

Prevalence of Adversarial Examples in Neural Networks: Attacks, Defenses, and Opportunities

Utku Özbulak

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors

Prof. Wesley De Neve, PhD* - Prof. Arnout Van Messem, PhD**

* Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

** Department of Applied Mathematics, Computer Science and Statistics
Faculty of Sciences, Ghent University

January 2022



Prevalence of Adversarial Examples in Neural Networks: Attacks, Defenses, and Opportunities

Utku Özbulak

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors

Prof. Wesley De Neve, PhD* - Prof. Arnout Van Messem, PhD**

* Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

** Department of Applied Mathematics, Computer Science and Statistics
Faculty of Sciences, Ghent University

January 2022



ISBN 978-94-6355-560-9

NUR 984

Wettelijk depot: D/2022/10.500/1

Members of the Examination Board

Chair

Prof. Filip De Turck, PhD, Ghent University

Other members entitled to vote

Prof. Joni Dambre, PhD, Ghent University

Prof. Bart Goossens, PhD, Ghent University

Prof. Adam Prugel-Bennett, PhD, University of Southampton, United Kingdom

Prof. Yvan Saeys, PhD, Ghent University

Prof. Steven Verstockett, PhD, Ghent University

Supervisors

Prof. Wesley De Neve, PhD, Ghent University

Prof. Arnout Van Messem, PhD, Ghent University

Acknowledgments

I would like to thank Professor De Neve and Professor Van Messem for giving me the chance to pursue this doctoral research under their supervision. For this, they have my deepest gratitude.

Only recently I have started to comprehend how this journey of pursuing a doctoral degree changed me as a person. It molded me, humbled me, and sometimes it put me in my place. The journey was fun, until it wasn't, then, it was miserable. Cheers to all who helped me even the slightest. Thanks to you and your kind act, I found the power in me to move forward.

Finally, I would like to thank the two special women who deserve recognition for all of this. Thank you mother, for giving me strength to start this journey. And thank you my wife, for making me see it through.

Ghent, January 2022
Utku Ozbulak

Samenvatting

Automatisering, het proces waarbij de hoeveelheid menselijk arbeid wordt geminimaliseerd voor de productie en de aflevering van goederen, is sinds de industriële revolutie een belangrijk onderdeel geworden van onze samenleving. Inderdaad, dankzij geautomatiseerde systemen die toelaten om snel en goedkoop goederen te produceren en af te leveren, is de welvaart van de mensheid er gedurende de vorige eeuw sterk op vooruit gegaan. Bovendien is er, wat betreft automatisering, de afgelopen jaren nog een stap verder gezet met behulp van gecomputeerde besluitvormingssystemen die interageren met mensen. Dergelijke systemen worden vaak ingezet om ingewikkelde problemen op te lossen met billijke beslissingen, en waarbij deze beslissingen mensen vaak rechtstreeks raken. Een recente trend bij het oplossen van dagdagelijkse problemen van mensen is het gebruik van technieken voor machinaal leren, met name neurale netwerken. Systemen die neurale netwerken integreren, vaak aangeduid als artificieel intelligente systemen (AI-systemen), maken het mogelijk om nauwkeurige oplossingen te vinden voor ingewikkelde problemen.

Alhoewel neurale netwerken, door gebruik te maken van grote hoeveelheden data, het mogelijk maken om ingewikkelde problemen op te lossen, werd onlangs ontdekt dat ze helaas een ernstige beveiligingsfout bevatten: ze zijn namelijk kwetsbaar voor *vijandige voorbeelden* (Eng. adversarial examples). In deze context verwijst de term “vijandige voorbeelden” naar invoer die is aangemaakt met een kwaadwillige intentie, met als doel geautomatiseerde besluitvormingssystemen te misleiden en te bedriegen.

