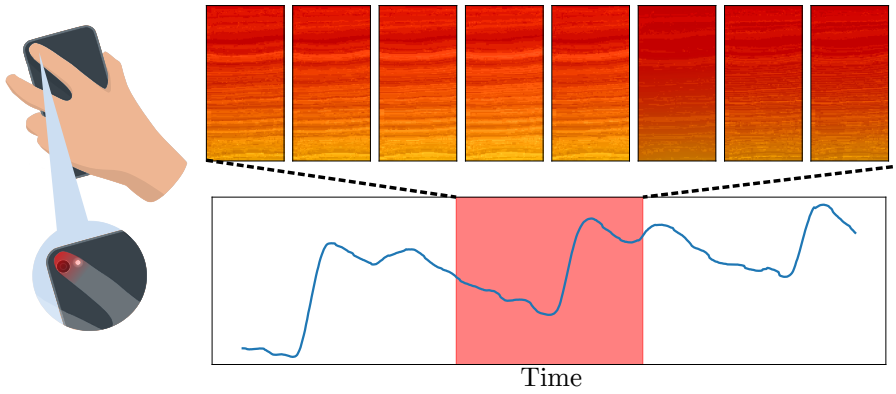


Figure 3.1: System overview. Users place their finger on the smartphone camera. The blood flow to the fingertip changes the light reflection properties of the skin creating subtle changes in the colour of the video frames (luma component).

3.1 Introduction

Biometric authentication is a popular approach to effortless user authentication, especially on mobile devices such as smartphones or tablets. While both fingerprint scanning and face recognition offer quick and accurate authentication, their modalities are also easy to observe for adversaries wanting to replicate a victim's trait. Once the secrecy of these traits is compromised, the door for attacks opens: adversaries can focus on replicating biometric samples with increasingly sophisticated forgeries in order to bypass the presentation attack detection measures in place. In recent years, photoplethysmography (PPG) has become particularly interesting in the context of mobile devices. PPG is a method to optically detect changes in blood flow volume and is often used in optical heart rate sensors of smartwatches. In the medical domain, PPG is used to monitor patients' heart rates and blood oxygen levels through pulse oximeters. Similarly to the electrocardiogram (ECG), PPG offers a range of distinctive biometric features derived from unique characteristics of an individual's cardiovascular system. Previous research in the field has relied on medical datasets or purpose-built PPG sensors. However, using these systems

is not practical due to the specialized hardware required (for example, users are unlikely to clamp a pulse oximeter on their finger for authentication).

In this chapter, we are proposing the first system to collect a user's PPG through a mobile phone camera and authenticate them through the extracted signal. Smartphone-based PPG authentication has two main advantages over established biometrics: unlike fingerprints or facial images, PPG is hard to observe, making it less susceptible to certain attacks. For example, Chapter 5 describes how a photo of a user's face is oftentimes enough knowledge to construct an attack, similarly, our work [207] shows similar properties for a sample of an individual's voice. Instead, during a PPG measurement, the user places their finger on the phone's camera while the finger is illuminated by the phone camera flashlight (see Figure 3.1), which is harder to observe in comparison. The hardware requirements for the PPG extraction are low and no specialized sensors are needed, which makes the technique even suitable for low-cost feature phones. We validate our approach through a user study with 15 participants and capture their PPG over 6-11 sessions. The multi-session nature of our experiment allows us to explore the time stability of the biometric, a component largely overlooked by previous work. In order to facilitate future work in the field, we make our code and the dataset available online[1].

3.2 Background

In this section, we will present the physiological foundations of PPG-based authentication, discuss different measurement methods and summarize related work.

3.2.1 Physiological foundations

A photoplethysmogram is an optically obtained plethysmogram that can be used to detect blood volume changes in the microvascular bed of tissue [183]. Having first been proposed in 1938, PPG is a relatively simple technique that only relies on a light source to illuminate the skin tissue, and a photo detector to measure the small variations in light intensity associated with changes in perfusion in the

[1] `https://github.com/ssloxford/seeing-red/`

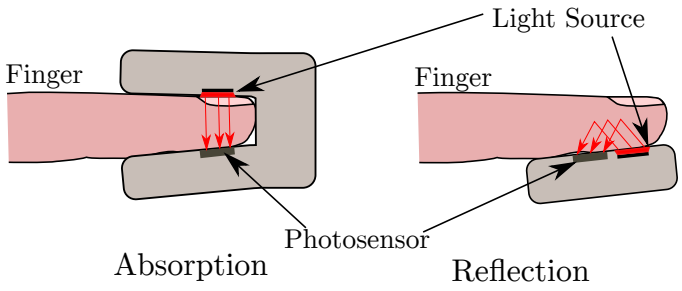


Figure 3.2: The two modes of PPG collection. The absorption mode is usually used by devices in a medical setting, such as pulse oximeters, whereas the reflection mode is used in smartwatches and other consumer systems. Our system operates in reflection mode.

catchment volume [125]. Within each heartbeat three distinct fiducial points can be identified that relate to different stages of the cardiac cycle (see Figure 3.4). Some biometric features used in this work (see Section 3.4.7) are directly based on these fiducial points and derive their distinctiveness from the uniqueness of the person's cardiovascular system.

The systolic peak is a result of the direct pressure wave travelling from the left ventricle to the periphery of the body [31]. Its amplitude relates to stroke volume [136] and is used to estimate continuous blood pressure [43]. The diastolic peak (or inflection) is a result of reflections of the pressure wave by arteries of the lower body [31]. The time difference between the systolic peak and diastolic peak has been previously used as a measure of large artery stiffness [133] The dicrotic notch is a small and brief increase in arterial blood pressure that appears when the aortic valve closes and is commonly used as an equivalent of end-systolic left ventricular pressure [49]. A more in-depth review of the PPG signal morphology and the medical reasons for the distinctiveness of individual components can be found in [63].

3.2.2 Measuring PPG

In a clinical setting, PPG is typically measured with a pulse oximeter, usually clamped to a person's finger, and consists of a light source on the top and a photosensor on the bottom, as shown in Figure 3.2. The sensor then measures changes in light absorption, allowing it to distinguish between oxygenated and deoxygenated blood. A PPG sensor used in absorption mode must be located on the body at

a site where transmitted light can be detected [153]. Consequently, measurement sites are limited to the extremities of the body, such as the fingertips or earlobes.

In recent years, PPG sensors have become increasingly popular in wearable devices (smartwatches in particular) to allow continuous heart rate measurements without the need for a chest strap. PPG-enabled watches use a set of LEDs to illuminate the skin tissue and a photosensor to capture the resulting reflection, as seen in Figure 3.2. The key difference to pulse oximeter measurements is that the light source and photosensor are adjacent. This is required as the absorption mode is not feasible on the wrist.

3.2.3 PPG for Authentication

While PPG was first proposed in 1938, it has only been used for authentication since 2003. Work in this area differs in terms of measurement device (particularly reflection vs absorption mode), sample size, feature types and consideration of feature stability over time. A summary of the field with regard to these properties can be found in Table 3.1.

Most early work in the field uses relatively simple, low-dimensional feature sets [81, 82], focusing on the number of peaks and the slope of parts of the waveform. Bao et al. use heart rate variability derived from PPG as a sole feature [15].

More recent work has significantly extended the set of features to more comprehensively capture the distinctiveness of each constituent part of the PPG waveform and incorporates non-fiducial methods in order to avoid the difficulty of peak detection in noisy signals [95]. The latter is particularly useful for the reflective measurement mode where the the diastolic peak and dicrotic notch are often hard to identify.

Unlike earlier work, which collected data from a single measurement session, Kavsaoğlu et al. conducted three recording sessions with 30 participants [98]. In order to compare intra-session and cross-session performance, they report individual results for each of the sessions and for a combined dataset consisting of both sessions. The authors achieve identification rates of 90.44% and 94.44% for single-session

Table 3.1: Related work on PPG-based authentication.

Paper	*Measurement*	*Subjects*	*Dataset*	*Sessions*
[82]	reflection	17	own	1
[81]	reflection	17	own	1
[15]	unknown	12	own	1
[234]	absorption	3	own	1
[189]	absorption	14/15	[1]	1
[27]	absorption	44	own	1
[98]	reflection	30	own	3
[173]	absorption	23	[101]	1
[95]	absorption	42	[96]	1
[94]	absorption	42	[96]	1
[239]	absorption	20	own	1
Our work	reflection	15	own	6-11

tests and 87.22% for the combined dataset. However data from the sessions have not been separated in the evaluation, which positively biases the results compared to a real-world scenario [62].

The most important difference in our work is the measurement device. Previous work relies on either purpose-built hardware or medical-grade devices, whereas we use readily available consumer hardware. In addition, most research uses single-session data and performs training and testing within the same session. Due to the single-session nature of most experiments, it is possible that systems distinguish measurement sessions, rather than individuals. For example, PPG measurements are affected by skin temperature [90] and sensor contact pressure and position [39], both of which can easily introduce artefacts into the signal that would only be present in this recording.

3.3 Experiment Design

In this section we present our data collection method, which we subsequently use for developing our authentication systems.

3.3.1 System Requirements

Our system is based on collecting PPG data for authentication with a smartphone camera. Our goal is to replace other forms of smartphone-based user authentication which are less secure because of the observability of the biometric trait (see Section 3.1). While observability is not always objective scale, it has been shown that biometric traits such as faces, fingerprints and voices can be replicated with varying degrees of effort on the adversaries side [117, 87, 207]. In comparison, PPG is harder to observe: although a few examples are able to extract PPG remotely from (close-up) videos of users by monitoring subtle skin colour changes [200, 176], this presents several practical challenges which warrant the use of PPG as a less-observable trait. In principle, PPG offers good enough recognition accuracy to replace other commonly used traits: when measured under optimal conditions with a finger-worn pulse-oximeter, PPG measurements are discriminative enough to obtain around 1% equal error rates on single beats [171].

3.3.2 Collection Apparatus

As previously mentioned, a key contribution of this work is the ability to use PPG for authentication without purpose-built hardware. As such, our system makes use of a smartphone's camera and camera flash to act as the photosensor and light source respectively. One challenge in this setup is presented by the camera light: commercial PPG sensors in reflection mode use green light as its wave length leads to ideal penetration depth [125]. Conversely, the phone camera uses white light, which leads to noisier measurements. We collect the PPG data using a custom iOS application, running on an Apple iPhone X.

Users are instructed verbally to hold the phone in their dominant hand, and place their finger lightly over both the camera and camera flash. They are instructed to sit down whilst performing the capture, and to remain as still as possible throughout; the application automatically stops recording if too much movement is detected i.e., total acceleration exceeds 1.3g ($12.75m/s^2$).

We set the camera ISO and exposure time to be the minimum possible, given the camera FPS and resolution settings. Likewise, we set the white balance gain to be the maximum in the red channel, and minimum in the blue and green channels. By fixing these settings we ensure consistency between capture sessions, reducing noise in the signal and increasing the information captured about the colour changes.

A single capture session, as taken by the application, consists of a video recorded in 1280x720 resolution at 240 frames per second, for a duration of 30 seconds. The video is resized to 360x240 and uploaded to a server for validation. After each capture session, the researcher supervising the data collection takes the phone away from the user and returns it to them for any subsequent capture sessions, leading to users placing their finger on the camera differently each time. A validation function analyses the red light channel of the video once it has been uploaded to the server, and rejects the video if there are sudden jumps or a lack of red light, as these are likely caused by the user lifting their finger from the camera.

3.3.3 Collection Procedure

We collect data for a total of 15 participants, performing between 6 and 11 capture sessions (measurements) per participant. There were 13 male and 2 female participants. Of the participants, 13 identify as white, one as Indian and one as South-East Asian. Each participant is handed the device running the application and given the above instructions on how to use the application, as well as being given a visual prompt towards correct usage on the screen. Captures for each participant were taken across multiple sittings, with no more than three capture sessions occurring in the same sitting. Sittings were taken approximately a day apart, with a median of 32.1 hours between any pair of capture sessions.

3.3.4 Research Ethics

Before the data collection, we obtained informed consent from the participants. Our data collection process and experiment took place with ethical approval from our institution: reference SSD/CUREC1A CS_C1A_19_032.

3.4 Method

Our method consists of a pipeline of steps which transform the raw video captured by the camera into biometric samples which can be used for authentication. We design the pipeline with the aim of running the system in real-time, meaning that all processing steps can be run as the video is being captured (i.e., we never use processing that requires looking at a whole video file, only at the video recorded up to that point).

3.4.1 Signal Extraction

To obtain the signal from the raw video, we compute the mean of the pixel-wise luma component from the pixels in each video frame, so that if F is a video composed by a sequence of frames $\{f_1, ..., f_m\}$, then the signal originating from F is:

$$\begin{gathered} S = \{Y(f_1), ..., Y(f_m)\}, \text{where} \\ Y(f) = \frac{1}{n} \sum_{i,j \in f} [0.299 f_{i,j}^{(r)} + 0.587 f_{i,j}^{(g)} + 0.114 f_{i,j}^{(b)}]. \end{gathered} \tag{3.1}$$

In Equation 3.1, i and j iterate over the pixels of the image and the superscripts $_^{(r)}$ indicate the considered RGB channel of the frame, either red green or blue. The channel coefficients of Equation 3.1 are taken from the ITU-R BT.601 standard.[2]

3.4.2 Signal Preprocessing

After the signal from a video is extracted, we apply the following preprocessing steps to remove noise. First, in order to remove trends from the signal we compute and subtract a rolling average of the signal with the signal itself. We use a window size of one second for the rolling average. Then, we use a low pass filter to remove high frequency noise, with cut-off frequency at 4Hz, i.e., 240 beats per minute. See Figure 3.3 for a visualization of the processed signals.

[2] `https://en.wikipedia.org/wiki/Rec._601`

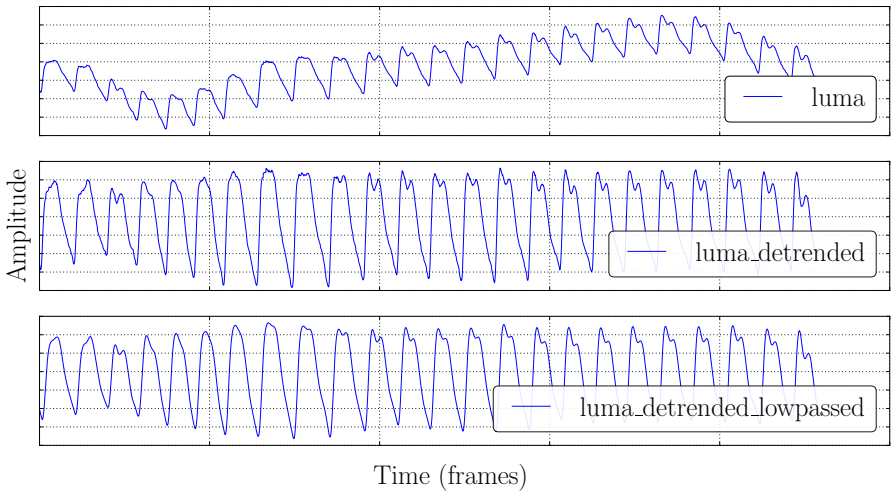


Figure 3.3: Signal over the course of the preprocessing steps. The first plot shows the luma component extracted by the video frames (Equation 3.1). The second plot shows the signal after de-trending with rolling average and the third plot shows the signal after the low-pass filtering, which is used in the rest of the pipeline.

3.4.3 Beat Separation

As mentioned above, in our approach we want to perform single beat authentication, therefore we design a beat detection algorithm to separate individual beats in the signal. We report in Algorithm 1 the procedure to obtain individual beats from a preprocessed signal. Algorithm 1 first finds the minimums of a smoothed version of the signal (moving average, Line 3), and then matches those minimums to the minimums in the original signal. The "arg rel min " instruction extracts all relative minimums indexes from the signal. We found that this algorithm leads to a more reliable beat separation, as the signals often presents random noise which survives the preprocessing above.

3.4.4 Fiducial Points Detection

After individual beats have been extracted, we detect fiducial points in the signal, which are used to extract features. We focus on three points in particular: (i) systolic peak, (ii) dicrotic notch and (iii) diastolic peak. In low-noise signals, these points are easily found by finding maximums and minimums in the signal and its first derivative, see Figure 3.4. However we found that this procedure needs

Algorithm 1 - Beat Separation. Given a signal detects individual heartbeats boundaries.

Input: preprocessed signal S, max_bpm U, sampling rate fps, smoothing window size ws.
$g = 60 \cdot \frac{fps}{U}$ ▷ minimum inter-beat gap
$A = \text{moving_average}(S, ws)$
$mins^{(S)} = \text{arg rel min } S$
$mins^{(A)} = \text{arg rel min } A$
$B = \{\}$ ▷ Individual Beats
for $i \in mins^{(A)}$ **do**
    $S_i = \{s_{i-g}, ..., s_{i+g}\}$
    $j = \underset{i-g,...,i+g}{\arg\min} S_i$
    $B = B + \{j\}$
end for
return B

to account for noisy signals, so we craft a more robust algorithm to detect them which falls back to best guess points.

3.4.5 Beat Signal Quality

Oftentimes, slight hand or finger movements may cause noise in the signal which bypasses our filtering pipeline. We therefore design a set of quality criteria for individual beats with the goal of excluding noisy beats from further processing. These criteria are the following:

- **max bpm**: all beats which correspond to > 120 bpm, as these are caused by noise which bypasses the separation of Algorithm 1;
- **number of peaks**: all beats with more than three distinct maximums;
- **distance from reference**: all beats that are too different from a reference expected beat wave. We compute this difference using dynamic time warping [170] between the current beat and the reference and we set a threshold of 2.0, i.e., beats with distance higher than 2.0 are discarded. The reference wave is the average beat wave in our dataset.

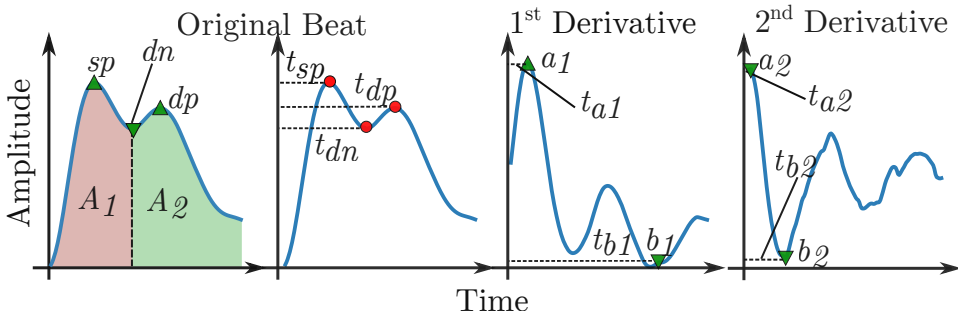


Figure 3.4: Visualization of fiducial point based features for a single beat. Time span features are based on time from start of an individual heartbeat, and are normalized to account for different bpm.

All beats that match at least one of these conditions are flagged as failed to acquire (FTA). Using these quality filters leads to better authentication accuracy at the cost of slightly longer authentication time. In the analysis, we report the results including and excluding these beats.

3.4.6 Feature Extraction

We consider four groups of features: statistical, curve widths, frequency domain and fiducial points based. To make features independent of a user's bpm at the time of capture, after the physiological features are computed, we re-sample beats onto a fixed sampling rate of 1,000 Hz and we normalize them so that the amplitude values lie in [0, 1]. Features are computed on a beat-basis, so that each beat leads to a sample.

Statistical. These are straightforward statistical features that are based on the physiological traits of the users and for a single beat include: (i) maximum value, (ii) minimum value, (iii) the difference max-min and (iv) the length of the beat. Given that our camera settings are fixed, features (i)-(iii) are a result of the absolute values found in the luma component of the video frames, which are related to skin tone and amount of blood flowing through the finger. Feature (iv) is just related to the individual's heart rate at rest, which has some variability, but we found it to be a distinctive features in our case.

Curve Widths. These features are the widths of the curve defined by the beat at a set of pre-defined heights. For a certain height $h \in [0, 1]$, where 1 is the maximum height and 0 the minimum, we consider the curve that lies above the threshold h and compute the distance between the extremes of this curve, see Figure 3.4. We choose 18 different heights evenly spaced in [0.05, 0.95] and compute one width feature for each of them.

Frequency Domain. For each beat we compute the discrete Fourier transform and use it as features. Section 3.4.7 discusses how we reduce the dimensionality of these features with feature selection.

Fiducial Points. We add a set of features based on the location of the fiducial points discussed above based on related work, see Figure 3.4 for examples of the features used.

3.4.7 Feature Selection

After feature extraction, each beat is represented by a total of 541 features, 4 physiological, 18 curve widths, 500 frequency domain and 19 from the fiducial points. To reduce dimensionality and remove redundant features, we apply multi step feature selection.

Step 1. We use principal component analysis (PCA) [201] to reduce the number of features in the frequency domain and the curve width groups. For the frequency features, we fit a PCA with 100 components and retain only the first n components so that these n components describe most of the space variance. For the curve width features, we do the same but fit a PCA model with 15 components. In practice, we choose to retain enough components to describe 99% of the variance, we do not extensively investigate this hyperparameter in our experiments and stick to 99% as a good compromise between the number of retained dimensions and the variance described. We find that n=5 components are necessary to retain this amount of variance for the frequency group (depending on the dataset) while n=9 are sufficient for the width group.

Step 2. In this step, we combine two different techniques for feature selection on the remaining features. At first, we compute the pairwise correlation coefficient between features and for a pair of feature distributions (f_1, f_2) we drop one feature from the dataset at random if the correlation coefficient between them is $r_{f_1,f_2} > .95$. Then, to avoid the effect of outliers, we remove all feature values which lie outside the 1^{st} and 99^{th} percentile of the feature distribution. Then we use minimum redundancy maximum relevance (mRMR) feature selection [152] to select the top 60% performing features (differently from the original paper, we choose the Mutual Information Quotient criteria in the algorithm). Afterwards, we select the top 60% features ranked on their relative mutual information (RMI). We use the non-parametric approach to estimate RMI introduced in [167] to avoid imprecisions arising from modelling continuous distributions with discrete bins. Finally, we choose all the features that pass both selections (RMI and mRMR) to create our final feature set. Section 3.5 presents and discusses the selected features set.

3.5 Results

In this section we present the results for a set of authentication use-cases.

Preliminaries. In each use-case, we mainly monitor the equal error rate (EER) of the authentication, that is the point at which False Reject Rate (FRR) and False Accept Rate (FAR) are equal. In each experiment, rather than using the classification output from the classifier as the decision (accept/reject), we use the distance from the decision boundary (or prediction probability) to make the classification decision, i.e., we list all distances coming from the positive (the user) and negative (other users) samples, and find the EER by changing the threshold.

Aggregation. We report results for aggregating the decision on multiple samples, i.e., we choose an aggregation window size n, collect n samples and compute the decision based on the aggregated samples. The aggregation function we use is the mean.[3] To gather more data points, whenever we aggregate samples together,

[3]We also tried median but it performed worse on average.

Table 3.2: Features retained and their importances (RMI column) after the feature selection step, for the ALL dataset.

Feature	mRMR	RMI
A_2/A_1	1.29	0.14
t_{b_1}	1.11	0.13
b/a_2	1.11	0.10
b_2	1.02	0.13
max	1.02	0.17
min	0.88	0.23
a_2	0.86	0.12
t_{b_2}	0.82	0.12
t_{sp}	0.77	0.12
fft_{PCA_0}	0.75	0.09
length	0.72	0.12
fft_{PCA_1}	0.70	0.08
t_{dn}	0.69	0.15
t_{a_2}	0.67	0.08
fft_{PCA_5}	0.66	0.08
b_1	0.64	0.12
t_{a_1}	0.64	0.10
A_1	0.61	0.11

we randomly draw sample sequences of length n for 100 times from the positive (genuine) class. For a certain user, to construct the negative (impostor) class, we randomly draw sample sequences of length n for 10 times for each other user. This way, each EER value for a user is computed using $100 + 10 \times (|U| - 1) = 240$ authentication attempts.

At the end of our data collection, we obtain a dataset of a total of 3,836 samples (one sample equals one beat) with at least 178 samples per user. Of the samples, 3,529 (92%) pass the beat signal quality checks discussed in Section 3.4.5. Throughout the evaluation, we refer to these two separate datasets as **ALL** and **Post-FTA** and we report results for both. After feature selection, we retained the top 18 features for both ALL and Post-FTA, see Table 3.2 for details.

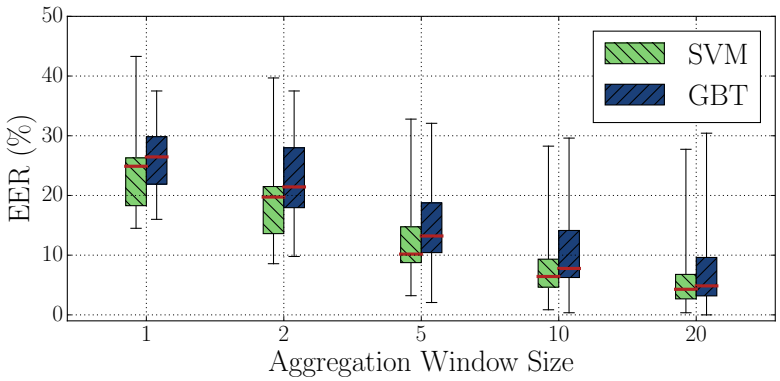


Figure 3.5: EERs in the multi-class case, for various classifiers and aggregation window sizes. Values are averaged over the two folds before plotting. Whiskers show the range of the data, horizontal lines in boxes show the median.

3.5.1 Multi-class Case

Setup. We randomly split the data into two stratified folds and train two different classifiers, support vector machine (SVM) with a radial basis function kernel and gradient boosted trees (GBT). For SVM we use a standard scaler to normalize the data before feeding it into the classifier, and only use the training part to fit the scaler.

Results. We report in Figure 3.5 the obtained EERs. We found that single sample EER lies around 15%, but decreases quickly to less than 5% when using an aggregation window size > 10. Out of the tested models we found that SVM performs the best, with an EER of $<1\%$ at aggregation window size of 20.

3.5.2 One-class Case

Setup. As a PPG-based authentication system leveraging the smartphone camera would reside on the user's device, the system would likely only have access to the legitimate user's data (provided at enrolment). We therefore consider the case where the training algorithms do not have access to any negative class data. We use two algorithms which can be fit only to a single class: Isolation Forest (IsFrst) [113] and a one class SVM (OSVM). The training data is therefore only provided at enrolment, when the user first uses the system, and as such we vary the number of samples provided at enrolment. This helps us provide insights on how many samples

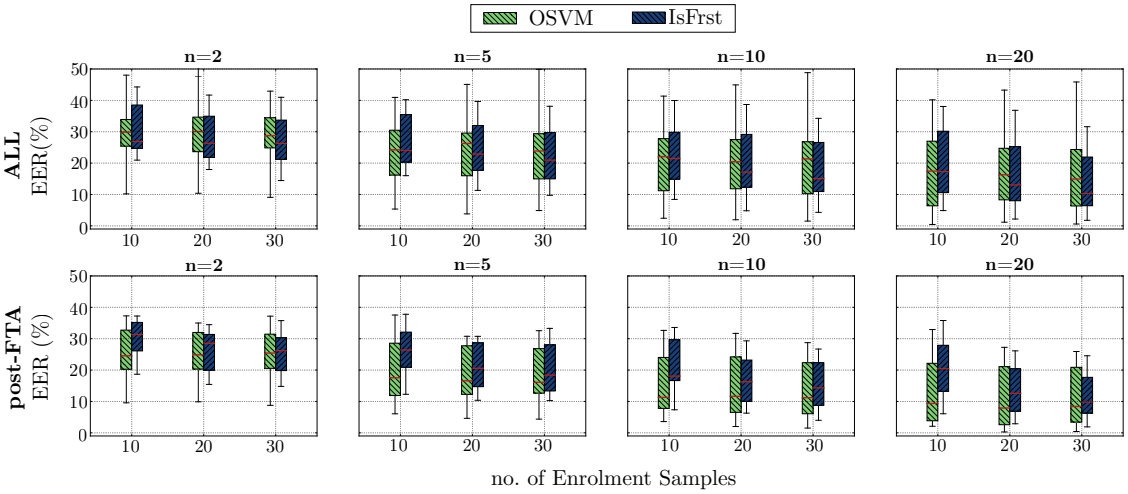


Figure 3.6: EERs in the one-class case, for various classifier, aggregation window sizes and no. of enrolment samples. Values are averaged over the number of random enrolments first and then across users. The first row of plots reports the results for the ALL dataset, while the second row reports the post-FTA dataset. Each plot reports the aggregation window size n used at the top, i.e., either 2, 5, 10 or 20. Whiskers show the 95% confidence intervals.

are necessary to capture sufficient intra-user variation. The rest of the experiment follows the same method described at the beginning of Section 3.5, with the difference that we repeat the enrolment 10 times to ensure that the enrolment samples do not bias the results (selecting particularly good/bad samples for enrolment will greatly influence the results). To compute the EER, we use the distance from the decision boundary for OSVM and the anomaly score for IsFrst. The results are averaged over the number of repeated enrolments first and then over the individual users.

Results. We report the results of the analysis in Figure 3.6. The figure shows the distributions of EERs across: (i) ALL or Post-FTA dataset, (ii) number of enrolment samples, (iii) aggregation window size. The number of enrolment samples represents a small fraction of the available user data (for each user we have at least 178 samples, and we use as few as 10 for training). We found that OSVM performs slightly better on average but not in a consistent manner. Using the Post-FTA dataset leads to better recognition results (as low as 4% at 20 aggregated samples) and smaller variations in the EER distributions, while the average EER for the ALL dataset is above 10% for most configurations with the exception of using IsFrst and aggregating 20 samples.

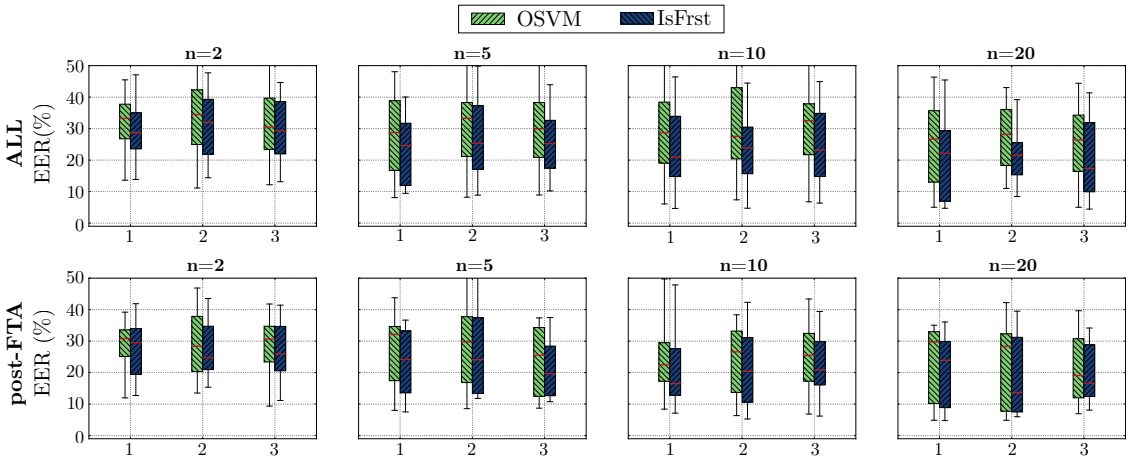


Figure 3.7: EERs in the one-class cross-session case, for various classifier, aggregation window sizes and no. of enrolment sessions. Values are averaged over the number of random enrolments first and then across users. The first row of plots reports the results for the ALL dataset, while the second row reports the post-FTA dataset. Each plot reports the aggregation window size n used at the top, i.e., either 2, 5, 10 or 20. Whiskers show the 95% confidence intervals.

3.5.3 One-class Cross-Session

Setup. The evaluation methodology is identical to the one described in Section 3.5.2, with the difference that now we select enrolment data by choosing a number of sessions (we have at least 6 sessions per user, with a mean of 9.3 sessions and median of 10 sessions) and test on the remaining sessions. A key difference to the work by Kavsaouglu et al. [98] is that we draw training and testing samples from separate sessions. This makes our analysis more realistic, as our results are not positively biased by features that are not stable between sessions.

Results. We report the results of the analysis in Figure 3.7. As in the previous section, the figure shows the distributions of EERs across: (i) ALL or Post-FTA dataset, (ii) number of enrolment sessions, (iii) aggregation window size. As mentioned in Section 3.3.3, we have 6-11 measurement sessions per user, the number of enrolment sessions (which varies between 1-3) represents between 10%-50% of available data per user. We found that the EERs are much higher in this case, with most scenarios scoring over 20% EER on average and high variance in the results. This suggests that each PPG measurement is not only unique to the user,

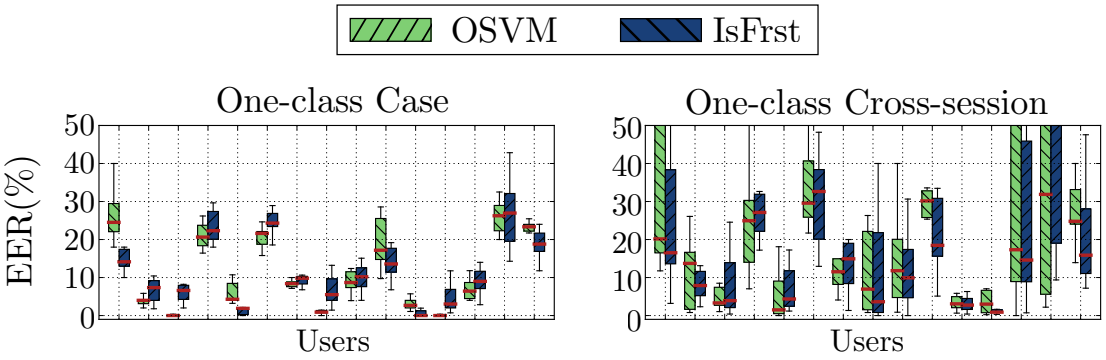


Figure 3.8: Distribution of EERs across users in our dataset, for the one-class and cross-session case, the Post-FTA dataset and an aggregation window size of 20. Whiskers show the 95% confidence intervals.

but also to the way the measurement is being taken, leading to most of our features being inconsistent over separate sessions.

3.5.4 EER User Distribution

We investigate how EER distributes over different users, showing the results in Figure 3.8. The figure shows how the distribution is uneven across users, with the worst performing user contributing the most to the overall EER. User 6 has over 20% EER in the One-class case and above 30% in the cross session case, while the authentication is relatively stable for a subset of users even in the cross-session scenario (e.g., User 3, 5, 11, 12 have below 5% EER). We found that such results are highly susceptible to the quality of samples used for enrolment, suggesting that stricter quality metrics in this phase could further improve robustness. Equally this effect could be caused due to specific, unknown characteristics of given individuals, as has been shown to be the case in biometric systems previously [53].

3.6 Discussion

Observability. Because the observability of the PPG trait is at the base of its security, such characteristic should be estimated in more detail. It has been shown that it is possible to extract an individual's pulse from remote videos thanks to the blood subtly changing the light absorption properties of ones' skin [200, 14, 176], in

what is known as remote PPG. This capability can be translated into an attack to uncover the PPG trait of a user from videos, so should be considered in further security analysis of PPG-based authentication systems. If efficient and accurate methods to observe PPG remotely were discovered, this would harm the security guarantees of such a system, as presentation attacks could be easily enabled with light-modulating screens (such as smartphone screens).

Practical Challenges. We observed that a set of factors greatly affect the fidelity of the signal. Firstly, we found that the warmth of the fingertip changes the amount of blood flow, leading to slightly different measurements. Secondly, breathing has an effect on the signal: heartbeats are stronger when inhaling compared to exhaling [16]. Additionally, slight hand and/or finger movements can generate noise that may be (in part) corrected by using accelerometer and gyroscope measurements. Further investigations should also consider the effects and limits of using PPG on different fingers and under different activities (e.g., sitting, walking, running). A larger data collection spanning a larger number of sessions and users may allow the use of deep models to derive more robust features in an unsupervised manner.

Final Considerations. These results suggest that at this time, camera-based PPG would not be considered an extremely secure biometric trait, and should be therefore used with the awareness of its limitations (see for example the Strong/Weak/Convenience distinction made by Android APIs [5] for biometric sensors). Nevertheless, PPG could be combined together with other traits to increase its security, or even in continuous fashion thanks to the increasing popularity of smartwatches equipped with PPG sensors [239].

3.7 Summary

This chapter proposed a system that enables PPG-based authentication by using a smartphone camera: the signal is collected by recording a video from the camera as the user is resting their finger on top of the lens. From the next chapter, we will take a look at what types of attacks an attacker can perform by targeting the vision

part of the system (either altering inputs directly or corrupting the sensor, see Figure 1.2). Here, rather than describing an adversary, following Research Question 2, we suggest an alternative way to perform user recognition which could be used in systems which are more vulnerable to real-world attacks. In fact, using PPG has the benefit of being less observable than other biometric traits such as face recognition which makes attacks more costly on the attacker's side, as discussed in Section 2.2. While the performance of the proposed system is not comparable with other established biometric traits, these results highlight the fact that further investigation is needed to improve PPG recognition in this camera-based scenario.

4

Poisoning Attacks against Unsupervised Template Updating

Contents

4.1 Introduction 42
4.2 Background 44
4.2.1 Template Update 44
4.2.2 Poisoning Attacks 44
4.2.3 Differences from Previous Work 46
4.3 System and Threat Model 48
4.3.1 System Model 48
4.3.2 Adversary Model 49
4.4 Attack Concept 51
4.4.1 Overview 51
4.4.2 Challenges 52
4.5 Attack Method 53
4.5.1 Poisoning Sample Generation 53
4.5.2 Poisoning Attack Injection 56
4.6 Evaluation 58
4.6.1 Experiment Design 58
4.6.2 Attack Success and Effect on Error Rates 61
4.6.3 Transferability 64
4.7 Poisoning Countermeasures 65
4.7.1 Detecting Adversarial Samples 65
4.7.2 Template Anomalies 66
4.7.3 Angular Similarity Detection 67
4.7.4 Detection Evaluation 68
4.8 Discussion 71
4.9 Summary 72

4.1 Introduction

In the previous chapter, we saw how the unobservability property of PPG is beneficial when users want to protect themselves from adversaries attempting presentation attacks. In comparison, for face recognition, simple photos of a user's face might reveal their trait for adversaries to copy permanently. Nevertheless, state-of-the-art face consumer face recognition mostly prefers to increase security guarantees of the authentication by increasing the bar for an attacker to produce a valid biometric sample – mostly with strong liveness detection. For example, Apple Face ID [11], checks for liveness of the presented sample by (i) capturing several frames, (ii) checking for 3D profile with a depth camera, (iii) checking for attention in the user's gaze. This is paired with current trends in biometric systems also prioritizing high protection of biometric *templates*, i.e., the user's stored biometric information. Templates in fact represent sensitive user data and their leak might compromise the secrecy of the biometric trait in a permanent way. To protect the template's secrecy, for example, both Apple's TouchID and FaceID store templates in the protected secure enclave and the data never leaves the user's device [11]. Due to these protection mechanisms, attackers can not modify the template directly, since that would require them to compromise the device.

Nevertheless, we noticed another trend in modern biometric systems: the usage of *template updating* procedures. During template update, the system replaces or adds recently seen samples to the user template. Template update allows systems to cope with the inherent variability of biometric traits, such as physiological changes or sensor measurement noise, by collecting additional user samples. The presence of template updating leaves another attack vector available for an adversary: they can exploit the updates to cause the system to adapt the template, thereby modifying the data within.

By exploiting the template update procedure, an attacker can inject adversarial samples (a backdoor) into the user template to carry out a *template poisoning* attack. In this attack, the adversary corrupts the user template allowing themselves to impersonate the user with his own biometric trait. The poisoned template is hard to detect and creates an inconspicuous and stealthy backdoor for the adversary in the long-term. Once placed, the backdoor allows the adversary to access the system without requiring them to modify their appearance.

In this chapter, we will show how an adversary can combine partial knowledge about the system and of the deep neural networks vulnerability to adversarial examples to carry out this poisoning attack by altering their own facial features (i.e., the camera-captured scene), even when bound by a physical realizability constraint. We will also propose a new countermeasure for the detection of this poisoning attack, allowing for biometric systems to benefit from template updates without having to worry about such attackers.

Contributions. The contributions of this chapter are as follows:

- We propose a method to plant *biometric backdoors* by poisoning the legitimate user's template in limited knowledge (known network, update policy and population data, unknown user template) and limited injection capabilities scenarios.

- We evaluate the attack on state-of-the-art recognition pipelines, including white- and black-box models. We show that the error rates of the system hardly change when such a backdoor is present, making the attack inconspicuous.

- We introduce a poisoning detection method that thwarts poisoning attacks without affecting legitimate template updates, and we investigate these trade-offs on a large face dataset.

As previously, the code used in this chapter is available online.[1]

[1] `https://github.com/ssloxford/biometric-backdoors`

4.2 Background

In this section, we give an overview of the reasoning behind using template updates and a broader view of data poisoning attacks in systems using machine learning models.

4.2.1 Template Update

Template updating is a recent trend in modern biometric systems: in template updating newly seen user samples are added to the user biometric template under some policy. The reason systems increasingly rely on template updates, is that while physiological might not show significant intra-user variations, when using sensor-captured data, a series of factors may influence one's trait appearance to the sensor. For example, for face recognition, well known variation factors include age, pose (viewing angle), lighting environment, facial hair and facial accessories. Some of these variations can be accounted for at enrolment, for example asking the user to rotate their head as some samples are collected or shining additional light onto the users face to account for the effect of external lighting conditions. As an example, both Apple's FaceID [11] and TouchID [10] perform template update procedures. FaceID performs template updates either after a new sample is accepted, or when a rejected sample is followed promptly by a correctly entered backup passcode.

4.2.2 Poisoning Attacks

Poisoning attacks is a blanket term that covers a variety of attacks that are based on altering data in a malicious way, so that an automated system using the data will misbehave [24]. These attacks have become more common since systems started increasingly rely on machine learning algorithms based on data in order to make decisions. In *backdoor* attacks, such as the one in this chapter, attackers attempt to go unnoticed, therefore they aim to retain the accuracy of the model on clean samples, while inserting the malicious behaviour. In biometric systems, poisoning attacks may be categorized into two different categories: (i) poisoning the DNN, (ii) poisoning the user template. When the DNN is the target of the

Table 4.1: Comparison with previous work in template poisoning attacks. A dash (—) indicates that the property does not apply directly to the work.

Work	template update	limited injection capabilities	unknown template	state-of-the-art recognition	unknown network (black-box)	stealthiness	uncompromised training phase
Kloft et al. [100]	✓	✗	—	—	✗	—	✓
Biggio et al. [21]	✓	✗	✗	✗	✓	✗	✓
Biggio et al. [22]	✓	✗	✗	✗	✗	✗	✓
Garofalo et al. [71]	✗	✗	✗	✓	✗	✗	✗
Liu et al. [115]	✗	—	✓	✓	✗	✓	✗*
This chapter	✓	✓	✓	✓	✓	✓	✓

*network weights are changed after training

poisoning, there is no assumption on controlling the DNN inputs, but rather the adversary has an increased control over the training phase, i.e., can directly edit the network weights [115] or change the training data [80]. However, as DNNs are increasingly used as black-box feature extractors in a classification pipeline rather than directly as classifiers, analysis of poisoning of the template database needs to be included in the security analysis. A preliminary analysis of template poisoning attacks on biometric systems was proposed by Biggio et al. [22]. The authors present a poisoning attack where an adversary injects a set of samples that gradually shifts the user template towards the biometric trait of the adversary. The same authors extended their work reducing the knowledge assumptions of the adversary, showing that the attack is feasible even considering a black-box scenario and limited knowledge of the victim template [21].

4.2.3 Differences from Previous Work

We use the previous work in physically-robust adversarial samples as one of the building blocks of our work, which allows us to control and carry out the poisoning attack as a whole. We report a comparison to previous work in Table 4.1, where we focus on the following properties.

Template Update. We consider that template updates are limited by self-update thresholds. However, we consider penalties for consecutive rejected authentication attempts. This is a reasonable security policy as sequences of failed authentication attempts might correspond to attackers trying to impersonate the user, and falling back to a different authentication method can thwart such attempts, e.g., FaceID allows five failed attempts before switching to personal identification number (PIN) input. In comparison, previous works [22, 21, 100] did not consider or investigate the frequency of rejected attempts.

Limited Injection Capabilities. We consider only cases that account for the physical realizability of the adversarial sample creation and injection. Adversarial examples have been shown to be realizable in the real-world [67, 179, 104], but their flexibility and effectiveness is greatly reduced in that case. Assuming perfect injection capabilities, where the adversary can freely inject samples and *digitally* manipulate exactly each pixel in the input is an unrealistic assumption in our scenario. In fact, being able to feed arbitrarily manipulated samples to the system, which bypass liveness detection (such as the images in [71, 22, 21]) would mean that the adversary could simply be able to authenticate inconspicuously at any time just by having a printed image of the user. This defeats the purpose of carrying out a more complex poisoning attack.

Unknown Template. Previous work focused on full template knowledge [22, 71]. Biggio et al. [21] considered a partial knowledge scenario where the known sample is chosen as the closest to user's centroid out of *all* the samples in the test set. In a biometric system, this assumption would require the adversary to have availability of several samples from the user, which might not be easily obtainable. Therefore,

we focus on an adversary that does *not* have any knowledge of the template stored in the user device, but only has a single image of the user.

Unknown Network. While we start our analysis with a white-box scenario, we extend it to black-box ones using the principle of transferability of adversarial samples [147, 51, 149]. A black-box attack was shown to be feasible in [21], however the system relied on non-state-of-the-art face recognition methods (i.e., EigenFaces [206]) that are nowadays greatly outperformed by deep neural networks.

Stealthiness. We consider a poisoning attack to be *stealthy* when it can be carried out without compromising the normal error rates of the system. This is a desirable property as changes in false acceptances or false rejections (i.e., legitimate user cannot authenticate, other users can authenticate) would be suspicious: users might stop using the biometric system or re-enrol their template in an attempt to reset the system performance. This approach separates us from previous work, such as [21, 22, 71], which aims to *replace*, rather than alter the legitimate user's template, blocking them out of the system. With enrolment being a procedure that the user can carry out independently, such attacks would not guarantee long-term access to the adversary, as victims would re-enrol into the system as soon as they are inconvenienced by the reduced recognition rates.

Uncompromised Training Phase. Unlike previous work, we assume no control over training data [71] or network weights [115]. While these assumptions may lead to more effective attacks, they also require the adversary to be able to affect the network training process. In our scenario, it is reasonable that a network would be trained beforehand, and would not be accessible once on the device (e.g., Apple FaceID is stored and executed in a secure element [9]).

These properties lead our experimental design and evaluation. Our goal is to bridge the gap between a purely theoretical poisoning attack and the challenges that an adversary would have to face in practice, allowing us to better understand the feasibility of such attacks in the real-world.

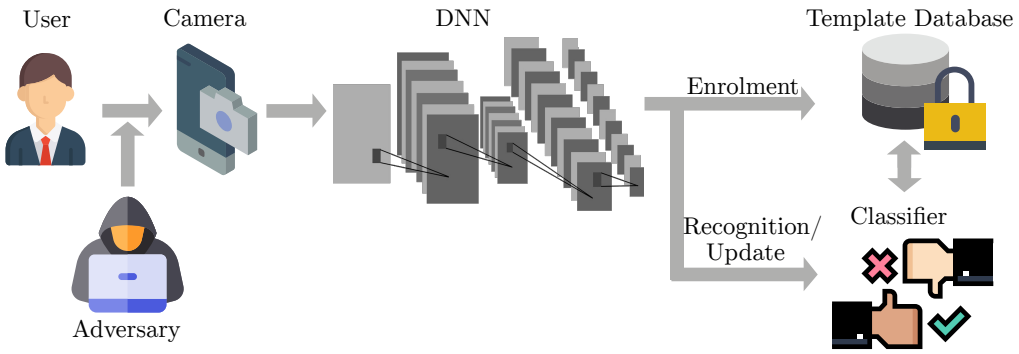


Figure 4.1: System and threat model. The adversary has physical realizability constraints, meaning that the adversarial samples need to survive through the sensor measurement. Captured images undergo a feature extraction step. The samples' features are then used for enrolment, and a classifier processes them during recognition and update.

4.3 System and Threat Model

In this section we first give an overview of the system model and later talk about the considered adversary.

4.3.1 System Model

Modern biometric recognition has four phases: development, enrolment, recognition and template updating.

Development. During development, a deep neural network is trained on a large labelled dataset of biometric samples. The objective of the DNN is to learn how to compute features from user samples so that the extracted features lie in a reduced-dimensionality linearly separable hyperspace. The extracted features can be either *embeddings* [175] when triplet loss is used for training, or they can be the features extracted from the layer preceding the last softmax layer (*logits*). This pre-trained network is then deployed with the biometric system.

Enrolment. At enrolment, the users present their biometric traits to the sensor. One or more measurements are taken, processed and fed into the DNN. The resulting samples are stored in the template database, and are known as the *user template*. These samples are kept in a secured space in the user device and are used as a reference whenever an authentication attempt is made. An illustration of this process is given in Figure 4.1.

Recognition/Update. During the recognition phase, whenever a user presents his trait to the sensor, the system measures it, feeds it into the DNN to extract the features and attempts to match the resulting sample and the known biometric template with a threshold-based classifier. If the match occurs, the user is successfully authenticated. In the update procedure the system might decide to add the newly seen sample into the template database if it is "confident enough" (based on the similarity to enrolment data) that the sample belongs to the legitimate user. There are scenarios where the update procedure might use a more conservative threshold than the authentication procedure i.e., of the samples that pass authentication, only those that are very similar to the known template will be added to it. However, current authentication systems generally set the same threshold for both procedures in order to efficiently keep up with changes in the user template (e.g., [187, 11]), we therefore focus on this scenario.

4.3.2 Adversary Model

The adversary's overarching goal is to place a "biometric backdoor" which grants them stealthy long-term access to the secured system without requiring further effort once the backdoor is in place. In this section, we define the attacker's objectives, knowledge and capabilities.

Objectives. The attacker's goals are to:

- Cause modifications of the user template that leads to the attacker being accepted.
- Maintain the stealthiness of the attack, i.e., minimize the changes to false rejects and false accepts.
- Minimize the number of physical accesses to the system required to plant the backdoor.
- Minimize the number of samples rejected by the system.

Capabilities. The adversary has physical access to the sensor and can therefore feed samples to it. The adversary does not have digital access to the sensor, as in that case he would be able to perfectly and effortlessly control the inputs, making a poisoning attack unnecessary. The adversary can therefore alter biometric sensor measurements only in the physical domain, and these will be subject to both sensor and presentation noise. Unlike previous work, we do *not* assume that the attacker has access to the system during enrolment. The adversary is also unable to directly change the template or training data (e.g., by removing or replacing user samples).

Template Knowledge. In line with related work, we assume that the adversary has at least one picture of the user's face. Being a physiological feature, face appearance is notoriously difficult to keep secret. Social media in particular is a plentiful source for videos or photographs [230]. Differently from previous work, where the best matching picture was chosen out of the test data [21], we do not set any requirement on the known picture other than it being accepted by the system. This assumption entails two obstacles for the adversary: he does not know enrolment data nor the resulting decision boundaries of the system's classifier.

System Knowledge. As our goal is to evaluate the effectiveness of poisoning attacks and construct possible defences, we first assume a strong attacker with white-box access who has access to the DNN used to extract the embeddings after it has been trained. The adversary additionally knows what algorithms are being used by the recognition pipeline i.e., the type of threshold-based classifier and the template update threshold. We will show how the adversary can use population data (ideally publicly available, e.g. for face, [32, 89]) to learn certain distribution properties of users' embeddings distances. We further investigate a black-box attacker using the principle of transferability of adversarial samples [147]. In this case, the adversary has knowledge of a surrogate model (which replaces the DNN) that he can use to optimize the attack on the black-box network.

4.4 Attack Concept

Using the assumptions made in the system and threat model, we present an overview of the attack and of its challenges.

4.4.1 Overview

The concept behind the poisoning attack is that the adversary adds adversarial samples to the legitimate user template in order to change the decision boundary of the classifier. Figure 4.2 shows a two-dimensional representation of how the attack works. There are three categories of samples:

- *user (victim) samples*: legitimate user samples;
- *attacker samples*: samples coming from the biometric trait of the adversary;
- *poisoning samples*: samples algorithmically crafted by the adversary.

Figure 4.2 shows how the user and attacker samples are well separated in the feature space, due to the uniqueness of their biometric traits. At enrolment, the classifier learns the distribution of the user samples creating a boundary around it, shown by the darker blue area. The classifier is able to correctly discriminate between attacker and user samples, rejecting the adversary in an impersonation attempt.

Knowing his own template and a user sample as the starting point, the adversary crafts the poisoning samples accordingly. As the self-update threshold is in place, the adversary must make sure that the crafted samples lie within the current accepted region (shaded blue area), otherwise they would be rejected as anomalous. By injecting one poisoning sample at a time, the adversary shifts the decision boundary towards his own sample distribution. With sufficient poisoning samples, the adversary will move the decision boundary enough so that his own samples will fall inside it, and can therefore impersonate the user with his own trait.

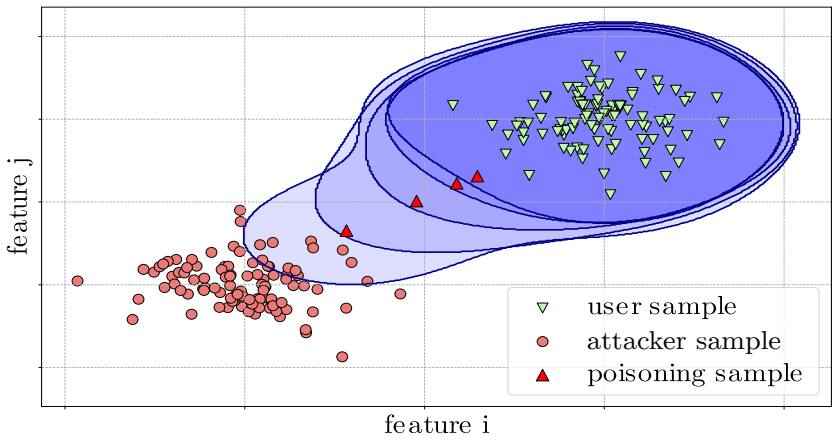


Figure 4.2: Changes in the classifier decision boundary (shaded areas) with the addition to the template of adversarial crafted samples. With enough poisoning samples, the classifier will recognize the adversary as the legitimate user.

4.4.2 Challenges

The challenges in designing the attack are based on the limited injection capabilities (which bring input uncertainty) and on the unknown user template and location of decision boundaries (which cause failures in the injection).

Input Uncertainty. Figure 4.2 shows a simplified 2-dimensional example of the user distribution. This distribution is actually in an n-dimensional feature space that consists of hundreds of dimensions. If the adversary had exact control over the location of samples in this space, he could simply craft and inject the poisoning samples that lie on the n-dimensional vector between the user and adversary distribution centroids. However, the adversary only controls the raw biometric measurements (input to the sensor) and furthermore needs to deal with the uncertainty that inevitably occurs with the biometric measurement (i.e., intra-user variation and input noise). Considering the fact that adversaries cannot alter the DNN, this further restricts the domain of available input manipulations.

Failed Injections. As adversaries only know one victim sample that is not part of the template, they need to overcome the uncertainty brought by the unknown location of the decision boundary. In fact, the adversary has no information about the training data and consequently the learned decision boundary (shaded

area in Figure 4.2). This means that the adversary does not know whether a crafted poisoning sample would be accepted by the system until he physically carries out the injection (i.e., crafting a poisoning sample, presenting the sample to the system). The procedure of presenting poisoning samples to the system is an expensive action for the attackers: they might raise suspicion and sample crafting might require resources. Additionally, the system could also block the template updating procedure after repeated failed attempts. Therefore, accounting for rejected attempts is fundamental in designing a strategy for the poisoning, as the risk of being detected grows with every attempt.

4.5 Attack Method

We adopt the idea of facial accessories from Sharif et al. [179] as a building block of our attack: we imagine the adversary can craft coloured glasses and wear them in order to carry out the attack. The glasses should be re-crafted at each injection step in order to achieve the correct location in features space shown in Figure 4.2. In our attack method, we have to re-define parts of the method in [179] in order to obtain a better resilience against the two challenges mentioned above: input uncertainty and failed injections.

4.5.1 Poisoning Sample Generation

Formulation. Formally, given a deep model f, a starting sample from the adversary $\vec{x}$, a target sample from the user $\vec{y}$ and their respective feature vectors computed by the model $f(\vec{x})$ and $f(\vec{y})$, we want to find a perturbation on the input features δ_x that minimizes the following:

$$\arg\min_{\delta_{\vec{x}}} \|f(\vec{x} + \delta_{\vec{x}}) - f(\vec{y})\|_p, \tag{4.1}$$

where $\|\cdot\|_p$ is a norm function. In order to limit the perturbations into the area of the glasses, we use a binary mask M_x that has the size of the input space and filters out the pixels that do not lie on the glasses frame:

$$\arg\min_{\delta_{\vec{x}}} \|f(\vec{x} \cdot (1 - M_x) + M_x \circ \delta_{\vec{x}}) - f(\vec{y})\|_p, \tag{4.2}$$

To account for the glasses smoothness, we look for perturbations that minimize total pixel variation of the added perturbation, computed as a function of each pixel value $p_{i,j}$:

$$V(\delta_{\vec{x}}) = \sum_{k,j}((p_{k,j} - p_{k+1,j})^2 + (p_{k,j} - p_{k,j+1})^2)^{\frac{1}{2}} \text{ [127].} \tag{4.3}$$

To account for perturbation printability, we account for the non-printability score (NPS) of the adversarial perturbation. Given a set of printable tuples, P, NPS is given by:

$$NPS(\delta_{\vec{x}}) = \sum_{p \in \delta_{\vec{x}}} \prod_{p' \in P} |p - p'| \text{ [179].} \tag{4.4}$$

In order to account for the limited injection capabilities and input noise, rather than optimizing individual samples separately we optimize all samples together in a batch. We then prioritize perturbations that minimize the standard deviation of the samples' distances to the target. To make the optimization faster, we introduce a weight that prioritizes samples that are further away from the distribution of samples' distance to the target. So for a set of samples from the adversary $\{\vec{x}_1, ..., \vec{x}_n\}$ and a perturbation $\delta_{\vec{x}}$ the distance d from the mean for sample i is:

$$d_i = |\|f(\vec{x}_i \cdot (1 - M_{x_i}) + M_{x_i} \circ \delta_{\vec{x}}) - f(\vec{y})\|_p - \mu|, \tag{4.5}$$

$$\mu = \frac{1}{n} \sum_{i \in n} \|f(\vec{x}_i \cdot (1 - M_{x_i}) + M_{x_i} \circ \delta_{\vec{x}}) - f(\vec{y})\|_p, \tag{4.6}$$

so that the optimization becomes:

$$\begin{aligned} \arg\min_{\delta_{\vec{x}}} \sum_{i \in n} d_i \|f(\vec{x_i} \cdot (1 - M_{x_i}) + M_{x_i} \circ \delta_{\vec{x}}) - f(\vec{y})\|_p \\ + NPS(\delta_{\vec{x}}) + V(\delta_{\vec{x}}). \end{aligned} \tag{4.7}$$

It should be noted that as M_{x_i} changes across different images (i) because of the different location of the glasses (depending on eyes landmarks), we refer to $M_{x_i} \circ \delta_{\vec{x}}$ as the operation that fills the positive elements of the mask with the current coloured glasses frame $\delta_{\vec{x}}$.

Optimization. To find the optimal perturbations, we combine gradient descent with a per-feature iterative approach. We initialize the glasses frame $\delta_{\vec{x}}^{(0)}$ at

Algorithm 2 - Poisoning Samples Generation: neural network f, batch of samples from the adversary $\{\vec{x}\}_i$, target victim sample $f(\vec{y})$, perturbation regularization λ, no. of features to update at each step m, $\{M_{\vec{x}}\}_i$ masks for glasses location.

INPUT: $f, \{\vec{x}\}_i, f(\vec{y}), \{M_{\vec{x}}\}_i, \lambda, m$

1: $\delta_{\vec{x}}^{(1)} =$ `random_init()` ▷ initialize random glasses
2: $j \leftarrow 1$
3: **for** $(j = 1, j < N, 1)$ **do**
4: $\quad \{\delta_{\vec{x}}\}_i =$ `back_prop`$(f, \{\vec{x}\}_i, \{M_{\vec{x}}\}_i, \delta_{\vec{x}}^{(j)}, f(\vec{x}_c))$
5: $\quad \Delta\delta_{\vec{x}}^{(j)} =$ `sel_top_f`$(\{\delta_{\vec{x}}\}_i, m)$
6: $\quad \delta_{\vec{x}}^{(j+1)} = \delta_{\vec{x}}^{(j)} + \lambda\Delta\delta_{\vec{x}}^{(j)}$.
7: **end for**
8: **for** $(j = N, j < 2N, 1)$ **do**
9: $\quad \{\delta_{\vec{x}}\}_i =$ `back_prop`$(f, \{\vec{x}\}_i, \{M_{\vec{x}}\}_i, \delta_{\vec{x}}^{(j)}, f(\vec{y}))$
10: $\quad \Delta\delta_{\vec{x}}^{(j)} =$ `sel_top_f`$(\{\delta_{\vec{x}}\}_i, m)$
11: $\quad \delta_{\vec{x}}^{(j+1)} = \delta_{\vec{x}}^{(j)} + \lambda\Delta\delta_{\vec{x}}^{(j)}$.
12: **end for**
13: **return** $\delta_{\vec{x}}^{(j)} \ \forall j \in [N, 2N]$ ▷ return intermediate results

random, avoiding too bright or too dark colours and use a Gaussian filter to reduce neighbouring pixels variability. At each step, we first back-propagate the gradient from the network. Then we weight the backpropagated gradient of sample x_i by d_i. At this point we choose the top m input features (based on the gradient value) which shift the current samples in the desired direction selecting them as those which minimize the total variability and non-printability of the glasses frame, i.e., $NPS(\delta_{\vec{x}_i}) + V(\delta_{\vec{x}_i})$. This way we obtain the changes to apply to the glasses at this step $\Delta\delta_{\vec{x}}^{(j)}$ which we apply to each sample in the batch:

$$\delta_{\vec{x}}^{(j+1)} = \delta_{\vec{x}}^{(j)} + \lambda\Delta\delta_{\vec{x}}^{(j)}. \tag{4.8}$$

We use a regularization parameter λ to control the magnitude of pixel changes, to avoid too large steps in the optimization which could lead to sub-optimal adversarial samples. We report the full procedure in Algorithm 2.

We perform the optimization twice. After the random initialization, we first optimize the glasses using the adversary's centroid in feature space x_c (Algorithm 2 Line 4) as target. In fact, we want to make sure that at the beginning of the generation of poisoning samples the adversaries wearing the glasses still maintain

Algorithm 3 - Poisoning Attack: neural network f, victim user template $Y = \{f(\vec{y})\}_i$, known victim sample $\vec{y} \notin Y$, adversary template $\{\vec{x}\}_i$, classifier C, perturbation delta θ.

INPUT: $f, \vec{y}, \{\vec{x}\}_i, \lambda, N, C, Y$

C is trained with Y
$\{M_{\vec{x}}\}_i \leftarrow$ `find_eyes_landmarks(`$\{\vec{x}\}_i$`)`
$\{\delta_{\vec{x}}\}^{(j)} \leftarrow$ `gen`$(f, \{\vec{x}\}_i, f(\vec{y}), \{M_{\vec{x}}\}_i, \lambda, m)$ ▷ Alg 1
IAR $\leftarrow$ `count`$(C.\text{predict}(\{\vec{x}\}_i) > C.\text{thresh})$
while IAR $< \theta_1$ **do**
    failures $\leftarrow 0$
    $j \leftarrow$ HEUR$(..)$ ▷ find perturbations with heuristic
    $\{\vec{x}^*\}_i \leftarrow \{\vec{x} \cdot (1 - M_x) + M_x \circ \delta_{\vec{x}}^{(j)}\}_i$ ▷ glasses
    IAR* $\leftarrow$ `count`$(C.\text{predict}(\{\vec{x}^*\}_i) > C.\text{thresh})$
    while IAR* $< \theta_2$ **do**
        failures $\leftarrow$ failures $+ 1$
        $j \leftarrow j + 1$ ▷ increase perturbations
        $\{\vec{x}^*\}_i \leftarrow \{\vec{x} \cdot (1 - M_x) + M_x \circ \delta_{\vec{x}}^{(j)}\}_i$
        IAR* $\leftarrow$ `count`$(C.\text{predict}(\{\vec{x}^*\}_i) > C.\text{thr})$
    end while
    $Y \leftarrow Y + \{f(\vec{x}_k^*)\}$ ▷ s.t. $C.\text{predict}(\vec{x}_k^*) > C.\text{thr}$
    IAR $\leftarrow$ `count`$(C.\text{predict}(\{\vec{x}\}_i) > C.\text{thr})$
end while
return `True`

their original appearance, i.e., the glasses do not affect the location of the attacker's samples in feature space. After the first optimization has ended, we perform a second optimization where we further modify the glasses so that they now change the adversary appearance onto the victim one, i.e., the glasses shift the location of the attacker's samples in feature space towards the victim. Algorithm 2 returns a list of *intermediate* glasses that correspond to the poisoning samples shown in Figure 4.2. In the next section we use these samples to construct the poisoning attack.

4.5.2 Poisoning Attack Injection

After using Algorithm 2, all the necessary intermediate poisoning samples are available for the adversary to use. However, adversaries need to decide at what point in the optimization process the glasses would lead to an accepted attempt. Here, we first explain how the poisoning works and then we show how the uncertainty on the decision boundaries can be overcome using population data.

Poisoning Algorithm. We report the full algorithm for the poisoning in Algorithm 3. The algorithm uses Algorithm 2 to obtain intermediate samples (Algorithm 3 Line 3). We use the impostor acceptance rate (IAR) as an indicator for injection (i.e., the proportion of attacker samples that are accepted by the system as legitimate). We consider that the adversary can successfully inject a sample when at least a θ_2 fraction of the attackers samples (as they are wearing the glasses, $\vec{x}^*$) are accepted by the system (Line 10). Whenever the adversary attempts to inject a sample, if less than θ_2 of his samples are accepted by the system we consider the attempt a failure. In this case, the adversary will increase the amount of perturbations on the glasses (move closer to the user's template, see Figure 4.2), and attempt again. In the cases where more than θ_2 samples are accepted, then we consider the attempt successful and inject one of these accepted samples into the current user template (chosen at random, Line 16). The algorithm stops when at least θ_1 fraction attacker samples while wearing no glasses ($\vec{x}$), is accepted by the system (Line 5).

Injection Heuristic. In order to minimize the number of attempted injections we develop a *heuristic* to estimate whether a crafted poisoning sample would be accepted given limited information about the training data and classifiers. The heuristic is based on the intuition that deep models distribute feature space evenly across different users, attempting to separate each user by a similar distance from the others. Therefore, adversaries can use population data to understand the dynamics of the decision boundaries. The heuristic is then based on two factors known by the adversary: (i) the number of perturbations applied to the input sample (j in Algorithm 2) and (ii) the L_2 distance between the poisoning sample and the single known user sample $\|f(\vec{x}^*) - f(\vec{y})\|_2$.

In practice, the adversary can run the attack for a set of users in the population for which he has knowledge of the template (i.e., excluding the actual victim). It should be noted that this does not assume white-box knowledge of the model but simply query access (free injection failures). This poses no significant challenge as user templates can be gathered through social media or just by using publicly

Table 4.2: Details of models and datasets used for the poisoning attack evaluation. The *Network Output* column reports the dimensionality and type of the layer we used to extract facial features. Accuracy refers to the accuracy computed on the labelled faces in the wild dataset [89] (LFW), and the figures are taken from the respective papers.

Model	*Dataset*	*no. of Identities*	*Network Output*	*Accuracy*
FaceNet [175]	VGGFace2	8,631	512 (embed)	99.65%
VGG16 [151]	VGGFace	2,622	4,096 (logits)	98.95%
ResNet-50 [32]	VGGFace2	8,631	2,048 (logits)	>99%

available datasets. In this process, he runs Algorithm 3 but replaces Line 7 with an iterative search over the intermediate samples generated by Algorithm 2 until he reaches θ_2 accepted samples. He then collects the number of perturbations applied so far and the L_2 distance to the known samples in the template. With this information, he can find a "sweet spot" in the two-dimensional space (i.e., the number of perturbations, distance to samples in the template) that indicates the likelihood of the current poisoning sample $\vec{x}^*$ of being accepted by the template update function.

Using the data gathered from the population, the adversary now has access to a good estimator of sample acceptance. In particular, the closer the current poisoning sample $\vec{x}^*$ is to the centre of the two-dimensional distribution, the most likely the sample will be accepted by an unseen template update function. The adversary therefore uses the heuristic to decide which $\vec{x}^*$ to inject and falls back to an iterative approach whenever the heuristic fails (as described in the previous paragraph).

4.6 Evaluation

In this section, we first describe the experiment, we then evaluate the attack, show its effect on the error rates and present the results for attack transferability across networks.

4.6.1 Experiment Design

Classifiers. we decide to consider three different classifiers: (i) `centroid` [21], (ii) `maximum` [21] and (iii) one-class Support Vector Machine (`SVM`). The `centroid`

Table 4.3: Performance of the face-recognition models in terms of EER. The classifiers thresholds are set at the EER on a separate subset of 100 users that are not used in the attack evaluation.

	`centroid`		`maximum`		`SVM`	
Model	EER	*Threshold*	EER	*Threshold*	EER	*Threshold*
FaceNet	0.9%	0.959	1.3%	0.901	1.0%	0.121
VGG16	2.2%	0.590	3.0%	0.564	1.9%	0.041
ResNet-50	1.3%	0.923	1.9%	0.876	1.7%	0.098

classifier computes the distance between the new sample and the centroid of the training data. The `maximum` classifier computes the distance between the new sample and the closest training sample. Both classifiers use L_2 distance to compute distances between samples; we tried using L_1 and we obtained similar results. All three classifiers perform an authentication decision comparing the computed distance to a pre-set threshold. We decide to use a linear kernel for `SVM` as this was the best performing one in terms of recognition performance. To account for the age of samples in the template, i.e., older samples might be considered less relevant than recent samples, we include two different weighting schemes: (i) *flat*, (ii) *sigmoid*. In the flat weighting schemes, each sample in the template is considered equally important in the decision, while in the sigmoid scheme the weight of each sample w_i is computed with a sigmoid function prioritizing recent samples: $w_i = \frac{1}{(1+e^{-x_i})}$, where $x_i \in [-5, 5]$ indicates how recent the sample is (the lower the older the sample is). Intuitively, prioritizing recent samples in the decision will make the poisoning attack faster.

Update Policy. For the template update procedure, we choose to use self-update with *infinite window update* policy [22, 100]. In other words, whenever a new sample is accepted by the classifier it is subsequently added to the training set. This choice is similar to the unsupervised update process used by Apple's FaceID or Siri, where newly obtained samples are added to the template if the sample quality (i.e., low level of noise) is sufficient [11, 187].

Considered DNN. We consider three different state-of-the-art convolutional network architectures: FaceNet [175], the VGG16 Face descriptor [151] and ResNet-50 [32]. FaceNet uses the Inception-ResNet-v1 architecture from [197], and is trained with triplet loss to produce 512-dimensional embeddings. The VGG-Face descriptor is based on the VGG16 architecture [186] and has been trained on the VGGFace dataset, which contains 2,622 identities. Both ResNet-50 and FaceNet are trained on the newer VGGFace2 [32] dataset which contains 8,631 different identities with on average over 340 images per individual. For VGG16 and ResNet-50 we use the ℓ_2-normalized *logits* layer as output rather than training the models with triplet loss. The details of the models and datasets are reported in Table 4.2.

Input Preprocessing. All images are first aligned using the landmark based model in [238]. All the remaining preprocessing follows the models' guidelines: FaceNet inputs are pre-whitened, VGG16 and ResNet-50 inputs use fixed image standardization. We don't use any random cropping or flipping for the images. The images dimensions in input are 160x160 for FaceNet and 224x224 for the remaining models. When testing images across different models we resize them to the correct size using bilinear interpolation.

Experiment Description. All the experiments make use of the testing part of the VGGFace2 dataset, which contains 500 separate users. Before we evaluate the attack's effectiveness, we first set up the system as follows. First, we randomly choose 100 users to compute the classifiers' thresholds. For each user we use 10 samples for training and test against all the remaining user samples and a randomly selected sample of different users (>100,000 authentication attempts are used to find the EER threshold). The results are reported in Table 4.3. Afterwards, we split the 400 remaining users into two equally sized groups and we treat the two groups as adversaries and victims separately. We randomly choose 1,000 attacker-victim pairs for the evaluation. These chosen pairs are the same across each model considered.

Attack Implementation. We always use Algorithm 3 for the evaluation. We define the attack as successful when at least half of the adversary template is accepted by the classifier and we use the same to compute the injection success rate, i.e.,

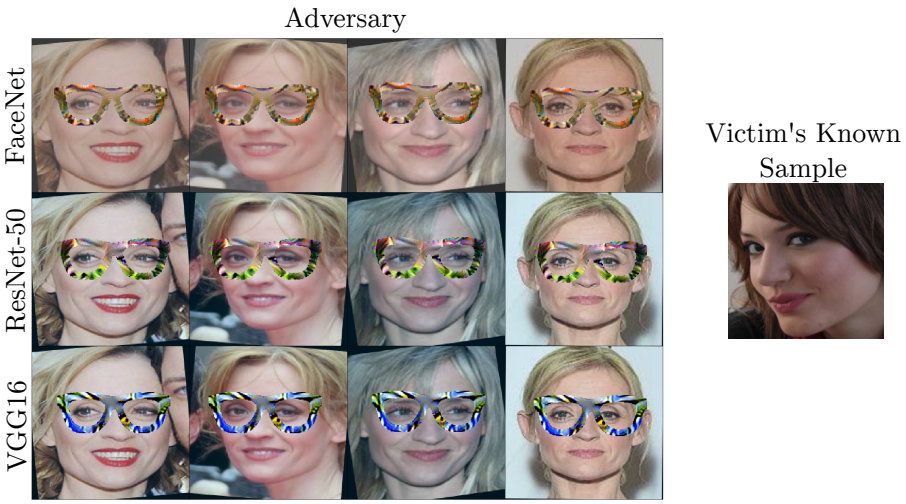


Figure 4.3: Poisoning samples for an adversary-victim pair. The figure shows how the *same* glasses can be applied to the user across different intra-user variations (e.g., pose).

$\theta_1 = \theta_2 = 0.5$. As mentioned in Section 4.3, the update and authentication thresholds are identical (found in Table 4.3), so that when a sample passes authentication it is automatically added to the template. The heuristic of Section 4.5.2 is computed using only 10 randomly chosen attacker-victim pairs. The glasses are positioned based on the eyes location (computed using a face landmark extractor) and occupy on average 8.59% of the total pixels in input. Figure 4.3 shows a set of poisoning samples for an attacker and the relative victim. We only use front facing images from the adversary (up to 50 images), as adversaries have control over their pose as they carry out the attack, and do not need to optimize for their own intra-user variation when attempting the attack. At the moment of injection, the injected sample is chosen at random from the currently accepted samples. We use $\|\cdot\|_2$ for the optimization of Equation 4.7 as L_2 is used by the classifiers for the decision. The parameter λ is set to 4, which indicates that the maximum change for a pixel in an iteration is 4 out of a range of $[0, 255]$.

4.6.2 Attack Success and Effect on Error Rates

Since the attack is generally successful in the long-term (as more malicious samples enter the template), we focus the analysis on the number of injection attempts required and the effect on the recognition rates as the poisoning progresses. For a

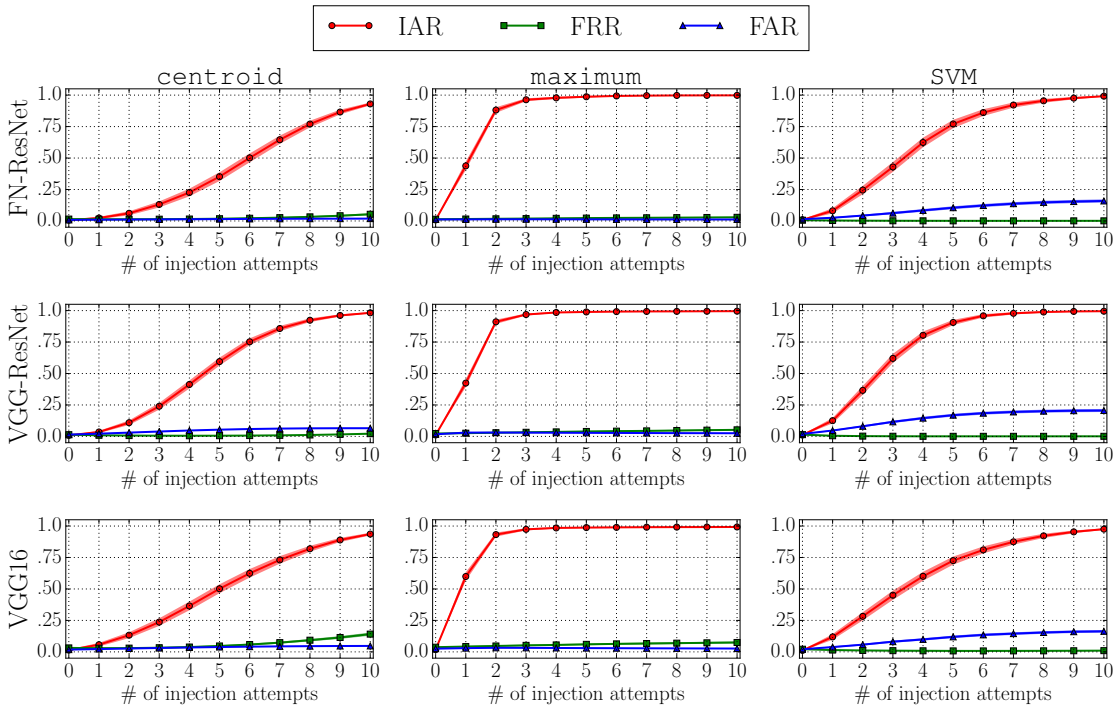


Figure 4.4: IAR, FAR and FRR changes over the course of the poisoning attack. Shaded areas show the standard deviation over each adversary-victim pair in the dataset. The shown results are for the `centroid` classifier with flat weights.

Table 4.4: Attack success rates in percentages (%) for each considered model and classifier. Success is defined as >50% IAR after i samples injections (either one, three or ten). Each figure is calculated on the (same) 1,000 randomly chosen pairs of attacker-victim.

	`centroid`						`maximum`						`SVM`					
	flat			*sigmoid*			*flat*			*sigmoid*			*flat*			*sigmoid*		
Model	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}
FN-ResNet	1	9	77	6	55	95	40	93	95	86	97	97	6	35	81	10	49	73
VGG16	4	19	85	12	66	100	61	98	100	92	100	100	9	42	94	12	50	71
VGG-ResNet	2	20	97	10	86	99	38	95	96	98	99	99	9	54	87	13	63	80

report on the attack success rate as defined in Section 4.5, see Table 4.4. We found that the heuristic presented in Section 4.5.2 can compensate for the limited decision boundaries knowledge of the attacker: every time the attacker used the heuristic to craft an intermediate sample to inject, the sample was accepted in more than 79% of cases (79%, 83% and 82% for FaceNet, VGG16 and ResNet-50, respectively). The results show the performance of the attacker *including* failed injections.

We monitor three different rates: (i) false accept rate, (ii) false reject rate,

(iii) impostor accept rate. For an attacker-victim pair, FAR is computed as the proportion of samples belonging to "other users" (users that are not the attacker or the victim) that are accepted by the classifier. Similarly, FRR is the proportion of victim's samples that are rejected by the classifier, excluding the 10 samples in the training set. IAR is defined as the proportion of samples belonging to the adversary that is accepted by the classifier without wearing glasses. Ideally, the attack should make sure that IAR increases as fast as possible while not changing FRR and FAR. Increases in FRR are suspicious because the legitimate user cannot authenticate smoothly anymore (lots of rejections). Consequently, users might contact the system administrator to report the problem or switch to more convenient authentication factors (when other options are available). Increases in FAR are less suspicious, depending on how often other users attempt impersonation attacks with their own biometric trait.

Figure 4.4 shows how the above-mentioned rates vary during the poisoning attack, for the flat weighting scheme. The three models respond similarly to the poisoning, while different classifiers show different error rate changes. Comparing the classifiers with each other shows that `maximum` classifier is particularly vulnerable to this attack: two injected samples lead to IAR >90% on average, only with marginal increments in the success rate for additional injections.

Overall the changes in FRR and FAR are minimal. A small FAR increase can be seen for the `SVM` classifier. This might be due to the linear kernel function used for the classifier, which cannot fit the training data well when poisoning samples are added and therefore includes wider areas of the feature space. The minimal changes in FRR are in contrast with what has been shown in previous work [21], where FRR (referred to as genuine accept rate) quickly increased to values above 40% after 5 injected samples and over 90% over 10 injected samples. This is mainly due to a combination of the initial enrolment data size and the type of window used for the self-update procedure. As noted previously [100], fewer samples in the training data lead to faster poisoning (ten in our case, five in [21]) and the choice of discarding older samples [21] with the finite window policy leads to the

Table 4.5: Poisoning attack success rates (%) in the transferability scenario across different models. The reported figures are the success rates of the attack uniquely using information from the source model (source is on the rows, target is on the columns). Bold values refer to same-model success rates (also found in Table 4.4). The rates correspond to the success at the 10^{th} injection attempt. The heuristic is always fit on the target system (i.e., both target model and classifier).

	Target Model											
	FaceNet			FaceNet-CASIA			ResNet-50			VGG16		
Source Model	centroid	maximum	SVM	centroid	maximum	SVM	centroid	maximum	SVM	centroid	maximum	SVM
FaceNet	**77.2**	**95**	**81**	22	27	24	12	12	14	9	11	9
FaceNet-CASIA	9	14	10	**85**	**99**	**88**	11	11	12	9	10	9
ResNet-50	13	20	14	23	30	27	**97**	**96**	**87**	13	16	13
VGG16	5	8	6	11	16	16	14	17	18	**85**	**100**	**94**

user being removed from the template. As mentioned in Section 4.6.1, it should be noted that modern systems tend to reflect the design choices in this work, both for speaker and face recognition [187, 11]). A more detailed and long-term analysis is required to investigate the trade-offs between these choices and the performance of the system in this scenario, which we leave for future work.

4.6.3 Transferability

We analyse the transferability of the attack across the models, reporting results in Table 4.5. In addition to the previously considered models, we add FaceNet-CASIA which has the same architecture as FaceNet, but is trained on the CASIA Webface dataset [235] instead of the VGGFace2 dataset. This allows us to investigate the influence of the training data on the transferability. Each attack repetition uses the source model as a surrogate: using the source to compute the poisoning samples (Algorithm 3 Line 3), but using the target model to compute the sample acceptances (Algorithm 3 Line 4, 9 and 14).

Table 4.5 reports the success rate, defined as >50% IAR after 10 injected samples, for source-target model pairs. We found that the low success rate is mostly caused by the fact that the optimization of Algorithm 2 did not lead to

accepted samples (85% of attacker-victim pairs) rather than not enough samples being accepted after 10 injections (2%). Comparing the white-box results with the across models ones, we see that in most cases there is a tenfold decrement in the success rates, which could still make the attack viable in black-box scenarios. We don't find particular differences across classifiers, with the `maximum` being slightly weaker against this attack. These findings are in line Sharif et al. [180] who report similar transferability for dodging attacks across different networks.

Table 4.5 does not show any particularly evident trend in the transferability across architectures, showing that some transferability applies for each pair of models. Additionally, while it would seem that higher EER would lead to higher chance of attacks, we find that baseline EER and success rate are only weakly correlated (r=-0.18). This suggests that transferability properties rely on less intuitive combinations of both training datasets and architectures. It should be noted that the success rate of attacks across different models could be improved, e.g., by using optimizations that are not tailored to an individual networks [180] or by obtaining better approximations of the user template.

4.7 Poisoning Countermeasures

Here, we discuss possible countermeasures and propose and evaluate a new detection method based on angular similarity.

4.7.1 Detecting Adversarial Samples

One indirect approach to limit the feasibility of a poisoning attack is to increase the difficulty of crafting adversarial samples with the required properties. As an example, in adversarial training [148, 38, 198], the training data includes adversarial samples specifically crafted to increase the model resilience against them at test time. With gradient masking, the network gradient is hidden during training, making the network robust to small perturbations in the input [149, 123, 79]. In [72] the approach aims to approximate the functions learnt by neural networks with abstract primitives, where security guarantees can be certified with sounder methods.

Similarly, when an attack vector is identified, such as wearing coloured glasses, measures that are specific to the detection of such attack can be implemented (e.g., detecting whether the pixels in input present anomalous sharpness).

4.7.2 Template Anomalies

Analysing anomalies in the template is another approach to stop poisoning attacks, where the goal is to detect whether sets of samples that are added to the template are anomalous. Biggio et al. [23] propose a technique named *template sanitization* which consists in defining a sanitization hypersphere around the current template distribution. Whenever series of k consecutive updates quickly drift the current user centroid outside of the current hypersphere, then such updates are discarded and the previous centroid is restored. The underlying assumption is that genuine template updates exhibit a less biased and more random behaviour. A theoretical analysis of the security of online template updates is given by Kloft et al. [100], where a protection mechanism based on monitoring the false positive rate is introduced. Such method suggests that the system could monitor the number of false rejections using a *hold-out* dataset to check whether these exceed a fixed thresholds. In other words, when the users legitimate samples start to be rejected as the poisoning progresses, updates might be discarded and the centroid reset to a previous "safe" state.

Detection Trade-off. Both works [100, 23] show that such countermeasures can successfully thwart the progress of a poisoning attack. The underlying assumptions of the methods is that poisoning samples present a behaviour that is not shown by legitimate updates: either quickly drift the centroid outside of a set hypersphere [23] or increase the number of false rejections [100]. However, neither study offers an analysis of the trade-off between poisoning detection and legitimate template update acceptance i.e., legitimate samples might show the behaviour that is labelled as anomalous.

For [100], we have shown that false rejections in our scenario present only marginal increases over the course of the poisoning, and the method additionally assumes the availability of an hold-out dataset for the user, which is not available

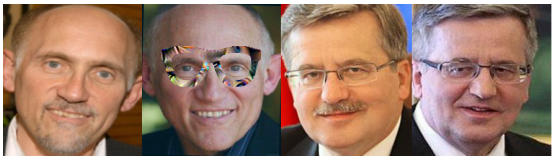
Table 4.6: Percentages (%) of successful poisoning attack detections. The top row indicates which model is considered for the evaluation: $a \rightarrow b$ indicates that the detection is trained on a and applied to b. The values are computed on the `centroid` with flat weights. For same-model cases we report EER. For across-model cases we report the FAR and FRR for the detection on the target model obtained after choosing the threshold at EER on the source model. The "All" row is computed using all the updates from each other factor.

	Same Model			*Across Models*							
	FaceNet	ResNet-50	VGG16	*FaceNet → ResNet-50*		*ResNet-50 → FaceNet*		*FaceNet → VGG16*		*VGG16 → ResNet-50*	
Factor	EER	EER	EER	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
Age	6.8	7.1	4.5	7.4	6.8	6.6	7.2	7.0	2.2	4.8	12.0
Glasses	14.3	14.6	11.3	14.5	14.6	14.3	14.3	18.7	5.8	7.7	27.0
Facial hair	7.6	7.1	4.8	6.8	7.5	8.0	7.2	6.9	2.6	4.6	12.6
Pose	6.3	6.2	4.0	6.2	6.1	6.3	6.3	6.5	1.9	4.0	11.1
All	**7.7**	**7.9**	**5.4**	**7.7**	**7.6**	**7.9**	**7.9**	**7.7**	**2.6**	**5.4**	**13.9**

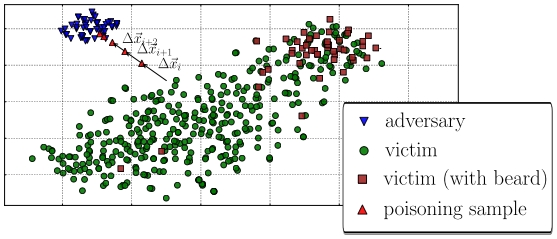
in a biometric authentication scenario. For [23], we notice that several intra-user variation factors that might affect the location of the sample in feature space in a predictable and consistent way, such as: facial hair, pose, age. Figure 4.5b shows an example of such behaviour: samples where the user has facial hair (i.e., a moustache, see Figure 4.5a) cluster together in a specific region of feature space. Such a region may be separate from the known user template, meaning that limiting the template updates to a set hypersphere would stop legitimate updates in this scenario [23]. In the following, we present a new poisoning countermeasure and evaluate its detection trade-off with legitimate samples.