Vijandige voorbeelden worden algemeen erkend als één van de grootste tekortkomingen van neurale netwerken op het vlak van veilig-

heid, gegeven dat het een uitdaging is om ware invoer te onderscheiden van deze vijandige voorbeelden, vooral in de context van afbeeldingen. De verstoring die een waar voorbeeld omzet in een vijandig voorbeeld staat bekend als een *vijandige verstoring* (Eng. adversarial perturbation). Deze vijandige verstoringen worden voortgebracht met zogenaamde *vijandige aanvallen* (Eng. adversarial attacks), gericht op het creëren van chaos of het uitbuiten van geautomatiseerde besluitvormingssystemen, doorgaans in het voordeel van een tegenstander.

In dit doctoraal proefschrift hebben we het fenomeen van vijandige voorbeelden in de context van neurale netwerken in meer detail onderzocht, met als doel ons begrip van deze kwaadaardige datapunten te verbreden. Door nieuwe inzichten te verwerven in vijandige voorbeelden, en de bijbehorende vijandige aanvallen en verdedigingen, hopen we de ontwikkeling van meer veilige neurale netwerken te bevorderen. In dat verband heeft ons doctoraatsonderzoek aandacht besteed aan de volgende onderwerpen:

Eigenschappen van vijandige voorbeelden – Na de ontdekking van vijandige voorbeelden en hun nadelige impact op neurale netwerken, richtten veel onderzoeksinspanningen zich op het vinden van meer diverse aanvallen om deze datapunten op een zorgvuldige manier te vervaardigen, waarbij deze studies vaak de vaardigheid aantonen van nieuwe aanvallen met doorgaans empirische resultaten. Om de kloof tussen het empirisch en het theoretisch begrip van vijandige aanvallen te overbruggen, en om een vergelijking tussen verschillende types aanvallen te vergemakkelijken, hebben we de eigenschappen onderzocht van verschillende verliesfuncties die vaak gebruikt worden door een aantal populaire vijandige aanvallen, evenals de kenmerken van de voortgebrachte vijandige verstoringen. Dankzij dit onderzoek hebben we vastgesteld dat het gebruik van verlies op basis van *kruis-entropie* (Eng. cross-entropy), de meest gebruikte verliesfunctie voor het aanmaken van vijandige voorbeelden, nadelig is wat betreft de beschikbare optimalisatieruimte voor de creatie van deze vijandige voorbeelden. Anderzijds hebben we ook vastgesteld dat het gebruik van verlies op basis van kruis-entropie gunstig is als het gaat om het aanmaken van vijandige voorbeelden met minder stappen (iteraties), in vergelijking met verliezen die gebaseerd zijn op de ruwe voorspellingswaarden voortgebracht door een neurale netwerk (*logits*). Verder hebben we vijandige verstoringen en hun impact onderzocht wanneer ze werden toegepast op verschillende gebieden in een afbeelding. Hierbij hebben we vastgesteld dat regionale aanvallen, die een afbeelding in bepaalde interessegebieden verstoren, het mogelijk maken om vijandige voor-

beelden te creëren met minder verstoringen wanneer die verstoringen worden gemeten met $\ell_{\{0,2,\infty\}}$ -normen. Bovendien hebben we aangetoond dat een groot deel van de vijandige voorbeelden die met dergelijke regionale verstoringen zijn aangemaakt, ook de overdraagbaarheid van hun vijandigheid van model naar model blijven behouden, dit in vergelijking met hun globaal verstoorde tegenhangers.