4.7.3 Angular Similarity Detection

We propose a new detection technique, based on the rationale that poisoning samples will all lie in a predetermined direction in feature space with respect to the current legitimate user centroid. The direction is determined based on the location of the



(a) from the left, adversary, adversary during the poisoning attack, victim with facial hair and without facial hair.



(b) embedding space visualized.

Figure 4.5: Poisoning detection overview. The figure shows victim's, adversary's and poisoning samples. Samples' locations are computed by chaining principal component analysis and t-SNE [210] reduction. Figure 4.5b that consecutive poisoning samples have consistent angular similarity with one another when compared with the victim centroid. Similarly, other intra-user variation factors (e.g., beards) might reside in a specific direction when compared with the centre of the victim distribution.

attacker samples. This is visualized in Figure 4.5b. Given the user's current centroid $\vec{x}_c$ and a set of template updates $\{\vec{x}_i, \vec{x}_{i+1}, ..., \vec{x}_{i+n}\}$, which we refer to as an *update sequence*, we compute the direction of the update at time i: $\Delta\vec{x}_i = \vec{x}_c - \vec{x}_i$, and we can obtain the directions at each step as $\{\Delta\vec{x}_i, ..., \Delta\vec{x}_{i+n}\}$. We then compute the angular similarity for pairs of consecutive updates with the cosine similarity as:

$$\cos\theta_{i+1} = \frac{\Delta\vec{x}_i \cdot \Delta\vec{x}_{i+1}}{\|\Delta\vec{x}_i\|\|\Delta\vec{x}_{i+1}\|}. \tag{4.9}$$

The underlying intuition is that $\cos\theta_i$ will be higher for pairs of poisoning samples compared to legitimate updates because the attack needs to shift user centroid towards the adversary's, which lies in a specific pre-defined direction, see Figure 4.5b for reference.

4.7.4 Detection Evaluation

Setup. We use the Google Vision API[2] to extract attributes for four intra-user variation factors: pose, facial hair and (sun)glasses. Since the API does not return a value for age, we use the age-estimator in [233]. For pose we only consider the *pan* angle as a factor, that is the horizontal angle of the face (a pan of 90 corresponds to a face looking sideways). For age we group samples into subgroups of samples with the same age, using ranges of three years for each subgroup. This way, for each

[2]https://cloud.google.com/vision/

of the 1,000 attacker-victim pairs considered in the previous section, we consider sequences of updates using the samples in the victim's test set as follows:

- **age**: update sequences are in 3-years age span;
- **pose**: we create two update sequences by choosing samples with pan angle ≥ 30 or samples with pan angle ≤ 30;
- **facial hair**: we create an update sequence where the user's samples have facial hair;
- **glasses**: we create an update sequence where the user is wearing glasses or sunglasses;
- **poisoning**: we consider sequences of poisoning samples obtained from carrying out the attack of Section 4.6.

We choose a random order for the samples in these sequences, excluding the poisoning one, and we evaluate the angular similarity of consecutive samples with Equation 4.9. We consider 1,132,994 legitimate updates, distributed across the attacker-victim pairs. By comparing legitimate and poisoning sequences we choose a threshold for the detection of too similar subsequent updates. We then show how IAR changes during the poisoning attack with detection in place. We present the results of our analysis on the `centroid` classifier with flat weights; other classifiers performed similarly.

Results. We report in Table 4.6 the results for the poisoning detection for each of the considered factors. Each row reports the results for a specific factor, that is, all pairs in the update sequences of that factor are evaluated against the pairs in the poisoning sequences; the last *all* row reports the result of using all the measurements together. We see that the detection performance in terms of EER lies around 6-7% for most factors and models (VGG16 model performs better), with the exception of glasses, which create an increase in the EER up to 14%. We find that glasses, in particular sunglasses, create a very predictable and consistent effect on the location of samples in feature space, leading to greater angular similarity

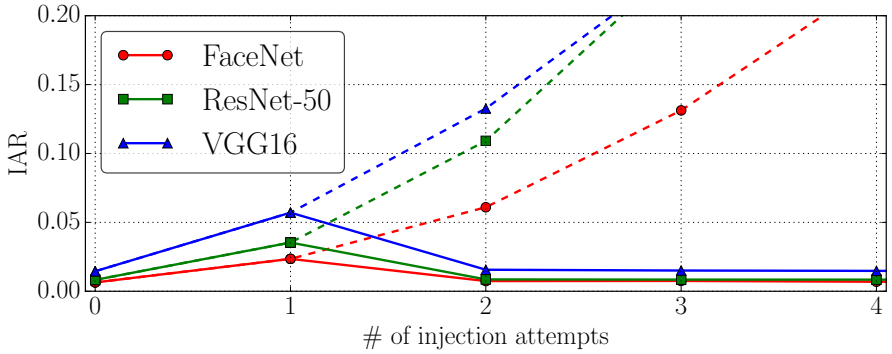


Figure 4.6: Poisoning attack success rate with poisoning detection in place. The plot shows the IAR comparing when the poisoning detection is in place (solid lines) and when it is not (dashed lines), for the `centroid` classifier with flat weights and the different models (same-model scenarios). The angular similarity threshold is set at EER on all the samples in the considered update sequences.

of consecutive updates. This can be explained with sunglasses being less frequent in the training data compared to the other factors such as pose and age, that the network learns how to ignore during optimization.

We test the transferability of the detection by applying the threshold learned on one model to different models. The last four columns of Table 4.6 show that this is the case: the obtained FAR and FRR on the target model closely resemble the EER obtained in the source model, excluding VGG16 where FRR is slightly higher.

We report in Figure 4.6 the IAR over the course of the poisoning with and without detection in place, setting the threshold at EER for all the considered update sequences. We find that >99% of attacks are detected at the second injection attempt, when the IAR is still low for the attacker to authenticate consistently (<6%). When consecutive poisoning samples are detected, the system removes the malicious samples from the template, resetting the baseline IAR.

Considerations. Note that the defence proposed in this section could be overcome by an adaptive attacker (i.e., aware of the defence) who chooses to spread the malicious template updates across a larger number of samples. In this case, the adversary can present consecutive samples which move in a "zig-zag" fashion in the embedding space, thus not presenting the angular similarity shown by the vanilla poisoning sequence discussed above. A more sophisticated defence could additionally

combine the centroid drifts discussed in Section 4.7 with the angular similarity detection discussed here. In general, defending from such attack will always involve a trade-off between allowing legitimate drifts and detecting malicious ones, and could be further measured in terms of number of required samples for a successful attack.

4.8 Discussion

In this section we discuss future research directions, including how to enhance the attack practicality and the attack transferability in black-box scenarios.

Attack Practicality. Using coloured glasses has been shown to be a practical way of fooling face authentication [179]. However our poisoning attack requires greater control over the embedding space compared to simple impersonation as the exact distance between samples in embedding space influences the attack outcome. Adversaries could improve such control by accounting for as many factors of intra-user variation as possible during the optimization (Algorithm 2), such as pose, lighting, facial accessories and backgrounds. In other words, adversaries would build a training dataset which captures all factors of variations of their own face appearance and optimize the glasses by using all the samples in the dataset. Correctly capturing these factors would lead to better handling of the input uncertainty at the time of injection and consequently to a better control over the injected samples. Adversaries could also obtain "offline" insights on whether a pair of glasses generalizes well against the various factors that cause input uncertainty as follows: split the above mentioned dataset into a training and testing part, use the training part for Algorithm 2 but using the testing part to simulate the actual injection in Algorithm 3. This would allow the adversary to refine and evaluate the poisoning samples before the time of attack to minimize the risk of failed attempts.

Improving Transferability. We found that our attack weakly transfers to different networks in the black-box case, which could limit the effectiveness of the attack in practice. Recently, a number of works have looked at the transferability of adversarial examples, providing insights on how to improve the generalization of such examples

to unseen models. One way to increase transferability is re-defining the optimization so that the adversarial perturbations can be applied across the entire distribution of legitimate input [242, 228, 240]. A straightforward method to obtain this is to use the *Expectation over Transformation* method (EOT) introduced in [13]. EOT has been used successfully in order to generate adversarial examples that are robust to real-world environmental changes. It has also been shown that transferability is affected by the local smoothness of the loss function used for computing the perturbations [226]. Enforcing such smoothness on the surrogate model during training leads to adversarial perturbations that better generalize to unseen models.

Limitations. Note that, in Section 4.3 we designed an authentication systems to the best of our knowledge using real-world biometric authentication systems as a reference, mainly Apple's FaceID and its voice counterpart Siri. To be able to evaluate such a system, we made a number of assumptions in Section 4.3 and Section 4.6.1 which condition the realizability of an attack in practice. These include: (i) the choice of models (DNN plus classifier), (ii) the update policy (i.e., infinite window update policy). Under different system setups adversaries might have to adjust the methods proposed here. In particular, (ii) allows attackers to obtain the inconspicuous "backdoor" effect: both adversary and victim authenticate successfully. On the other hand, using an update policy which removes old samples from the template would add an obstacle to such attack, as removing old samples might decrease the adversary's IAR or increase the victim's FRR.

4.9 Summary

In this chapter, we have presented a template poisoning attack that exploits the template update procedure to place a biometric backdoor, which grants inconspicuous long-term access to the biometric system. In view of the considerations about attack realizability in Section 1.1, we designed the attack to cope with limited knowledge and limited injection capabilities, reducing the number of assumptions made in prior work. The results show that a successful attacks can be obtained even in black-box models thanks to the principle of transferability of adversarial examples.

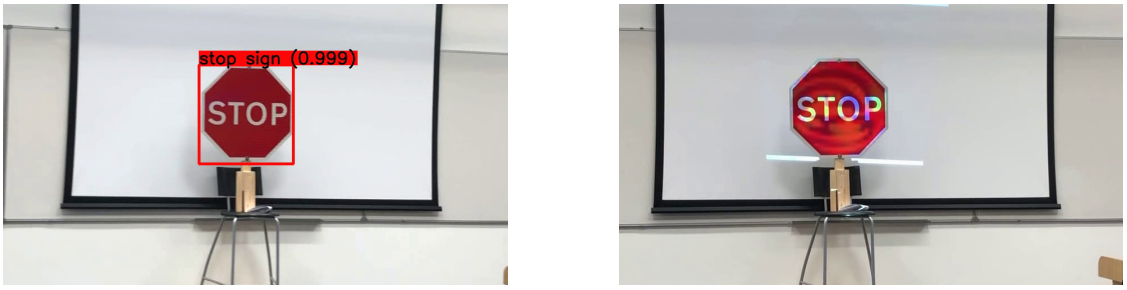
These results highlight the importance of identifying possible attack vectors for a system. For a biometric system, the security analysis should always include each way an adversary can alter the data the system relies on, such as biometric templates. In fact, simply protecting direct access to such data (e.g., with secure enclaves) might not always be sufficient if there are other measures in place that allow an adversary to edit the data, such as template updates. These results suggest that increased attention should be given to the update procedure, as this represents an opportunity for attackers to compromise the authentication system. While this chapter used a digital attack with self-imposed constraints, the following two chapters will investigate attacks which target the physical domain, moving rightwards on the Realizability scale of Figure 1.1.

5

Short-Lived Adversarial Perturbations for Physical Adversarial Examples

Contents

5.1 Introduction 76
5.2 Background 78
5.2.1 Projector technology 78
5.2.2 Detection of Physical Adversarial Examples 79
5.3 Threat Model 80
5.4 Method 81
5.4.1 Modelling projectable perturbations 81
5.4.2 AE Generation 85
5.4.3 Training Data Augmentation 87
5.4.4 Remarks 89
5.5 Evaluation 90
5.5.1 Experimental Setup 90
5.5.2 Indoor Results 94
5.5.3 Road Driving Test 96
5.5.4 Defences 97
5.5.5 Attack Transferability 99
5.6 Discussion 101
5.7 Summary 103



(a) Non adversarial scenario. **(b)** Adversarial projection.

Figure 5.1: Short lived adversarial perturbations attack visualized. A projector shines a specific pattern on the stop sign causing an object detector (Yolov3 in this picture) to misdetect the object.

5.1 Introduction

In this chapter, we further explore DNN's vulnerability to adversarial examples, but we develop the attack further to the point of physical world realizability. Thanks to the recent advances in computational capabilities and machine learning algorithms, DNN are used across a variety of vision tasks beside biometric authentication as seen in the previous chapter. In particular, DNNs are increasingly used in security- or safety-critical scenarios, such as autonomous driving, which is the focus in this chapter. This vulnerability raises important research topics: increasing the understanding of the AE creation process, as well as devising methods to make existing networks more robust to adversarial samples.

As mentioned previously in Section 1.1, we consider an adversary bounded by a physical realizability constraint: meaning that its AE must survive in the real-world environmental conditions, such as varied viewing distances or angles. The main vector to realize real-world AE has been the printing of patches (e.g. as stickers or glasses in the case of face recognition), or replacing an entire object by overlaying the object with a printed version of it with subtle changes. While adversarial patches [86, 30, 67] have been the most explored way to alter a scene in an adversarial way, patches have limitations. Firstly they generate highly salient areas in the network inputs, which makes them detectable [42]. Secondly, these patches are static, so they cannot be changed: in the autonomous driving scenario, sticking patches on a traffic sign leads to continuous misdetection of such signs,

which is equivalent to removing the sign from the road or covering it, an attack that does not require deep technical knowledge.

In this chapter, we explore a new attack vector that uses a common RGB light projector to alter a camera-captured scene in order to achieve *Short-Lived Adversarial Perturbations* (SLAP), a novel AE approach that allows adversaries to realize robust, dynamic real-world AE from a distance; an example is in Figure 5.1. SLAP-generated AE provide the attacker with multiple benefits over existing patch-based methods, in particular giving fine-grained control over the timing of attacks, allowing them to become *short-lived*. We implement and carry out such attack in practice on a variety of vision models, including object detectors and traffic signs classifiers. We further explore how to defend against such attack, showing that SLAP bypasses previously proposed defences tailored specifically to real-world AE detection, and finding that adversarial learning seems to be a promising way to defend against such attack, although with an accuracy trade-off in legitimate scenarios.

Contributions.

- We propose SLAP, a novel attack vector for the realizability of AE in the physical world by using an RGB projector. This technique gives the attacker new capabilities compared to existing approaches, including short-livedness and undetectability.
- We propose a method to craft robust AE designed for use with a projector. The method models a three-way additive relationship between a surface, a projection and the camera-perceived image. We enhance the robustness of the attacks by systematically identifying and accounting for varying environmental conditions during the optimization process.
- We evaluate the SLAP attack on two different computer vision tasks related to road safety for autonomous driving: (i) object detection and (ii) traffic sign recognition. We conduct an extensive empirical evaluation, including in- and outdoors, showing that under favourable lighting conditions the attack leads to the target object being undetected.

- We evaluate countermeasures. We firstly show that SLAP AE bypass locality-based detection measures such as SentiNet [42], which is tailored for the detection of physical AE. We then show that an adaptive defender using adversarial learning can thwart most of the attacks.

The experiments code and data have been validated by the USENIX Security '21 artifacts review process and are available online.[1]

5.2 Background

We start by introducing the necessary background on LCD projectors and in physical adversarial examples detection.

5.2.1 Projector technology

A common liquid crystal display (LCD) projector works by sending light through a series of dichroic filters in order to form the red, green and blue components of the projected images. As the light passes through, individual pixels may be opened or closed to allow the light to pass, creating a wide range of colours, depicted in Figure 5.2. The total amount of light that projectors emit (measured in lumens), as well as the amount of light per area (measured in lux) is an important factor for determining the image quality, with stronger output leading to more accurate images in a range of conditions. Common office projectors are in the range of 2,000-3,000 lumens of emitted light, while the higher-end projectors can achieve up to tens of thousands of lumens (e.g., the projectors used during the London 2012 Olympics [29]). As lumens only measure the total quantity of visible light emitted from the projector, the current ambient light perceived on the projection surface has an important role in determining the formed image contrast and colour quality. The brighter the ambient light, the less visible will the image formed by a projector be due to weaker contrast and narrower range of colours.

As an example, a 2,000 ANSI lumens projector can emit enough light to obtain a light intensity of 2,000 lux on a square meter area (measured for white light [191]).

[1] `https://github.com/ssloxford/short-lived-adversarial-perturbations`

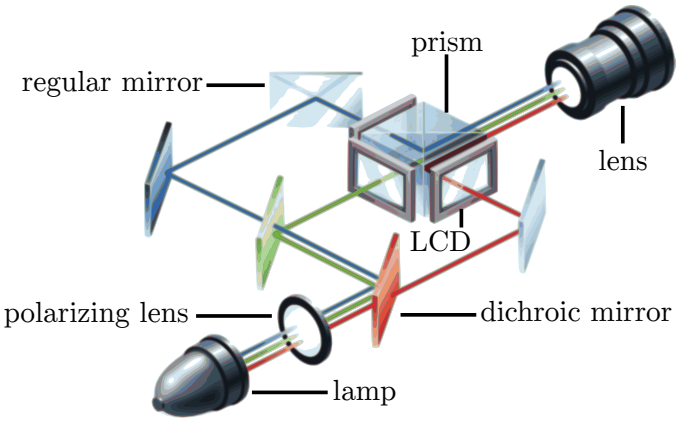


Figure 5.2: Depiction of an LCD projector inner workings. A lamp shines a light through a polarizing lens and then through a series of dichroic filters that separates light to three polysilicon mirrors – one each for the red, green and blue components of the video signal. As polarized light passes through the mirrors, individual pixels can be opened or close, allowing for an RGB image to be produced in output.

Such a projector would reproduce an image in an office quite well (ambient ~500 lux), but could hardly make the image visible if it was placed outside in a sunny day (~18,000 lux). Additionally, projectors are generally used and tested while projecting on a (white) projection screen, which are designed to optimize the resulting image quality. When projecting on different materials and non-white surfaces, the resulting image will vary greatly given that light propagation significantly changes depending on the material in use and the background colour. In Section 5.4.1 we explain how we model such changes in an empirical way that accounts for many variability factors.

5.2.2 Detection of Physical Adversarial Examples

Differently from a digital scenario, where input changes are simply limited by ℓ_p-norms, the realization of physically robust AE is more constrained. Adversarial patches are one technique for physical AE, however, they have drawbacks which enable their detection. In fact, Chou et al. [42] exploited the locality of adversarial patches to create an AE-detection method named SentiNet, which detects physical AE leveraging the fact that adversarial patches generate localized areas of high saliency in input, as shown in Figure 5.3. These highly salient areas successfully capture the adversarial patch in input, and therefore can be used for the detection of an AE by using the fact that such salient areas will cause misclassifications



Figure 5.3: Example of an adversarial patch attack [80]. The network has been compromised and reacts to the sunflower being placed in the input by misclassifying the stop sign. SentiNet [42] leverages the locality of the patch to detect regions with high saliency, and can therefore detect the attack. The figure is taken from Figure 5 in [42].

when overlaid onto other benign images. For example, Figure 5.3 shows that an adversarial patch shaped as a flower will cause the stop sign to be misclassified as a warning sign. The same flower patch can be applied to different images and will also cause misclassifications in other classes, which is an unusual behaviour which can be detected. SentiNet can capture this behaviour just by looking at the saliency masks of benign images, and fitting a curve to the accepted behaviour range, rather than fitting a binary model for the detection. This way SentiNet can adapt for unseen attacks and therefore claims to generalize to different attack methods. Further details can be found in Appendix A.3 and in the original paper [42]. In this chapter, we show how AE generated with SLAP can bypass such detection.

5.3 Threat Model

We focus on an autonomous driving scenario, where cameras are placed in vehicles and the vehicle makes decisions based on the cameras' inputs. The vehicle uses camera(s) to detect and track the objects in the scene, including traffic signs.

Goal. The adversaries' goal is to cause a stop sign to be undetected by the neural networks processing the camera feeds within the car, which will cause vehicles approaching the stop sign to ignore them, potentially leading to accidents and dangerous situations. The adversary may want to target specific vehicles, therefore

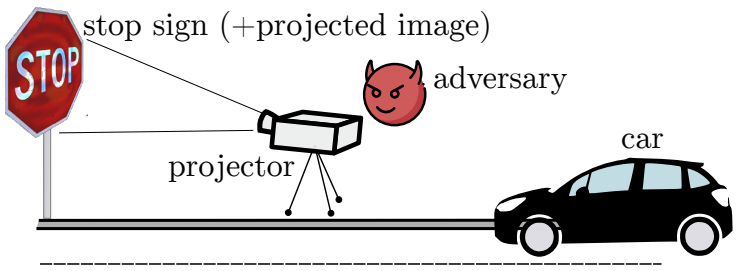


Figure 5.4: Attack scenario. An adversary points a projector at a stop sign and controls the projection in order to cause the sign to be undetected by an approaching vehicle.

using adversarial patches to stick on the stop sign is not a suitable attack vector. Patches would lead to the stop sign always being undetected by each passing vehicle and would cause suspicion among the occupants realizing that cars did not stop because of an altered sign.

Capabilities and Knowledge. The adversary has access to the general proximity of the target stop sign and can control a projector so that it points to the sign, see Figure 5.4. We note that the adversary does not necessarily need to have direct physical access to the sign itself – rather to a position from which a visual line of sight exists. In the following we analyse both adversaries with white-box knowledge and a black-box scenario based on the transferability of adversarial examples.

5.4 Method

In this section, we explain our method to carry out the attack.

5.4.1 Modelling projectable perturbations

Often, to realize physical AE, researchers use the non-printability-score introduced by Sharif. [179], which models the set of colours a printer is able to print. In our case, when shining light with the projector, the resulting output colour as captured by a camera depends on a multitude of factors rather than just the printer (as in NPS). These factors include: (i) projector strength, (ii) projector distance, (iii) ambient light, (iv) camera exposure, (v) colour and material properties (diffusion, reflections) of the surface the projection is being shone on (hereafter, *projection surface*). The achievable colour spectrum is significantly smaller than the spectrum available to

printed stickers as a result of these factors (e.g., a patch can be black or white, while most projections on a stop sign will result in red-ish images). In order to understand the feasibility of certain input perturbations, we model these phenomena as follows.

Formalizing the problem. We wish to create a model which, given a certain projection and a projection surface, predicts the resulting colours in output (as captured by a camera). We describe this model $\mathcal{P}$ as follows:

$$\mathcal{P}(\theta_1, S, P) = O, \tag{5.1}$$

where S is the projection surface, P is the projected image, O is the image formed by projecting P on S and θ_1 are the model parameters, respectively.

Finding a perfect model would require taking all of the factors listed above into account, some of which may not be available to an adversary and is also likely to be time consuming due to the volume of possible combinations. Therefore, we opt for a sampling approach, in which we iteratively shine a set of colours on the target surface (the target object) and collect the outputs captured by the camera. We then fit a model to the collected data, which approximates the resulting output colour for given projected images and projection surfaces.

Collecting projectable colours. We define *projectable colours* for a given pixel in S as the set of colour which are achievable in output for that pixel given all possible projection images. To collect the projectable colours, we do as follows:

1. collect an image of the projection surface (S in Equation 5.1). This is an image of the target object.

2. select a colour $c_p = [r, g, b]$, shine an image of that colour P_{c_p} over the projection surface, collect the output O_{c_p}.

3. repeat the previous step with different colours until enough data is collected.

In practice, with r, g and $b \in [0, 255]$ we choose a certain quantization per-colour channel and project all possible colours consecutively, while recording a video of the projection surface. This allows us to collect enough information about the full



Figure 5.5: Camera light sensor noise visualized. The first two images show consecutive frames, while the third image shows the absolute pixel-wise difference (×20) between the two frames. Such sensor noise is accounted for with smoothing over many frames during the data collection step.

colour space. With this method, we found that a quantization of 127 is enough to obtain sufficient accuracy for our method, so that we only need to project $3^3 = 27$ colours to obtain enough data for our model.

Camera noise. In order to collect accurate data, our modelling technique has to account for noise that is being introduced by the camera. At first, we remove noise originating from the sensitivity of the light sensor (ISO [130]), shown in Figure 5.5. In fact, in non-bright lighting conditions, the camera increases the light-sensitivity of image sensor, which generates subtle pixel changes across consecutive (static) frames [83]. To overcome this factor, instead of collecting individual frames for S, P_{c_p}, O_{c_p}, we collect 10 consecutive frames and compute and use the median of each pixel as our final image, the camera is static during this process.

Secondly, we found that there is a smoothing over-time effect in the sensor readings while recording the video, so that the sensor does not update immediately when a certain colour is being shown. Figure 5.7 shows how the average pixel colour per channel changes over time in relation to the timing of certain projections being shown. The camera does not immediately stabilize to the resulting colour when a projection is shown, but adjusts over a few frames. To account for this adaptation, during the data collection, we interleave each projected colour with 10 frames of no projection, so that the camera re-adapts to the unaltered image of the projection surface.

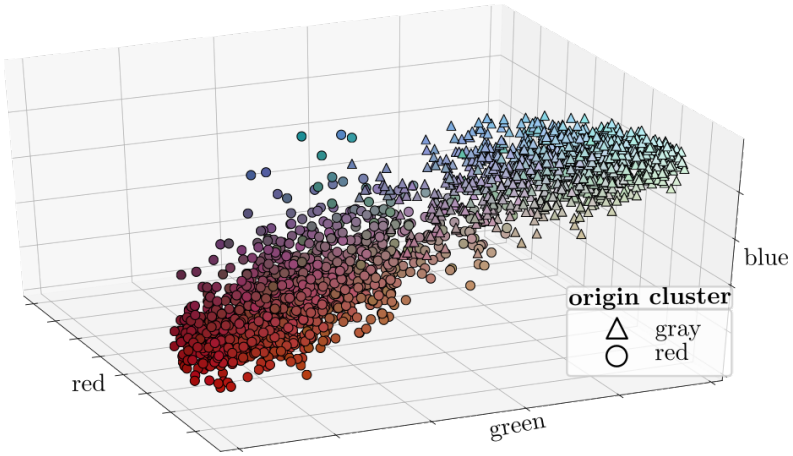


Figure 5.6: Plot showing the output space of the learned projection model. Each data point correspond to a colour in S and its colour is the model output $\mathcal{P}(c_s, c_p)$ for a random c_p.

Fitting a projection model. Once we have collected a set of S, P_{c_p}, O_{c_p} for the chosen set of colours, we construct a training dataset as follows. First we group together pixels of the same colour by creating a mask for each unique colour in the projection surface. In other words, we find the set of unique colours present in S, i.e., $c_s \in S_{unq}$ and then create a mask for each colour $M^{(c_s)} = \{i_j, ..., i_k\}$ such that:

$$i \in M^{(c_s)} \text{ i.f.f. } i^{th} \text{ pixel in } S == c_s.$$

Then, for each unique source colour c_s, we extract all the mask-matching pixels from the output O_{c_p}, average their colours to get an output colour $c_o^{(s,p)}$, and save the following triple for our training data $\{c_s, c_p, c_o^{(s,p)}\}$. A triple indicates that by projecting c_p on pixels of colour c_s we obtained (on average) the colour $c_o^{(s,p)}$. We then use the triples to fit a neural network composed of two hidden layers with ReLU activation, we re-write Equation 5.1 as an optimization problem as follows:

$$\mathcal{L}_{\mathcal{P}} = \arg\min_{\theta_1} \sum_{\forall c_s, c_p} \|\mathcal{P}(c_s, c_p) - c_o^{(s,p)}\|_1, \tag{5.2}$$

where $\mathcal{P}$ is the model. We optimize the network using gradient descent and Adam optimizer. Using $\mathcal{P}$ we have a differentiable model which can be used to propagate the derivatives through it during the AE generation, see Section 5.4.2.

Visualizing the Learned Model. When the projection surface S is a stop sign (as mainly investigated in this chapter), pixels in S generally can be separated into

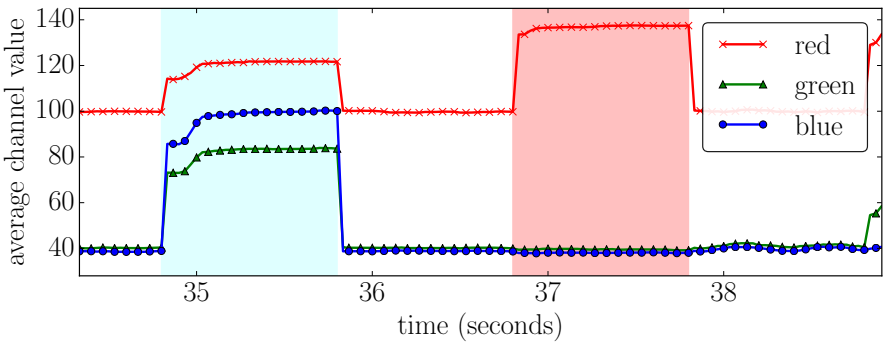


Figure 5.7: Plot showing how the average value of a pixel (RGB) changes when a certain projection is shown. Immediately after the projection is shown, the camera requires a few frames (the lines are marked every 2 frames) to converge to a stable value. The two shaded areas mark the time the projection is being shown and are coloured with the projection colour.

two clusters based on their colour, corresponding to the "red" and "white" part of the sign. The presence of these two clusters is reflected in the outputs of the projection model, as different colours will be achievable in output for the red and white parts of the stop sign. We visualize the outputs of the projection model in Figure 5.6, where we use a learned projection model $\mathcal{P}$, the captured source image S and we compute a set of output colours for random projection colours c_p. Each data point in Figure 5.6 corresponds to the colour of an output pixel and is marked by a different marker (either triangle or circle) based on whether the corresponding source pixel was into the red or white cluster. Figure 5.6 shows that the model learns a different function for red or white source pixels, obtaining in output more blue tones for white pixels while different shades of red for the remaining red pixels.

5.4.2 AE Generation

In this section we describe our method for generating the adversarial projection. As a starting point, we combine the projection model described in Section 5.4.1 with the target network and use gradient descent along both to optimize the projected image. In its basic form, we optimize the following loss function:

$$\arg\min_{\delta_x} \; J(f(t + \mathcal{P}(x, \delta_x))) \quad \text{s.t.} \;\; 0 \leq \delta_x \leq 1,$$

where δ_x is the projected image, f the detection network, $\mathcal{P}$ the projection model, x the input image background, x a stop sign image, and J the detection loss, described later. In the following we describe how we augmented the loss function in order to facilitate the physical feasibility of the adversarial perturbation and the convergence of the optimization.

Physical Constraints. We improve the physical realizability of the projection with two steps. In order to maintain the physical realizability of the projection we have two two steps. At first, we restrict the granularity of the projection in a fixed grid of $n \times n$ cells, so that each cell contains pixels of the same colour. This allows us to use the same projection for different distances of viewing the stop sign. Secondly, we include the *total variation* of the projection in the loss function in order to reduce the effect of camera smoothing and/or blurring [127].

Variable Substitution. Since the optimization problem for the projection is bounded in [0,1] (space of RGB images) to ease the flowing of gradients when backpropagating we remove this box constraint. Given the image to project δ_x, we substitute δ_x with a new variable w such that

$$w = \frac{\tanh \delta_x}{2} + 0.5$$

and instead optimize for w. Since $\tanh \delta_x$ is bounded in $[-1, 1]$ we find that this substitution leads to faster convergence in the optimization.

Loss Function. We also limit the amount of perturbation in our loss so that our final optimization looks as follows:

$$\arg\min_{w} \; J(f(t + \mathcal{P}(x, w))) + \lambda \|\mathcal{P}(x, w) - x\|_p + \mathrm{TV}(w),$$

where λ is a parameter used to control the importance of the p-norm $\|\cdot\|_p$ and TV is the total variation described above. As mentioned in Section 2.3, since we operate on both object detectors and traffic sign recognizers, we use two different losses J depending on the target network. For object detectors, we consider that the network returns a finite set of boxes $b \in B$ where for each box there is an associated probability output of the box containing a semantic object of class j, i.e.,

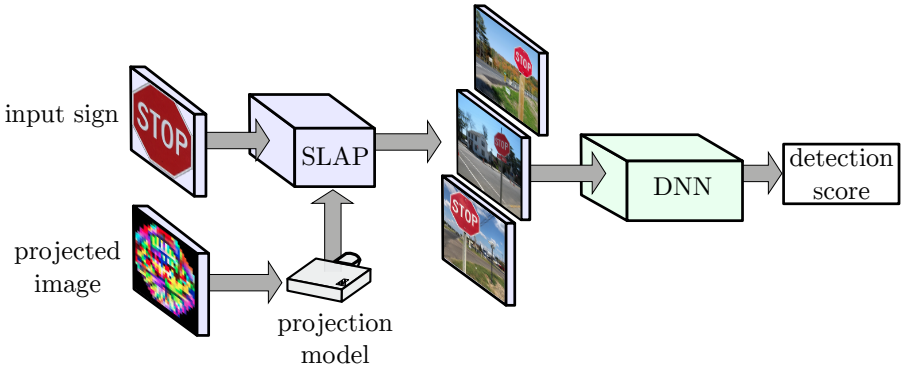


Figure 5.8: Overview of the adversarial samples generation pipeline. We optimize the projected image which passes through the projection model in order to minimize the target detection score on a given DNN for a set of randomly generated permutations of the input.

$p_j^{(b)}$. For traffic sign recognizers, the network returns a probability vector containing the probability of the input image being traffic sign of class j, i.e., p_j. We then use the following loss functions in the two cases:

- **Object Detectors:** the loss is the sum of the detection probabilities for stop signs, i.e., $\sum_{b \in B} p_j^{(b)}$;
- **Traffic Sign Classification:** the loss is the probability for the stop sign class p_j.

5.4.3 Training Data Augmentation

Generating adversarial examples that work effectively in the physical world requires taking into account different environmental conditions. Adversarial examples computed with straightforward approaches such as in [198] do not survive different viewing angles or viewing distances [188]. In order to enhance the physical realizability of these samples, different input transformations need to be accounted for during the optimization. We use the expectation over transformation method [67], which consists in reducing the loss over a set of training images computed synthetically. These training images are generated using linear transformations of the desired

input, i.e., an image containing stop signs, so that different environmental conditions can be accounted for during the optimization. Using EOT, our final loss becomes:

$$\mathcal{L}_f = \arg\min_{w} \ \mathbb{E}_{t_i \sim T, m_j \sim M} \ \ J(f(t_i + m_j \cdot \mathcal{P}(x, w))) + \lambda \|\mathcal{P}(x, w) - x\|_p + \mathrm{TV}(w), \quad (5.3)$$

where T is a distribution over several background images and M is an alignment function that applies linear transformations to the perturbed sign. In this chapter, we augment the set of the transformations to account for additional environmental conditions that are disregarded in previous work. We report in Figure 5.8 an overview of the complete optimization used in our method.

Background and Traffic Sign Post. Similarly to [240] we select a set of road backgrounds and carefully place the stop sign on a post at the edge of the road. In [240] it is shown that the post provides useful information to the detector and should therefore be included when crafting the adversarial perturbation.

Perspective. We vary the angle at which the camera is looking at the stop sign. Since we do not want to account for all perspective transforms, we use the following observations. Firstly, a traffic sign is mostly placed on one side of the lane (to the right in right-driving countries), meaning that rarely a camera mounted on a car would see a sign on the left-part of the frame. Secondly, traffic signs are mounted at specific heights (e.g., 5 or 7 feet in the United States [209]), which normally exceed the height of cars for better visibility. Given these two observations, we prioritize perspective transforms that match these conditions.

Distance. As the car is approaching the stop sign, the sign will appear with different sizes in the camera frame. Our goal is for the car to misclassify the stop sign in every frame, therefore we place stop signs with different sizes during the optimization. We test the detection of the stop sign in non-adversarial settings with decreasing stop sign sizes and we set the minimum size of the sign to be the smallest size at which the sign is detected with high confidence. In other words, we only optimize for signs sizes that are large enough to be detected by the classifier.

Rotation. As shown in [64], simple rotations may lead to misclassifications when those transformations are not captured in the training dataset. We therefore add rotation to the stop sign when crafting the adversarial perturbation.

Brightness. The colour of the stop sign changes based on a combination of ambient light and camera settings, e.g., in sunny days the colours appear brighter to the camera. To account for this, we apply different brightness transformations to the stop sign, so that we include a wider range of colour tones. Since different colours contribute differently to an image brightness, we transform the stop sign image from RGB to YCrCb format [214], increase the luma component (Y) by a specified delta and then bring the image back into RGB.

Camera Aspect Ratio. We observe that popular object detectors resize the input images to be squared before being processed by the network (e.g., Yolov3 resizes images to 416x416 pixels), to speed up the processing. However, the typical native aspect ratio of cameras, i.e., the size of the sensor, is 4:3 (e.g., the Aptina AR0132 chip used in the front-viewing cameras by Tesla, has a resolution of 1280x960 [12]). This leads to objects in the frames to being distorted when the frames are resized to squared. To account for this distortion, we choose the dimension of the stop sign so that its height is greater than its width, reflecting a 4:3 to 1:1 resizing.

5.4.4 Remarks

We use AdamOptimizer to run the AE generation. We optimize a single variable that is the image to project with the projector (its substitute, see Section 5.4.2). We use batches of size 20. All the training images are created synthetically by placing a stop sign on a road background and applying the transformations described in the previous section. We do not use a fixed pre-computed dataset, a new batch with new images is created after every backward pass on the network. The parameters for the transformations are chosen uniformly at random in the ranges shown in Table 5.1. For all operations that require resizing, we use cubic interpolation, finding that it provides more robust results compared to alternatives. We run the optimization for 50 epochs, in one epoch we feed 600 generated images containing a stop sign

Table 5.1: Parameters used for the AE generation and the training data augmentation. The values for brightness, perspective, rotation, aspect ratio indicate the ranges for the applied transformations. All parameters are picked uniformly at random (with the exception of perspective) during the AE generation for each sample in the generated training data.

Parameter	Yolov3	Mask-RCNN	Lisa-CNN	Gtsrb-CNN
learning rate	0.005	0.005	0.05	0.05
brightness	$[-13, +13]$ (with range [0, 255])			
perspective	x-axis $[-30°, +30°]$, y-axis $[-30°, +30°]$			
rotation	$[-5°, +5°]$			
aspect ratio	from 4:3 to 16:9			
sign size	[25, 90] pixels			
grid size	25×25			

in the network. For each epoch we optimize the 20% worst-performing batches by backpropagating twice, convergence is usually reached before the last epoch. Compared to similar works [240], our method runs significantly faster requiring only 50 modifications of the perturbation (compared to 500), which takes less than 10 minutes on an NVIDIA Titan V GPU for Yolov3.

5.5 Evaluation

In this section, we test the feasibility of the attack in practice.

5.5.1 Experimental Setup

Projector Setup. To test our projection, we buy a real stop sign of size 600x600mm. For all of our experiments, we use a Sanyo PLC-XU4000 projector [158], which is a mid-range office projector (roughly $1,500) with 4,000 maximum lumens. We carry out the experiment in a large lecture theatre in our institution. We measure the projector light intensity with a Lux Meter Neoteck, following the 9-points measuring procedure used to measure ANSI lumens [191], which reports that in default settings the projector emits around 2,200 lumens. For the experiments, we place the projector 2 meters away from the stop sign, which, at maximum zoom, allows us to obtain roughly 800 lux of (white) light on the stop sign surface. We

Table 5.2: Preliminary results for the various light settings considered in the experiment. The camera exposure is the exposure of the camera used for profiling (set automatically). The table shows the optimization losses: $Loss_{\mathcal{P}}$ refers to the loss in Equation 5.2, while $Loss_f$ refers to the loss in Equation 5.3.

			$\mathcal{L}_f$			
Lux	*Camera Exposure* (ms)	$\mathcal{L}_{\mathcal{P}}$	Yolov3	Mask-RCNN	Gtsrb-CNN	Lisa-CNN
120	33	0.020	0.09	0.08	0.01	0.06
180	25	0.023	0.11	0.52	0.00	0.07
300	18	0.017	0.68	0.86	0.89	1.03
440	12	0.015	1.44	4.24	5.31	2.45
600	9	0.011	1.80	5.92	9.12	8.16

use this 800 lux white value to make considerations on the attack feasibility in Section 5.6. A similar amount of projected light can be obtained from greater distances by using long throw projectors, available for few thousand dollars (e.g., $3,200 for Panasonic PT AE8000 [157], see Section 5.6). We align the projection to match the stop sign outline by transforming the perspective of the image.

Ambient Light. As mentioned in Section 5.4.1, the amount of ambient light limits the control on the input space for the adversary. In fact, as the ambient light increases, fewer colours are achievable as the projector-emitted light becomes less in the resulting appearance of the sign. To account for different ambient light levels, we conduct our experiments indoor and we control the amount of light hitting the stop sign (Section 5.5.2). We further evaluate the attack outdoors with a road driving test (Section 5.5.3). To reproduce various light settings indoors, we use both the ceiling lights mounted in the indoor hall and by using an additional 60 Watts LED floodlight pointed at the sign. We measure the attack in five different light settings: 120, 180, 300, 440 and 600 lux. The darker setting (120 lux) corresponds to slightly dimming the ceiling lights only. The 180 lux setting corresponds to the normal indoor lighting found in the lecture theatre where we carry out the measurements. Higher settings are achieved by adding the LED floodlight pointed directly at the

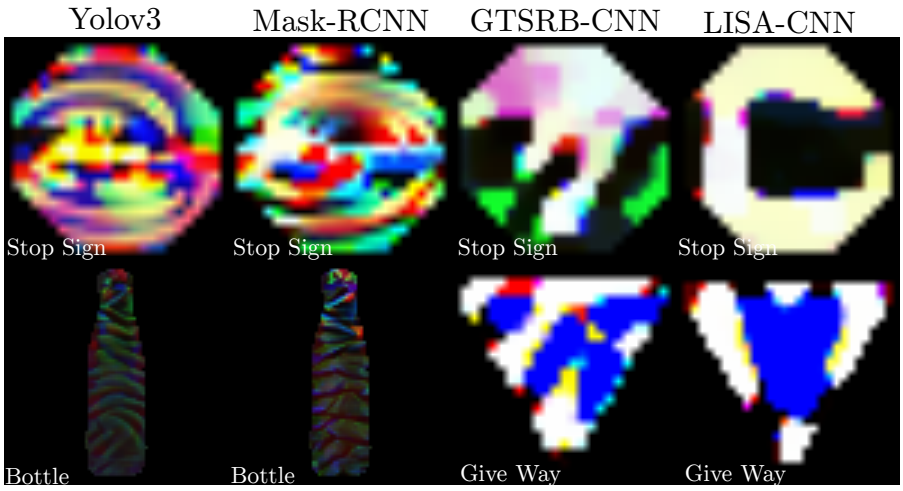
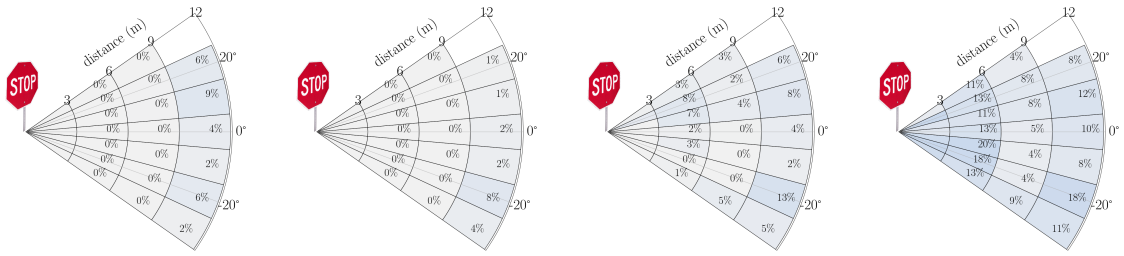


Figure 5.9: Examples of the projected images computed with the optimization. Bottom-right of each image specifies the target class fed to Equation 5.3. These images are computed within the 180 lux setting.

sign at different distances (from roughly 4m away at 300 lux to less than 2m away at 600 lux). For reference, on a clear day at sunrise/sunset the ambient light is roughly 400 lux, while on an overcast day at the same hours there are roughly 40 lux [224].

Networks and Detection Thresholds. We consider four different networks in our experiments: two object detectors, (1) Yolov3 and (2) Mask-RCNN, two traffic sign recognizers, (3) Lisa-CNN and (4) Gtsrb-CNN. For Yolov3, we use the Darknet-53 backbone of the original paper [162]. For Mask-RCNN, we use Resnet-101 as a backbone and feature pyramid network [111] for the region proposals. We download the weights for Lisa-CNN and Gtsrb-CNN from the GitHub of the paper authors [67]. As Mask-RCNN and Yolov3 return a list of boxes with a confidence score threshold for the output class, we set the threshold for detection at 0.6 and 0.4 respectively (i.e., we count detection as "there is a box labelled stop sign with score higher than x"). These are the thresholds that bring the highest mean Average Precision (mAP) in the MS COCO object detection benchmark [112]. For Lisa-CNN and Gtsrb-CNN we set the detection threshold as 0.5. The input images are resized to 416x416 for Yolov3 and Mask-RCNN and to 32x32 for Lisa-CNN and Gtsrb-CNN.