Aanvallen op modellen voor segmentatie van biomedische afbeeldingen – Vanwege zijn missiekritische aard wordt het vakgebied dat biomedische beeldanalyse bestudeerd aanzien als één van de domeinen dat het meest te lijden heeft onder de kwetsbaarheid van neurale netwerken voor vijandige voorbeelden. Daarom hebben we, afstand nemend van de veelgebruikte datasets waarvoor regelmatig vijandige aanvallen worden besproken, de kwetsbaarheid van beeldsegmentatiemodellen onderzocht voor vijandige voorbeelden. In deze context hebben we de gevolgen besproken van een door ons ontworpen vijandige aanval, namelijk de *Adaptive Segmentation Mask Attack* (ASMA), die het mogelijk maakt om tegenstrijdige voorbeelden aan te maken voor modellen die beeldsegmentatie beogen. In het bijzonder laat ASMA toe om twee soorten vijandige voorbeelden te genereren voor beeldsegmentatiemodellen: (1) vijandige voorbeelden die de nauwkeurigheid van de segmentatie minimaliseren en (2) vijandige voorbeelden met gefabriceerde voorspellingen die andere datapunten in de beschikbare dataset nabootsen. Met behulp van meerdere modellen voor semantische beeldsegmentatie en twee datasets die biomedische afbeeldingen bevatten, hebben we een gedetailleerd onderzoek uitgevoerd naar de weerbaarheid van de gebruikte segmentatiemodellen ten aanzien van vijandige voorbeelden. Hierbij hebben we vastgesteld dat deze segmentatiemodellen kwetsbaar zijn voor vijandige voorbeelden die met ASMA werden aangemaakt. Door de vijandige voorbeelden die zijn aangemaakt in de context van classificatie te vergelijken met de vijandige voorbeelden die zijn aangemaakt in de context van segmentatie, hebben we bovendien ook het volgende vastgesteld: vijandige voorbeelden die gericht zijn op het aanvallen van segmentatiemodellen worden gekarakteriseerd door enigszins unieke verstoringen voor elke afbeelding, terwijl vijandige voorbeelden die gericht zijn op het aanvallen van classificatiemodellen worden gekarakteriseerd door zout-en-peperruis. Ten slotte hebben we de weerbaarheid gemeten van de verstoringen die met ASMA werden aangemaakt ten aanzien van twee methoden die toelaten om de invoer aan te passen, en waarbij deze twee methoden erop gericht zijn om vijandige verstoringen teniet te doen: encoding met JPEG en Gaussiaans vervagen. In deze context

hebben we vastgesteld dat de twee methoden voor het verwijderen van ruis, alhoewel ze het mogelijk maken om het risico van vijandige voorbeelden gedeeltelijk op te heffen, niet in staat zijn om de vijandigheid in de invoer volledig weg te nemen.

Aanvallen op modellen voor het herkennen van activiteiten via radarbeelden – Het domein van natuurlijke afbeeldingen is niet het enige domein dat baat heeft bij de vooruitgang die geboekt wordt door het onderzoek naar neurale netwerken. Dankzij hun flexibele aard worden neurale netwerken ook meer en meer gebruikt door onderzoekers die werken met systemen die gebaseerd zijn op radar. Om de zwakheden te identificeren van besluitvormingssystemen die gebruikmaken van zowel neurale netwerken als radarbeelden voor het herkennen van menselijke activiteiten, hebben we de kwetsbaarheid geanalyseerd van de onderliggende modellen voor veelgebruikte vijandige aanvallen. Verder hebben we een nieuwe aanval ontworpen die het mogelijk maakt om voorspellingen te wijzigen door vijandige verstoringen enkel toe te passen op *opvulbeelden* (Eng. padding frames) in de invoer, in plaats van deze vijandige verstoringen toe te passen op de radarbeelden waar een zekere activiteit plaatsvindt. Ten slotte hebben we de relatie onderzocht tussen de interpreteerbaarheid van modellen en vijandige verstoringen. Hierbij hebben we vastgesteld dat beelden die belangrijk worden geacht door de Grad-CAM-techniek voor interpreteerbaarheid ook deze beelden zijn die meer worden verstoord door vijandige aanvallen, wat een verband aantoont tussen interpreteerbaarheid en vijandigheid.

Verdedigen tegen vijandige voorbeelden – Met behulp van de kennis die werd verkregen door onze eerdere onderzoeksinspanningen, hebben we een aantal methoden onderzocht om ons te verdedigen tegen vijandige aanvallen, en waarbij deze methoden het detecteren van vijandige voorbeelden beogen. We hebben in eerste instantie de verdedigingsmogelijkheden onderzocht van *vijandige training* (Eng. adversarial training), dat een neurale netwerk laat leren met zowel vijandige voorbeelden als ware datapunten, teneinde een onderscheid te kunnen maken tussen de twee. Met behulp van vijandige voorbeelden die zijn aangemaakt met verschillende verliesfuncties, hebben we vastgesteld dat vijandige training een effectieve methode is om vijandige voorbeelden te detecteren. Onze experimenten toonden echter ook aan dat vijandige training data-hongerig is en dat vijandige training alleen effectief is als een groot aantal vijandige voorbeelden wordt gebruikt, samen met de ware datapunten. Anders zijn de resultaten die worden verkregen door middel van vijandige training zeer onsta-

biel. Verder hebben we als verdediging tegen vijandige voorbeelden een eenvoudige maar effectieve methode ontwikkeld voor het afwijzen van vijandige invoer. In het bijzonder maakt deze nieuwe verdedigings-techniek het mogelijk om een welbepaald type vijandige voorbeelden te detecteren, namelijk die vijandige voorbeelden die gepaard gaan met hoge *logit*-waarden. Deze hoge *logit*-waarden worden in de praktijk gedetecteerd met behulp van *drempels* (Eng. thresholds), waarbij een drempel wordt bepaald voor elke categorie die in een dataset aanwezig is, en waarbij het aantal valse positieven wordt geminimaliseerd. Door middel van uitgebreide experimenten die gebruikmaken van drie datasets en vijf modellen, hebben we aangetoond dat de voorgestelde verdediging zowel flexibel als effectief is in het detecteren van het type vijandige voorbeelden dat hierboven is aangehaald.

Om af te sluiten heb ik een aantal reflecties geformuleerd over het onderzoek dat ik de afgelopen jaren op het vlak van vijandige voorbeelden heb uitgevoerd. In deze context kaart ik een aantal zorgen aan omtrent de integriteit van onderzoeksinspanningen naar vijandig machinaal leren. Verder bespreek ik eveneens een aantal pistes voor toekomstig onderzoek.

Summary

Automation, the process of minimizing human labor to produce and deliver goods, has been a crucial part of our society since the industrial revolution. Indeed, thanks to automated systems that allowed for rapid and cheap production and delivery of goods, human welfare has increased dramatically in the past century. In recent years, automation has been taken a step further with the help of automated decision-making systems that interact with humans. Such automated decision-making systems are employed to come up with fair decisions that are able to address complex problems, with these decisions often directly affecting humans. A recent trend in solving complex problems regarding everyday human life is the usage of machine learning techniques, specifically, neural networks. Systems that integrate neural networks, often labeled as artificial intelligence (AI) systems, facilitate the discovery of accurate solutions to complicated problems.

Although neural networks, by leveraging large amounts of data, make it possible to solve rather complex problems, it was recently discovered that they harbour a serious security flaw, namely their vulnerability to *adversarial examples*. In this respect, the term “adversarial examples” refers to inputs created with a malicious intent, with the goal of misleading and deceiving automated decision-making systems.

Given that it is challenging to distinguish genuine inputs from adversarial examples, especially in the context of images, adversarial examples are acknowledged to be a major security flaw of neural networks. Here, the perturbation that converts a genuine input to an adversarial example is typically referred to as *adversarial perturbation*. These adversarial perturbations are produced by so-called *adversarial attacks*, aiming at the creation of chaos or the exploitation of auto-

mated decision-making systems, usually to the benefit of an adversary.