Metrics and Measurements. For object detectors (Yolov3 and Mask-RCNN), we feed each frame into the network and we count how many times a stop sign is



(a) Yolov3. **(b)** Mask-RCNN. **(c)** Gtsrb-CNN. **(d)** Lisa-CNN.

Figure 5.10: Baseline mis-detection rate in absence of the adversary for the 180 lux setting at different angles, distances and for different networks, as the percentage of frames where a stop sign is **not** detected. Brighter shades represent higher detection rates. Percentages for 0-3m are omitted for clarity, but the corresponding cone section is coloured accordingly.

detected in the input. For traffic sign recognizer (Gtsrb-CNN and Lisa-CNN), the network expects a cut-out of a traffic sign rather than the full frame. In order to obtain the cut-out, we manually label the bounding box surrounding the stop sign and use a CSRT tracker [121] to track the stop sign over the frames. We then count how often the predicted label is a stop sign. In order to monitor viewing angle and distance from the sign, we reconstruct the angle of view and distance based on the distortion on the octagonal outline of the sign and our recording camera field-of-view. We use the default camera app on an iPhone X to record a set of videos of the stop sign at different distances and angles, with the projection being shone. The iPhone is mounted on a stabilizing gimbal to avoid excessive blurring. As mentioned in Section 5.4.3, to match the 4:3 aspect ratio, we crop the 1080p video from the iPhone X (which has a resolution of 1920x1080) to 1440x1080 by removing the sides.

Experimental Procedure. Experiments follow this pipeline:

- **Step 1:** We setup the stop sign and measure the amount of lux on the stop sign surface;
- **Step 2:** We carry out the profiling procedure to construct a projection model (Section 5.4.1); this uses a separate Logitech C920 HD Pro Webcam rather than the iPhone X camera (on which the attack is later evaluated).
- **Step 3:** We use the projection model to run the AE generation (Section 5.4.2) and optimize the image to project;

- **Step 4:** We shine the image on the sign and we take a set of videos at different distances and angles.

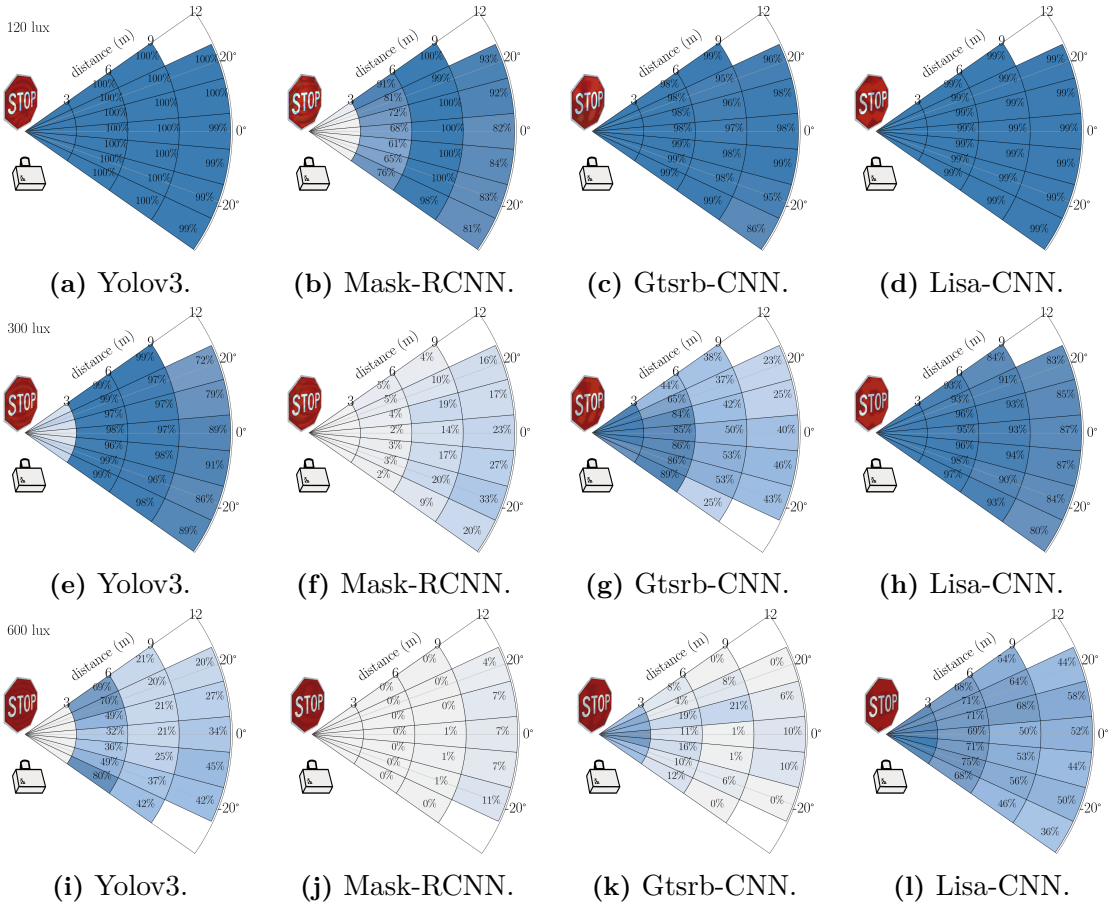
The parameters used for the optimization (Step 3) are those of Table 5.1. Recording the profiling video of Step 2 requires less than 2 minutes, so does fitting the projection model.

Preliminary Results. Table 5.2 shows parameters and resulting value of the loss functions at the end of the optimizations for the various light settings. The table shows that our projection model fits the collected colour triples: $Loss_{\mathcal{P}} < 0.03$ shows that the error in the predicted colours, is less than 1% per channel. As expected, we found that the results of the optimization match the reduced capability to reproduce colours for higher ambient light settings: $Loss_f$ goes from roughly 0 to higher values as the light increases. Note that the reported $Loss_f$ is summed over the batches of 20 and computed before non-maximum suppression. We report in Figure 5.9 examples of projected images output of the optimization process, for three different target objects: stop sign, give way sign and bottle. Whilst we limit the rest of the experiments to attacks on stop signs, we report considerations and results on generalizing the attack to various objects in Appendix A.1 and in Section 5.6.

5.5.2 Indoor Results

In this section, we present the results of the detection for the controlled indoor experiment. We also report in Figure 5.10 the baseline results of using the networks to detect/classify the stop sign, by recording videos of the stop sign unaltered. Figure 5.10 shows that all networks work quite well in non-adversarial conditions, with the exception of Lisa-CNN which shows a few misdetections.

We report in Figure 5.11 the results of the detection for the 120, 300 and 600 lux setting for the different networks, as the percentage of frames where the stop sign was not detected by the network. The minimum number of frames tested for a single model is 3,438, see Table A.1 for exact figures. Figure 5.11 shows that the attack is extremely successful in dimmer lighting conditions, obtaining >99% success rate for all networks except Mask-RCNN, which presents additional



(a) Yolov3. (b) Mask-RCNN. (c) Gtsrb-CNN. (d) Lisa-CNN.

(e) Yolov3. (f) Mask-RCNN. (g) Gtsrb-CNN. (h) Lisa-CNN.

(i) Yolov3. (j) Mask-RCNN. (k) Gtsrb-CNN. (l) Lisa-CNN.

Figure 5.11: Attack success rate at different angles, distances and for different networks, as the percentage of frames where a stop sign is not detected. Darker shades represent higher success rates. Percentages for 0-3m are omitted for clarity, but the corresponding cone section is coloured accordingly. The images of the stop signs in the figure are computed using the projection models for the two light settings, so they resemble what the adversarial stop sign looks like in practice.

resilience at shorter distances. The figure also shows how our method is able to create AE that generalize extremely well across all the measured distances from 1 to 12m and viewing angles -30° to 30°. As the ambient increases, the success rate quickly decreases accordingly. Already at 300 lux, the attack success rate is greatly reduced for Mask-RCNN and Gtsrb-CNN, while Yolov3 and Lisa-CNN still remain vulnerable, but the attack degradation becomes evident at 600 lux.

Overall, we found that Mask-RCNN is consistently more resilient than the other networks in the detection. In particular we found that Mask-RCNN sometimes recognizes stop signs just based on the octagonal silhouette of the sign or even just with faded reflections of the sign on windows. This could be a combination

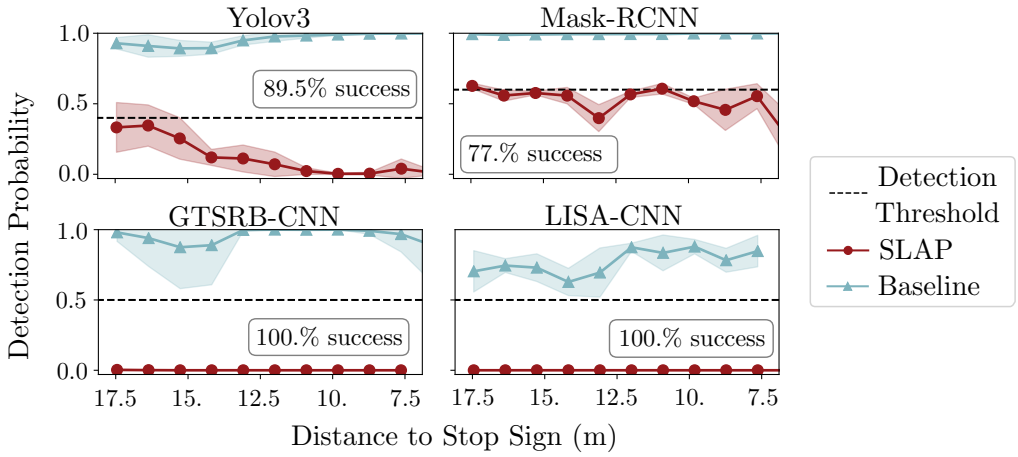


Figure 5.12: Detection probability for the stop sign during the road driving test. During the test the car approaches the stop sign while the attack is being carried out, the ambient light during the measurements is ∼120 lux, the car headlights are on. The data are grouped into 10 distance bins, the shaded areas indicate the standard deviation of the probability within that distance bin.

of Mask-RCNN learning more robust features for the detection (possibly thanks to the higher model complexity) and of using a region proposal network for the detection [84]. Nevertheless, such robustness comes at the cost of execution speed: Mask-RCNN requires up to 14 times the execution time of Yolov3 (300ms vs 22ms).

5.5.3 Road Driving Test

To further test the feasibility of the attack, we carry out the attack outdoors in moving vehicle settings.

Setup. The experiment is carried out on a section of private road at our institution. We mount the stop sign at 2m height and set the projector in front of it at a distance of approximately 2 metres. The experiment was conducted shortly prior to sunset in early October, at coordinates 51.7520° N, 1.2577° W. At the time of the experiment the ambient light level measured at the surface of the sign is ∼ 120 lux. We use a car to approach the stop sign at 10-15km/h, with the car headlights on during the approach. Videos are recorded using the same iPhone X mounted inside the car at 240fps. We follow the same pipeline described in Section 5.5.1. However, rather than carrying out the profiling step (Step 2 of the experimental procedure), we *re-use* the 120 lux projections that were optimized for the controlled indoor conditions.

Table 5.3: Percentages (%) of attack success rate across the various evaluated defences, models and lux settings. Figures are reported as the percentage of frames in which the attack is successful, i.e., a stop sign is not detected. Differently, (*) figures for SentiNet are reported as percentage out of the 100 adversarial frames extracted from the videos, both overlaying patterns Random and Checkerboard are reported.

	Ambient	*Attack*	*Adversarial*	*Input*	*SentiNet [42]*	
Network	*Light* (lx)	*Success*	*Learning [198]*	*Rand. [227]*	Rand.	Checkerb.
Gtsrb-CNN	120	99.96	20.23 (-79.73)	99.55 (-0.40)	93.43	95.45
	180	90.53	23.57 (-66.97)	90.02 (-0.51)	93.19	93.72
	300	56.51	48.18 (-8.33)	86.78 (+30.27)	96.97	96.46
	440	56.34	40.24 (-16.10)	82.96 (+26.61)	95.81	96.34
	600	12.79	10.91 (-1.88)	51.37 (+38.58)	95.29	95.29
Lisa-CNN	120	100.00	0.06 (-99.94)	100.00 (+0.00)	94.24	95.29
	180	99.95	0.88 (-99.07)	99.90 (-0.05)	100.00	100.00
	300	99.81	0.00 (-99.81)	99.98 (+0.17)	94.76	96.86
	440	98.44	0.59 (-97.85)	99.95 (+1.51)	100.00	100.00
	600	69.05	0.04 (-69.01)	95.71 (+26.67)	95.81	96.86

Results. We report the results from the driving test in Figure 5.12, which shows the probability of detection for stop sign as the car approaches the sign. The experiment measures up to 18m away to roughly 7m, when the stop sign exits the video frame (we keep the camera angle fixed during the approach). The results closely match the findings indoor, with the attack being successful for most networks along the whole approach: we obtain 100% success rate for Lisa-CNN and Gtsrb-CNN and over 77% for Mask-RCNN and Yolov3. These results also confirm the generalizability of optimized projections: simply re-using projections without having to re-execute Step 2 and Step 3 of the experimental procedure at the time of attack led to similar success rates. This means that adversaries could easily pre-compute a set of projections and quickly swap between them depending on the current light conditions.

5.5.4 Defences

Generally, AE defences are aimed at detecting AE in a digital scenario, where adversaries have the capability to arbitrarily manipulate inputs, but are limited to an ℓ_p-norm constraint. In the case of physical AE, adversaries are not directly

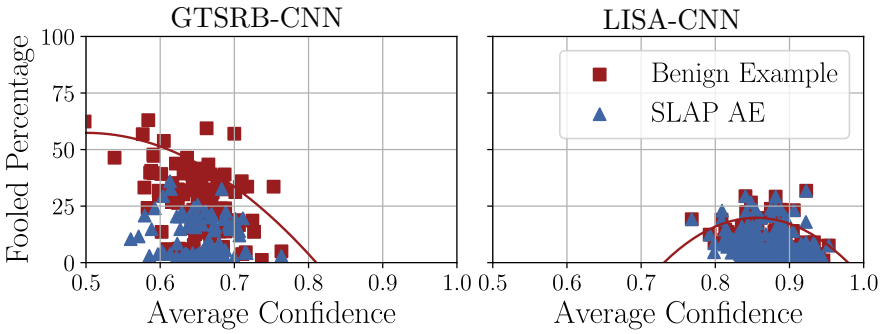


Figure 5.13: Visualization of the SentiNet detection results (from the 180 lux setting). The plot shows that the SLAP AE have a behaviour similar to benign examples across the two dimensions used by SentiNet, preventing detection.

limited by an ℓ_p-norm constraint but by the physical realizability of their AE. Defences that are tailored to physical AE have not been investigated as much as general-scope AE defences. For this reason, we specifically choose to evaluate our AE against Sentinet [42]: it is one of the few published works that addresses physical AE detection. Additionally, we evaluate our attack against two other defences which could be used in the autonomous driving scenario as they do not entail additional running time: the input randomization by Xie et al. [227] and adversarial learning [198]. In the following we describe our evaluation setup and results.

Setup and Remarks. We evaluate the three considered AE defences applied to Gtsrb-CNN and Lisa-CNN, as all three defences are designed to work in image classification scenario; at the time of writing, defences for object detectors are not as well explored. For SentiNet, we use 100 *benign* images taken from the GTSRB and LISA dataset to compute the threshold function, 100 *test* images where we overlay the suspected adversarial regions and 100 random frames containing a SLAP AE from the collected videos. For our Sentinet implementation, we use XRAI [93] to compute saliency masks as the original method used (GradCam [40]) led to too coarse grained masks (see Appendix A.3). For the input randomization of [227], we set the maximum size of the padded image to be 36 (from 32). For adversarial learning [198], we re-write the Lisa-CNN and Gtsrb-CNN models and we train them on the respective datasets from scratch adding a fast-gradient sign method (FGSM)-based adversarial loss to the optimization. We use Adam with learning rate

0.001, the weight of the adversarial loss is set to 0.2, the FGSM step size to 0.2, we use ℓ_∞-norm and train for 50 epochs. Adversarially trained models present a slight accuracy degradation on the test set compared to training them with categorial cross-entropy, Gtsrb-CNN goes from 98.47% to 98.08% (-.39%) while Lisa-CNN from 95.9% to 95.55% (-.35%). For input randomization and adversarial learning we run the inference on all the collected video frames of the experiment.

Results. We report the results in Table 5.3. The table shows the attack success rate computed as the percentages of frames where a stop sign was not detected. We also report the legitimate attack success for comparison. We found that input randomization does not detect our attack. This is expected given that any type of input augmentation-defence is intrinsically compensated for by our optimization (see Section 5.4.3). Even worse, such method actually degrades the accuracy of the model, showing that the original models for Lisa-CNN and Gtsrb-CNN taken from [67] were not trained with sufficient data augmentation. As expected, thanks to the larger affected areas of the SLAP AE, these adversarial samples can bypass detection by SentiNet in over 95% of the evaluated frames, with no significant difference across the overlay pattern used (either Random or Checkerboard). We also report a visualization of the threshold function fit in SentiNet in Figure 5.13, showing that the behaviour of SLAP AE resembles those of normal examples. We found that adversarial learning is a more suitable way to defend against SLAP, stopping a good portion of the attacks. Nevertheless, the fact that we only evaluate an adaptive defender (not an adaptive adversary) and that adversarially-trained models suffer from benign accuracy degradation (performance of the model with no attack in place) highlights how SLAP still remains a potential threat.

5.5.5 Attack Transferability

Setup. In this section, we test the transferability of our attack across networks, testing all pairwise combinations of our models, including adversarially trained ones. We also use the Google Vision API[77] to test our projections against their proprietary models. The API returns a list of labelled objects in the image with

Table 5.4: Transferability results. We test all the frames from the collected videos with a certain projection being shone against a different target model, figures in bold are white-box pairs. (*) For Google Vision we only test one frame every 30 frames, i.e., one per second. We also remove all frames that are further than 6m away as Google Vision does not detect most of them in a baseline scenario. $_^{(a)}$ indicates adversarially trained models.

			Target Model						
Lux	*Source Model*	*no. Frames*	Yolov3	Mask-RCNN	Gtsrb-CNN	Gtsrb-CNN$^{(a)}$	Lisa-CNN	Lisa-CNN$^{(a)}$	Google Vision*
120	Yolov3	4587	**100.0**	73.4	0.0	0.0	21.5	0.0	100.0
	Mask-RCNN	3765	98.7	**97.1**	0.0	0.0	15.5	0.0	100.0
	Gtsrb-CNN	3760	40.5	37.0	**99.9**	16.1	51.4	0.0	72.4
	Lisa-CNN	4998	29.4	28.1	6.8	0.0	**100.0**	0.0	77.1
300	Yolov3	5169	**96.5**	3.6	2.5	0.0	2.3	0.0	72.3
	Mask-RCNN	3543	32.0	**14.0**	0.1	0.0	10.4	0.0	65.9
	Gtsrb-CNN	3438	2.0	2.9	**48.0**	43.1	44.0	0.0	47.6
	Lisa-CNN	4388	0.7	4.9	8.6	0.0	**100.0**	0.0	25.0
600	Yolov3	5507	**17.8**	0.2	32.5	0.0	27.4	0.0	23.7
	Mask-RCNN	5058	0.1	**0.4**	5.3	0.0	4.6	0.0	16.7
	Gtsrb-CNN	4637	0.0	0.9	**7.2**	7.5	4.9	0.0	21.1
	Lisa-CNN	4714	0.0	0.9	8.6	0.0	**57.5**	0.0	15.8

associated confidence scores and bounding boxes, "stop sign" is one of the labels. We set the detection threshold for Google Vision API as 0.01, i.e., we count that a stop sign is detected in a frame if the API replies with a stop sign object with confidence greater than 0.01.

Results. We report the results in Table 5.4. The table shows the source (white-box) model on the left, which identifies the projection shown in the tested videos. We also report the number of frames tested, taken from the videos from the indoor experiment. Table 5.4 reports success rates of the attack as a percentage of the frames where the stop sign was undetected. Table 5.4 shows that our attack transfers well for low light settings, but the transferability degrades quickly for the 300 lux setting and above. We find that Mask-RCNN transfers better to

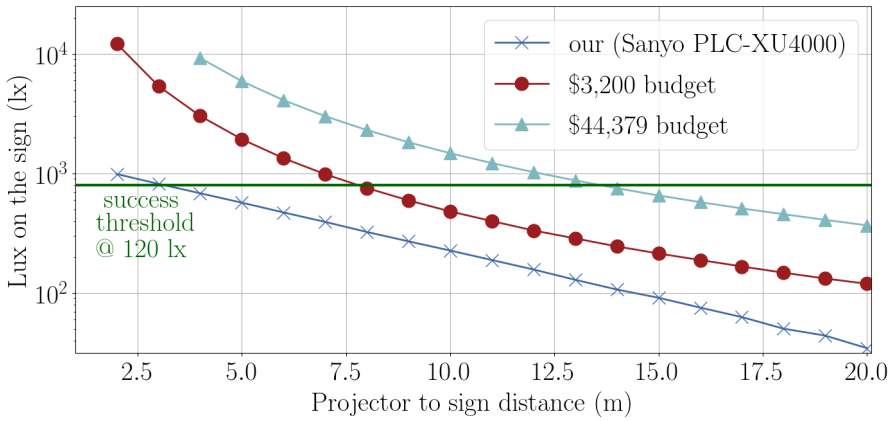


Figure 5.14: Amount of lux achievable on the stop sign surface for increasing projection distances and different projectors. The horizontal line shows the threshold for success measured in our experiments (800 lux at 120 lux ambient light).

Yolov3 compared to the opposite direction, the same happens for Gtsrb-CNN and Lisa-CNN, suggesting that fitting AE on complex models favours the attacker. Table 5.4 also shows that adversarially trained model have benefits by reducing the transferability of attacks fit on surrogate models.

5.6 Discussion

In this section, we discuss the attack feasibility.

Attack Feasibility. Our experiments demonstrate that increasing ambient light quickly stops the feasibility of the attack in bright conditions. In practice, during daytime, the attack could be conducted on non-bright days, e.g., dark overcast days or close to sunset or sunrise, when the ambient light is low (<400 lux). Regarding the effect of car headlights, our outdoor experiments show that the car headlights-emitted light is negligible compared to the projection luminosity and does not influence the attack success. While car headlights on high-beam would compromise the projection appearance and degrade the attack success rates, we did not consider these lights to be on as stop signs would be mainly present in urban areas, where high-beam headlights would be off. In general, the amount of projector-emitted light that reaches the sign depends on three factors: (i) the distance between projector and sign, (ii) the throw ratio of the projector and

(iii) the amount of lumens the projector can emit. We report in Figure 5.14 a representation of how the distance between the projector and the stop sign relates to the attack success rate. We consider two additional projectors with long throw distance, the Panasonic PT-RZ570BU and the NEC PH1202HL1, available for $3,200 and $44,379 respectively. We use the projector's throw ratios (2.93 and 3.02) and their emitted lumens (5,000 and 12,000 lumens) to calculate how many lux of light the projector can shine on the sign surface from increasing distances. We consider the success as measured in 120 lux ambient light, where obtaining 800 lux of light on the sign with the projector is sufficient to achieve consistent attack success (see Section 5.5.1). Figure 5.14 shows that the attack could be carried out from 7.5m away with the weaker projector and up to 13m away with the more expensive one. Additionally, adversaries could also use different lenses to increase the throw ratio of cheaper projectors (similarly to [129]).

Attack Generalizability. We show results for attacks on other objects (give way sign, bottle) in Appendix A.1, however, to extend the attack to *any* object, the adversary will have to consider the distortion introduced by the projection surface (not necessary for flat traffic signs). The attacker will have to augment the projection model used in this chapter with differentiable transformations which model the distortion caused by the non-flat surface. In general, the size of the projectable area limits the feasibility of the attack against certain objects (e.g., hard to project on a bike); this drawback is shared across all vectors that create physically robust AE, including adversarial patches. We also found that the properties of the material where the projection is being shone will impact the attack success: traffic signs are an easier target because of their high material reflectivity. When executing the attack on other objects, we found that certain adaptations lead to marginal attack improvements, in particular context information (e.g., the pole for the stop sign, the table where the bottle is placed). Generally, for object detectors, adversaries will have to tailor certain parameters of the optimization to the target object.

5.7 Summary

In this chapter we presented SLAP, a new attack vector to realize short-lived physical adversarial examples by using a light projector to alter the content of the scene captured by an RGB-camera. We investigate the attack in the context of road safety, where the attacker's goal is to change the appearance of a stop sign by shining a crafted projection onto it so that it is undetected by the DNNs mounted on autonomous vehicles. Following the principle of realizability (Section 1.1), we evaluated the attack by reproducing it in the real-world under white-box adversarial knowledge, showing limited effectiveness even in black-box settings. The novel capability of modifying the environment combined with the capability of carrying out opportunistic attacks, makes SLAP a powerful attack vector, which should be included in the security analysis of vision-based systems. In the following chapter, rather than focusing on adversaries altering the scene, we take a deeper look at the inner workings of image sensing in cameras, which are integral part of the overall correct behaviour of the system.

6

Vulnerabilities of Cameras using Rolling Shutters

Contents

6.1 Introduction 106
6.2 Image Sensing Background 108
6.2.1 Photodiode Arrays 109
6.2.2 Scanning Element 109
6.3 Threat Model 111
6.4 Exploiting the Rolling Shutter 113
6.4.1 Controlling the Distortion Size 114
6.4.2 Estimating Parameters 116
6.5 Rolling Shutter Model Evaluation 118
6.5.1 Experimental Setup 118
6.5.2 Accuracy of the Shutter Model 121
6.6 Attack on Object Detection 122
6.6.1 Evaluation on Video Datasets 124
6.6.2 Comparison with Blinding Attack 129
6.7 Countermeasures 131
6.7.1 Hardware Countermeasures 131
6.7.2 Sensor Redundancy 131
6.7.3 Deep Learning Based Detection 132
6.8 Limitations 134
6.8.1 Evaluation Limitations 134
6.8.2 Attack Limitations 135
6.9 Summary 136

6.1 Introduction

The previous chapter combined a weakness in DNNs with a practical attack vector which allowed for an attack to be physically realized. Nevertheless, many visions systems not only rely on DNNs, but also on the correct acquisition of data (images), which depends on digital RGB-cameras (see the threat model in Figure 1.2). In fact, digital cameras are undoubtedly ubiquitous today, with units now integrated into most mobile and internet-of-things (IoT) devices and already dominant in photography, monitoring, and surveillance. With the ongoing integration of cameras into the decision-making process of intelligent systems, such as consumer home security [78, 165], commercial security [44], blanket surveillance [182, 146], construction site safety [66], retail [190], and vehicle automation [154, 231], the integrity of captured images is becoming a cornerstone of the correct behaviour of the system.

As seen in Section 2.4, academic work has produced many examples of disrupting camera-based vision systems by altering the (camera-)captured environment [141, 244, 67, 120]. On the other hand, attacks inhibiting the correct usage and integrity of camera footage by targeting the camera itself are well known and have been demonstrated in different settings [154, 231]. Naïve disruption of cameras using powerful light projected *at the camera* is straightforward and often effective, shown in academic work [154] and real-world settings [48]. This is because such attacks target the image sensor at the heart of the digital camera.

Here, we combine knowledge of (most widespread) CMOS image sensors with the use of DNN in vision systems to show that an adversary can obtain small-scale control over the captured images, in an attack which we refer to as *rolling shutter attack*. This is in fact possible because CMOS image sensors typically implement an electronic rolling shutter mechanism that captures the image in sequential rows instead of all-at-once. As small-scale, high-frequency information plays a crucial role in these systems [215], such as for edge detection, the injection of fine-grained



(a) Object detection. **(b)** Instance segment. **(c)** Face detection. **(d)** Person tracking.

Figure 6.1: Examples of computer vision use-cases and the impact of a rolling shutter attack on the models' predictions. The distortions introduced by the attack compromise the accuracy of the vision task. The models used were, from left to right, FasterRCNN [163], MaskRCNN [84], RFCN [50] and SSD [114].

distortion has a more profound effect on performance than naïve blinding attacks. As such, attacks based on rolling shutter artefacts have consequences for a wide range of vision-based intelligent systems, see examples in Figure 6.1.

This chapter introduces rolling shutter attacks and studies them in practice. We model the dynamics of the electronic rolling shutter, showing that all the parameters necessary to carry out an attack can typically be inferred from the camera's technical specifications. We validate our modelling with empirical data collected from executing the rolling shutter attack on two cameras, showing how such data matches the expectations. We then investigate the inconspicuousness of the rolling shutter attack, showing that when compared to a blinding attack, the rolling shutter attack causes only small changes in a set of commonly-used metrics for measuring image distortion. Nevertheless, our evaluation on object detection reveals that rolling shutter attacks can significantly disrupt the performance of state-of-the-art networks, as in Figure 6.1a. Finally, we discuss countermeasures for the attack detection, including re-purposing the backbone network used for object detection in order to detect the presence of a rolling shutter attack, and we outline the attack limitations.

Contributions We make the following contributions:

- We model the mechanisms behind the rolling shutter, we show that an attacker can accurately control the attack-injected distortions simply by inferring

parameters from the target camera's technical specifications. We validate our modelling against empirical data by collecting a large dataset of attack executions with real hardware.

- We perform an extensive evaluation of the rolling shutter attack's effect on the object detection task, by partially simulating it on the BDD100K [236] and the VIRAT [145] dataset, showing that the integrity of object detection can be compromised.

- We assess the inconspicuousness of the attack across several metrics on BDD100K and VIRAT video datasets, our analysis highlights how the rolling shutter attack is harder to detect in comparison to a blinding attack.

- We discuss a range of countermeasures and propose an accurate and lightweight enhancement to the backbone network of an object detector to detect rolling shutter attacks.

The experiments code and data used in Section 6.5, 6.6, 6.7.3 and B.2 have been validated by the Annual Computer Security Applications Conference '22 artifacts review process and are available online.[1]

6.2 Image Sensing Background

In this section, we describe the underlying technical concepts and terminologies of image sensing technologies. We first explain how the image sensor captures image information and how the accumulated signal charge is recovered from the photosensitive array. Then, we outline how the scanning technique, used to recover the pixels' signal charge, enables a rolling shutter attack.

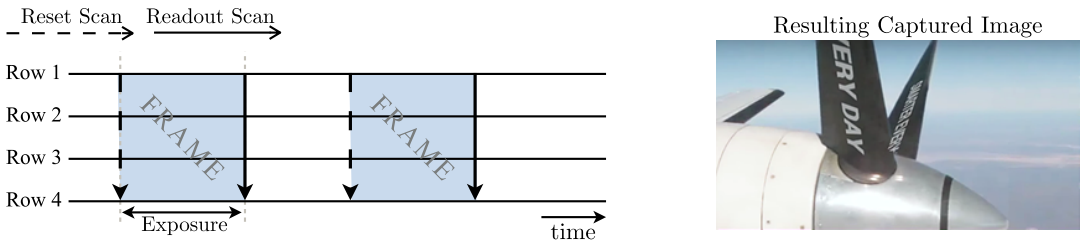
[1] `https://github.com/ssloxford/they-see-me-rollin`

6.2.1 Photodiode Arrays

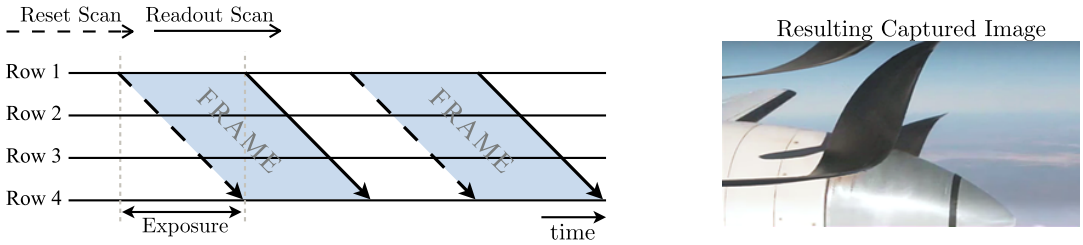
An image sensor is composed of a two-dimensional array of unit cells, also known as pixels. The central component of each pixel is a photodiode, which absorbs the incident photons and converts them into a signal charge depending on the intensity of the light beam. An image is captured by exposing each pixel in the photosensitive array to incident light for a certain amount of time. The period in which a photodiode is exposed, and signal charge is generated, is known as the integration period [58], or *exposure time*. The longer the integration period, the more photons are converted into signal charge and the brighter the resulting image. To control the amount of light that hits the image sensor and is absorbed by the photodiode, a shutter mechanism is necessary. For reasons of speed and accurate control, most cameras nowadays implement an electronic rather than mechanical shutter mechanism [137]. With electronic shutters, the image sensor is constantly exposed to light and signal charge is continuously generated. However, the accumulated charge is flushed before reading, such that only the light gathered during the specified integration time, between the reset and readout scan, represents the final image [99]. The length of the integration time is typically managed by an automatic exposure control system, in which the measurements from an integrated light meter are used to optimize the exposure period.

6.2.2 Scanning Element

Due to their respective sensor layout, the two most common image sensor technologies, Charge-Coupled Device (CCD) and CMOS, use different techniques to recover the accumulated signal charge from the image array. The main difference between the two architectures is the arrangement of scanning elements and measurement units in the chip. Measurement units quantify the signal charge and convert it into electrical signals, whereas scanning elements control the order in which the captured signal charges are read [105]. While CCD sensors have a single measurement unit, CMOS image sensors have a measurement unit in each pixel [137]. This means that in CMOS image sensors, the signal charge is amplified and converted individually in



(a) Electronic global shutter timing.



(b) Electronic rolling shutter timing.

Figure 6.2: Comparison of the frame acquisition in the time-domain between the electronic global shutter and the electronic rolling shutter. Rolling shutter effects are sometimes visible on captured images or frames, in the example, a rotating airplane propeller fan is captured with distortions by the electronic rolling shutter.

each pixel. In the following, we describe the two electronic shutter implementations of modern image sensors [105], *global* and *rolling* shutters, and how they relate to the CMOS and CCD image sensors.

Global Shutter. Image sensors that implement an electronic global shutter must simultaneously expose the entire photodiode array to incident light and also read out the accumulated signal charge from it at once. In CCD image sensors, all photodiodes are flushed at the same time and thus, the exposure time for all pixels starts simultaneously. Figure 6.2a illustrates the readout of a CCD image sensor with an electronic global shutter in the time-domain. The simultaneous readout is possible because, in CCD image sensors, the individual pixels are interconnected via vertical and horizontal shift registers, more specifically charge-coupled devices, hence the name CCD image sensor [105]. Those CCDs work as memory buffers and store the readout charge while the new integration period has already begun. The use of such memory buffer weighs on production cost and chip size [243, 221], which lead to CCD image sensors being most common in industrial and professional use [58]. In

comparison, in CMOS image sensors, the individual pixels are connected via metallic wires that cannot be used to store the signal charge [105]. Even though it is possible to implement an electronic global shutter in CMOS image sensors by changing the circuit and adding additional memory buffer, it is often not economically viable and would defeat the main advantage — cheap manufacturing that enables widespread use in consumer and semi-professional applications.

Rolling Shutter. Image sensors with an electronic rolling shutter expose and read each row of the photodiode array one by one. As a result, the readout of the last pixel row is slightly delayed compared to the first one. An exemplary readout scan showing the time delay between the rows is depicted in Figure 6.2b. Since CMOS image sensors do not have a memory buffer, they can only implement an electronic rolling shutter and thus suffer from image distortions as a consequence of the delay between reading the first and last pixel row — banding and geometric distortion [105]. The most common phenomena are skewed or compressed objects, which can be observed particularly well when fast-moving objects are captured. Another frequently occurring distortion is horizontal banding, which is especially common in indoor environments where dimmable LED lights are used. Since LEDs are often dimmed by switching them on and off at a high frequency, bright horizontal bars might appear in the captured image [52]. In our proposed rolling shutter attack, we exploit this phenomenon to inject controlled distortions into the captured image.

6.3 Threat Model

We focus on defining a general basis for a rolling shutter attack on camera-based intelligent systems. This section describes and motivates the threat model used in our analysis.

Goal. The adversary's general goal is to reduce the reliability and performance of camera-based intelligent systems in a way that is *stealthier* than a blinding attack (denial-of-service). In the simplest case, where the camera is within the attacker's reach, it is possible to use paint, bullets, or direct physical force to prevent a camera

from working properly. However, these truly brute-force attacks are mitigated by keeping a camera out-of-reach or using tamper sensors in the housing. They are also unattractive as they risk the attacker being caught on camera while conducting the attack. As pointed out in Section 6.1, it is straightforward to perform a blinding attack against a camera by shining a very bright light at it. However, this approach suffers from two key problems. First, it is easily detectable with even simple methods such as 'scene too bright/dark' thresholds or 'image global change' comparisons, or by monitoring automatic exposure levels [196]. Second, in bright environments, it often provides incomplete disruption. The image is completely disrupted in one area, but the effect rapidly diminishes away from that point; leaving much of the image unobscured. The residual colour tint has little effect on computer vision systems such as object detection or face recognition, that typically make greater use of high-frequency image components [215]. By contrast, the attack considered here is a *stealthy* attack that minimizes observable disruption to the system, while in fact, impairing its function.

Capabilities. We assume the attacker has access only to off-the-shelf equipment to drive and modulate the laser and to aim it appropriately. We consider adversaries that have no control or access over the target camera, although they *may* be able to obtain knowledge of the camera hardware specifications, either from a datasheet or from a replica device. However, knowledge of camera parameters is not a requirement, as we explain in Section 6.4 and demonstrate on a range of cameras in Section 6.5.1. Similar to previous work [154, 34, 195] and as shown in [33], we consider that the adversary has line-of-sight with the target camera and the capabilities of aiming the laser at it. In contrast to [108, 174], which assume a rolling shutter attack in a static scene (i.e., single object, fixed distance and static lighting), we focus on a more general threat model. Following the assumption that the adversary cannot access the video of the target camera, we consider the attack to be executed in a black-box setting without any feedback channel. As such, distortions can only be injected at random locations, making it infeasible to

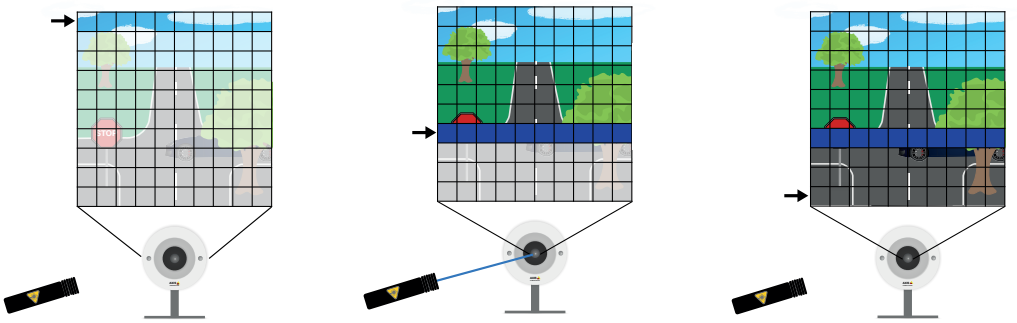
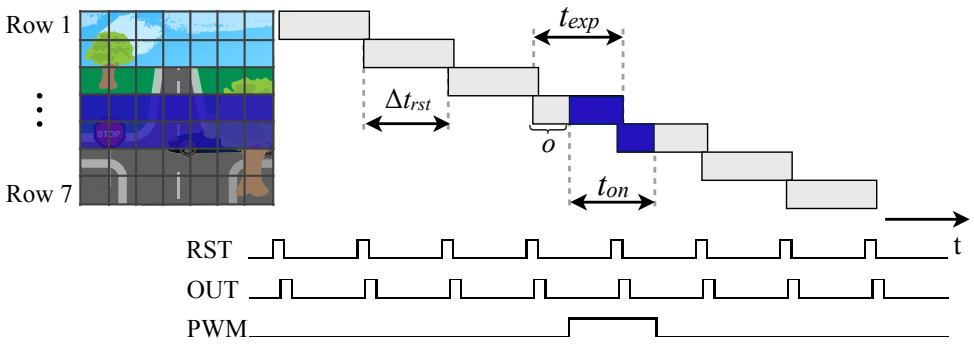


Figure 6.3: Illustration of the row-wise acquisition of a single frame. The arrow on the left indicates the row that is currently exposed and readout. While reading the frame, a laser injects light into the sensor during the exposure of a specific row. As a result, most of the image rows are left unchanged, but distortions appear at the compromised row, obscuring important scene information necessary for the autonomous vision system.

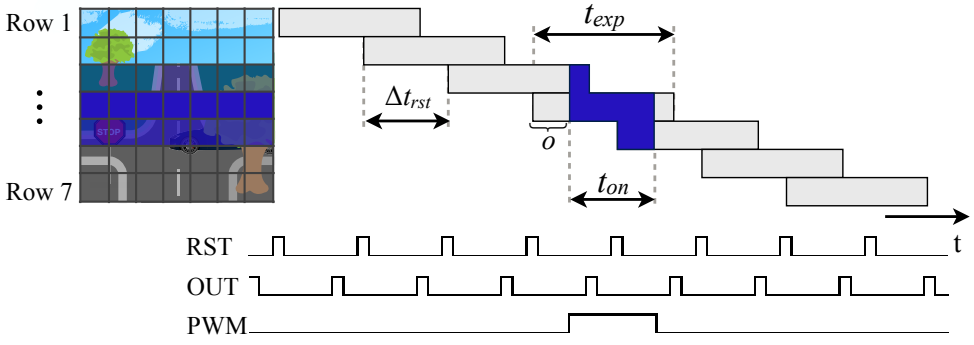
target specific objects or apply optimization techniques. However, our approach makes the attack applicable to dynamic multi-object scenes.

6.4 Exploiting the Rolling Shutter

Preliminaries. The rolling shutter attack is based on the consecutive exposure and readout of individual pixel rows. Each row is exposed for a short period t_{exp}, often in the range of microseconds, before the exposure of the next row starts. The time delay Δt_{rst} between the reset pulses of consecutive lines can be exploited to inject noise that causes partial distortion of the captured image. If a modulated bright light, e.g., a laser, is pointed at the target camera, additional light can be selectively injected into rows of the image. The proportion of the image that is disturbed is controlled by the time t_{on}, which determines how long the laser is switched on (i.e., is emitting light). The attack is illustrated in Figure 6.3, which depicts the row-wise acquisition of a single frame. In Figure 6.3a, the readout starts at the top-most row, the laser is inactive, and consequently, no image disturbances occur. At some point, the laser is turned on and the row that is currently exposed, e.g., the 7^{th} row in Figure 6.3b, is illuminated and the original image information is perturbed. Before the next line is exposed, the laser is switched off, and the rest of



(a) Short exposure time.



(b) Long exposure time.

Figure 6.4: Overview of the different parameters that influence the size of the injected distortion. The signals RST (reset) and OUT (readout) represent the clock cycles for the reset and readout scan, while the pulse-width modulated (PWM) signal indicates when the laser is on. Since some of the rows are only partially illuminated, the distortion becomes lighter towards the edges.

the frame is captured without any further interference, see Figure 6.3c.

In the following, we first introduce the principles behind the attack and then we show how an adversary can accurately control the number of rows affected by simply reading parameters from a camera's technical datasheet.

6.4.1 Controlling the Distortion Size

Quickly shining a modulated light onto the camera sensor will inject light in sets of adjacent pixel rows; we refer to each injection as a *distortion*. Depending on the use-case, the attacker might want to introduce large or narrow distortions. The appearance of the distortions depends on several factors. Without loss of generality, we focus on rolling shutters that scroll top-to-bottom along the image sensor. The same modelling applies for left-to-right rolling shutters, although the distortion

would appear vertically rather than horizontally (as in [108]).