In this dissertation, we investigated the phenomenon of adversarial examples in detail in the context of neural networks, with the goal of broadening our understanding of these malicious data points. Indeed, by obtaining new insights into adversarial examples, and corresponding adversarial attacks and defenses, our hope is to further the development of safer neural networks. In that regard, our doctoral research has paid attention to the following topics:

Properties of adversarial examples – After the discovery of adversarial examples and their adverse effects on neural networks, many studies focused on introducing more diverse attacks to carefully generate such data points, with these studies often demonstrating the proficiency of novel attacks through mostly empirical results. In order to bridge the gap between the empirical and theoretical understanding of adversarial attacks, and in order to facilitate comparability between different types of attacks, we investigated the properties of several objective functions used by a number of popular adversarial attacks, as well as the characteristics of the adversarial perturbation generated. Through this investigation, we found the cross-entropy loss, the most frequently used loss function for the creation of adversarial examples, to be detrimental when it comes to the available optimization space for creating adversarial examples. Conversely, we identified cross-entropy loss to be beneficial when it comes to creating adversarial examples with fewer iterations, compared to logit-based losses. Furthermore, we investigated adversarial perturbations and their impact when exercised on various image regions, finding that regional attacks, which only perturb an image in particular areas of interest, allow creating adversarial examples with less perturbation when those perturbations are measured with $\ell_{\{0,2,\infty\}}$ norms. Moreover, we demonstrated that a large portion of adversarial examples created with such regional perturbations also maintain their model-to-model adversarial transferability, again compared to their globally perturbed counterparts.

Attacks on models for biomedical image segmentation – Due to its mission-critical nature, the field of biomedical image analysis is recognized as one of the fields that suffers the most from the vulnerability of neural networks to adversarial examples. As such, moving away from the datasets that are commonly used for discussing adversarial attacks, we investigated the vulnerability of biomedical image segmentation models to adversarial examples. We discussed the repercussions of a newly introduced adversarial attack, namely the Adaptive Segmentation Mask Attack (ASMA), which allows creating

adversarial examples in the context of image segmentation models. Specifically, ASMA makes it possible to generate two types of adversarial examples for image segmentation models: (1) adversarial examples that minimize the segmentation accuracy and (2) adversarial examples with fabricated predictions that imitate other data points in the dataset available. With the help of multiple semantic image segmentation models and two biomedical image datasets, we performed a detailed investigation of the robustness of biomedical image segmentation models against adversarial examples, finding the segmentation models to be vulnerable to adversarial examples created with ASMA. Furthermore, comparing the adversarial examples created in the context of classification to the ones created in the context of segmentation, we discovered that adversarial examples targeting segmentation models come with somewhat unique perturbations for each image, whereas adversarial examples targeting classification models tend to come with salt-and-pepper type noise. We concluded our research on this topic with measuring the robustness of the perturbation created with ASMA against two input-modification methods: JPEG encoding and Gaussian blurring. Through this analysis, we found that, although such methods for noise removal are able to partially alleviate the risk induced by the adversarial perturbation, they cannot completely get rid of the adversariality present in the input.

Attacks on models for radar-based activity recognition –

The domain of natural images is not the only domain to benefit from advancements in the area of neural network research. Thanks to their flexible nature, neural networks also saw a soaring interest from researchers working with radar-based systems. In order to identify the weaknesses of radar-based decision-making systems that employ neural networks, we analyzed the vulnerability of models for radar-based human activity recognition to commonly used adversarial attacks, finding that adversarial examples created by such attacks are indeed effective against radar-based neural networks. Furthermore, we discussed a newly introduced attack that is particularly dangerous to the radar domain, making it possible to change predictions when adversarial perturbation is only exercised on the padding frames of the input, rather than the radar frames in which the activity of interest takes place. Lastly, we investigated the connection between model interpretability and adversarial perturbation, finding that radar frames that the Grad-CAM interpretability technique deems to be important are also those frames that are perturbed more by adversarial attacks, thus demonstrating a link between interpretability and adversariality.