Mechanisms behind Distortion Size. The number of pixel rows that are illuminated by a distortion (*distortion size)* depends on four factors: (i) the absolute exposure time of individual rows t_{exp}, (ii) the time the laser is switched on t_{on}, (iii) the time between RST pulses Δt_{rst}, which is determined by the clock speed of the analog-to-digital converter, and (iv) the time delay between the last RST pulse and the moment the laser is switched on, o. Figure 6.4 shows how the four parameters interact to define the distortion size. Based on the four parameters, the adversary can precisely calculate the distortion size:

$$N_o = \left\lceil \frac{t_{exp} - o}{\Delta t_{rst}} \right\rceil + \left\lceil \frac{t_{on} + o}{\Delta t_{rst}} \right\rceil - 1,\ o \in [0, \Delta t_{rst}). \tag{6.1}$$

Since the attacked has no feedback channel, the laser cannot be synchronized with the rolling shutter of the target camera, so the offset o is unknown. Consequentially, the attacker can only learn an upper and lower bound for the number of consecutive rows that will be illuminated, that is:

$$N_{min} = \left\lceil \frac{t_{exp}}{\Delta t_{rst}} \right\rceil + \left\lceil \frac{t_{on}}{\Delta t_{rst}} \right\rceil - 2 \tag{6.2}$$

$$N_{max} = \left\lceil \frac{t_{exp}}{\Delta t_{rst}} \right\rceil + \left\lceil \frac{t_{on}}{\Delta t_{rst}} \right\rceil, \quad \text{with} \tag{6.3}$$

$$N_{min} \leq N_o \leq N_{max}.$$

In modern CMOS image sensors, the exposure time is generally much higher than the clock speed, $t_{exp} > \Delta t_{rst}$, meaning that the exposure times for consecutive rows overlap [160]. As a consequence, it is impossible to affect *individual* pixel rows with the laser, as the additional illumination will affect multiple rows at once, depending on the current t_{exp}. Figure 6.4 shows the frame acquisition for two different t_{exp}, a short and a long one, showing how larger t_{exp} leads to wider overlap across consecutive rows.

Distortion Repetition. The adversary can increase the number of injected distortions per-frame by increasing the laser frequency f. Intuitively, if the laser frequency f matches the frame rate F of the camera, each camera frame will present exactly one injected distortion. To cause interference at multiple parts in the frame at the same time, the attacker can increase f. The number of distortions per frame can be calculated as follows:

$$N_D = \frac{f}{F}. \tag{6.4}$$

When f is not an integer multiple of F, a distortion will span multiple frames and the location of the distortions will change over consecutive frames. For example, when f is slightly higher than F, the location of the laser-injected light will appear to slowly scroll downwards across consecutive frames. Depending on the attack scenario, the adversary can pick f accordingly to enhance, reduce or even prevent the movement of the distortions.

6.4.2 Estimating Parameters

While the mechanism to control the distortion size is consistent, in practice each camera has different parameters. This means that the adversary needs to estimate these parameters for a target camera to control the distortion size, using Equation 6.3 and Equation 6.4. It should be noted that while the attack can be carried out without accurately estimating these parameters, a better estimate will lead to the attack outcome being more predictable (i.e., more effective for the attacker). Across the introduced parameters, t_{on} and f are controlled directly by the adversary, Δt_{rst} and F are camera-dependent but static, t_{exp} is camera-dependent and dynamic. In the following, we describe how an adversary can infer the camera-dependent parameters using the information on the technical datasheet of a camera model.

Estimating Δt_{rst} and F. The operating frame rate F is generally available in the camera specifications (common examples are 25fps or 30fps). The clock speed

Δt_{rst} is also sometimes available, nevertheless, when it is not, it can be estimated precisely using F and the full resolution of the image sensor:

$$\Delta t_{rst} = \frac{1}{F \cdot N_{rows}}, \tag{6.5}$$

where N_{rows} is the number of pixel rows of the image sensor. For some cameras, the size of the physical resolution of the photodiode array (total resolution) is larger than the maximum resolution that can be used to record videos. For example, a 1920×1080 full-HD video may have been recorded on a camera with $N_{\text{rows}} > 1080$. The rolling shutter will still go through *all* N_{rows} pixel rows to capture additional image data, such as colour information, but ignore a portion of them to read frames in a smaller resolution than the physical resolution; the ignored rows are known as *dead areas*. If the target image sensor has dead areas, Equation 6.4, used to calculate the number of injected distortions, has to be extended by the ratio of visible rows $N_{visible}$ and total rows N_{rows}:

$$N_D = \frac{f}{F} \cdot \frac{N_{visible}}{N_{rows}}. \tag{6.6}$$

The sensor's total resolution N_{rows} and video resolution $N_{visible}$ can be found in the camera's technical specifications.

Estimating t_{exp}. During operation, t_{exp} is set automatically by an auto-exposure mechanism that balances the camera t_{exp} with the current ambient light level: dimmer scenes will require larger t_{exp} to produce sufficient quality frames. Different cameras generally have differing light sensitivities and lens apertures, and therefore use different t_{exp} for the same lighting conditions. To estimate t_{exp}, the adversary can use the *minimal luminous exposure* parameter H_v required by the camera to capture a usable frame and measure the current ambient light level E_v, with a light meter, at the time of the attack. Based on E_v and H_v, the exposure time t_{exp} can be calculated:

$$t_{exp} = \frac{H_v}{E_v}. \tag{6.7}$$

In case H_v is not available, the adversary can measure the ambient light level E_v versus t_{exp} relationship by using a replica of the target camera and performing a set

of measurements in controlled conditions. At the time of the attack, the adversary needs to measure E_v, lookup the respective t_{exp} that would be set by the target camera, and adjust the attack parameters accordingly. In Section 6.5.2, we demonstrate that imprecise estimates of t_{exp} do not affect the distortion size substantially.

6.5 Rolling Shutter Model Evaluation

Here we test our assumptions on the rolling shutter functioning by performing a large data collection in a controlled environment.

6.5.1 Experimental Setup

Here, we describe the equipment used to collect the dataset.

Equipment. We only used off-the-shelf hardware that we bought online for less than $90. We report the various components of the recording setup in Figure B.1 All the components were powered via a single portable power bank, making the setup portable and easy to use at any arbitrary location. For aiming we used a 3-way Pan-and-Tilt head [131] together with a camera tripod [166]. We tested different lasers in terms of quality, wavelength (colour), and power. The lasers ranged from cheap 650nm (red), 535nm (green) and 405nm (violet/blue) laser pointers with an output power of 5mW, usually used for presentations [4], to more expensive and powerful semi-professional 532nm (green) and 445nm (blue) continuous wave (CW) diodes with power ratings of up to 1.8W [3]. We noticed that an adversary might use green lasers to leverage the fact that most image sensors use a Bayer-Matrix colour filter array and are therefore more sensitive to wavelengths at around 530nm [17].

Laser Modulation. The laser was connected to a constant current laser driver with Transistor-Transistor-Logic (TTL) support that enabled us to operate the laser in pulsed mode rather than continuous-wave. The TTL was controlled via the hardware Pulse Width Modulation (PWM) controller of a Raspberry Pi using the `pigpio` library [110]. A PWM modulated signal has two variables that can be changed — the frequency and the duty cycle. The frequency specifies how

Table 6.1: Range of variation of the attack parameters we collected for the two cameras, Logitech and Axis. The last column reports whether the parameter can be controlled by the adversary at attack time.

Param.	*Range* [min, max]	*Symbol*	*Adversary Controlled*
Camera Exposure		t_{exp}	✗
- Logitech	[100μs, 2,500μs]		
- Axis	[32μs, 1,000μs]		
Laser on-time		t_{on}	✓
- Logitech	[320μs, 16,000μs]		
- Axis	[50μs, 400μs]		
Laser Frequency		f	✓
- Logitech	30Hz		
- Axis	25Hz		
Camera fps		F	✗
- Logitech	30fps		
- Axis	25fps		

often the PWM signal is set to high per second, in our case, how often the laser is switched on. The duty cycle, given in percent, determines the duration of a pulse, i.e., how long the laser is switched on each time the signal is set to high. In other words, the duty cycle specifies t_{on}. Since t_{on} depends on both frequency f and the duty cycle, we will use t_{on} in the rest of the chapter rather than the duty cycle to make it easier to compare the injected distortions. Equation 6.8 shows how t_{on} is calculated, given the frequency f and the duty cycle D.

$$t_{on} = \frac{1}{f} D. \tag{6.8}$$

Data Collection. We picked two different cameras for our analysis, namely the Logitech C922, a common webcam, and the Axis M3045-V, a dome surveillance camera. We systematically reproduced the attack in controlled conditions by changing t_{exp} and t_{on}, which affect the resulting distortions. While we verified the attack against a more extensive set of cameras that use the rolling shutter mechanism (reported in Appendix B.1), we only performed data collection on these two cameras because they allowed us to control the camera exposure time t_{exp}. We placed the camera and the laser in a closed environment with little or no light,

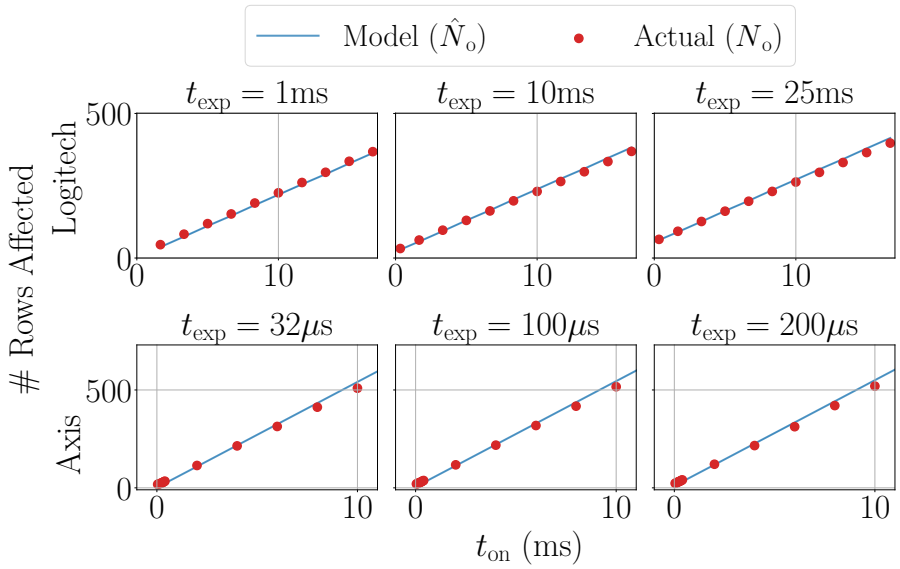


Figure 6.5: Comparison between the model-predicted number of affected rows and the empirical value found by measuring the number in practice on the two cameras.

and we collected a set of videos by varying the parameters of the attack, which are reported in Table 6.1. We set the laser frequency f to be slightly offset from the camera frame rate F so that over consecutive video frames, the location of the pattern slowly iterated over the image rows. This way, the set of collected frames covered all possible locations of the distortion in the image. For each parameter configuration, we extracted the generated *pattern* of distortions from the captured video, which was easily visible as the only coloured and bright pixels on a black background. We did so by creating a mask that filters out black pixels with a threshold, i.e., pixels with per-colour channel value ≤ 10 (range [0, 255]). To account for dead areas, we only considered frames where the entire distortion was visible, and not frames where it was partially hidden in the dead area.

Research Ethics. Data collection involved shining laser pointers onto different cameras, powerful lasers can cause vision loss. For safety reasons, the data collection of this chapter happened in a strictly controlled and enclosed environment indoor, while the experimenters wore appropriate protection.

6.5.2 Accuracy of the Shutter Model

By knowing the shutter model parameters, the adversary can estimate the distortion size N_o. We show here that even inaccurate estimations of t_{exp} only lead to minor changes in the distortion size.

Estimated vs Empirical N_o. In this case, to estimate N_o with Equation 6.1 the adversary only needs Δt_{rst}; as t_{on} is set by them and t_{exp} is known in this experiment (in the next paragraph we analyse the estimation of t_{exp}). We used Equation 6.5 to estimate Δt_{rst}. For the Logitech camera, this led to $\Delta t_{rst} = 46.3\mu s$. In contrast, the Axis camera has dead areas and requires N_{rows} to be determined before Δt_{rst} can be calculated. For most cameras and image sensors, manufacturers provide detailed information about the number of total, effective and recording pixels. However, for the Axis camera used in our experiment this information was not available. By calculating the ratio between visible and invisible distortions, we estimated N_{rows} for the Axis camera to be 2,160 (1,080 visible and 1,080 invisible rows). We therefore use $N_{rows} = 2160$ for the Axis camera, resulting in $\Delta t_{rst} = 18.5\mu s$. We report the result in Figure 6.5, which shows how the prediction of our model closely matches the empirical outcome of the attack, for various t_{exp}, t_{on}, and the two considered cameras. We find this model to be precise regardless of the part of the image that is being hit. Figure 6.5 also shows the confidence intervals for the number of rows affected by the attack, computed over every frame of the data collection videos.

Incorrect Estimation of t_{exp}. As mentioned in Section 6.4.2, the estimation of t_{exp} can be imprecise if information from the datasheet is missing, leading to changes in the resulting distortion size N_o. Using the rolling shutter model of the previous section, we computed the effect of an incorrect estimate of t_{exp} on the resulting N_o. For simplicity, we used N_{max} here (see Equation 6.3) as the effect of o is negligible. Given an estimate of t_{exp} as $\hat{t}_{exp}$, and the resulting actual and estimated number of affected rows as N_o and $\hat{N}_o$, we report the relationship between these four variables in Figure 6.6. The figure shows that in the worst case, where the adversary underestimates $\hat{t}_{exp}$ and t_{exp} is longest, this can lead

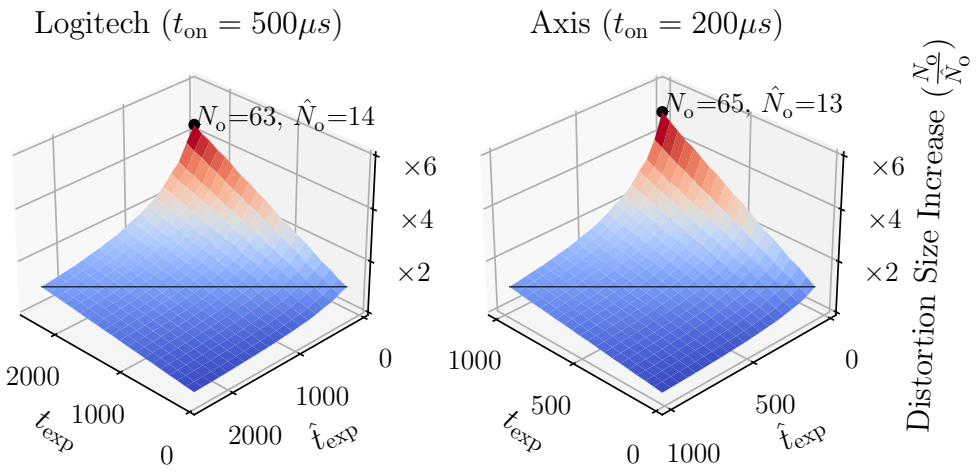


Figure 6.6: Increment in the expected distortion size $\hat{N}_o$ and actual size N_o as the adversary's exposure time estimate t_{exp} diverges from the true exposure time value $\hat{t}_{exp}$.

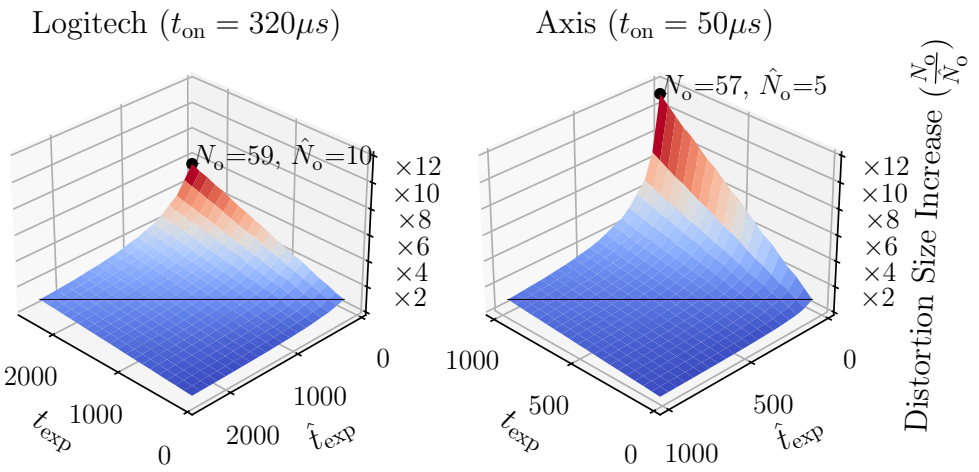
to a distortion that is over 4× larger than expected. The relationship depicted in Figure 6.6 is also dependent on t_{on}.

We report the best and worst divergence observed for the parameters used in our data collection, which depends on t_{on}. The shortest t_{on} is reported in Figure 6.7a, the longest t_{on} is reported in Figure 6.7b. The figures show that for larger t_{on} (which also generally produce better attack success in our evaluation, Section 6.6), incorrect estimates tend to lead to diminishing divergence in expected versus actual distortion size. For Logitech we get a worst-case distortion size increase of 1.6 (down from six in the shortest t_{on} setting), while the same increase for Axis goes from a factor of 11 (in the shortest t_{on} setting) down to a factor of three.

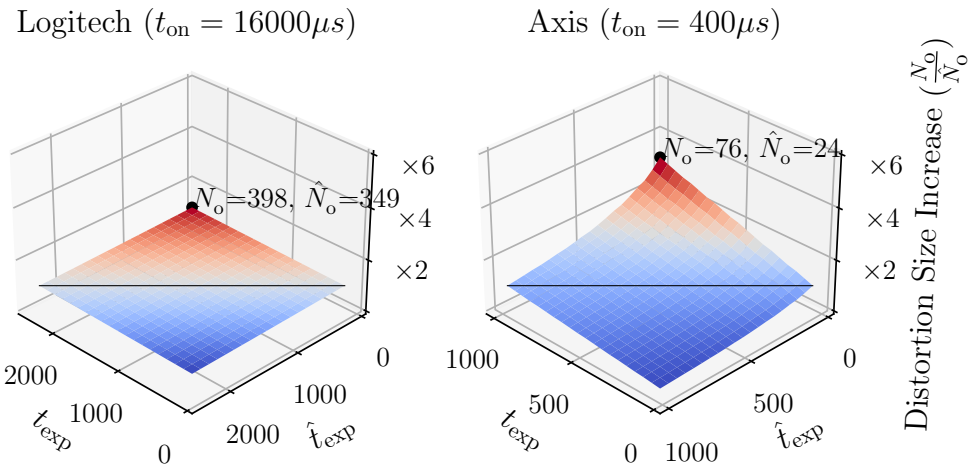
Overall, we find that roughly 70% of the difference in distortion size is within a factor of two of the expected value (i.e., either twice as small or twice as large). This shows that large deviations occur only when the adversary's estimate is far off the real exposure value, showing that the attack's outcome is quite predictable under an educated guess.

6.6 Attack on Object Detection

In this section, we use our collected rolling shutter patterns to evaluate the attack's effect on the *object detection* task, i.e., locating and identifying semantic objects in



(a) Shortest t_{on} used in the data collection of Section 6.5.



(b) Longest t_{on} used in the data collection of Section 6.5.

Figure 6.7: As in Figure 6.6, increment in the expected distortion size $\hat{N}_o$ and actual size N_o as the adversary's exposure time estimate t_{exp} diverges from the true exposure time value $\hat{t}_{exp}$.

frames. We first evaluate the attack effect in principle on two well-known datasets, then we measure how the attack might affect an autonomous driving platform as a whole, where object detection is used together with other components. Most of this evaluation uses the Axis camera as a reference, however, we report examples of an end-to-end attack realization in Appendix B.1 in Figure B.2, where a printout of an image is placed in front of the camera, and the attack is carried out with the laser.



Figure 6.8: Attack evaluation on object detection. A network (Single-shot detector [114]) detects objects in input images, shown with green boxes on the left. We overlay the attack distortions on the image and compute the differences in the detected objects. On the right, colours outline what happened to the object: red=hidden, orange=misplaced, green=unaltered.

6.6.1 Evaluation on Video Datasets

In this section, we first outline the preliminaries for the evaluation and then present the results, along with some insights into the attack, on the basis of two different video datasets.

Defining Attack Success. The goal of object detectors is to identify and locate objects in the input image with 2D boxes, which we refer to as $\{b\}_i^{(o)}$ when they belong to a legitimate image and $\{b\}_i^{(c)}$ when they belong to the attack-corrupted image. Figure 6.8 shows an example of the attack effect on the detected objects: in the rolling shutter attack-corrupted image many objects are mis-detected when compared to a clean input. In practice, to compute the attack effect, for each box, we categorize the effect of the attack on each object box based on the Intersection-over-Union (IoU) between original and corrupted boxes. IoU is a measure of the overlapping area of two boxes, as a proportion of the total, combined area; it is commonly used to evaluate the accuracy of a box prediction compared to a ground truth box. We measure the attack outcome with the effect that the distortions have on the detected boxes with the following:

- **Hidden**: for a box $b_i^{(o)}$ its object has been *hidden* by the attack if there is no box in the corrupted image with $\text{IoU}(b_i^{(o)}, b_j^{(c)}) > 0.5$.

- **Misplaced**: for a box $b_i^{(o)}$ its object has been *misplaced* by the attack if there is a box in the corrupted image with IoU smaller than 0.95 and same output class.

- **Appeared**: for a box (in the corrupted image) $b_j^{(c)}$, its object has *appeared* if there is no box in the original image with $\text{IoU}(b_j^{(c)}, b_i^{(o)}) > 0.5$.

We define the attacker's goal to be the hiding of objects, the pattern which hides the most objects will be preferred in the attack phase.

Attack Pattern Collection. We simulate the attack to find the best configuration of adversary-controlled parameters (i.e., t_{on} and f): the one that leads to the highest number of hidden objects. To collect the generated distortions for the simulation, we repeated a data collection procedure on the Axis camera, as in Section 6.5.1. Given that the Axis camera's frame rate is 25 fps (F=25), we collected patterns for four different laser frequencies, f = 25, 250, 500, and 750Hz. Since the image sensor of the Axis camera has dead areas, the number of distortions can be calculated using Equation 6.6. We collected six different duty cycles, from 0.1% to 40%,f and duty cycle lead to different t_{on} (see Equation 6.8). It should be noted that t_{on} refers to the time the laser is on per-distortion rather than per-frame (each frame will present multiple distortions based on Equation 6.6). We empirically found that the exposure times the camera would pick on a sunny and cloudy day outdoors are 32μs and 200μs, respectively, we therefore collected patterns with this range of t_{exp}.

Simulation Setup. We simulate the attack effect on a subset of two different video datasets, namely BDD100K [236], a large and diverse driving video dataset, and VIRAT [145], a dataset of surveillance footage. We randomly pick 50 videos from the BDD100K and, due to the longer duration, 25 videos from the VIRAT dataset, for each video, we extract every 10^{th} frame. For each parameter configuration (f, t_{exp}, t_{on}), we extracted randomly selected patterns, four for BDD100K and ten for VIRAT. We then generate corrupted frames by overlaying the extracted patterns over each extracted legitimate video frame, and we calculate the attack effect (see Figure 6.8).

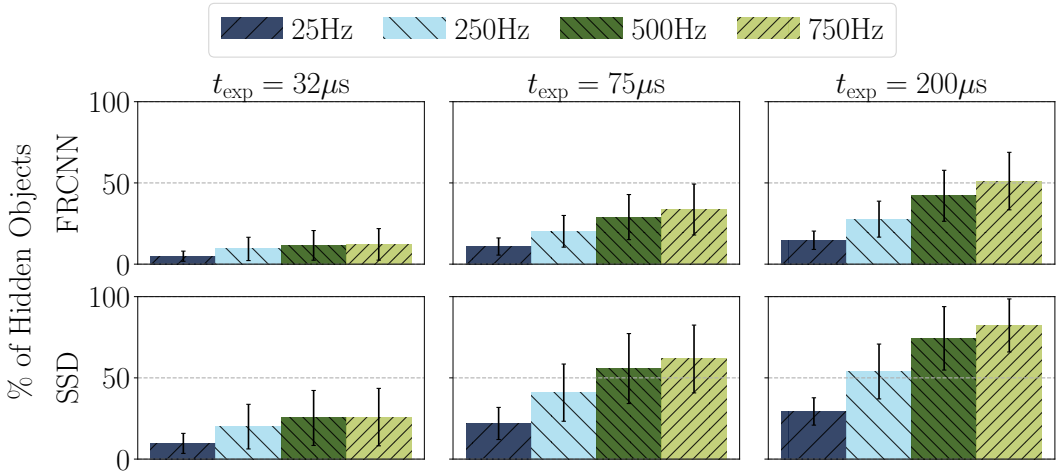


Figure 6.9: Choosing attack parameters: percentage of hidden objects caused by the rolling shutter attack for frequencies $f = 25$, 250, 500, and 750Hz, various t_{exp} and the two object detection models on the BDD100K and VIRAT dataset. Error bars show the standard deviation over the results per-video.

We evaluated two well-known state-of-the-art object detection models, Single Shot Detector (SSD) [114] and Faster-RCNN [163] (FRCNN), both using Inception v2 backbone network, both taken from the Tensorflow Model Zoo. All images were scaled down to 640×360 before being input into the object detectors; it is common to use lower resolutions for performance reasons (in fact, SSD further downsizes images to 300×300 pixels).

Results

Here, we present the results of the simulations and discuss which parameter configuration performs best.

Choosing Attack Parameters. We report the percentage of objects hidden in the input frames for the various frequencies and exposure times in Figure 6.9. It should be noted that for all four frequency settings, the image area covered by the laser-injected light is the same. As the frequency increases, t_{on} decreases, which results in smaller, but more frequent distortions (see Equation 6.8). However, the sum of t_{on} per frame stays the same. Figure 6.9 shows a clear increment in the ratio of hidden objects when using larger f. For $f = 750$Hz, we obtained the largest number of hidden objects. At $t_{exp} = 200\mu$s, 51% and 82% of objects

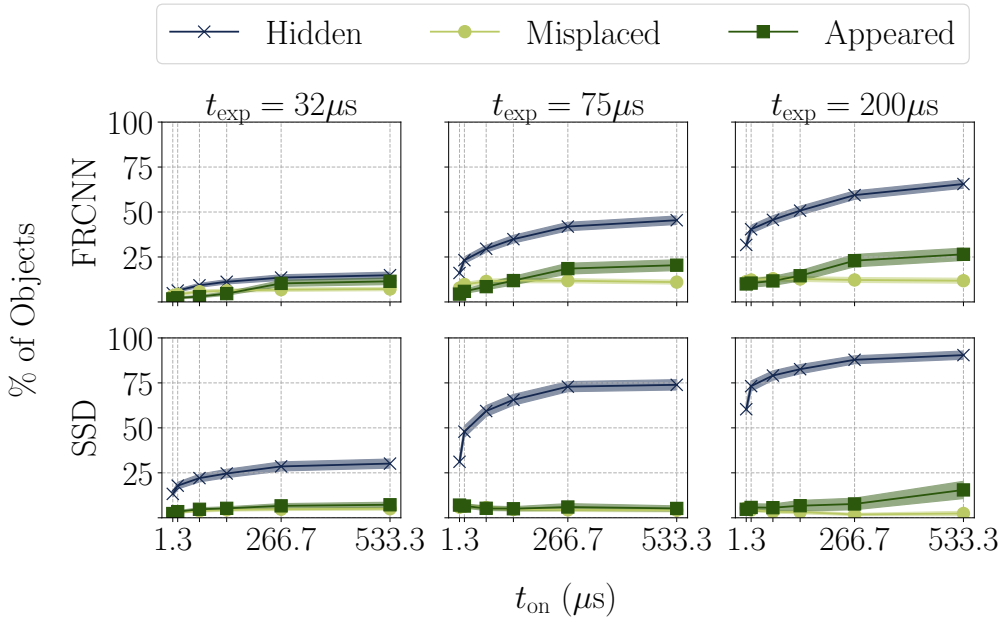


Figure 6.10: Percentage of hidden, misplaced, and appeared objects in the evaluation of the BDD100K and the VIRAT dataset, for various t_{on}, exposure times t_{exp}, and the two models used in the evaluation (SSD and FRCNN). Shaded areas show 99% confidence intervals.

were hidden for FRCNN and SSD, respectively. We noted that the narrower distortions generated by higher frequencies strongly affect the object detection task, in particular, for small objects. Given these results, an adversary would select 750Hz as the modulation frequency for the attack. In line with this, we limit the presentation of the remaining results to this 750Hz configuration.

Effect of Exposure. As the adversary cannot control the exposure value t_{exp} of the camera during the attack (see Section 6.4), we investigated the effect of varying exposures. Figure 6.10 shows the ratio of hidden, misplaced, or appeared objects for various exposure times t_{exp} and laser on-times t_{on}. With increasing t_{exp} and thus greater distortion, more objects are misplaced or hidden. As expected, an increasing t_{on} also leads to more objects being hidden. However, the number of misplaced or appeared objects does not change significantly. In contrast, for short t_{exp} less of the injected light is absorbed, leading to less intensive distortions and fewer hidden objects. While the weak illumination could be compensated with a more powerful laser, this highlights how in dimmer ambient light settings, the attack

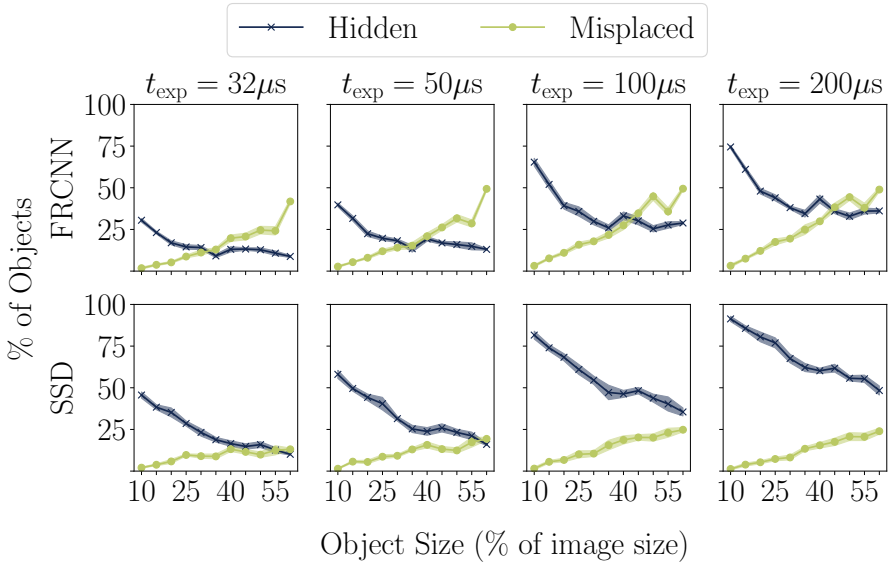


Figure 6.11: Percentage of hidden and misplaced objects in the evaluation, for increasing object sizes expressed in percentage of image size, various exposure times t_{exp} and the two models used in the evaluation (SSD and FRCNN). In all cases t_{on} was set to 266.6μs.

requires fewer resources on the adversary's side. For the largest evaluated exposure ($t_{exp} = 200\mu$s) and $t_{on} = 533\mu$s, our attack can hide up to 90% of objects in an image for SSD and up to 65% for FRCNN. This is in line with the fact that SSD is designed with more focus on speed of execution rather than prediction accuracy: SSD can execute one inference every 30ms, while FRCNN needs 89ms.

Effect of Varying Object Size. In order to estimate which type of objects are more or less affected by the attack, we investigated the attack effect on various object sizes. We grouped objects based on the size of the ground-truth 2D-box containing them, computed as the length of the box diagonal. We consider the relative size of the object box diagonal with respect to the diagonal of the full image, for example, an object of size 10% means that its diagonal length is 10% of the diagonal of the full image. The ratio of hidden and misplaced objects with increasing dimensions is shown in Figure 6.11. We can see a clear trend that larger objects tend to be misplaced rather than hidden, while small objects tend to be hidden in both models. If the attacker wants to hide a specific object from a non-stationary camera, for example, hiding traffic signs from approaching cars, they must take into account that although the attack can easily interfere with the detection of small,

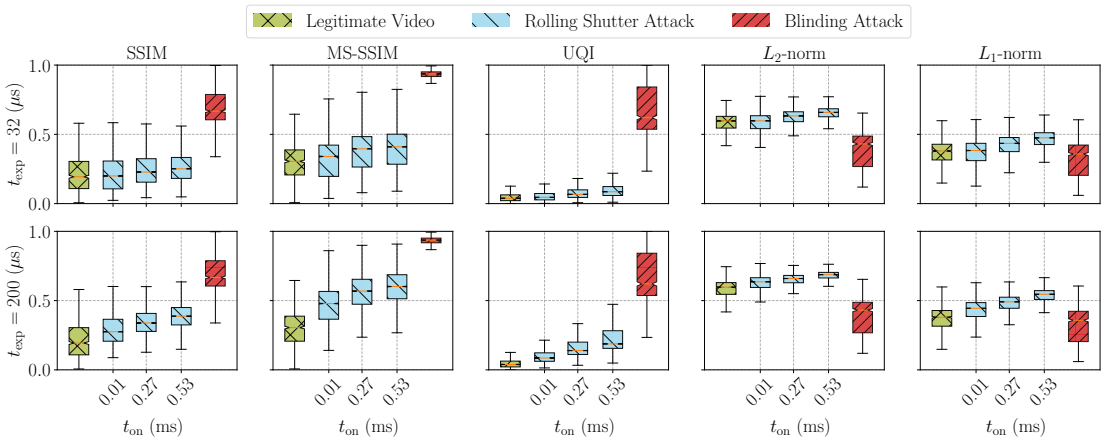


Figure 6.12: Image perturbation metrics for the selected frames in the BDD100K driving video dataset. The plots report the different perturbations measured across consecutive frames on (i) legitimate videos, (ii) when applying the rolling shutter attack, and (iii) when applying a blinding attack. The upper and lower part report the results for $t_{exp} = 32\mu$s and $t_{exp} = 200\mu$s, respectively. For both exposures, f is set to 750Hz. High values correspond to highly perturbed images. The values for SSIM, MS-SSIM and UQI are flipped on the y-axis to represent *dissimilarity*.

more distant objects, the object is more easily detected the closer it gets to the camera. In line with what was shown in the previous paragraph, Figure 6.11 also shows how FRCNN is more resilient to the attack compared to SSD, in particular for smaller objects. For objects with size $< 10\%$, FRCNN results in roughly 20% more detections under attack compared to SSD.

6.6.2 Comparison with Blinding Attack

In this section, we compare the image perturbation introduced by the rolling shutter attack, during which the laser is modulated and only small parts of the image are affected, to a blinding attack, where the laser is constantly on and the entire frame is disrupted. We show that our attack introduces only minor interference, which is harder to detect across consecutive video frames.

Method. To measure the perturbation caused by our attack, we repeated parts of the simulation on the BDD100K dataset as described in Section 6.6.1. We randomly selected 100 videos from the `validation` set and extracted 10 evenly spaced frame pairs, whereas each pair consisted of two consecutive frames. For clarity, we refer to those frames as *previous* and *current*. We then calculated the amount of

perturbation between those consecutive frames to evaluate how the *current* frame has changed compared to the *previous* frame during (i) normal operation, (ii) a rolling shutter attack, and (iii) a blinding attack. For (ii) and (iii), we generated corrupted frames by overlaying the malicious patterns, as described in Section 6.6.1, onto the *current* frame of each pair. We measured the image perturbation with several metrics from adversarial machine learning literature [178]. For each scenario and pair of frames, we calculated five different metrics: the Structural Similarity (SSIM), Multi-Scale SSIM (MS-SSIM), Universal Image Quality Index (UQI), ℓ_2- and ℓ_1-norm. The values for SSIM, MS-SSIM and UQI are bounded in [0, 1] and have been inverted in our case to represent *dissimilarity* rather than similarity. In addition, we removed perturbations with a value of 0, as these correspond to stationary frames. To measure the detection performance for pairs of anomalous frames, we set up a simple binary-threshold approach, i.e., if the dissimilarity metric value is higher than a certain threshold, we label the frame pair as anomalous. We report detection results with the area under the receiver operating characteristic curve (ROC-AUC), where a score of 1.0 corresponds to perfect detection while a score of 0.5 corresponds to random guess.

Results. Figure 6.12 shows the frame-to-frame perturbation introduced by the blinding and the rolling shutter attack for $t_{exp} = 32\mu$s and $t_{exp} = 200\mu$s. The results indicate that for short t_{exp} our attack is causing interference that is similar across all metrics to the expected level of perturbation seen in consecutive legitimate video frames. In contrast, the blinding attack generates much larger perturbations compared to our attack, which are easily detected. We found that SSIM, MS-SSIM and UQI could detect almost 100% of blinding attacks with the binary-threshold approach, i.e., ROC-AUC of 0.99, 1.0 and 1.0, respectively. While higher t_{exp} causes more perturbation, the level of interference is still closer to that for legitimate video frames than for blinding. In comparison, for most cases our attack was indistinguishable from legitimate frame-to-frame changes when using the same threshold approach. The ROC-AUC scores for the detection of the rolling shutter

attack with $t_{exp} = 32\mu$s and $t_{on} = 0.53$ms are 0.61, 0.68 and 0.80, for SSIM, MS-SSIM and UQI, respectively. Similar results (0.67, 0.79, 0.81) were observed for $t_{exp} = 200\mu$s and $t_{on} = 0.01$ms. However, as expected, the attack is more easily detected with increasing t_{on}. Nevertheless, in most cases our attack can bypass detection that is based on common structural similarity image metrics. We found that ℓ_2- and ℓ_1-norm also do not perform well in the detection (similar scores as UQI), although they slightly outperform MS-SSIM and SSIM. However, it should be noted that L_2 and L_1 only look at pixel-wise changes rather than the image structure. While they are commonly used in adversarial machine learning settings, they might be unsuited for the task of detecting anomalies in videos.

6.7 Countermeasures

The vulnerability that enables the rolling shutter attack is inherent to CMOS image sensors, meaning that attacks cannot be prevented entirely without changing the hardware design. However, detection in software can mitigate the attacks' impact, while other countermeasures are applicable in some special cases.

6.7.1 Hardware Countermeasures

A clear design option is to use a camera with a global shutter mechanism. This does not stop the injected light but prevents rolling shutter attacks as the image rows are exposed at the same time, making the attack more similar to a blinding attack. However, this requires a design-time choice and also carries additional component costs for incorporating frame memory or using CCD image sensors. These costs increase dramatically for existing systems, as the replacements must be retrofitted as well.

6.7.2 Sensor Redundancy

While camera redundancy could quickly increase manufacturing costs, it is a straightforward approach to protect against rolling shutter attacks. If multiple cameras were in place, in particular in the case of moving targets (e.g., cameras

mounted on autonomous cars), both aiming and syncronization of the attack would become a complex practical obstacle for the adversary. Similarly, the existing use of multiple sensors allows the system to continue operating effectively despite the performance degradation caused by the rolling shutter attack (e.g., in autonomous driving, with multiple cameras or LiDAR). However, it is not trivial to reliably determine which sensor input is malicious or untrustworthy in order to stop relying on it. Furthermore, while a system may be tolerant to the degradation of its sensor inputs, there is still some finite budget for degradation that it can withstand before it becomes unsafe or ineffective.

6.7.3 Deep Learning Based Detection

We explore the re-use of the deep network used for the computer vision task to detect the presence of the attack. We re-use the backbone network using the insight that appearance of the distortions can be captured by early convolutional filters, designed to capture low-level input features. In the following we describe our custom detection architecture and evaluate it on our dataset.

Setup and Training. Given the backbone network, we added a network head as follows: on top of the first convolutional layer (after pooling, when present) we added one additional convolution layer with 32 filters, then we used global maximum pooling and added a binary classification layer which predicts whether the input is corrupted by a rolling shutter attack. We also used dropout and selected leaky rectified linear unit (ReLu) as the activation function, which has been found to lead to faster convergence and improved performance [229]. We created a dataset to train our detection by splitting both collected patterns and video frames into training, validation and testing subsets, with ratios 60%, 20% and 20%, respectively. As in Section 6.6.1, corrupted frames were obtained by overlaying the pattern onto the legitimate frame. We used all patterns collected for the Axis camera ($\sim$6,500, see Section 6.5.1 and 180 videos randomly selected from BDD100K, which we sampled every 10^{th} frame (20,880 frames). We used cross entropy loss and trained the added parameters with Adam, with learning rate 10^{-4} and for 2 epochs.

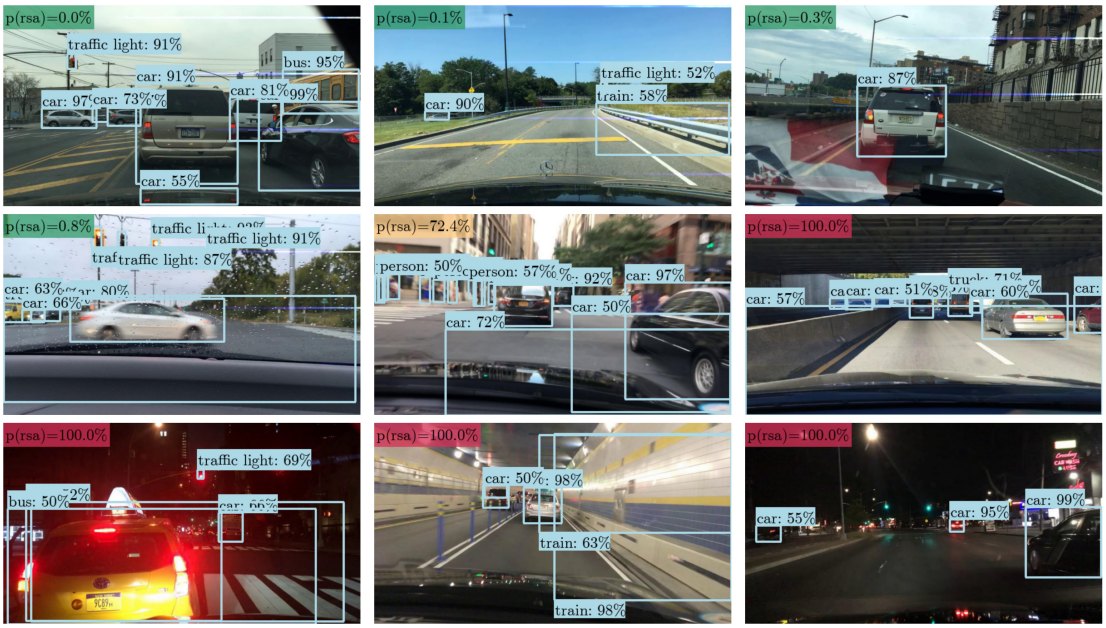


Figure 6.13: Examples of misclassified frames by the rolling shutter attack detection head. Each image reports the output of the detection head in the top-left of the image, as the probability of the frame being compromised by an attack. A number of false positives occur with very thin distortions, false negatives happen in low-light conditions, where other artificial light is present.

Table 6.2: Rolling shutter attack detection performance for the backbone networks. False Negatives and False Positives are calculated over 3,560 test frames. The column *Overhead* represents the computational overhead for the detection mechanism in percentages on the baseline execution (without the additional network head), with the standard deviation in parentheses.

Model	*Backbone*	*Accuracy*	*FN*	*FP*	*Overhead in % ($\pm\sigma$)*
SSD	Mobilenet v2 [172]	0.9882	32	10	+0.57 ($\pm$4.059)
SSD	Inception v2 [199]	0.9949	18	0	+0.13 ($\pm$4.242)
FRCNN	Inception v2 [199]	0.9955	7	9	-0.11 ($\pm$5.045)
FRCNN	Resnet-50 [85]	0.9899	25	11	+1.38 ($\pm$5.119)

Results. The test set accuracies for the two object detectors with different backbones are presented in Table 6.2. The detection is efficient, all networks obtain over 98.8% accuracy on the task, although MobileNet v2 [172] performs slightly worse due to design compromises for faster execution speed. Table 6.2 also reports the number of False Negatives (FN) and False Positives (FP). This shows that setting the detection threshold to 0.5 would generally tend to favour low false positives, but a different trade-off can be sought by changing the threshold for

detection. We report examples of misclassified frames in Figure 6.13.

To analyse the performance impact of the proposed detection mechanism, we measured the required inference time for all images in the test set (batch size = 1) with and without the added layers on an NVIDIA TITAN RTX. The last column of Table 6.2 shows the computational overhead of the detection, indicating that it is negligible. As such, this detection measure could be applied as a software patch to protect existing systems. Additionally, the method represents a general approach that can be applied to protect any computer vision network from rolling shutter attacks.

6.8 Limitations

In this section, we provide an overview of the limitations of our evaluation and the rolling shutter attack itself.

6.8.1 Evaluation Limitations

The evaluation presented in this paper is a first step in raising awareness about the rolling shutter attack and its impact. Nevertheless, our attack evaluation has some limitations that an attacker might need to consider when performing the attack. Although the patterns we used have been collected from real cameras under different close to real-world settings, our simulations do not perfectly reflect the dynamic behaviour of the target camera under attack. Depending on the camera, numerous factors and mechanisms, may influence the appearance of the injected distortion. We mimicked the behaviour by exhaustively simulating the attack with different patterns that correspond to different exposure times. Nevertheless, depending on the scenario, an attacker might not always freely choose parameters f, t_{on} while maintaining the desired brightness of the injected distortions. For instance, in bright settings, short t_{on} and non-powerful lasers might not generate enough light to be captured by the camera. Furthermore, we have not explicitly considered the effect of auto-exposure and auto-focus mechanisms. While we found that these are generally not triggered for most of our attack configurations, we cannot exclude

that these mechanisms might come into play as the attack is executed (this will depend on camera-specific factors). This will have an effect on the appearance of the distortions and may cause warnings in the vision-based system.

6.8.2 Attack Limitations

Due to the characteristics of the rolling shutter attack, there are some limitations that cannot be easily solved.

Limitations in Dim Conditions. Contrary to what humans perceive, the amount of ambient light does not scale linearly between what is perceived as bright versus what is perceived as dark. For example, on a clear day, the amount of ambient light at midday is >30k lx, at sunset, it is ∼400 lx while at night it is close to 0 lx. Cameras are no exception: in poor lighting conditions, cameras have to increase the exposure time *dramatically* to allow enough photons to hit the sensor. This leads to a very large overlap during the exposure of consecutive pixel rows resulting in fine-grained distortions not being achievable. Therefore, in dim conditions, even a very short t_{on} will impact a significant proportion of image rows, limiting the amount of high-frequency distortions from which the attack benefits.

Timing and Synchronization. While it is, in principle, possible to target specific rows during the attack, the adversary would require tight synchronization with the modulation signal and the row reset signal of the image sensor in order to do so. In other words, if the adversary wants to alter the content of the i-th row of the frame to hide a specific object, they need to know exactly (to the microseconds) when the target row is exposed. In case the attacker has access to the video feed, such synchronization could be obtained through an iterative process. However, as in our threat model, it is generally reasonable to assume that the attacker does not have this information and therefore needs to design their attack accordingly.

Frame Rate Accuracy. In order to control the movement of distortions between consecutive frames, the laser modulation must vary in relation to the camera frame rate, or a multiple of it. Accurate knowledge of the frame rate allows the attacker to keep the progression of the pattern in line with their intentions. The

frame rate F of a camera is typically specified in the datasheet, although minor timing inaccuracies might cause the frame rate to deviate slightly (e.g., 30.0025 fps instead of 30 fps). However, this only causes the pattern to drift very slowly if the attack is carried out over an extended period of time and is negligible for typical attacks, with short duration.

6.9 Summary

In this chapter, we have highlighted an inherent vulnerability of the electronic rolling shutter mechanism implemented in most modern CMOS image sensors, which can be exploited to inject fine-grained distortions into the captured frames by targeting the camera sensor directly. Compared to prior work, we limited the number of assumptions made in the threat model to the minimum. We were able to show that even in black-box settings and with no other capability than purchasing off-the-shelf-equipment, an attacker is able to disrupt the correct behaviour of a computer vision model simply by corrupting the camera inputs (in fact, the attack considered in this chapter is rather realizable as depicted in Figure 1.1). At the same time, we showed that our attack is stealthier than a global blinding attack across several anomaly metrics, making it non-trivial to detect. Our findings outline a weakness in using cameras with CMOS images sensors in settings with potentially-adversarial camera inputs, and particularly for computer vision applications in safety-critical systems. In the following chapter, we explore the threat of altering a scene with adversarial patches (see Section 2.4), but rather than focusing on classical DNN architectures we consider the novel perception architectures known as Vision Transformers.

7

Dot-Product Attention Vulnerability for Adversarial Patch Robustness

Contents

7.1 Introduction 137
7.2 Robustness of Vision Transformers 140
7.3 Preliminaries and Threat Model 142
7.4 Robustness of Dot-Product Attention 144
7.4.1 Gradient of Dot-Product Self-Attention 144
7.4.2 Robustness of Dot-Product Attention Weights 146
7.5 Attention-Fool 147
7.6 Evaluation of Attention-Fool 150
7.6.1 Evaluation Setup 150
7.6.2 Attention-Fool on Vision Transformer 150
7.6.3 Attention-Fool on DETR 153
7.7 Limitations 155
7.8 Summary 156

7.1 Introduction

The previous chapters focused on convolutional networks as these have been the predominant tools for image processing tasks in recent years. Nevertheless, recent breakthroughs in image recognition have highlighted how transformers [212], common in natural language processing, have the potential to improve performances

even in the image domain [56, 202, 116]. These successes have inspired various architectures' design for tackling image-tasks such as object detection [35, 18], semantic segmentation [192, 241], image synthesis [218, 65, 91], video understanding [194, 74, 74], and low-level vision tasks [232, 109, 219]. In transformers, the attention mechanism – network layers which enhance parts of input data while diminishing other parts – plays a prominent role in their success for different language and image processing tasks. Among other mechanism, the dot-product attention [212], is one of the most common components of architectural design that model global interactions among different input or feature patches. Understanding robustness of these dot-product attention-based networks against adversarial attacks targeting security-critical vulnerabilities is important for their deployment into real-world applications.

The increasing interest of research in transformers across vision tasks has motivated several recent works [138, 128, 2, 139, 19, 222, 20, 177, 7, 6] to study their robustness against adversarial attacks. Some prior works [138, 2, 19, 177] have hypothesized that transformers are more robust than convolutional neural networks against these adversarial attacks. On the other hand, [128, 222, 7, 6] have shown that vision transformers are not an exception and are also prone to adversarial attacks. In particular, [222] shows that an adversarial attack can be tailored to transformers to achieve high adversarial transferability. These findings indicate that robustness evaluation protocols (attacks) designed for CNNs might be suboptimal for transformers. In the same line of work, here we identify a principled vulnerability in the widely-used dot-product attention in transformers that can often be exploited by image-based adversarial patch attacks.

We propose a family of adversarial losses and attacks called *Attention-Fool* that directly acts on the dot-product outputs (pre-softmax dot-product similarities). These losses optimize the adversarial patch to maximize the dot-product similarity of all the queries to a desired key (typically the key whose token corresponds to the input region with adversarial patch, see Figure 7.1). This approach maximizes the number of queries that attend to the targeted key in an attention head, refer to Figure 7.2 for an illustration. We apply these losses at multiple attention heads and

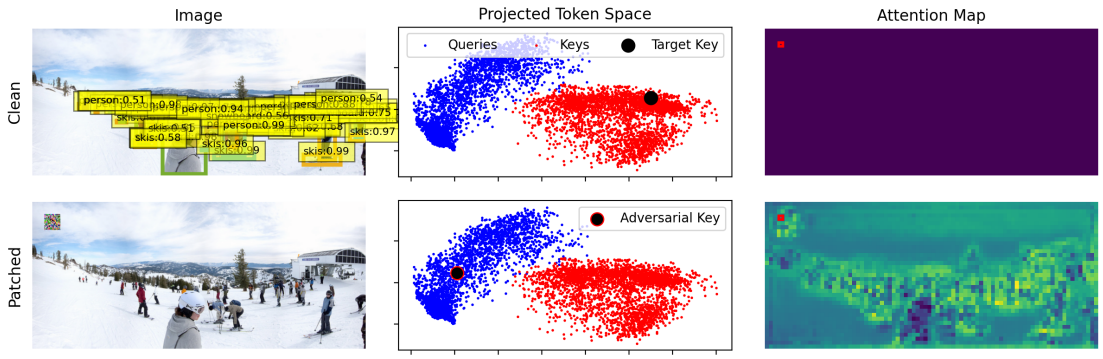


Figure 7.1: Comparison of clean and adversarially patched input for DETR [35]. The patch shifts a targeted key token towards the cluster of query tokens (middle column). For dot-product attention, this effectively directs the attention of all queries to the malicious token and prevents the model from detecting the remaining objects. The right column compares queries' attention weights to the target key, marked by a red box, between clean and patched inputs and highlights large attention weights drawn by this adversarial key.

layers in transformers to misdirect the model's attention from the image content to the adversarial patch in order to encourage wrong predictions. We show that our *Attention-Fool* adversarial losses improve gradient-based attacks against different vision transformers ViTs [56] and the data-efficient vision transformer (DeiT) [202] for image classification and also significantly improve the attack's effective on the object detection model DETR [35].

Contributions Our contributions can be summarized as follows: We

- Identify necessary conditions for the existence of a vulnerability in dot-product attention weights.
- Provide reasons why this vulnerability may not be fully exploited by vanilla gradient-based attacks.
- Introduce Attention-Fool, which is based on losses defined directly on the dot-product similarities.
- Show that transformers for image classification and object detection are highly sensitive to small adversarial patches.

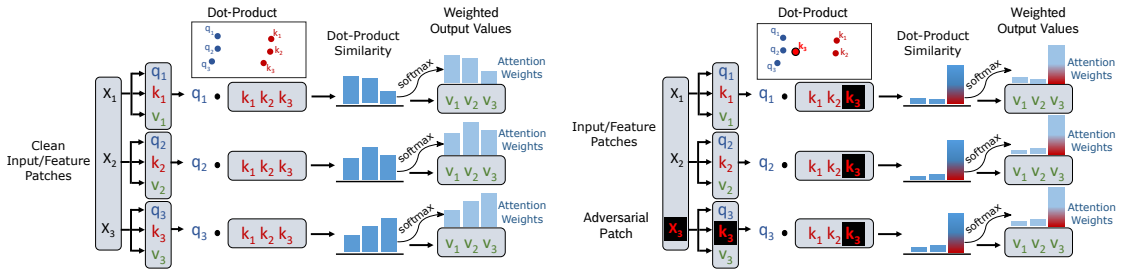


Figure 7.2: Example of dot-product (self-)attention mechanism for clean (left) and adversarial patch attack (right) settings. Here, q, k, and v stand for projected queries, keys, and value tokens of input features. Left: dot-product attention computes dot-product similarities of a query with all keys, which is later normalized using softmax to obtain per token attention weights. These are multiplied with value tokens to control their contributions in an attention block. Right: *Attention-Fool* losses optimize the adversarial patch in input at X_3 to maximize dot-product similarity of all the queries to the key k_3 (marked in red/black) , which corresponds to moving k_3 closer to the queries cluster. The increase in dot-product similarity of queries with k_3 misdirects the model's attention from image content to adversarial patch.

7.2 Robustness of Vision Transformers

Recent successes of ViT [56] across different vision tasks have inspired several works [138, 128, 2, 139, 19, 222, 177, 7, 6] that study their robustness against adversarial attacks. While some works [2, 19, 177, 138, 138] have hypothesized that ViTs are more robust than CNNs under different white- and black-box transfer attack settings (including universal adversarial patches), others [128, 139, 222, 20] have claimed that ViTs are at least as vulnerable as CNNs. In particular, [2] attributed ViT's presumed robustness to its ability to capture global features of the input. [19, 177] analyzed that ViTs are less sensitive to high frequency perturbations and rely on low-frequency features that are robust to adversarial perturbations. [177] have stated that ViTs have better certified robustness than CNNs when tested with denoised randomized smoothing technique and have further suggested that adversarially trained ViTs don't bring further advancements and have comparable robustness to their CNN counterparts. However, they observed catastrophic overfitting for ViTs when using fast adversarial training [225], suggesting the need for improvements in adversarial training.

On the other hand, [20] have claimed that ViTs pretrained on larger datasets are at least as vulnerable as CNN counterparts. [128, 139, 222] suggested that

adversarial transferability between ViTs and from ViTs to CNNs can be improved by carefully tailoring adversarial attacks to the transformer architecture. [128] explored ensembling of CNNs and ViT models to improve attack transferability. The authors of [139] proposed a self-ensemble technique: split a single ViT model into an ensemble of networks to improve the transferability. [222] proposed an attack that skips attention gradients to generate highly transferable adversarial perturbations.

In this chapter, we aim to understand robustness of the widely-used dot-product attention in transformers and expose its vulnerability to adversarial patch attacks. These are constructed by tailoring adversarial objectives to specifically fool the dot-product attention mechanism. Concurrent to our work, [7, 6] also fool the attention mechanism in transformers using image patches. [6] uses a conventional adversarial patch attack to mislead the model attention to the perturbed patch to promote wrong predictions. They also show that the adversarial patch generalizes to different positions in the image. Patch-Fool [7] attack the model by increasing the attention weight of an adversarial patch at a predefined attention layer. [7] selects a salient image patch to contain the adversarial patch, based on how much attention the image patch draws in the clean image, and optimizes the adversarial patch to increase its attention weights (post-softmax) from all other patches to attack the model. We discuss limitations of optimizing post-softmax attention weights as in [7] and propose to optimize directly the pre-softmax dot-product similarities of query and keys to draw the attention to the adversarial patch.

Besides optimizing the content of the patch, there exist also methods for optimizing the location of the patch in the input. Joshi et al. [92] select the location of adversarial patches based on token saliency. Fu et al. [7] selects a salient image patch to contain the adversarial patch, based on how much attention the image patch draws in the clean image. Our work abstracts from selecting the patch location and focuses on the loss for the optimization of the patch content. Any patch location-selection method is orthogonal and could be combined with our Attention Fool-losses.

This work focuses on dot-product attention, widely used in transformers. We leave investigation of vulnerabilities of other attention mechanisms [107, 245, 217] to future work.

7.3 Preliminaries and Threat Model

In this section we introduce the threat model, the objective for patch robustness evaluation, then we summarize an optimization algorithm for finding adversarial patches and finally we recap scaled dot-product attention.

Adversarial Patch Threat Model. Compared to previous chapters, this chapter considers a more generic threat model, where an adversary is able to introduce a small (i.e., in terms of number of image pixels) arbitrarily-coloured patch in input images. The adversaries' goal is to cause misclassifications of objects, both scenarios of image classification (Section 7.6.2) and object detection (Section 7.6.3) are considered. As in previous chapters, we consider a white-box attacker which has access to the model's internals (this includes intermediate network layers outputs). As in [7, 46], we do not consider the imperceptibility of the patch to be a requirement. We also focus on a single-patch fixed-location threat model. Note that methods for choosing the patch location [7, 92] could be combined with the proposed approach. As discussed in Section 1.1, while this threat model may not necessarily correspond to a directly realizable attack (in fact this chapter appears earlier on the realizability scale of Figure 1.1), finding worst-cases outcomes under these assumptions is useful to estimate a lower bound to model safety in practice [36].

Generic Objective Formulation. Given a (normalized) image $x \in [0,1]^{3\times h\times w}$ and associated label y, we craft an adversarial patch $p \in [0,1]^{3\times p_h\times p_w}$ with $p_h \ll h, p_w \ll w$ that maximizes the following objective:

$$\arg\max_p \ \mathcal{L}(f(\mathcal{F}(x,p,\mathrm{L})),y), \tag{7.1}$$

with L specifying the location of the patch p within the larger image x, $\mathcal{F}$ a function to apply the patch onto the image (i.e., overwriting an input region for a given

size), and f being the target model. For classification tasks, we are interested in the 0-1 loss $\mathcal{L}(x, y) = \mathcal{L}_{0,1}(x, y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases}$, which corresponds to finding patches that maximize misclassifications. We note that the constraint $p \in [0, 1]^{3 \times p_h \times p_w}$ can be rewritten as $||p - 0.5||_\infty \leq 0.5$.

Optimization Algorithm. For the patch optimization, throughout the paper, we use Projected Gradient Descent (PGD) [124] for ℓ_∞-norm bounded perturbations:

$$p^{t+1} = p^t + \alpha \cdot \text{sgn}\left(\nabla_p \mathcal{L}(\mathcal{F}(x, p, \text{L}), y)\right). \tag{7.2}$$

We initialize p^0 uniform randomly from $[0, 1]^{3 \times p_h \times p_w}$. Since the 0-1 loss $\mathcal{L}_{0,1}$ is piecewise-constant, it is not suited for gradient-based optimization. Accordingly, a surrogate loss such as the cross-entropy $\mathcal{L}{=}\mathcal{L}_{\text{ce}}$ is used typically. However, we propose alternative losses in Section 7.5.

Dot-Product Attention. In its basic form, dot-product attention [122, 212] computes, for every query, attention weights as the dot-product of the query to all keys. The softmax function is then applied over the key dimension. These attention weights are then multiplied by the values:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^\top)V. \tag{7.3}$$

Here, $Q \in \mathbb{R}^{n \times d_{\text{model}}}$, $K \in \mathbb{R}^{n \times d_{\text{model}}}$, and $V \in \mathbb{R}^{n \times d_{\text{model}}}$ are the matrices of n queries, keys, and values, respectively. According to Vaswani et al. [212], for large values of d_{model}, the dot-product between queries and keys may grow large in magnitude. This has the effect of pushing the softmax function into the saturated regime, where it has extremely small gradients. This is due to the exponentiation of individual query-key dot-products in the softmax function. Because this can be harmful for training, they introduce scaled dot-product attention, where $QK^\top$ is scaled by $1/\sqrt{d_{\text{model}}}$.

In practice, using $H > 1$ attention heads by linearly projecting queries, keys, and values H times to d_k, d_k, and d_v dimensions was found to be beneficial [212]. The output of the h-th attention head (AH) becomes:

$$\text{AH}_h(Q, K, V) = \text{softmax}\left(\frac{QW_Q^h(KW_K^h)^\top}{\sqrt{d_k}}\right)VW_V^h, \tag{7.4}$$

where $W_Q^h \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_K^h \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_V^h \in \mathbb{R}^{d_{\text{model}} \times d_v}$ are (learned) projection matrices. The outputs of individual attention heads are then concatenated and multiplied by another learned projection matrix $W_O \in \mathbb{R}^{Hd_v \times d_{\text{model}}}$.

A special case is self-attention with $Q = K = V \in \mathbb{R}^{n \times d_{\text{model}}}$, which is typically used in encoder layers of image recognition models. We define the attention weights of the h-th head on X via $A_h(X) = \text{softmax}(\frac{XW_Q^h(XW_K^h)^\top}{\sqrt{d_k}}) \in \mathbb{R}^{n \times n}$. The h-th self-attention head becomes:

$$\text{SelfAH}_h(X) = A_h(X)XW_V^h \tag{7.5}$$

7.4 Robustness of Dot-Product Attention

In this section, we first discuss why (scaled) dot-product self-attention is challenging for gradient-based adversarial attacks such as PGD. We then provide an example of an adversarial vulnerability in the attention weights themselves.

7.4.1 Gradient of Dot-Product Self-Attention

For computing $\nabla_p \mathcal{L}(\mathcal{F}(x, p, \text{L}), y)$ in Eq. 7.2, it is necessary to backpropagate through the entire model. This requires the gradient $\nabla_X \text{SelfAH}_h(X)$ for every attention layer and head h. With the product-rule we obtain:

$$\nabla_X \text{SelfAH}_h(X) = \left[(\nabla_X A_h(X))X + A_h(X)1_X\right] W_V^h,$$

where 1_X is a matrix of ones with the same shape as X.

An important property of the gradient $\nabla_X \text{SelfAH}_h(X)$ is accordingly the element-wise ratio $|\frac{(\nabla_X A_h(X))X}{A_h(X)1_X}|$. We summarize the median value of this ratio over tokens and heads in Table 7.1 for different models and layers. As seen, the typical regime (that is, for $> 50\%$ of the cases) is that $(\nabla_X A_h(X))X$ is smaller than $A_h(X)1_X$ by a factor of ≈ 20 for ViTs [56], DeiTs [202], and for the inner encoder layers of DETR [35]. In this setting, one can approximate $\nabla_X \text{SelfAH}_h(X) \approx (A_h(X)1_X)W_V^h$, that is: the gradient considers the attention weights $A_h(X)$ as effectively constant. Accordingly, gradient-based attacks such as PGD based on an end-to-end loss such as $\mathcal{L}_{ce}$ would be

Table 7.1: Median of $|(\nabla_X A_h(X))X)/(A_h(X)1_X)|$ over tokens and heads, for models on 12 encoder layers. We report the mean of the ratio medians over 100 randomly selected images from the MS COCO 2017 validation set [112] for DETR (DC5-R50), and over 100 randomly selected images from the ImageNet 2012 [169] dataset for all remaining models.

Layer	DETR	ViT-T	ViT-S	ViT-B	DeIT-T	DeIT-S	DeIT-B
1	0.208	0.079	0.042	0.072	0.035	0.045	0.053
2	0.035	0.040	0.051	0.080	0.045	0.045	0.044
3	0.039	0.031	0.027	0.047	0.044	0.033	0.042
4	0.066	0.043	0.028	0.037	0.035	0.032	0.057
5	0.098	0.037	0.029	0.033	0.035	0.033	0.044
6	0.147	0.044	0.027	0.044	0.070	0.033	0.040
7	n/a	0.036	0.031	0.040	0.045	0.033	0.040
8	n/a	0.045	0.046	0.046	0.050	0.037	0.039
9	n/a	0.171	0.087	0.064	0.076	0.052	0.049
10	n/a	0.308	0.126	0.061	0.082	0.077	0.083
11	n/a	0.428	0.280	0.099	0.080	0.100	0.192
12	n/a	0.442	0.469	0.249	0.212	0.198	0.104

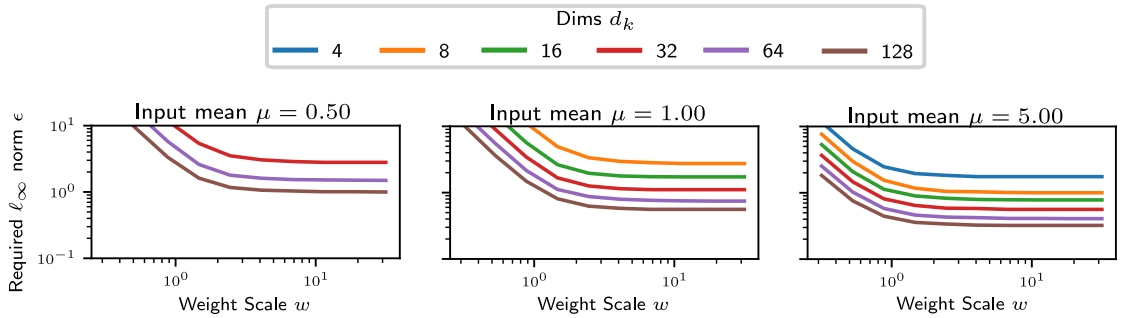


Figure 7.3: Minimum L_∞ norm perturbation ϵ required for reaching the attacker's goal on the controlled setting from Section 7.4.2. Increased weight scale w, higher embedding dimension d_k, and input mean norm μ simplify the attack and allow to achieve smaller ϵ.

biased towards focusing on adversarial effects in X that can be propagated (linearly) via the values $V = XW_V^h$ in self-attention, while effectively ignoring potentially adverse (and non-linear) effects of X propagated via the attention weights $A_h(X)$. We note that also later stages of model training with gradient-based optimizers can be negatively affected by this property of dot-product attention. Studying this in more detail is left to future work.

7.4.2 Robustness of Dot-Product Attention Weights

We now study to which extent attention weights $A_h(X)$ of dot-product attention can be affected by an adversarial patch attack. For this, we use a controlled setting with normally distributed X, where each feature has mean μ and variance 1: $X_j \sim \mathcal{N}(\mu \cdot \mathbf{1}, \mathbf{1})$. Moreover, we choose $d_k = d_{\text{model}}$ and diagonal $W_Q = -w \cdot \mathbb{I}_{d_k}$ and $W_K = w \cdot \mathbb{I}_{d_k}$, that is W_Q and W_K having both scale w but opposite signs.

We study a simple threat model: the adversary can only modify X_0, the first of n entries of X (X can be considered as embedding of patches and X_0 as corresponding to the embedded adversarial patch), with the constraint $||X_0^{adv} - X_0||_\infty \leq \epsilon$. Moreover, the adversaries goal is to achieve $\left[\frac{1}{n}\sum_j \left(A_h(X^{adv})_{0j} > 0.99\right)\right] > 0.95$, that is: at least 95% of the queries need to attend to the first key with attention weights greater or equal 0.99. By design, setting $X_0^{adv} = X_0 - \epsilon$ is a strong attack for $\mu > 0$ because $-\mathbf{1} \cdot \epsilon$ corresponds to the direction of $(W_Q - W_k)(\mu \cdot \mathbf{1})$, the difference of projected query and key mean.

We study how ϵ needs to be chosen as a function of μ, w, and d_k for a successful attack with the above attack and threat model. Results are shown in Figure 7.3. In general, one can observe that a successful attack with a smaller perturbation amount ϵ requires: (a) increasing the scale w of the projection matrices, (b) higher embedding dimensions d_k, and (c) less centered inputs X (larger $|\mu|$). The findings (a) and (b) can be attributed to higher dimensions and larger weights allow the effect of minor perturbations in each input dimension to be amplified. Upon closer inspection (see Section C.1 in supplementary material), we can attribute finding (c) to a separation of projected keys and queries into distinct clusters for less centered inputs: all queries are close to each other, all keys are close to each other, but query-key pairs are all distant from each other. In this case, a single key can be made to be the most similar to each query, just by moving this key in the direction of the query cluster (see Figure 7.1 for a real data illustration) .

In which regime does dot-product attention typically operate in trained image transformers? For many architectures, d_k is relatively large by design. Moreover, the input of dot-product attention is typically not centered (zero-mean) due to enabled

Table 7.2: Largest singular value of the projection weight matrices $W_Q(W_K)^\top$ for randomly initialized and trained models. Trained models tend to have largest singular values, indicating that training leads to large weight scales, which is shown to lead to larger vulnerability in the toy example in this section.

Layer	ViT/DeiT-B *random*	ViT-B *trained*	DeiT-B *trained*	DETR Encoder *random*	DETR Encoder *trained*
1	0.80	60.96	175.28	1.27	7.93
2	0.79	22.67	65.86	1.29	10.61
3	0.80	11.76	54.19	1.25	15.35
4	0.79	5.88	44.95	1.27	45.26
5	0.79	4.91	29.36	1.28	49.82
6	1.21	4.83	28.72	1.26	29.30

affine transformations within the normalization layers before self-attention. Many implementations, e.g. [159], also use affine rather than linear query/key projections in the heads, where different biases in the projection of key and queries will have a similar effect as non-centered inputs. Lastly, we also observe that for many vision transformers, the product of projection weight matrices $W_Q(W_K)^\top$ has very large maximal singular value at convergence, compared to the randomly initialized projection weight matrices (see Table 7.2). Large singular values can be considered to be analogous to the "large weight scales" in our controlled setting. We thus expect that dot-product attention of trained vision transformers typically operates in a setting where the attention weights $A_h(X)$ can be a source of vulnerability to patch attacks, but gradient-based attacks such as PGD are biased towards ignoring this vulnerability (as discussed in Section 7.4.1).

7.5 Attention-Fool

We have discussed that attention weights $A_h(X)$ can be largely affected by adversarial patches but we have also shown that gradient-based attacks are impaired in exploiting this vulnerability. In order to encourage fooling of attention weights, we introduce a family of losses that are defined directly on the pre-softmax attention

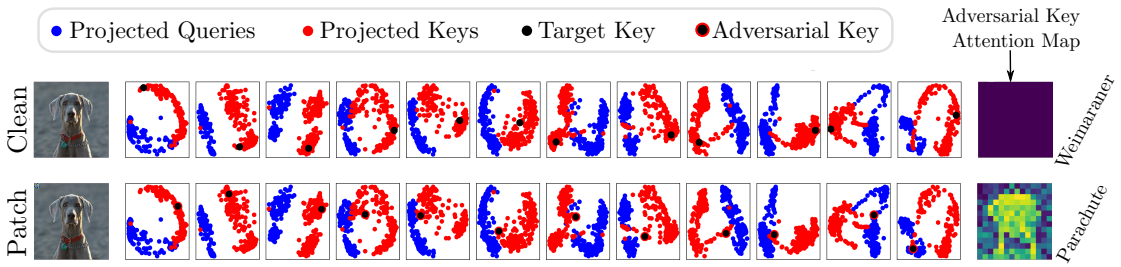


Figure 7.4: Embedded projected key and query tokens for clean and patched input images on each of 12 ViT layers of the attention head which showed the largest attention change. We design Attention-Fool to target all layers and heads at once. The last column reports the attention map weights of the adversarial key on the last layer – showing that the key draws a large amount of attention from queries.

logits $B_h(X) = \frac{XW_Q^h(XW_K^h)^\top}{\sqrt{d_k}}$[1]. We denote attacks based on these losses as *Attention-Fool*. We note that Attention-Fool losses are always maximized by the attacker, regardless of whether an attack is targeted or untargeted. Moreover, while this paper focuses on self-attention, it is straightforward to extend these losses to cross-attention.

$\mathcal{L}_{kq}^{hl}$: Attacking a single attention head. We first focus on adversarially modifying attention weights of a specific head h in a specific self-attention layer l. We propose a loss $\mathcal{L}_{kq}^{hl}$ that aims at maximizing the query attention to the $i^\star$-th projected key, that is: maximize the number of queries that devote the majority of their attention to this key. Naturally, $i^\star$ is chosen such that its token corresponds to the region in the input where the adversarial patch has been placed.

We denote projected queries by $P_Q^{hl} = X^{hl}W_Q^{hl} \in \mathbb{R}^{n\times d_k}$ and projected keys by $P_K^{hl} = X^{hl}W_K^{hl} \in \mathbb{R}^{n\times d_k}$. We have $B^{hl} = \frac{P_Q^{hl}(P_K^{hl})^\top}{\sqrt{d_k}} \in \mathbb{R}^{n\times n}$, where the first dimension indexes queries and the second one keys. Each element of B^{hl} quantifies the dot-product similarity between a key and a query in the respective attention head h and layer l. We now set $\mathcal{L}_{kq}^{hl} = \frac{1}{n}\sum_j B_{ji^\star}^{hl}$, where j indexes the queries. Maximizing $\mathcal{L}_{kq}^{hl}$ thus corresponds to maximizing average dot-product similarity between queries and the target key.

[1]Note that similar effects could be obtained by estimating the true gradient by replacing the softmax operation with identity in a backward pass or by blocking gradient flow (`.detach()` in PyTorch) in value paths in the attention blocks.

$\mathcal{L}_{kq}$: Attacking all attention layers and heads simultaneously. In the previous paragraph, we have introduced a loss that targets a single head and layer. However, in general fooling a single head might not be sufficient and it is also a priori unclear which head would be the most vulnerable one. The same applies to the choice of the layer, when there are multiple layers using dot-product attention. Because of this, we propose applying the loss to all layers and heads simultaneously. However, simply averaging $\mathcal{L}_{kq}^{hl}$ over all heads and layers might not be optimal because it may favour many smaller head-wise changes overall rather than successfully fooling a subset of heads and layers to a larger extent. Because of this, we will utilize a smooth maximum $\text{smax}(x) = \log \sum_i e^{x_i}$ over $\mathcal{L}^{hl}$. Specifically, we define $\mathcal{L}_{kq}^{l} = \log \sum_h e^{\mathcal{L}_{kq}^{hl}} \quad \forall l$ and $\mathcal{L}_{kq} = \log \sum_l e^{\mathcal{L}_{kq}^{l}}$. We empirically compare the choice of this smooth maximum over mean and hard maximum in Section 7.6.2. We also visualize $\mathcal{L}_{kq}$'s effect on each network layer in Figure 7.4.

In order to make the losses of different heads and layers commensurable, we also propose to normalize scales of projected keys and queries via $\bar{P}_Q^{hl} = {P_Q^{hl}}/{\frac{1}{n}||P_Q^{hl}||_{1,2}}$ and $\bar{P}_K^{hl} = {P_K^{hl}}/{\frac{1}{n}||P_K^{hl}||_{1,2}}$, where $||X||_{1,2} = \sum_i^n \sqrt{\sum_j^{d_k} X_{ij}^2}$ is the $\ell_{1,2}$ norm. Note that the normalization is applied per individual head and makes the average ℓ_2 norm of queries and keys equal to 1. B^{hl} is then computed based on $\bar{P}_Q^{hl}$ and $\bar{P}_K^{hl}$, making the losses $\mathcal{L}_{kq}^{hl}$ commensurable. We also empirically evaluate the effect of this normalization in Section C.2 of the supplementary material.

$\mathcal{L}_{kq^\star}$: Targeting a special class token. The $\mathcal{L}_{kq}$ loss treats all queries equally and aims at misdirecting the attention of the majority of queries towards the adversarial key token. However, for many architectures not all queries are created equally, for instance for ViTs [56] and DeiTs [202], there exists a special class token, which is supposed to accumulate class evidence over the layers. We propose a version of the $\mathcal{L}_{kq}$ loss that specifically targets the attention of a certain query, for instance the one corresponding to the class token. Let $j^\star$ be the index of the query that shall be targeted. We define $\mathcal{L}_{kq^\star}^{hl} = B_{j^\star i^\star}^{hl}$ and generalize to $\mathcal{L}_{kq^\star}$ as above using smooth maximum over heads and layers.

In the following experiments, we will use combinations of $\mathcal{L}_{kq}$ or $\mathcal{L}_{kq^\star}$ with standard cross-entropy loss $\mathcal{L}_{ce}$ for the adversarial patch optimization of Equation 7.1. We denote the resulting set of attacks leveraging the weakness in dot-product attention as *Attention-Fool*. Note that Attention-Fool is different from concurrent work Patch-Fool [7] that defines a loss on the post-softmax attention weights, averaged of heads h and queries j: $\mathcal{L}_{PF} = \sum_h \sum_j A_h(X)_j$. In particular, Attention-Fool (in contrast to Patch-Fool) is not affected by the small gradient issue described in Section 7.4.1 since it is defined on pre-softmax attention weights.

7.6 Evaluation of Attention-Fool

We evaluate the robustness of different vision transformers against adversarial patches generated using our proposed Attention-Fool adversarial losses. We first investigate the robustness of ViT [56] and the improved DeiT [202] in Section 7.6.2. Then we show how Attention-Fool generalizes to DETR [35] in Section 7.6.3, which uses a hybrid CNN + Transformer architecture to perform object detection.

7.6.1 Evaluation Setup

We use PGD as the optimizer to solve Equation 7.1. We set PGD's initial step size $\alpha^0 = 8/255$ and schedule it with a cosine decay: $\alpha^{t+1} = \alpha^t \frac{1}{2}(1 + \cos(\pi \frac{t}{\mathrm{N}}))$. Additionally, we add a stronger baseline to the adversarial patch robustness evaluation, we use an improved version of PGD by adding normalized-momentum in the optimization [55]:

$$m^{(t)} = \beta m^{(t-1)} + (1-\beta)^{\nabla_p^t}/_{||\nabla_p^t||_2},$$

and use $m^{(t)}$ instead of the gradient ∇_p^t in PGD.

7.6.2 Attention-Fool on Vision Transformer

We use the new Attention-Fool attack described in Section 7.5 to compare the effectiveness of adversarial patches optimized with the new losses $\mathcal{L}_{kq}$ and $\mathcal{L}_{kq^\star}$ with adversarial patches optimized using cross-entropy loss $\mathcal{L}_{ce}$. We use 1,000 images

Table 7.3: Robust accuracies (%) under adversarial patch attacks on 1000 ImageNet images. Clean performance, as well robust accuracy against Patch-Fool loss [7] and Patch-RS [46] are shown as baselines. All rows in the bottom block are computed using PGD^{250} with step size α=8/255 and crossentropy loss $\mathcal{L}_{ce}$. An Attention-Fool term is added to the loss (either $\mathcal{L}_{kq}$ or $\mathcal{L}_{kq\star}$) and optionally momentum is added to PGD. Numbers in parenthesis report the improvement or degradation in robust accuracy w.r.t. the $\mathcal{L}_{ce}$ baselines. All models use 224×224 resolution unless marked by "384" in the model name, then they use 384×384. Patch size is always 16×16.

	ResNet50	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
Clean	80.6	73.5	85.0	86.4	69.4	82.0	82.0
Patch-Fool [7]	-	0.3	17.5	47.8	3.2	52.9	68.2
Patch-RS [46]	70.8	18.6	49.8	65.9	44.0	57.3	71.2
$\mathcal{L}_{ce}$	55.1	0.1	13.5	31.2	19.8	36.0	58.8
$+\mathcal{L}_{kq}$	-	0.5(+0.4)	5.0(–8.5)	18.0(–13.2)	13.1(–6.7)	35.5(–0.5)	55.1(–3.7)
$+\mathcal{L}_{kq\star}$	-	0.3(+0.2)	2.6(–10.9)	13.0(–18.2)	11.7(–8.1)	33.7(–2.3)	57.2(–1.6)
+Momentum	49.0	0.0	3.1	13.2	1.5	16.8	41.7
$+\mathcal{L}_{kq}$	-	0.0(-0.0)	0.1(–3.0)	2.5(–10.7)	0.0(–1.5)	19.3(+2.5)	39.8(–1.9)
$+\mathcal{L}_{kq\star}$	-	0.0(-0.0)	0.1(–3.0)	1.9(–11.3)	0.0(–1.5)	13.1(–3.7)	40.6(–1.1)

from the ImageNet 2012 [169] dataset and perform an untargeted attack, where y in Eq. 7.1 is the image ground truth. We also report the results on a targeted attack in Section C.3 in Appendix. For the experiment, we place an adversarial patch of 16×16 pixels in the top left corner. For the evaluation, we use pre-trained ViT models from `timm` [223]. We choose models which split input images into 16×16 image sub-patches; this way the adversarial patch occupies exactly one of the image sub-patches. We consider models with input image resolution of 224×224 or 384×384; the adversarial patch covers 0.5% and 0.17% of the input, respectively. For comparison, we also perform the attack on a CNN, ResNet50, using the same setup but optimizing adversarial patches with $\mathcal{L}_{ce}$ only. We report the resulting robust accuracies in Table 7.3. We find that Attention-Fool's $\mathcal{L}_{kq}$ and $\mathcal{L}_{kq\star}$ improve on the baseline cross-entropy on most models regardless of momentum of PGD. While Table 7.3 shows that $\mathcal{L}_{kq}$ already decreases robust accuracies across most settings, using the architecture-specific Attention-Fool loss-variant $\mathcal{L}_{kq\star}$ brings more stable robust accuracy down among all models, often with large performance gains over cross-entropy alone. Attention-Fool can bring accuracies down to 0% in ViT-T and DeiT-T models, and as low as 1.90% even in ViT-B-384, where the patch

Table 7.4: Robust accuracies (%) under adversarial patch attacks with Attention-Fool losses when choosing different ways of aggregating $\mathcal{L}_{kq}^{hl}$ across encoder layers and attention heads. All rows are computed using PGD^{250} with momentum and step size α=8/255. Numbers in parenthesis report the improvement or degradation in robust accuracy w.r.t. the smax,smax default choice outlined in Section 7.5.

Layer	*Heads*	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
smax	smax	0.0	0.1	2.5	0.00	19.3	39.8
smax	mean	0.0 (-0.0)	0.2 (+0.1)	2.7 (+0.2)	0.1 (+0.1)	17.8 (–1.5)	40.6 (+0.8)
smax	max	0.0 (-0.0)	3.3 (+3.2)	5.8 (+3.3)	0.0 (-0.0)	36.6 (+17.3)	45.9 (+6.1)
mean	smax	0.0 (-0.0)	0.0 (–0.1)	2.9 (+0.4)	0.0 (-0.0)	18.3 (–1.0)	40.4 (+0.6)
mean	mean	0.0 (-0.0)	0.1 (-0.0)	3.5 (+1.0)	0.0 (-0.0)	17.5 (–1.8)	39.6 (–0.2)
mean	max	0.0 (-0.0)	2.4 (+2.3)	9.2 (+6.7)	0.0 (-0.0)	23.8 (+4.5)	43.6 (+3.8)
max	smax	0.1 (+0.1)	15.6 (+15.5)	26.7 (+24.2)	1.2 (+1.2)	27.1 (+7.8)	45.7 (+5.9)
max	mean	0.2 (+0.2)	13.8 (+13.7)	24.6 (+22.1)	1.5 (+1.5)	26.1 (+6.8)	46.1 (+6.3)
max	max	3.7 (+3.7)	45.5 (+45.4)	59.7 (+57.2)	1.5 (+1.5)	58.5 (+39.2)	58.8 (+19.0)

occupies only 0.17% of the input image. Notably, in contrast to prior work (see Section 7.2) we find all ViT/DeiT variants to be considerably *less* robust than a ResNet50 when the patch is placed in the top left corner, whereas placing the patch in the centre could have a larger effect on a CNN.

We include additional comparisons with the "Attention-Aware Loss" of Patch-Fool (Section 4.4 in [7]). For a controlled comparison with Patch-Fool, we perform two adjustments: (i) we disregard the saliency-based selection of the patch location (Section 4.3 in [7]) and (ii) we replace Adam with PGD (momentum = 0.9). This abstracts from attack specifics and allows us to compare losses directly. The results show that Patch-Fool's Attention-Aware loss underperforms the Attention Fool losses; confirming the limitations of optimizing post-softmax attention weights (discussed in Section 7.4.1). Moreover, we compare to Patch-RS [46], a random-search based black-box patch attack. The inferior results of Patch-RS indicate that gradient-free attacks are no viable alternative for exploiting the existent vulnerabilities in attention weights.

Ablations on Reduction over Heads and Layers. As discussed in Section 7.5, when multiple attention layers and attention heads are present, one can aggregate per-layer and per-head $\mathcal{L}_{kq}^{hl}$ in different ways. To study the influence of these aggregation choices, in this section we empirically test three of them: (i) smax (as

in Section 7.5), (ii) hard maximum max and (iii) mean. Since we are aggregating along two dimensions, layers and heads, this sparks nine combinations of the three aggregations, which we empirically test on ViT models. As shown in Table 7.4, across different aggregation methods, choosing smax over both layers and heads generally performs well, resulting in low robust accuracies. Intuitively, using smax gives more flexibility to the optimization to choose the most vulnerable layers and heads. We also find that mean aggregation across both layers and heads is an effective choice, presumably in part due to the fact that in ViT inputs to attention layers are pre-emptively normalized, leading to smaller difference in per-head per-layer behaviour. In comparison the hard maximum max seems to be the weakest choice, leading to consistently worse results. We note from Table 7.4 that the best combination is rather model-specific, which suggests that a similar ablation study could be run on a model-basis. In fact, we will show in the next section how on DETR [35], Attention-Fool with max aggregation results in better performance.

7.6.3 Attention-Fool on DETR

Since Attention-Fool targets dot-product attention layers, it can be adapted to target a various tasks and architectures using this attention. Here, we apply Attention-Fool to object detection with DETR [35], which combines a CNN backbone with a Transformers encoder-decoder.

Attention-Fool Configuration. While $\mathcal{L}_{kq\star}$ showed superior results for ViT/DeiT in Section 7.6.2, it is not applicable to DETR due to the absence of a class token in DETR. In contrast, we use $\mathcal{L}_{kq}^{(1)}$: a loss targeting all queries and heads but only the first transformer encoder layer ($l = 1$) rather than all at once. We focus on $l = 1$ because we hypothesize that the transition from CNN to transformer encoder in DETR is specifically brittle due to large activations from the CNN backbone. For a similar reason, we aggregate losses across attention heads using the maximum function rather than smax used in Section 7.4.

Evaluation Setup. We evaluate a targeted Attention-Fool attack on DETR by choosing the target to be the background class, which effectively forces missed

Table 7.5: Mean Average Precisions (mAP) in presence of adversarial patches computed with $\mathcal{L}_{ce}$ and patches computed with Attention Fool $\mathcal{L}_{kq}^{(1)}$, on DETR models. The Baseline Clean mAP is obtained on the clean set of images. Three different patch sizes and four different DETR models are considered, based either on R50 or R101 backbone and with and without dilation DC5.