Defending against adversarial examples—Leveraging the knowledge obtained through our previous research efforts, we investigated a number of methods to defend against adversarial attacks, with these methods taking the form of adversarial example detection. We first evaluated the defensive capabilities of adversarial training, which trains a neural network with adversarial examples in conjunction with genuine data points in order to make a distinction between the two. With the help of adversarial examples created through the use of various loss functions, we found adversarial training to be an effective method to detect adversarial examples. However, our experiments also indicated that adversarial training is data hungry and that adversarial training is only effective if a large number of adversarial examples are provided together with the genuine images. Otherwise, we found that the results obtained through adversarial training can be highly unstable. Furthermore, in the context of adversarial defenses, we developed a simple yet effective input rejection method that allows detecting adversarial examples that come with large logit predictions. In particular, with the help of this defense, it is possible to separate genuine images from adversarial examples by only making use of logit predictions and thresholds, and where a threshold is determined for each category in a dataset, hereby minimizing the number of false positives. Through extensive experimentation with three datasets and five models, we demonstrated that the proposed defense is flexible and effective in detecting the aforementioned type of adversarial examples.

Finally, I reflected on the research I conducted in this field for the past years, raising a number of concerns regarding the integrity of adversarial machine learning research, as well as providing several directions for future research.

List of Abbreviations

Machine Learning

AdaGrad	Adaptive gradient algorithm
Adam	Adaptive momentum
AI	Artificial intelligence
CAM	Class activation mapping
CE	Cross-entropy
CNN	Convolutional neural network
CV	Computer vision
DNN	Deep neural network
ELU	Exponential linear unit
GD	Gradient descent
Grad-CAM	Gradient-weighted class activation mapping
IOU	Intersection over union
LSTM	Long short-term memory
ML	Machine learning
Nadam	Adam with Nesterov momentum
NLL	Negative logarithmic loss
NN	Neural network
ReLU	Rectifier linear unit
RmsProp	Root mean square propagation
RNN	Recurrent neural network
SGD	Stochastic gradient descent
SVM	Support vector machine

Adversarial Attacks

ASMA	Adaptive segmentation mask attack
BPDA	Backward pass differentiable approximation
BIM	Basic iterative method
CW	Carlini & Wagner's attack
DAG	Dense adversary generation
FGS	Fast-gradient sign
IFGS	Iterative fast-gradient sign
LAVAN	Localized and visible adversarial noise
PGD	Projected gradient descent

Miscellaneous

GPU	Graphics processing unit
PNG	Portable graphics format
CCTV	Closed-circuit television
Radar	Radio detection and ranging
LOS	Line of sight
FMCW	Frequency-modulated continuous-wave
LIDAR	Laser imaging, detection, and ranging
RD	Range-doppler
MD	Micro-doppler
ILSVRC	The ImageNet Large-scale Visual Recognition Challenge
CIFAR	Canadian Institute For Advanced Research
MNIST	Modified National Institute of Standards and Technology

Mathematical Notation

Numbers, Arrays, and Indexing

a	A scalar
\mathbf{x}	A vector
\mathbf{X}	A matrix
\mathbf{X}	A tensor
x_i	i th element of the vector \mathbf{x} , with indexing starting at 1
$X_{i,j}$	Element at i th row and j th column of the matrix \mathbf{X}
$\hat{\mathbf{x}}^{(k)}$	$\hat{\mathbf{x}}$ obtained after the k th iteration of an iterative operation

Sets

\mathbb{A}	A set
$\{a, b\}$	A set containing the two elements a and b
$\{1, \dots, m\}$	A set containing all integers from 1 to m
$\{1, \dots, m\} \setminus \{c\}$	A set containing all integers from 1 to m , excluding c
$[a, b]$	A real interval including both a and b
(a, b)	A real interval including a and excluding b
$\mathbb{A} \setminus \mathbb{B}$	Set subtraction, with the resulting set containing the elements of \mathbb{A} that are not in \mathbb{B}