	R50	DC5-R50	R101	DC5-R101
Clean mAP	53.00	54.25	54.41	56.74
64×64: $\mathcal{L}_{ce}$	34.91	17.67	31.76	34.71
$+\mathcal{L}_{kq}^{(1)}$	21.03 (-13.88)	7.34 (-10.33)	2.07 (-29.70)	4.19 (-30.52)
$\mathcal{L}_{kq}^{(1)}$ only	29.52 (-5.39)	15.03 (-2.64)	2.05 (-29.71)	10.18 (-24.53)
56×56: $\mathcal{L}_{ce}$	37.74	24.13	32.06	38.28
$+\mathcal{L}_{kq}^{(1)}$	28.19 (-9.55)	14.68 (-9.45)	3.30 (-28.76)	6.82 (-31.46)
$\mathcal{L}_{kq}^{(1)}$ only	38.14 (+0.40)	20.18 (-3.95)	5.60 (-26.46)	14.27 (-24.01)
48×48: $\mathcal{L}_{ce}$	42.08	27.69	37.89	41.32
$+\mathcal{L}_{kq}^{(1)}$	34.88 (-7.20)	18.94 (-8.75)	9.80 (-28.09)	17.05 (-24.26)
$\mathcal{L}_{kq}^{(1)}$ only	42.19 (+0.12)	32.60 (+4.91)	12.85 (-25.04)	17.55 (-23.77)

detections (i.e., false negatives). We evaluate four pre-trained DETR models from the official paper repository [68], with backbone either Resnet50 or Resnet101, and with or without a dilation in the Resnet layer 5 convolution (DC5). Note that DC5 models have twice the resolution in the backbone-extracted feature map, which is used as the encoder input. We select 100 images from the MS COCO 2017 validation set [112], and use the default DETR validation image loader which re-scales images to have shortest side 800 pixels. We evaluate Attention-Fool by targeting a single key token – a single unit in the CNN feature map – out of the entire token sequence which is at least 2,500- and 625-tokens long for DC5 and non-DC5 models (with final length depending on the input image resolution). We provide an ablation in Section C.4 in the supplementary material where we attribute the adversarial effect to this token alone. We place the adversarial patch in the image top-left centered around the 80,80 pixel. Accordingly, we target the key token with index 2,2 in non-dilated models and 4,4 in dilated ones – this ensures that the key token receptive field is centered around the adversarial patch location in the image. We record the primary COCO challenge metric, the mAP, which averages precision-recall curves scores at various IoU thresholds. To not bias the evaluation

towards local patch effects, we ignore detection boxes whose intersection with the patch box is >50% of the detection box area. We do the same for ground truths for a fair comparison between clean and patched inputs.

Results. Similarly to the previous section, we use $\mathcal{L}_{ce}$ as an attack baseline and test the improvement (degradation) in mAP as we combine it with the Attention-Fool loss $\mathcal{L}_{kq}^{(1)}$. We test three different patch sizes, 64x64, 56x56, and 48x48, which occupy <0.64%, <0.49% and <0.36% of the input image, respectively. Given the higher complexity of DETR (compared to ViT), we increase the number of PGD iterations to 1000 in this experiment. We report the resulting mAPs in Table 7.5: baseline mAP on clean images, mAP under a targeted attack which uses $\mathcal{L}_{ce}$, and the mAP change when we use the combined Attention-Fool $\mathcal{L}_{ce} + \mathcal{L}_{kq}^{(1)}$ loss. Table 7.5 shows that the addition of $\mathcal{L}_{kq}^{(1)}$ reduces the mAP performance across all models and all patch sizes. We find that larger models with DETR101 are more vulnerable to Attention-Fool, where the mAP can be reduced down to 2.07 – this corresponds to suppressing the detection of the vast majority of objects.

Table 7.5 also presents a setting where we do not use any loss on the model's output but solely focus on misdirecting the first encoder layer's attention ("$\mathcal{L}_{kq}^{(1)}$ only"). In most cases, this results in considerably lower robust accuracy than using $\mathcal{L}_{ce}$ directly, indicating that for DETR "fooling attention is all you need" for misclassification.

7.7 Limitations

We focused on dot-product attention. Other attention mechanisms [107, 245, 217, 116] are likely not affected to the same degree by the identified weakness. In particular, locality constraints on attention blocks, i.e., when tokens are mixed only with other spatially-near tokens rather than with all remaining tokens, will limit the attack effectiveness in practice. In that case, and akin to what happens in CNNs (where receptive field is limited), an adversarial patch would not be able to affect the entirety of the network output. Additionally, note that the "attention-drawing" is in principle detectable as it generates large, sparse activations in the attention

weight matrix. However, limiting attention to spatially-close tokens might be a better solution in practice rather than a detection module. Nevertheless, since dot-product attention is the predominant attention mechanism in transformers, our approach is broadly applicable.

7.8 Summary

In this chapter, in view of the recent interest for vision transformers, we revisited the robustness of these novel vision models against adversarial patch attacks. We identified properties of the dot-product attention's gradient that bias vanilla patch attacks to mostly ignore a core vulnerability of these attention weights. This presumably caused prior works to overestimate robustness – we identified all studied vision transformers as considerably less robust than a ResNet50. We propose Attention-Fool, which directly targets the dot-product attention weights and allows tighter robustness estimates. We hope that these findings can provide the base for more robust attention mechanisms which can mitigate the identified vulnerability of dot-product attention. In the following we summarize this thesis' results and give final conclusions.

8 Conclusions

Contents

8.1 Summary of Results 157
8.2 Final Conclusions 158

8.1 Summary of Results

This work investigated the security of intelligent computer vision systems that use deep neural networks in combination with camera-captured inputs. Rather than being part of a large body of research focusing on norm-bounded attacks under the constraint of imperceptibility, we focused on attacks in view of a realizability constraint, where we attempted to bridge the gap between theoretical attacks and practical ones. This includes the attacks proposed in Chapter 7, 4, 5 and 6, sorted in this order according to the realizability scale described in Figure 1.1. For most of these attacks, we outlined possible defences and countermeasures, which may be used to thwart attacks effectiveness, but have to be considered in view of a security-performance trade-off.

8.2 Final Conclusions

Assuring the safety or security of computer vision systems is a multi-faceted problem which does not yet have holistic solutions which can provide general valid guarantees. Many researchers have focused on generic, well-defined threat models (such as those considering ℓ_p-norm adversarial examples) and defences, but recent trends show how these tend to be bypassed by specific attacks and adaptive adversaries. We showed with a range of attacks, that the security (and safety) analysis for these vision systems would need to consider various potential weaknesses and components of the system, which are not trivial to enumerate in many cases. We hope that in the future, comprehensive assurance evaluations standards can be built for computer vision systems and models. These standards would include extensive analysis of possible threats along many dimensions. Many toolkits have started systematising digital adversarial attacks on the scene [45, 47, 161, 143, 132], nevertheless these could be expanded to include: (i) physical adversarial attacks on the scene and the camera (which would envelop the attacks presented in this thesis) (ii) robustness to common perturbations, (iii) resilience to poisoning attacks, and possibly any other type of tampering which can be done with the system's inputs, data or decision. While oftentimes we can not provide mathematically-strong security guarantees, we have learned a lot about how to analyse vision systems and we will be able to use heuristic guarantees to build reliable suites of standardized tests for the deployment of system in the real-world.

Appendices

A

Supplementary Material for Chapter 5

Contents

A.1 Attack on Different Objects 161
A.2 Additional Results 162
A.3 SentiNet Description 163

A.1 Attack on Different Objects

The introduced attack can generalize in principle to any kind of deep neural network which uses RGB-camera inputs to make decisions. To show how our attack generalizes, we also investigate the feasibility of the attack on different objects.

Setup. For Lisa-CNN and Gtsrb-CNN, we choose another traffic sign, "give way", while for Yolov3 and Mask-RCNN we choose the "bottle" class. For the give way sign and the bottle, we run a reduced evaluation: we execute all the experiment procedure steps reported in Section 5.5.1 and we test the correct (mis-)classification across a set of photos of the altered objects. Extending our method to other objects is straightforward, it only requires to change the input mask of the projection and re-profile the projectable colours. When projecting on non-flat surfaces, the



Figure A.1: Sample video frame extracted from the outdoor experiment, for the 120 lux setting. The image shows the stop sign undetected under the threshold set in the experiments. The blue 'tracker' box is set manually and tracks the location of the sign.

adversary will also have to consider the distortion introduced by those surfaces, this is discussed in Section 5.6.

Results. We report example frames of successful attack on other objects in Figure A.2 and in Figure A.3. These include legitimate frames where the classification works correctly. All the pictures are taken in 180 lux ambient light. For Mask-RCNN and Yolov3 we restricted the bottle size to [150, 250], meaning that the bottle is generally in the foreground.

A.2 Additional Results

We report extended results for the transferability-based cross-network attack in Table A.1. This includes each pair of the evaluated models, including Lisa-CNN$^{(s)}$ and Gtsrb-CNN$^{(s)}$ which are trained with cross-entropy loss from scratch. We report an example frame from the outdoor experiment in Figure A.1.

A.3 SentiNet Description

Rationale. We picked SentiNet for the evaluation because it was one of the few defences that was *specifically* designed to detect physical adversarial examples. In fact, there is a plethora of works that creates physical adversarial examples by using stickers (or patches) that are placed on the targeted objects. The insight behind SentiNet is that these patches are the most common way to create physical AE, but generate small image areas with large saliency. This is not only a *detectable* behaviour in general, but it is also *unavoidable* for the attacker to escape such behaviour when creating a physical AE (without replacing the entire object).

Description. SentiNet is a system designed to detect AE leveraging the intuition of locality of patches. If an adversarial sample contains a patch which causes a misclassification, then the saliency of the area containing the patch will be high. Therefore, the salient area will cause misclassifications on other legitimate samples when overlayed onto them. To compute the salient areas in input SentiNet uses GradCam++ [40], which backpropagates the outputs to the last convolutional layer of the network and checks which region of the input lead to greater activations. Since the resolution of this layer is only 4x4 for both Gtsrb-CNN and Lisa-CNN, we instead use XRAI [93], a newer and more accurate method to compute salient areas. We found that using GradCam made the output masks unusable as a resolution of 4x4 leads to coarse block like regions where salient areas cannot be accurately identified (resolution of this layer also is pointed out as a problem in the original paper [42]). XRAI on the other hand produces saliency regions at the input resolution, leading to more granular salient areas, using an algorithm that incrementally grows salient regions. As a consequence of this improved technique, XRAI has been shown to outperform older saliency algorithms, producing higher quality, tightly bound saliency regions [93].

SentiNet computes a threshold function which separates AE from benign images. The threshold function is computed using: (i) the *Average Confidence*, i.e., the average confidence of the network prediction made on benign test images where

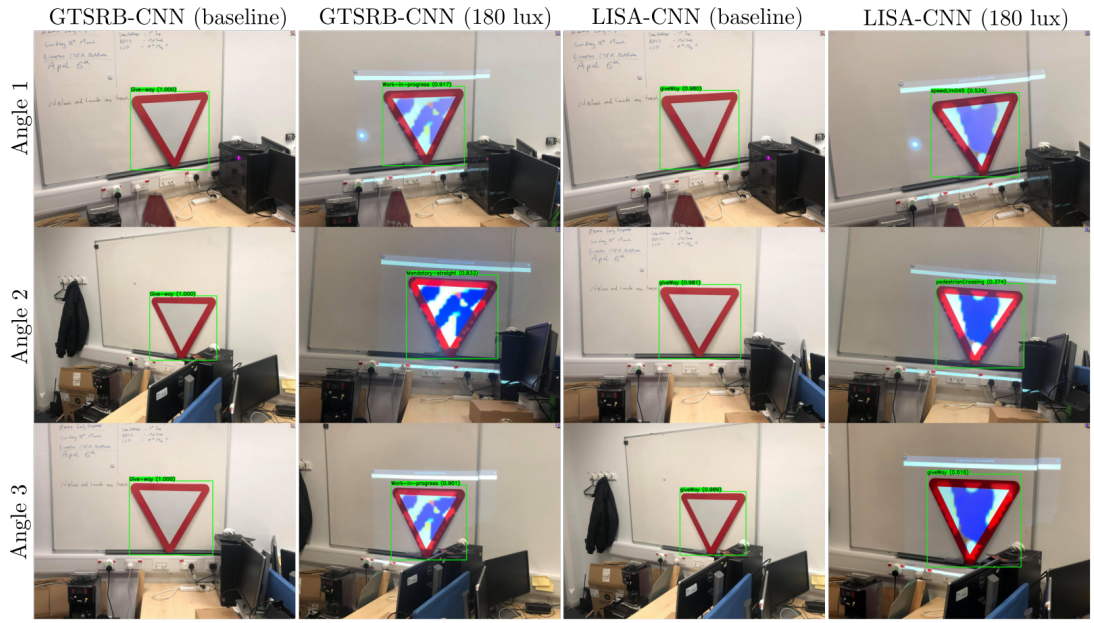


Figure A.2: SLAP attack on class "Give Way" for Gtsrb-CNN and Lisa-CNN.

salient masks are replaced with inert patterns added to them and (ii) the *Fooled Percentage*, i.e., the percentage of benign test images where overlaying the salient mask leads the network to predict the suspected adversarial class. These two scores characterize benign behaviour and can almost perfectly separate benign from adversarial inputs in SentiNet. We follow the same technique as in the original paper for fitting the threshold function that separates the malicious and benign data. Our SentiNet implementation is also available with the rest of the source code in the project repository.

Table A.1: Extended transferability results. We test all the frames from the collected videos with a certain projection being shone against a different target model, figures in bold are white-box pairs. (*) For Google Vision we only test one frame every 30 frames, i.e., one per second. We also remove all frames that are further than 6m away as Google Vision does not detect most of them in a baseline scenario. $_^{(a)}$ indicates adversarially trained models. $_^{(s)}$ indicates models we re-trained from scratch.

			Target Model								
lux	*Source Model*	*no. frames*	Yolov3	Mask-RCNN	Gtsrb-CNN	Gtsrb-CNN$^{(a)}$	Gtsrb-CNN$^{(s)}$	Lisa-CNN	Lisa-CNN$^{(a)}$	Lisa-CNN$^{(s)}$	Google Vision*
120	Yolov3	4587	**100.0**	73.4	0.0	0.0	0.0	21.5	0.0	0.0	100.0
	Mask-RCNN	3765	98.7	**97.1**	0.0	0.0	0.0	15.5	0.0	0.0	100.0
	Gtsrb-CNN	3760	40.5	37.0	**99.9**	0.0	16.1	51.4	0.0	0.0	72.4
	Lisa-CNN	4998	29.4	28.1	6.8	0.0	0.0	**100.0**	0.0	0.0	77.1
180	Yolov3	7862	**99.9**	4.0	14.6	0.0	0.0	17.5	0.0	0.0	90.6
	Mask-RCNN	4083	96.3	**91.0**	0.2	0.0	0.0	54.8	0.0	0.0	98.9
	Gtsrb-CNN	7426	12.3	2.0	**85.7**	0.0	27.7	13.4	0.0	0.0	44.4
	Lisa-CNN	6268	9.0	0.6	35.7	0.0	0.0	**100.0**	0.0	0.0	26.2
300	Yolov3	5169	**96.5**	3.6	2.5	0.0	0.0	2.3	0.0	0.0	72.3
	Mask-RCNN	3543	32.0	**14.0**	0.1	0.0	0.0	10.4	0.0	0.0	65.9
	Gtsrb-CNN	3438	2.0	2.9	**48.0**	0.0	43.1	44.0	0.0	0.0	47.6
	Lisa-CNN	4388	0.7	4.9	8.6	0.0	0.0	**100.0**	0.0	0.0	25.0
440	Yolov3	6716	**49.5**	0.8	40.3	0.0	0.0	40.9	0.0	0.0	35.3
	Mask-RCNN	6023	5.4	**3.3**	41.1	0.0	0.0	35.6	0.0	0.0	33.6
	Gtsrb-CNN	6565	0.7	0.7	**43.7**	0.0	44.1	35.6	0.0	0.0	33.1
	Lisa-CNN	6287	1.0	2.4	26.4	0.0	0.0	**97.4**	0.0	0.0	26.8
600	Yolov3	5507	**17.8**	0.2	32.5	0.0	0.0	27.4	0.0	0.0	23.7
	Mask-RCNN	5058	0.1	**0.4**	5.3	0.0	0.0	4.6	0.0	0.0	16.7
	Gtsrb-CNN	4637	0.0	0.9	**7.2**	0.0	7.5	4.9	0.0	0.0	21.1
	Lisa-CNN	4714	0.0	0.9	8.6	0.0	0.0	**57.5**	0.0	0.0	15.8

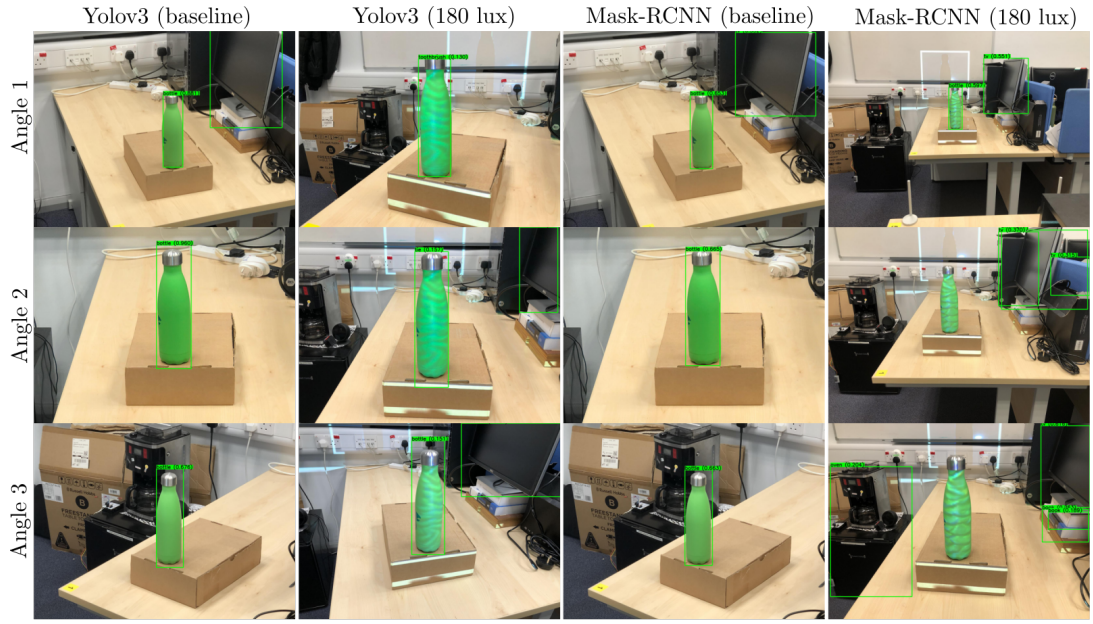


Figure A.3: SLAP attack on class "Bottle" for Yolov3 and Mask-RCNN. The detection thresholds used in the paper are 0.4 and 0.6, respectively.

B

Supplementary Material for Chapter 6

Contents

B.1 Experimental Setup and Tested Cameras 167
B.2 Effect on Autonomous Driving 167

B.1 Experimental Setup and Tested Cameras

We tested a wide variety of cameras, ranging from cheap IoT cameras (YI Home Camera) over mid-range smart cameras (Google Nest) to semi-professional surveillance cameras (Axis M3045-V). Moreover, we tested one camera with a CCD image sensor and an electronic global shutter mechanism. Finally, we tested an Apple iPhone 7. Table B.1 lists all cameras that we evaluated and whether they implement an electronic rolling shutter mechanism or not. We report in Figure B.2 examples of the physical realization of the attack, where a printout image is placed in front of the camera and the rolling attack is executed with the laser.

B.2 Effect on Autonomous Driving

In this section, we use the open-source simulator CARLA [57], in combination with the self-driving car platform Pylot [76] to illustrate the effects of the rolling

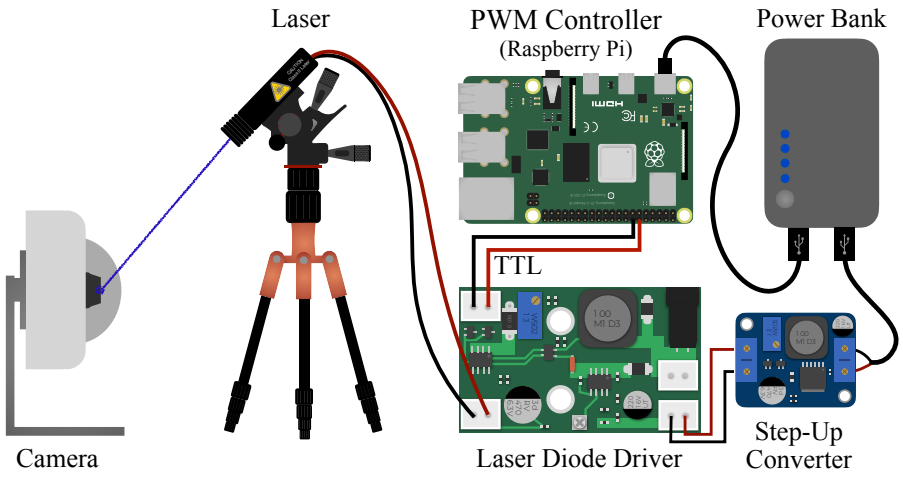


Figure B.1: Visualization of the experimental setup we used to collect the different patterns.

Table B.1: List of the cameras that we tested for the success of our attack. R or G indicate whether the camera uses an electronic rolling or electronic global shutter, respectively. The last column **A** indicates whether we could reproduce the rolling shutter attack for the camera (✓) or whether we could not (✗).

Camera	*Resolution*	*F*	*Shutter*	*A*
Logitech C922	1920×1080	30	R	✓
Axis M3045-V	1920×1080	25	R	✓
YI Home Camera 1080p	1920×1080	24	R	✓
V380 Camera 720P	1280×720	25	R	✓
Google Nest	1920×1080	24	R	✓
Netgear Arlo	1280×720	24	R	✓
Imaging Source DFM 25G445	1280×960	25	G	✗
Apple iPhone 7	3840×2160	30	R	✓

shutter attack on an autonomous system. By default, the Pylot agent is equipped with four sensors: (i) a wide-angle RGB camera for lane detection and object detection/tracking, (ii) a telephoto camera for traffic light detection, (iii) a LiDAR sensor for localization, and (iv) GPS for route planning. We uniquely targeted the wide-angle RGB camera with our attack, to measure how the system as a whole is affected.

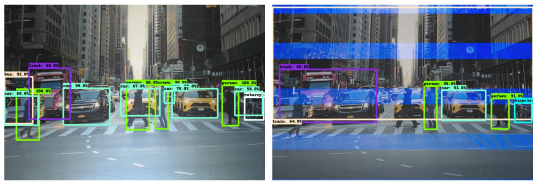
CARLA Setup. We utilized two predefined scenarios from the CARLA scenario runner, where an entity is crossing the road in front of the car: *Scenario 1* involves a pedestrian and *Scenario 2* involves a cyclist. We collected the baseline performance

of Pylot by simulating each scenario 50 times. We then repeated the simulation under the presence of the rolling shutter attack. We intercepted the frames captured by the wide-angle camera as they were forwarded to Pylot and overlayed the rolling shutter pattern, as described in Section 6.6.1. The attack parameters were as follows: $f = 750$Hz, $t_{exp} = 200\mu$s and $t_{on} = 0.53$ms. In line with the previous evaluation, we again used two object detectors, FRCNN and SSD, which are among the pre-implemented options in Pylot.

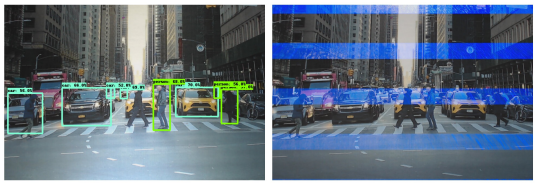
Results. Most of the key findings from the previous section can be transferred to the CARLA simulations. In particular, the performance difference between SSD and FRCNN stands out. While for *Scenario 1* during normal operation for both object detectors the probability of a collision was equal (5%), under the rolling shutter attack, SSD showed a 67% probability of a collision compared to 47% for FRCNN. Similar behaviour can be observed for the baseline of *Scenario 2*. FRCNN showed a 0% probability of a collision, while using SSD led to an increase to 15%. However, interestingly, under attack, FRCNN experienced a strong performance drop and fell behind SSD. The probability of a safety infraction increased to 97% compared to 65% for SSD.

In Figure B.3, the velocity curve of the autonomous vehicle for the two scenarios and object detectors is presented. The figure shows that in the baseline, the car immediately reduces speed as soon as the non-player character starts moving. In contrast, as the braking gap shows, during the attack, a noticeable delay in braking can be observed.

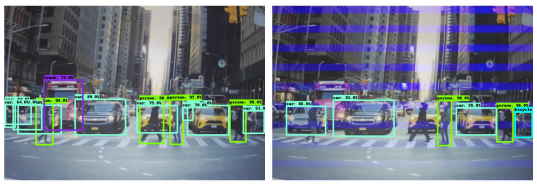
Figure B.2: Physical evaluation of the rolling shutter attack. These images were collected during a physical experiment where we printed images, placed them in front of the camera, and carried out the attack with the laser. The first column reports the object detection results on FRCNN, the second column on SSD.



(a) Logitech C922



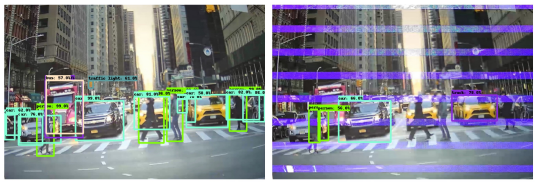
(b) Logitech C922



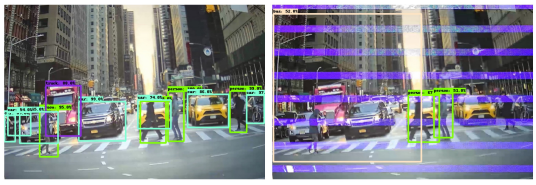
(c) Axis M3045-V



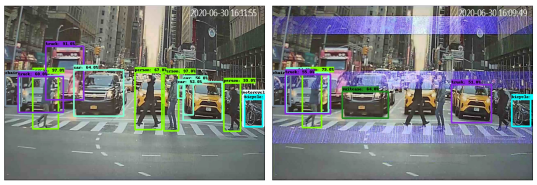
(d) Axis M3045-V



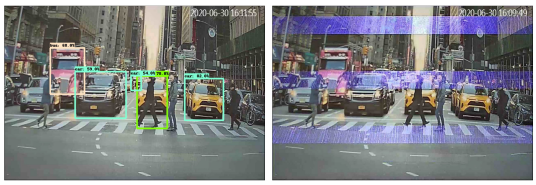
(e) YI Home Camera 1080p



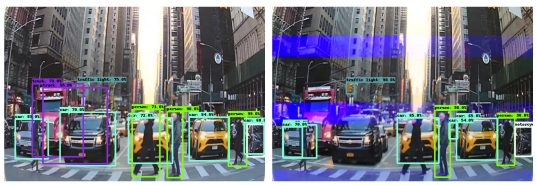
(f) YI Home Camera 1080p



(g) V380 Camera 720p



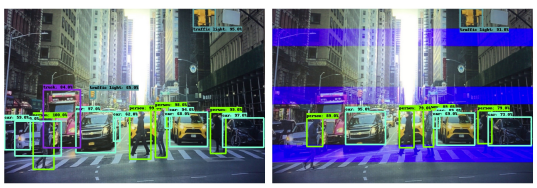
(h) V380 Camera 720p



(i) Netgear Arlo



(j) Netgear Arlo



(k) Apple iPhone 7



(l) Apple iPhone 7

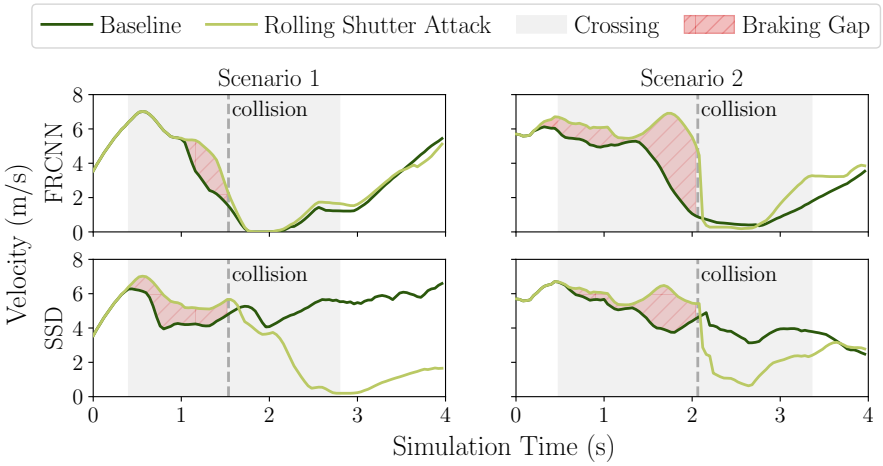


Figure B.3: Velocity curve of the agent during normal operation (baseline) and during the rolling shutter attack. The grey area shows the time when the pedestrian/cyclist crosses the road. The dashed line marks the time when the agent collides with the non-player character.

C

Supplementary Material for Chapter 7

Contents

C.1 Effect of Input Mean on Attention Weight Robustness 173
C.2 Ablation on Attention-Fool on ViT 175
C.2.1 Normalization 175
C.2.2 Patch Location 175
C.3 Ablation on Targeted Attacks on ViT 176
C.4 Adversarial Token Replacement 177
C.5 Additional Visualizations 178

C.1 Effect of Input Mean on Attention Weight Robustness

As discussed in in Section 7.4.2, less centered inputs X (larger absolute value of input mean $|\mu|$) make dot-product self-attention less robust to patch attacks on the controlled setting (when input variance is constant). We recap that the input mean is $\mu = \mu \cdot \mathbf{1}$, input standard deviation $\sigma = \mathbf{1}$, and $W_Q = -w \cdot \mathbb{I}_{d_k}$ and $W_K = w \cdot \mathbb{I}_{d_k}$.

We now denote the query mean by $\mu_Q = \text{Mean}(W_Q \cdot X) = -w\mu$ and key mean by $\mu_K = \text{Mean}(W_K \cdot X) = w\mu$. We note that the element-wise distance between query and key mean is $|\mu_K^{(i)} - \mu_Q^{(i)}| = |w\mu_i + w\mu_i| = 2|w||\mu|$. On the other hand, for

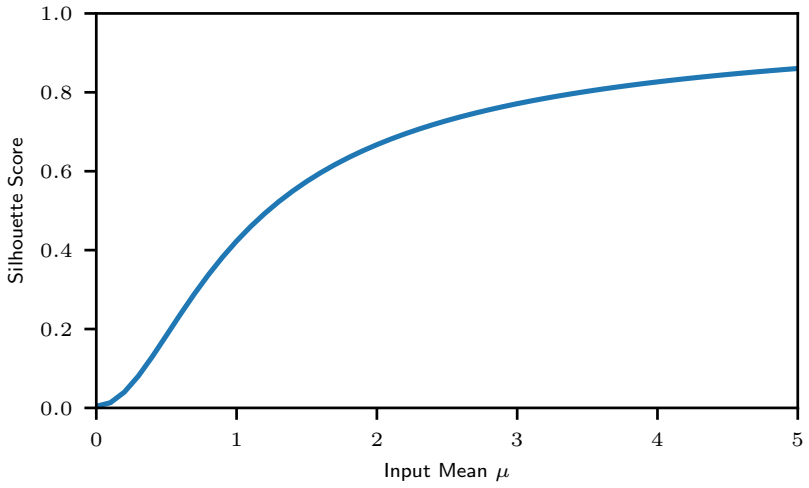


Figure C.1: Silhouette score between clusters of projected keys and queries for different choices of μ on synthetic data. Larger scores corresponds to keys and queries forming more distinct clusters.

query standard deviation, we have $\sigma_Q = \text{StdDev}(W_Q \cdot X) = \text{StdDev}(-wX) = w\mathbf{1}$ and for key standard deviation $\sigma_K^2 = \text{StdDev}(W_K \cdot X) = \text{StdDev}(wX) = w\mathbf{1}$.

As can be seen. increasing $|\mu|$ increases the distance between query and key cluster mean, while leaving the key's and query's standard deviation unchanged. As a result, it increases the separation of key and query clusters (see also Figure 7.1). On the other hand, increasing $|w|$ has no systematic effect on the separation of query and key cluster, as w's effect on mean and standard deviation cancels out.

We empirically quantify the query and key cluster separation using the Silhouette score [168]. Figure C.1 shows this score as a function of the input mean scale $|\mu|$ in the controlled setting. The plot confirms the above theoretical argument: increased input mean results in more separated keys and queries.

The result of this separation of keys and queries is that attention drawn by one key can increase for all queries when moving the key in the direction of the query mean (because in a sense, all queries lie in the same direction from the key as long as they are distant and have small variance). On the other hand, if keys and queries lie intermingled, than for any direction the key moves, it will get closer to some queries at the expense of increasing distance to other queries. Because

Table C.1: Robust accuracies (%) under Attention Fool adversarial patch attach using the un-normalized P_Q^{hl} and P_K^{hl} introduced in Section 7.5. All rows are computed using PGD^{250} with momentum and step size α=8/255. Without normalization, $\mathcal{L}_{kq}$ and $\mathcal{L}_{kq\star}$ do not improve on $\mathcal{L}_{ce}$ baselines and in fact perform much worse, likely because individual $\mathcal{L}_{kq}^{hl}$ losses are not commensurable with each other.

	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
$\mathcal{L}_{ce}$	0.1	13.5	31.2	19.8	36.0	58.8
$+\mathcal{L}_{kq}$	53.1(+53.0)	81.7(+68.2)	84.5(+53.3)	67.4(+47.6)	79.8(+43.8)	80.6(+21.8)
$+\mathcal{L}_{kq*}$	24.4(+24.3)	79.8(+66.3)	82.0(+50.8)	49.0(+29.2)	79.6(+43.6)	80.8(+22.0)
+ Momentum	0.00	3.1	13.2	1.5	16.8	41.7
$+\mathcal{L}_{kq}$	50.0(+50.0)	81.6(+78.5)	84.3(+71.1)	66.8(+65.3)	78.3(+61.5)	80.3(+38.6)
$+\mathcal{L}_{kq*}$	21.1(+21.1)	80.3(+77.2)	82.1(+68.9)	24.1(+22.6)	78.6(+61.8)	80.1(+38.4)

of this, for an adversarial patch attack on the attention weights, it is beneficial if keys and queries are well separated (as in Figure 7.1).

C.2 Ablation on Attention-Fool on ViT

C.2.1 Normalization

In Section 7.5 we introduced an $\ell_{1,2}$ normalization in the computation of $\mathcal{L}_{kq}$, where projected queries and keys are normalized as $\bar{P}_Q^{hl} = {P_Q^{hl}}/{\frac{1}{n}||P_Q^{hl}||_{1,2}}$ and $\bar{P}_K^{hl} = {P_K^{hl}}/{\frac{1}{n}||P_K^{hl}||_{1,2}}$. To evaluate the effect of this normalization we compute $\mathcal{L}_{kq}$ without it, repeating the experiment reported in Table 7.3, and report the results in Table C.1. We observe that this $\ell_{1,2}$ normalization is very crucial and without normalization, performance of $\mathcal{L}_{kq}$ and $\mathcal{L}_{kq^\star}$ deteriorates considerably, to levels clearly below $\mathcal{L}_{ce}$.

C.2.2 Patch Location

While Attention-Fool was designed to operate for arbitrary adversarial patch locations (as long as the adversarial patch aligns with ViT/DeiTs patch tiling), the location may have an effect on the resulting robust accuracies. In Section 7.6.2 we set the patch location to be the top left-most corner of the image. Here, we repeat the experiment but we place the patch in the image centre, and we target the corresponding key. Specifically, in 224×224-resolution models we place the patch top-left corner at 96,96, while in 384×384 models we place it at 176,176.

Table C.2: Robust accuracies (%) of ResNet50 and different vision transformers with the adversarial patch positioned at the centre of the images. The evaluation setting is similar to Table 7.3 except for the position of the patch. Here, the robustness of ResNet50 has further deteriorated compared to a patch placed at the image corners. In contrast, transformers demonstrate similar vulnerability with the patch positioned at the centre of the images and corner of the images under our Attention Fool loss variant $\mathcal{L}_{kq\star}$. These results indicate that the transformers are less sensitive to location of the adversarial patch compared to CNNs.

	ResNet50	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
$\mathcal{L}_{ce}$	41.5	0.1	14.5	28.9	23.5	44.2	66.5
$+\mathcal{L}_{kq}$	-	0.0(–0.1)	5.8(–8.6)	23.1(–5.8)	22.1(–1.3)	44.9(+0.7)	69.1(+2.6)
$+\mathcal{L}_{kq*}$	-	0.0(–0.1)	4.1(–10.4)	18.8(–10.1)	12.6(–10.9)	35.9(–8.3)	67.2(+0.7)
+ Momentum	31.1	0.0	2.4	11.1	1.6	20.9	42.3
$+\mathcal{L}_{kq}$	-	0.0(-0.0)	0.3(–2.1)	3.1(–8.0)	0.1(–1.4)	28.50(+7.6)	48.9(+6.6)
$+\mathcal{L}_{kq*}$	-	0.0(-0.0)	0.2(–2.2)	2.6(–8.5)	0.1(–1.5)	11.7(–9.2)	45.9(+3.6)

Table C.3: Adversarial patch attack success rate (%) for a targeted attack with target class "0". The evaluation setting is similar to Table 7.3. Here, the attack success rate in vision transformers is larger than that of the ResNet50 model, but the improvements obtained with Attention Fool in comparison to the cross-entropy baseline are not as consistent as in the untargeted attack setting.

	ResNet50	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
$\mathcal{L}_{ce}$	30.7	93.2	30.0	11.6	43.0	14.1	1.9
$+\mathcal{L}_{kq}$	-	94.6(-1.4)	20.0(+10.0)	12.2(-0.6)	48.4(-5.4)	14.8(-0.7)	1.7(+0.2)
$+\mathcal{L}_{kq*}$	-	94.3(-1.1)	19.5(+10.5)	13.0(-1.4)	49.1(-6.1)	14.6(-0.5)	2.3(-0.4)
+ Momentum	38.9	100.0	42.7	23.8	100.0	75.2	4.7
$+\mathcal{L}_{kq}$	-	100.0(+0.0)	43.1(-0.4)	21.1(+2.7)	99.1(+0.9)	74.6(+0.6)	4.6(+0.1)
$+\mathcal{L}_{kq*}$	-	100.0(+0.0)	43.6(-0.9)	22.4(+1.4)	98.4(+1.6)	77.1(-1.9)	4.0(+0.7)

We repeat the evaluation of Section 7.6.2 in this setting, and report the results in Table C.2. Notably, the table shows that we obtain a significant drop in ResNet50 robust accuracy because of the different location (from 49.00% in Table 7.3 to 31.10% in Table C.2), while the robust accuracies of vision transformers ViT and DeiT do not change significantly. Moreover, $\mathcal{L}_{kq\star}$ improves upon $\mathcal{L}_{ce}$ for all but the DeiT-B-384 model.

C.3 Ablation on Targeted Attacks on ViT

In Section 7.6.2, we reported the robust accuracies for vision transformers under an untargeted patch attacks. Here, we also evaluate a targeted attack on the same

models by selecting a target class in the optimization: rather than maximizing $\mathcal{L}_{ce}$ with the true image class y as in Equation 7.1, we replace y with a target class $y^\star$ and minimize the $\mathcal{L}_{ce}$ instead (note that the $\mathcal{L}_{kq}$ loss is still maximized in a targeted setting). The way we combine $\mathcal{L}_{ce}$ with the Attention-Fool variants is identical as in Section 7.6.2; here we choose $y^\star = 0$. In this case, rather than reporting robust accuracies, we report attack success rate, i.e., the percentage of times the addition of the adversarial patch changed the correct classification of an image into the target class $y^\star$. We report the results in Table C.3, note that the colours and the signs of improvements/degradation are inverted in comparison to the other tables. Table C.3 shows that the benefit of Attention-Fool in targeted attacks is less clear. We hypothesise that different weighting of $\mathcal{L}_{ce}$ and $\mathcal{L}_{kq}$ might be required in targeted settings, specifically because $\mathcal{L}_{ce}$ is bounded by zero when minimized (while it is unbounded when maximized in the untargeted setting). Additionally, specific properties of the target class "0" on certain models might bias the evaluation; a larger, more in-depth evaluation of targeted attacks is left for future work.

C.4 Adversarial Token Replacement

In Section 7.6.3 we showed how targeting a single key with Attention-Fool in DETR can lead to large degradation in mAP. Differently than in ViT models, because of DETR's hybrid architecture (CNN plus Transformer) an adversarial patch placed on the DETR image input affects a number of tokens in the encoder's input: all those backbone outputs (tokens) whose receptive field via the CNN overlaps with the input image patch. Here, to show that the attack performance can be attributed to the individual key token that is the Attention-Fool target, we replace this key with its clean counterpart. To do so, we compute all keys on the clean image and all keys in the patched image (by forwarding the image through the CNN backbone), and we replace the target adversarial key with its clean counterpart. This is either the key indexed by 2,2 or by 4,4 in non-dilated and dilated models, respectively. We report the resulting mAPs in Table C.4. The table shows how replacing this single token's key removes a large part of the adversarial effect in

Table C.4: Mean Average Precisions in presence of adversarial patches computed with $\mathcal{L}_{ce}$ and Attention-Fool's $\mathcal{L}_{kq}^{(1)}$ patches on DETR models. Differently than in Table 7.5, here we replace the adversarial target key – one of the units in the backbone feature map – with its clean counterpart (the same unit computed in the non-patched image). Replacing the adversarial key removes the entirety of the adversarial effect of Attention-Fool, showing how the mAP degradation under attack can be attributed to the target key alone. Three different patch sizes and four different DETR models are considered, based either on R50 or R101 backbone and with and without dilation.

	R50	DC5-R50	R101	DC5-R101
Clean mAP	53.00	54.25	54.41	56.74
64×64: $\mathcal{L}_{ce}$	38.96	17.51	33.35	35.67
$+\mathcal{L}_{kq}^{(1)}$	51.85 (+12.90)	52.92 (+35.41)	48.85 (+15.50)	56.61 (+20.94)
$\mathcal{L}_{kq}^{(1)}$ only	52.13 (+13.18)	53.48 (+35.97)	50.71 (+17.36)	57.11 (+21.44)
56×56: $\mathcal{L}_{ce}$	41.18	25.09	33.64	38.50
$+\mathcal{L}_{kq}^{(1)}$	53.34 (+12.17)	53.66 (+28.56)	53.51 (+19.87)	56.81 (+18.31)
$\mathcal{L}_{kq}^{(1)}$ only	52.73 (+11.55)	53.28 (+28.19)	52.37 (+18.74)	57.18 (+18.68)
48×48: $\mathcal{L}_{ce}$	43.61	27.84	40.45	42.16
$+\mathcal{L}_{kq}^{(1)}$	52.54 (+8.93)	52.97 (+25.13)	53.52 (+13.06)	56.90 (+14.73)
$\mathcal{L}_{kq}^{(1)}$ only	53.27 (+9.66)	53.99 (+26.16)	53.58 (+13.13)	56.90 (+14.74)

all rows using $\mathcal{L}_{kq}^{(1)}$ loss. In comparison, in rows using $\mathcal{L}_{ce}$, replacing this token's key still results in low mAPs, showing that in this case the adversarial effect is not attributable to the same adversarial token alone.

C.5 Additional Visualizations

We report additional visualizations akin to those in Fig. 7.1 and Fig. 7.4 in Figure C.3 Figure C.2, respectively. These are hand-picked as examples showcasing successful adversarial patches.

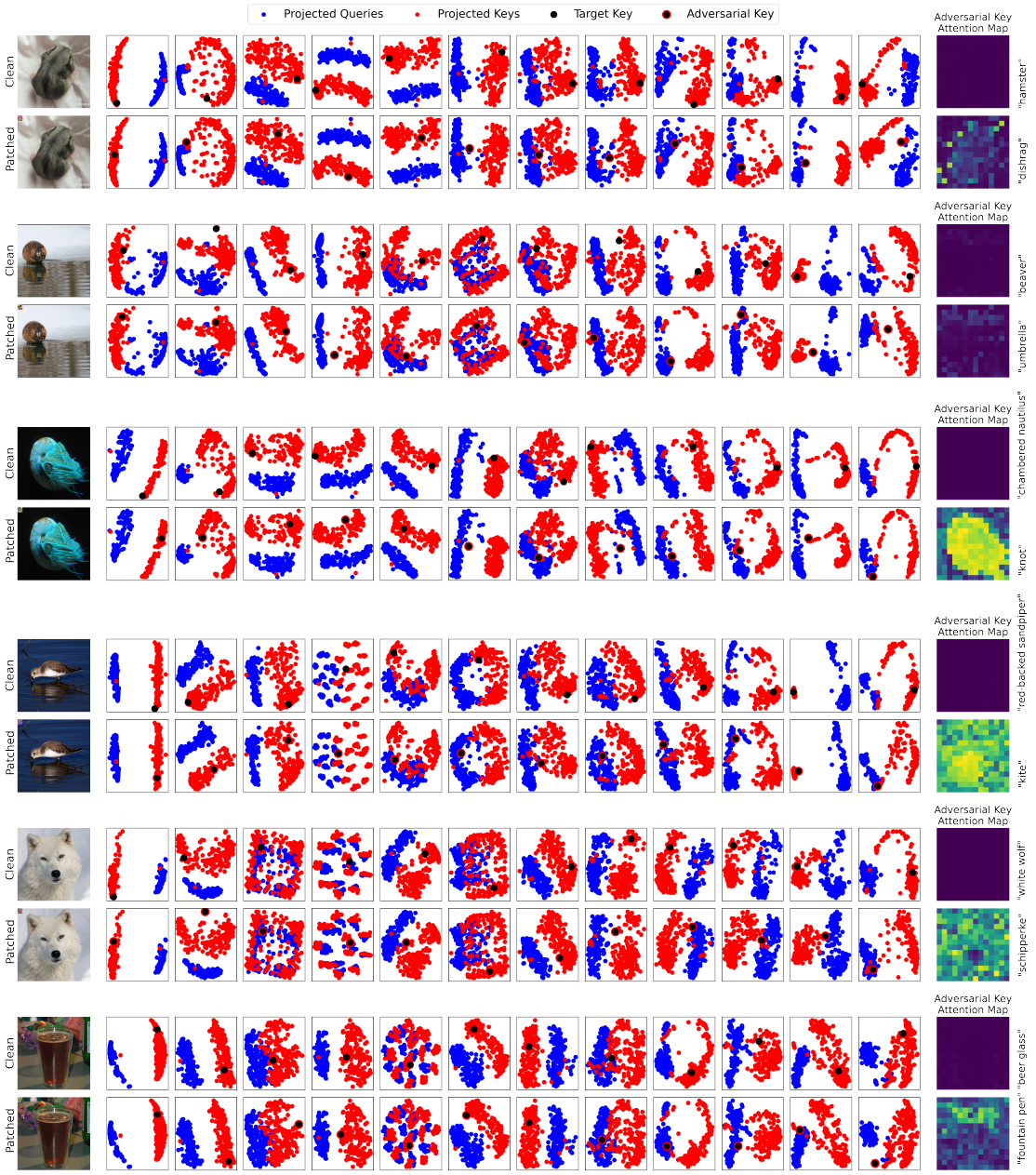


Figure C.2: Embedded projected key and query tokens for clean and patched input images on each of the 12 DeiT-B layers, for a single attention head. For each image we chose an attention head which showed large amount of changes. The last column reports the attention map weights of the adversarial key on the last layer – showing that generally the key draws a large amount of attention from queries. Note that while the adversarial patch tends to focus on drawing the attention on the last layer (which is visualized in the right-most column), it can target arbitrary layers – in fact, we obtain successful mis-classifications even if the last layer attention map only has minor changes.

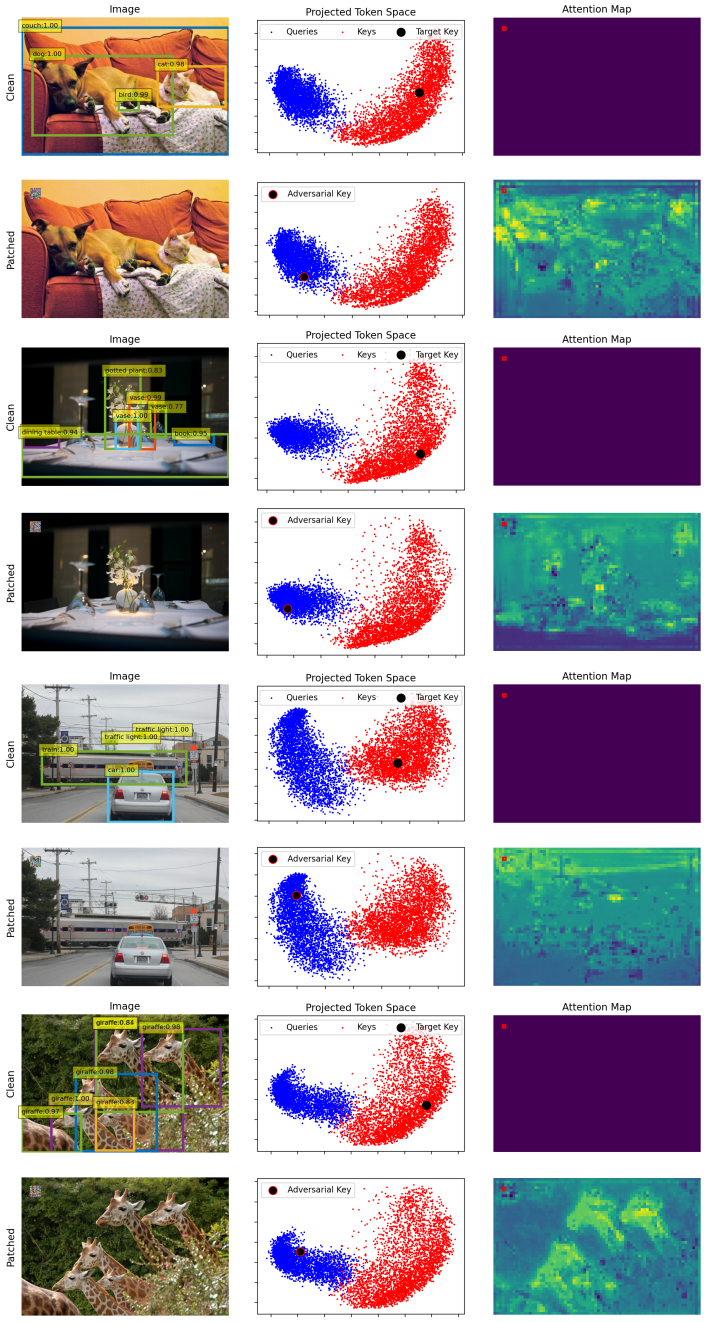


Figure C.3: More comparisons of clean and adversarially patched input for DETR [35]. The patch shifts a targeted key token towards the cluster of query tokens. In dot-product attention, this directs queries attention to the malicious token and prevents the model from detecting the remaining objects. The right-most column compares queries' attention weights to the adversarial key, whose location is marked by a red box, between clean and patched inputs. These images use DETR DC5-R50 and patches are optimized with $\mathcal{L}_{ce} + \mathcal{L}_{kq}^{(1)}$.

Bibliography

[1] OpenSignal PPG dataset. http://opensignals.net, 2017. [Online] Accessed 2018-04-17.

[2] Ahmed Aldahdooh, Wassim Hamidouche, and Olivier Deforges. "Reveal of vision transformers robustness against adversarial attacks". *arXiv preprint 2106.03734*, 2021.

[3] Amazon.com. 1,8 W A M140 445 Nm blue Diode. https://amazon.com/dp/B00UECEMEE. [Online] Accessed 2020-06-01.

[4] Amazon.com. JING TAI Powerful Pointer Beam Red/Green/Purple Color Starry Star Pen Beam Light 5MW 532nm. https://amazon.com/dp/B07FZCFL3V. [Online] Accessed 2020-06-01.

[5] Android. Measuring biometric unlock security. https://source.android.com/security/biometric/measure, 2022. [Online] Accessed 2022-05-31.

[6] Anonymous. "Are vision transformers robust to patch-wise perturbations?". In *Submitted to The Tenth International Conference on Learning Representations*, 2022. under review.

[7] Anonymous. "Patch-fool: Are vision transformers always robust against adversarial perturbations?". In *Submitted to The Tenth International Conference on Learning Representations*, 2022. under review.

[8] Apollo. Apolloauto - an open autonomous driving platform. http://github.com/apolloauto. [Online] Accessed 2021-02-19.

[9] Apple. About Face ID advanced technology. https://support.apple.com/en-gb/HT208108, 2017. [Online] Accessed 2022-02-01.

[10] Apple. About Touch ID advanced security technology. https://support.apple.com/en-gb/HT204587, 2017. [Online] Accessed 2018-04-18.

[11] Apple. Face ID Security. https://images.apple.com/business/docs/FaceID_Security_Guide.pdf, 2017. [Online] 2018-04-18.

[12] Aptina. 1/3-inch cmos digital image sensor ar0132at data sheet. http://datasheetspdf.com/pdf/829321/AptinaImagingCorporation/AR0132AT/1f. [Online] Accessed 2021-02-19.

[13] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. "Synthesizing robust adversarial examples". In *International Conference on Machine Learning*, pages 284–293. PMLR, 2018.

[14] Guha Balakrishnan, Fredo Durand, and John Guttag. "Detecting pulse from head motions in video". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3430–3437, 2013.

[15] Shu-Di Bao, Yuan-Ting Zhang, and Lian-Feng Shen. "Physiological signal based entity authentication for body area sensor networks and mobile healthcare systems".

In *IEEE Engineering in Medicine and Biology Annual Conference*, pages 2455–2458. IEEE, 2006.

[16] Dieter Barschdorff and Wei Zhang. "Respiratory rhythm detection with photoplethysmographic methods". In *Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 912–913. IEEE, 1994.

[17] B.E. Bayer. "Colour filter array". *United States of America patent*, 3971065, 1976.

[18] Josh Beal, Eric Kim, Eric Tzeng, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. "Toward transformer-based object detection". *arXiv preprint 2012.09958*, 2020.

[19] Philipp Benz, Soomin Ham, Chaoning Zhang, Adil Karjauv, and In So Kweon. "Adversarial robustness comparison of vision transformer and MLP-mixer to CNNs". *arXiv preprint 2110.02797*, 2021.

[20] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. "Understanding robustness of transformers for image classification". *arXiv preprint 2103.14586*, 2021.

[21] Battista Biggio, Luca Didaci, Giorgio Fumera, and Fabio Roli. "Poisoning attacks to compromise face templates". In *International Conference on Biometrics*, pages 1–7, 2013.

[22] Battista Biggio, Giorgio Fumera, Fabio Roli, and Luca Didaci. "Poisoning adaptive biometric systems". *Lecture Notes in Computer Science*, 7626:417–425, 2012.

[23] Battista Biggio, Giorgio Fumera, Paolo Russu, Luca Didaci, and Fabio Roli. "Adversarial biometric recognition: A review on biometric system security from the adversarial machine-learning perspective". *IEEE Signal Processing Magazine*, 32(5):31–41, 2015.

[24] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Poisoning attacks against support vector machines". In *International Conference on Machine Learning*, page 1467–1474. Omnipress, 2012.

[25] Battista Biggio and Fabio Roli. "Wild patterns: Ten years after the rise of adversarial machine learning". *Pattern Recognition*, 84:317–331, 2018.

[26] BMW. BMW TechOffice Munich. `http://github.com/BMW-InnovationLab`. [Online] Accessed 2020-02-19.

[27] Angelo Bonissi, Ruggero Donida Labati, Luca Perico, Roberto Sassi, Fabio Scotti, and Luca Sparagino. "A preliminary study on continuous authentication methods for photoplethysmographic biometrics". In *IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications*, pages 28–33. IEEE, 2013.

[28] Joseph Bonneau, Cormac Herley, Paul C. Van Oorschot, and Frank Stajano. "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes". In *IEEE Symposium on Security and Privacy*, pages 553–567. IEEE, 2012.

[29] Andy Boxall. From robots to projection mapping: Inside Panasonic's Tokyo 2020 olympic tech. `http://digitaltrends.com/mobile/panasonic-tokyo-2020-technology-interview/`. [Online] Accessed 2021-02-19.

[30] Tom Brown, Dandelion Mane, Aurko Roy, Martin Abadi, and Justin Gilmer. "Adversarial patch". *arXiv preprint 1712.09665v2*, 2018.

[31] Anne M. Brumfield and Michael E. Andrew. "Digital pulse contour analysis: investigating age-dependent indices of arterial compliance". *Physiological measurement*, 26(5):599, 2005.

[32] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. "Vggface2: A dataset for recognising faces across pose and age". In *IEEE International Conference on Automatic Face & Gesture Recognition)*, pages 67–74. IEEE, 2018.

[33] Yulong Cao, Jiaxiang Ma, Kevin Fu, Rampazzi Sara, and Morley Mao. "Automated tracking system for LiDAR spoofing attacks on moving targets". In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2021.

[34] Yulong Cao, Yimeng Zhou, Qi Alfred Chen, Chaowei Xiao, Won Park, Kevin Fu, Benjamin Cyr, Sara Rampazzi, and Z. Morley Mao. "Adversarial sensor attack on LiDAR-based perception in autonomous driving". In *ACM SIGSAC Conference on Computer and Communications Security*, pages 2267–2281, 2019.

[35] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-end object detection with transformers". In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[36] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. "On evaluating adversarial robustness". *arXiv preprint 1902.06705*, 2019.

[37] Nicholas Carlini and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods". In *Proceedings of the ACM Workshop on Artificial Intelligence and Security*, page 3–14. Association for Computing Machinery, 2017.

[38] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In *IEEE Symposium on Security and Privacy*, pages 39–57. IEEE, 2017.

[39] Denisse Castaneda, Aibhlin Esparza, Mohammad Ghamari, Cinna Soltanpur, and Homer Nazeran. "A review on wearable photoplethysmography sensors and their potential future applications in health care". *International journal of biosensors & bioelectronics*, 4(4):195, 2018.

[40] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks". In *IEEE Winter Conference on Applications of Computer Vision*, pages 839–847, 2018.

[41] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. "Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector". In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer, 2018.

[42] E. Chou, F. Tramèr, and G. Pellegrino. "SentiNet: Detecting localized universal attacks against deep learning systems". In *IEEE Security and Privacy Workshops (SPW)*, pages 48–54, 2020.

[43] Eric Chern-Pin Chua, Stephen J Redmond, Gary McDarby, and Conor Heneghan. "Towards using photo-plethysmogram amplitude to measure blood pressure during sleep". *Annals of biomedical engineering*, 38(3):945–954, 2010.

[44] Axis Communications. Network video solutions for all industries. `https://axis.com/en-us/solutions-by-industry`, 2020. [Online] Accessed 2022-02-01.

[45] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. "RobustBench: a standardized adversarial robustness benchmark". In *NEURIPS - Datasets and Benchmarks Track*, 2021.

[46] Francesco Croce, Maksym Andriushchenko, Naman D. Singh, Nicolas Flammarion, and Matthias Hein. "Sparse-RS: a versatile framework for query-efficient sparse black-box adversarial attacks". In *AAAI Conference on Artificial Intelligence*, 2022.

[47] Francesco Croce and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In *International Conference on Machine Learning*, pages 2206–2216. PMLR, 2020.

[48] Anthony Cuthbertson. Hong Kong protesters use lasers to avoid facial recognition cameras and blind police. `https://independent.co.uk/news/world/asia/hong-kong-protests-lasers-facial-recognition-ai-china-police-a9033046.html`, 2019. [Online] Accessed 2022-02-01.

[49] Gunnar Dahlgren, Flavio Veintemilla, Göran Settergren, and Jan Liska. "Left ventricular end-systolic pressure estimated from measurements in a peripheral artery". *Journal of cardiothoracic and vascular anesthesia*, 5(6):551–553, 1991.

[50] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. "R-FCN: Object detection via region-based fully convolutional networks". In *Conference on Neural Information Processing Systems*, pages 379–387, 2016.

[51] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. "Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks". In *USENIX Security Symposium*, pages 321–338. USENIX Association, 2019.

[52] Henry Dietz and Paul Eberhart. "Shuttering methods and the artifacts they produce". *Electronic Imaging*, 2019(4):590–591, 2019.

[53] George Doddington, Walter Liggett, Alvin Martin, Mark Przybocki, and Douglas Reynolds. Sheep, goats, lambs and wolves: A statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation. `https://apps.dtic.mil/sti/pdfs/ADA528610.pdf`, 1998. [Online] Accessed 2022-02-01.

[54] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. "Decaf: A deep convolutional activation feature for generic visual recognition". In *International Conference on Machine Learning*, pages 647–655. PMLR, 2014.

[55] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. "Boosting adversarial attacks with momentum". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018.

[56] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale". *arXiv preprint 2010.11929*, 2020.

[57] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An open urban driving simulator". In *Annual Conference on Robot Learning*, pages 1–16, 2017.

[58] Daniel Durini. *High performance silicon imaging: Fundamentals and applications of CMOS and CCD sensors*. Elsevier, 2019.

[59] Simon Eberz, Giulio Lovisotto, Andrea Patane, Marta Kwiatkowska, Vincent Lenders, and Ivan Martinovic. "When your fitness tracker betrays you: Quantifying the predictability of biometric features across contexts". In *IEEE Symposium on Security and Privacy*, pages 889–905. IEEE, 2018.

[60] Simon Eberz, Giulio Lovisotto, Kasper B. Rasmussen, Vincent Lenders, and Ivan Martinovic. "28 blinks later: Tackling practical challenges of eye movement biometrics". In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1187–1199, 2019.

[61] Simon Eberz, Kasper B. Rasmussen, Vincent Lenders, and Ivan Martinovic. "Looks like eve: Exposing insider threats using eye movement biometrics". *ACM Transactions on Privacy and Security*, 19(1):1–31, 2016.

[62] Simon Eberz, Kasper B. Rasmussen, Vincent Lenders, and Ivan Martinovic. "Evaluating behavioral biometrics for continuous authentication: Challenges and metrics". In *Asia Conference on Computer and Communications Security*, pages 386–399, 2017.

[63] Mohamed Elgendi. "On the analysis of fingertip photoplethysmogram signals". *Current cardiology reviews*, 8(1):14–25, 2012.

[64] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. "Exploring the landscape of spatial robustness". In *International Conference on Machine Learning*, pages 1802–1811, 2019.

[65] Patrick Esser, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021.

[66] EVERGUARD.ai. sentri360. https://everguard.ai/sentri360.php, 2020. [Online] Accessed 2022-02-01.

[67] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. "Robust physical-world attacks on deep learning visual classification". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[68] Facebook Research. DETR: End-to-End Object Detection with Transformers. `https://github.com/facebookresearch/detr`. [Online] Accessed 2021-11-16.

[69] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication". *IEEE Transactions on Information Forensics and Security*, 8(1):136–148, 2012.

[70] Antonio Fratini, Mario Sansone, Paolo Bifulco, and Mario Cesarelli. "Individual identification via electrocardiogram analysis". *Biomedical engineering online*, 14(1):1–23, 2015.

[71] Giuseppe Garofalo, Vera Rimmer, Davy Preuveneers, and Wouter Joosen. "Fishy faces: Crafting adversarial images to poison face authentication". In *USENIX Workshop on Offensive Technologies*, 2018.

[72] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. "AI2: Safety and robustness certification of neural networks with abstract interpretation". In *IEEE Symposium on Security and Privacy*, pages 3–18, 2018.

[73] Martin Georgiev, Simon Eberz, Henry Turner, Giulio Lovisotto, and Ivan Martinovic. "Common evaluation pitfalls in touch-based authentication system". In *Asia Conference on Computer and Communications Security*, 2022.

[74] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. "Video action transformer network". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019.

[75] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[76] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A. Wright, Joseph E. Gonzalez, and Ion Stoica. "Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles". *arXiv preprint 2104.07830*, 2021.

[77] Google. Google vision api. `https://cloud.google.com/vision`. [Online] Accessed 2020-10-12.

[78] Google Nest. Nest cam. `https://store.google.com/us/product/nest_cam_outdoor`, 2020. [Online] Accessed 2022-02-01.

[79] Shixiang Gu and Luca Rigazio. "Towards deep neural network architectures robust to adversarial examples". *arXiv preprint 1412.5068*, 2014.

[80] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "BadNets: Identifying vulnerabilities in the machine learning model supply chain". *arXiv preprint 1708.06733*, 2017.

[81] Y.Y. Gu and Y.T. Zhang. "Photoplethysmographic authentication through fuzzy logic". In *IEEE EMBS Asian-Pacific Conference on Biomedical Engineering*, pages 136–137. IEEE, 2003.

[82] Y.Y. Gu, Y. Zhang, and Y.T. Zhang. "A novel biometric approach in human verification by photoplethysmographic signals". In *IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine*, pages 13–14. IEEE, 2003.

[83] Phil Hall. The exposure triangle: Aperture, shutter speed and iso explained. `http://techradar.com/uk/how-to/the-exposure-triangle`. [Online] Accessed 2021-02-19.

[84] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2961–2969, 2017.

[85] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[86] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. "Universal adversarial perturbations against semantic image segmentation". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2755–2764, 2017.

[87] Alex Hern. "Hacker fakes german minister's fingerprints using photos of her hands". *The Guardian*, 30, 2014.

[88] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. "Detection of traffic signs in real-world images: The german traffic sign

detection benchmark". In *International Joint Conference on Neural Networks*, pages 1–8, 2013.

[89] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. "Labeled faces in the wild: A database for studying face recognition in unconstrained environments". *University of Massachusetts Amherst Technical Report*, 1:07–49, 2007.

[90] In Choel Jeong, Hyungro Yoon, Hyunjeong Kang, and Hojun Yeom. "Effects of skin surface temperature on photoplethysmograph". *Journal of healthcare engineering*, 5(4):429–438, 2014.

[91] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. "Transgan: Two transformers can make one strong GAN". *arXiv preprint 2102.07074*, 1(3), 2021.

[92] Ameya Joshi, Gauri Jagatap, and Chinmay Hegde. "Adversarial token attacks on vision transformers". *arXiv preprint 2110.04337*, 2021.

[93] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. "XRAI: Better attributions through regions". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4948–4957, 2019.

[94] Nima Karimian, Zimu Guo, Mark Tehranipoor, and Domenic Forte. "Human recognition from photoplethysmography (ppg) based on non-fiducial features". In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4636–4640. IEEE, 2017.

[95] Nima Karimian, Mark Tehranipoor, and Domenic Forte. "Non-fiducial ppg-based authentication for healthcare application". In *IEEE EMBS International Conference on Biomedical & Health Informatics*, pages 429–432. IEEE, 2017.

[96] Walter Karlen, Srinivas Raman, J. Mark Ansermino, and Guy A. Dumont. "Multiparameter respiratory rate estimation from the photoplethysmogram". *IEEE Transactions on Biomedical Engineering*, 60(7):1946–1953, 2013.

[97] Danny Karmon, Daniel Zoran, and Yoav Goldberg. "Lavan: Localized and visible adversarial noise". In *International Conference on Machine Learning*, pages 2507–2515, 2018.

[98] A Reşit Kavsaoğlu, Kemal Polat, and M. Recep Bozkurt. "A novel feature ranking algorithm for biometric recognition with PPG signals". *Computers in biology and medicine*, 49:1–14, 2014.

[99] T Kinugasa, M Noda, T Imaide, I Aizawa, Y Todaka, and M Ozawa. "An electronic variable-shutter system in video camera use". *IEEE Transactions on Consumer Electronics*, CE-33(3):249–255, 1987.

[100] Marius Kloft and Pavel Laskov. "Security analysis of online centroid anomaly detection". *Journal of Machine Learning Research*, 13:3681–3724, 2010.

[101] Sander Koelstra, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani, Touradj Ebrahimi, Thierry Pun, Anton Nijholt, and Ioannis Patras. "Deap: A database for emotion analysis; using physiological signals". *IEEE transactions on affective computing*, 3(1):18–31, 2011.

[102] Sebastian Köhler, Giulio Lovisotto, Simon Birnbach, Richard Baker, and Ivan Martinovic. "They see me rollin': Inherent vulnerability of the rolling shutter in cmos image sensors". In *Annual Computer Security Applications Conference*, page 399–413. Association for Computing Machinery, 2021.

[103] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". *Conference on Neural Information Processing Systems*, 25:1097–1105, 2012.

[104] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". *arXiv preprint 1607.02533*, 2016.

[105] Takao Kuroda. *Essential principles of image sensors*. CRC Press, 2017.

[106] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. "Backpropagation applied to handwritten zip code recognition". *Neural computation*, 1(4):541–551, 1989.

[107] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. "FNet: Mixing tokens with fourier transforms". *arXiv preprint 2105.03824*, 2021.

[108] Haoliang Li, Yufei Wang, Xiaofei Xie, Yang Liu, Shiqi Wang, Renjie Wan, Lap-Pui Chau, and Alex C Kot. "Light can hack your face! Black-box backdoor attack on face recognition systems". *arXiv preprint 2009.06996*, 2020.

[109] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. "SwinIR: Image restoration using swin transformer". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1833–1844, 2021.

[110] Pigpio Library. The pigpio library. `http://abyz.me.uk/rpi/pigpio/index.html`. [Online] Accessed 2022-02-01.

[111] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[112] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common objects in context". In *European Conference on Computer Vision*, pages 740–755, 2014.

[113] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest". In *IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[114] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single shot multibox detector". In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[115] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Authors Yingqi Liu, Weihang Wang, and Xiangyu Zhang. "Trojaning attack on neural networks". In *Network and Distributed System Security Symposium*, 2018.

[116] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "SWIN transformer: Hierarchical vision transformer using shifted windows". *arXiv preprint 2103.14030*, 2021.

[117] Giulio Lovisotto, Simon Eberz, and Ivan Martinovic. "Biometric backdoors: A poisoning attack against unsupervised template updating". In *IEEE European Symposium on Security and Privacy*, pages 184–197, 2020.

[118] Giulio Lovisotto, Nicole Finnie, Mauricio Munoz, Chaithanya Kumar Mummadi, and Jan Hendrik Metzen. "Give me your attention: Dot-product attention considered harmful for adversarial patch robustness". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[119] Giulio Lovisotto, Henry Turner, Simon Eberz, and Ivan Martinovic. "Seeing red: PPG biometrics using smartphone cameras". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 3565–3574, 2020.

[120] Giulio Lovisotto, Henry Turner, Ivo Sluganovic, Martin Strohmeier, and Ivan Martinovic. "SLAP: Improving physical adversarial examples with short-lived adversarial perturbations". In *USENIX Security Symposium*, 2021.

[121] Alan Lukezic, Tomas Vojir, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. "Discriminative correlation filter with channel and spatial reliability". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6309–6318, 2017.

[122] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". *arXiv preprint 1508.04025*, 2015.

[123] Chunchuan Lyu, Kaizhu Huang, and Hai Ning Liang. "A unified gradient regularization family for adversarial examples". In *IEEE International Conference on Data Mining*, 2016.

[124] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks". *arXiv preprint 1706.06083*, 2017.

[125] Y. Maeda, M. Sekine, T. Tamura, A. Moriya, T. Suzuki, and K. Kameyama. "Comparison of reflected green light and infrared photoplethysmography". In *Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2270–2272. IEEE, 2008.

[126] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. "Exploring the limits of weakly supervised pretraining". In *European Conference on Computer Vision*, pages 181–196, 2018.

[127] Aravindh Mahendran and Andrea Vedaldi. "Understanding deep image representations by inverting them". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5188–5196, 2015.

[128] Kaleel Mahmood, Rigel Mahmood, and Marten Van Dijk. "On the robustness of vision transformers to adversarial examples". *arXiv preprint 2104.02610*, 2021.

[129] Yanmao Man, Ming Li, and Ryan Gerdes. "Ghostimage: Perception domain attacks against vision-based object classification systems". *arXiv preprint 2001.07792*, 2020.

[130] Massimo Mancuso and Sebastiano Battiato. "An introduction to the digital still camera technology". *ST Journal of System Research*, 2(2), 2001.

[131] Manfrotto. XPRO geared three-way pan/tilt tripod head. `https://manfrotto.com/global/xpro-geared-three-way-pan-tilt-tripod-head-mhxpro-3wg/`. [Online] Accessed 2022-02-01.

[132] Marco Melis, Ambra Demontis, Maura Pintor, Angelo Sotgiu, and Battista Biggio. "secML: A python library for secure and explainable machine learning". *arXiv preprint 1912.10013*, 2019.

[133] Sandrine C. Millasseau, R.P. Kelly, J.M. Ritter, and P.J. Chowienczyk. "Determination of age-related increases in large artery stiffness by digital pulse contour analysis". *Clinical science*, 103(4):371–377, 2002.

[134] Andreas Mogelmose, Mohan Manubhai Trivedi, and Thomas B. Moeslund. "Vision-based traffic sign detection and analysis for intelligent driver assistance systems:

Perspectives and survey". *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497, 2012.

[135] Randall Munroe. Xkcd sketch-style drawing mode, 2021. `http://xkcd.com/`.

[136] Willie Bosseau Murray and Patrick Anthony Foster. "The peripheral pulse wave: information overlooked". *Journal of clinical monitoring*, 12(5):365–377, 1996.

[137] Junichi Nakamura. *Image sensors and signal processing for digital still cameras*. CRC Press, 2006.

[138] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. "Intriguing properties of vision transformers". *arXiv preprint 2105.10497*, 2021.

[139] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Fahad Shahbaz Khan, and Fatih Porikli. "On improving adversarial transferability of vision transformers". *arXiv preprint 2106.04169*, 2021.

[140] Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, and Yuval Elovici. "Phantom of the ADAS: Securing advanced driver-assistance systems from split-second phantom attacks". In *ACM SIGSAC Conference on Computer and Communications Security*, page 293–308, New York, NY, USA, 2020. Association for Computing Machinery.

[141] Dudi Nassi, Raz Ben-Netanel, Yuval Elovici, and Ben Nassi. "MobilBye: Attacking ADAS with camera spoofing". *arXiv preprint 1906.09765*, 2019.

[142] Thanh Trung Ngo, Yasushi Makihara, Hajime Nagahara, Yasuhiro Mukaigawa, and Yasushi Yagi. "The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication". *Pattern Recognition*, 47(1):228–237, 2014.

[143] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. "Adversarial Robustness Toolbox v1.2.0". *arXiv preprint 1807.01069*, 2018.

[144] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1520–1528, 2015.

[145] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, J.K. Aggarwal, Hyungtae Lee, Larry Davis, et al. "A large-scale benchmark dataset for event recognition in surveillance video". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3153–3160. IEEE, 2011.

[146] BBC Online. Met police to deploy facial recognition cameras. https://bbc.co.uk/news/uk-51237665, 2020. [Online] Accessed 2022-02-01.

[147] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples". *arXiv preprint 1605.07277*, 2016.

[148] Nicolas Papernot, Patrick Mcdaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. "The limitations of deep learning in adversarial settings". In *IEEE European Symposium on Security and Privacy*, pages 372–387, 2016.

[149] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. "SoK: Security and privacy in machine learning". In *IEEE European Symposium on Security and Privacy*, pages 399–414. IEEE, 2018.

[150] Youngseok Park, Yunmok Son, Hocheol Shin, Dohyun Kim, and Yongdae Kim. "This ain't your dose: Sensor spoofing attack on medical infusion pump". In *USENIX Workshop on Offensive Technologies*, 2016.

[151] O. M. Parkhi, A. Vedaldi, and A. Zisserman. "Deep face recognition". In *British Machine Vision Conference*, 2015.

[152] Hanchuan Peng, Fuhui Long, and Chris Ding. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[153] Tania Pereira, Nate Tran, Kais Gadhoumi, Michele M. Pelter, Duc H. Do, Randall J. Lee, Rene Colorado, Karl Meisel, and Xiao Hu. "Photoplethysmography based atrial fibrillation detection: a review". *npj Digital Medicine*, 3, 2020.

[154] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. "Remote attacks on automated vehicles sensors: Experiments on camera and LiDAR". *Black Hat Europe*, 11:2015, 2015.

[155] Fabien Petitcolas. La cryptographie militaire. `https://petitcolas.net/kerckhoffs/crypto_militaire_1.pdf`, 1883. [Online] Accessed 2022-02-01.

[156] Jonah Philion. "Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11582–11591, 2019.

[157] Projectorcentral.com. Panasonic pt-ae8000 projector. `http://projectorcentral.com/Panasonic-PT-AE8000.htm`. [Online] Accessed 2020-10-12.

[158] Projectorcentral.com. Sanyo plc-xu4000 projector. `http://projectorcentral.com/Sanyo-PLC-XU4000.htm`. [Online] Accessed 2020-10-12.

[159] PyTorch. PyTorch multi-head attention implementation. `https://pytorch.org/docs/1.10.0/generated/torch.nn.MultiheadAttention.html`, 2021. [Online] Accessed 2022-02-01.

[160] QImaging. Rolling shutter vs. global shutter. `http://tecnicaenlaboratorios.com/Qimaging/brochures/RollingvsGlobalShutter.pdf`. [Online] Accessed 2022-02-01.

[161] Jonas Rauber, Wieland Brendel, and Matthias Bethge. "Foolbox: A python toolbox to benchmark the robustness of machine learning models". In *Reliable Machine Learning in the Wild Workshop, ICML*, 2017.

[162] Joseph Redmon and Ali Farhadi. "YOLOv3: An incremental improvement". *arXiv preprint 1804.02767*, 2018.

[163] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks". In *Conference on Neural Information Processing Systems*, pages 91–99, 2015.

[164] Kenneth Revett, Hamid Jahankhani, Sergio Tenreiro De Magalhaes, and Henrique M.D. Santos. "A survey of user authentication based on mouse dynamics". In *International Conference on Global e-Security*, pages 210–219. Springer, 2008.

[165] Ring. Ring camera. `https://ring.com`, 2020. [Online] Accessed 2022-02-01.

[166] Rollei. Rollei compact traveler no. 1 carbon tripod orange. `https://rollei.com/rollei-compact-traveler-no-1-carbon-tripod-orange-22579-en-eur`. [Online] Accessed 2022-02-01.

[167] Brian C. Ross. "Mutual information between discrete and continuous data sets". *PLoS ONE*, 9(2):e87357, 2014.

[168] Peter J. Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[169] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge". *International journal of computer vision*, 115(3):211–252, 2015.

[170] Hiroaki Sakoe and Seibi Chiba. "Dynamic programming algorithm optimization for spoken word recognition". *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[171] Jorge Sancho, Álvaro Alesanco, and José García. "Biometric authentication using the PPG: a long-term feasibility study". *Sensors*, 18(5):1525, 2018.

[172] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[173] Abhijit Sarkar, A Lynn Abbott, and Zachary Doerzaph. "Biometric authentication using photoplethysmography signals". In *Conference on Biometrics Theory, Applications and Systems*, pages 1–7. IEEE, 2016.

[174] Athena Sayles, Ashish Hooda, Mohit Gupta, Rahul Chatterjee, and Earlence Fernandes. "Invisible perturbations: Physical adversarial examples exploiting the rolling shutter effect". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14666–14675, 2021.

[175] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[176] Robert Mark Seepers, Wenjin Wang, Gerard De Haan, Ioannis Sourdis, and Christos Strydis. "Attacks on heartbeat-based security using remote photoplethysmography". *IEEE journal of biomedical and health informatics*, 22(3):714–721, 2017.

[177] Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. "On the adversarial robustness of visual transformers". *arXiv preprint 2103.15670*, 2021.

[178] Mahmood Sharif, Lujo Bauer, and Michael K. Reiter. "On the suitability of lp-norms for creating and preventing adversarial examples". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1605–1613, 2018.

[179] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition". In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540, 2016.

[180] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. "Adversarial generative nets: Neural network attacks on state-of-the-art face recognition". *arXiv preprint 1801.00349*, 2017.

[181] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. "CNN features off-the-shelf: an astounding baseline for recognition". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 806–813, 2014.

[182] Sam Shead. Chinese residents worry about rise of facial recognition. https://bbc.co.uk/news/technology-50674909, 2019. [Online] Accessed 2022-02-01.

[183] K. Shelley and S. Shelley. *Pulse Oximeter Waveform: Photoelectric Plethysmography.* Clinical Monitoring. Saunders Company, 2001.

[184] Meng Shen, Zelin Liao, Liehuang Zhu, Ke Xu, and Xiaojiang Du. "VLA: A practical visible light-based attack on face recognition systems in physical world". *ACM Conference on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3):1–19, 2019.

[185] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications". In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 445–467. Springer, 2017.

[186] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". *arXiv preprint 1409.1556*, 2014.

[187] Siri Team. Personalized Hey Siri, 2018. [Online] Accessed 2019-05-01.

[188] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno. "Physical adversarial examples for object detectors". In *USENIX Workshop on Offensive Technologies*. USENIX Association, 2018.

[189] Petros Spachos, Jiexin Gao, and Dimitrios Hatzinakos. "Feasibility study of photoplethysmographic signals for biometric identification". In *International Conference on Digital Signal Processing*, pages 1–5. IEEE, 2011.

[190] Nick Statt. Amazon is expanding its cashierless go model into a full-blown grocery store. `https://theverge.com/2020/2/25/21151021/amazon-go-grocery-store-expansion-open-seattle-cashier-less`, 2020. [Online] Accessed 2022-02-01.

[191] David Stone. Spotlight on lumens: How they're measured, and why they're not all the same. `http://projectorcentral.com/Lumens-Explained.htm`. [Online] Accessed 2020-02-20.

[192] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. "Segmenter: Transformer for semantic segmentation". *arXiv preprint 2105.05633*, 2021.

[193] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. "Light commands: Laser-based audio injection attacks on voice-controllable systems". *arXiv preprint 2006.11946*, 2020.

[194] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. "Videobert: A joint model for video and language representation learning". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7473, 2019.

[195] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. "Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures". *arXiv preprint 2006.16974*, 2020.

[196] Bosch Security Systems. Fw 6.30 tamper detection. https://resources-boschsecurity-cdn.azureedge.net/public/documents/TN_VCA_tamper_detect_WhitePaper_enUS_22996235531.pdf, 2016. [Online] Accessed 2022-02-01.

[197] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning". In *AAAI Conference on Artificial Intelligence*, 2017.

[198] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Explaining and harnessing adversarial examples". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[199] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[200] Chihiro Takano and Yuji Ohta. "Heart rate measurement based on a time-lapse image". *Medical engineering & physics*, 29(8):853–857, 2007.

[201] Michael E Tipping and Christopher M Bishop. "Probabilistic principal component analysis". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[202] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. "Training data-efficient image transformers; distillation through attention". In *International Conference on Machine Learning*, volume 139, pages 10347–10357, 2021.

[203] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. "On adaptive attacks to adversarial example defenses". In *Conference on Neural Information Processing Systems*, 2020.

[204] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. "The space of transferable adversarial examples". *arXiv preprint 1704.03453*, 2017.

[205] Florian Tramèr. Measuring and enhancing the security of machine learning. https://purl.stanford.edu/yz747qq9787, 2021. PhD Thesis, [Online] Accessed 2021-11-16.

[206] Matthew Turk and Alex Pentland. "Eigenfaces for recognition". *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[207] Henry Turner, Giulio Lovisotto, and Ivan Martinovic. "Attacking speaker recognition systems with phoneme morphing". In *European Symposium on Research in Computer Security*, pages 471–492. Springer International Publishing, 2019.

[208] Henry Turner, Giulio Lovisotto, and Ivan Martinovic. "Generating identities with mixture models for speaker anonymization". *Computer Speech & Language*, 72:101318, 2022.

[209] U.S. Department of Transportation. Manual of uniform traffic control devices for street and highways. http://mutcd.fhwa.dot.gov/pdfs/2009r1r2/mutcd2009r1r2edition.pdf. [Online] Accessed 2022-02-01.

[210] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE.". *Journal of machine learning research*, 9(11), 2008.

[211] Adam Van Etten. "You only look twice: Rapid multi-scale object detection in satellite imagery". *arXiv preprint 1805.09512*, 2018.

[212] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In *Conference on Neural Information Processing Systems*, pages 5998–6008, 2017.

[213] VentureBeat. Tesla AI chief explains why self-driving cars don't need LiDAR. `https://venturebeat.com/2021/07/03/tesla-ai-chief-explains-why-self-driving-cars-dont-need-lidar/`, 2021. [Online] Accessed 2022-01-01.

[214] W3. Jpeg file interchange format. `http://w3.org/Graphics/JPEG/jfif3.pdf`. [Online] Accessed 2022-02-01.

[215] Haohan Wang, Xindi Wu, Zeyi Huang, and Eric P. Xing. "High-frequency component helps explain the generalization of convolutional neural networks". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8684–8694, 2020.

[216] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. "Understanding convolution for semantic segmentation". In *IEEE Winter Conference on Applications of Computer Vision*, pages 1451–1460. IEEE, 2018.

[217] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. "Linformer: Self-attention with linear complexity". *arXiv preprint 2006.04768*, 2020.

[218] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. "Sceneformer: Indoor scene generation with transformers". *arXiv preprint 2012.09793*, 2020.

[219] Zhendong Wang, Xiaodong Cun, Jianmin Bao, and Jianzhuang Liu. "Uformer: A general u-shaped transformer for image restoration". *arXiv preprint 2106.03106*, 2021.

[220] Ze Wang, Weiqiang Ren, and Qiang Qiu. "Lanenet: Real-time lane detection networks for autonomous driving". *arXiv preprint 1807.01726*, 2018.

[221] Martin Wany and Georg P. Israel. "CMOS image sensor with NMOS-only global shutter and enhanced responsivity". *IEEE Transactions on electron devices*, 50(1):57–62, 2003.

[222] Zhipeng Wei, Jingjing Chen, Micah Goldblum, Zuxuan Wu, Tom Goldstein, and Yu-Gang Jiang. "Towards transferable adversarial attacks on vision transformers". *arXiv preprint 2109.04176*, 2021.

[223] Ross Wightman. Pytorch image models. `https://github.com/rwightman/`, 2019. [Online] Accessed 2022-02-01.

[224] Wikipedia. Daylight. `https://en.wikipedia.org/wiki/Daylight`. [Online] Accessed 2020-02-20.

[225] Eric Wong, Leslie Rice, and J. Zico Kolter. "Fast is better than free: Revisiting adversarial training". *arXiv preprint 2001.03994*, 2020.

[226] Lei Wu, Zhanxing Zhu, Cheng Tai, and Weinan E. "Understanding and enhancing the transferability of adversarial examples". *arXiv preprint 1802.09707*, 2018.

[227] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. "Mitigating adversarial effects through randomization". *arXiv preprint 1711.01991*, 2017.

[228] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. "Improving transferability of adversarial examples with input diversity". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2725–2734, 2019.

[229] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. "Empirical evaluation of rectified activations in convolutional network". *arXiv preprint 1505.00853*, 2015.

[230] Yi Xu, True Price, Jan-Michael Frahm, and Fabian Monrose. "Virtual U: Defeating face liveness detection by building virtual models from your public photos". In *USENIX Security Symposium*, 2016.

[231] Chen Yan, Wenyuan Xu, and Jianhao Liu. "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle". *DEF CON*, 24(8):109, 2016.

[232] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. "Learning texture transformer network for image super-resolution". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5800, 2020.

[233] Tsun-Yi Yang, Yi-Hsuan Huang, Yen-Yu Lin, Pi-Cheng Hsiu, and Yung-Yu Chuang. "SSR-Net: A compact soft stagewise regression network for age estimation.". In *International Joint Conference on Artificial Intelligence*, pages 1078–1084, 2018.

[234] Jianchu Yao, Xiaodong Sun, and Yongbo Wan. "A pilot study on using derivatives of photoplethysmographic signals as a biometric identifier". In *International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4576–4579. IEEE, 2007.

[235] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. "Learning face representation from scratch". *arXiv preprint 1411.7923*, 2014.

[236] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. "BDD100k: A diverse driving dataset for heterogeneous multitask learning". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020.

[237] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. "Dilated residual networks". In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 472–480, 2017.

[238] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. "Joint face detection and alignment using multitask cascaded convolutional networks". *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[239] Tianming Zhao, Yan Wang, Jian Liu, Yingying Chen, Jerry Cheng, and Jiadi Yu. "TrueHeart: Continuous Authentication on Wrist-worn Wearables Using PPG-based Biometrics". In *IEEE International Conference on Computer Communications*, 2020.

[240] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. "Seeing isn't believing: Towards more robust adversarial attack against real world object detectors". In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1989–2004, 2019.

[241] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, and Philip H.S. Torr. "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers".

In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021.

[242] Tianhang Zheng, Changyou Chen, and Kui Ren. "Distributionally adversarial attack". In *AAAI Conference on Artificial Intelligence*, volume 33, pages 2253–2260, 2019.

[243] Zhimin Zhou, Bedabrata Pain, and Eric R. Fossum. "Frame-transfer CMOS active pixel sensor with pixel binning". *IEEE Transactions on Electron Devices*, 1997.

[244] Zhe Zhou, Di Tang, Xiaofeng Wang, Weili Han, Xiangyu Liu, and Kehuan Zhang. "Invisible mask: Practical attacks on face recognition with infrared". *arXiv preprint 1803.04683*, 2018.

[245] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. "Deformable DETR: Deformable transformers for end-to-end object detection". *arXiv preprint 2010.04159*, 2020.