Functions and Operations

$f : \mathbb{R} \rightarrow \mathbb{N}$	The function f with domain \mathbb{R} and range \mathbb{N}
$\log(x)$	Natural logarithm of x
$\max(\mathbf{x})$	The largest element of \mathbf{x}
$\arg \max(\mathbf{x})$	Index of the largest element of \mathbf{x}
$\mathbb{1}_{\text{condition}}$	Is 1 if the condition is satisfied, 0 otherwise
$\ \mathbf{x}\ _p$	ℓ_p norm of \mathbf{x}
$f(\mathbf{x}, \theta)$	A function of \mathbf{x} , parameterized by θ
\mathbf{X}^\top	Transpose of \mathbf{X}
$\mathbf{X} \odot \mathbf{Y}$	Element-wise (Hadamard) product of \mathbf{X} and \mathbf{Y}
$\det(\mathbf{X})$	Determinant of \mathbf{X}
$\nabla_x y$	Gradient of y with respect to \mathbf{x}

Table of Contents

1	Introduction	1
1.1	Human vision	2
1.2	Computer vision	4
1.3	Limitations of human vision	7
1.4	Limitations of computer vision	11
1.5	Focus of this dissertation	15
1.6	Organization of this dissertation	17
1.7	Contributions	19
2	Learning from data	23
2.1	Neural networks	25
2.1.1	Overview of neural networks	28
2.1.2	Trainable weights	31
2.1.3	Activation functions	36
2.1.4	Loss functions	39
2.1.5	Optimizers	41
2.2	Training and error quantification	42
2.3	Machine learning notation	45
3	Adversarial attacks on deep neural networks	47
3.1	Introduction	48
3.2	Threat scenarios	49
3.3	Adversarial examples	52
3.3.1	L-BFGS attack	53
3.3.2	Fast-gradient sign	54
3.3.3	Iterative fast-gradient sign	54
3.3.4	Projected gradient descent	54

3.3.5	Carlini & Wagner's attack	55
3.3.6	Other adversarial attacks	56
3.4	Why do adversarial examples exist?	58
3.5	Perturbation measurement	59
3.6	Box constraints	60
3.7	Adversarial transferability	60
3.8	Investigating the loss functions of adversarial attacks . .	61
3.8.1	Generalizing loss functions	61
3.8.2	Cross-entropy loss	64
3.8.3	Cross-entropy sign loss	68
3.8.4	Experiments	69
3.8.4.1	Use of a controllable 2-D environment .	69
3.8.4.2	Use of ImageNet	72
3.9	Regional adversarial perturbation	78
3.10	Global to regional conversion	79
3.10.1	Framework	80
3.10.2	Experimental setup	80
3.10.3	Experimental results	82
3.11	Conclusions and future work	84
4	Adversarial attacks on biomedical image segmentation models	89
4.1	Introduction	90
4.2	Adaptive segmentation mask attack	92
4.2.1	Fabricating a new segmentation object	95
4.2.2	Reducing the segmentation accuracy	96
4.3	Data and deep learning models	97
4.4	Experiments	100
4.4.1	Quantifying the threat of adversarial attacks . .	100
4.4.2	Robustness of the generated perturbations	105
4.5	Discussion	107
4.6	Conclusions and future work	109
5	Adversarial attacks on radar-based activity recognition systems	111
5.1	Introduction	112
5.2	Framework	113
5.2.1	Data	113
5.2.2	Models	115
5.3	Threat model	120
5.4	Threat model evaluation	124

- 5.4.1 Evaluating common adversarial attacks 125
 - 5.4.2 Adversarial padding for radar data 128
- 5.5 Relation of adversarial attacks to interpretability 133
- 5.6 Recent developments 140
- 5.7 Conclusions and future work 141
- 6 Detecting adversarial examples 143**
 - 6.1 Introduction 144
 - 6.2 Adversarial retraining 147
 - 6.2.1 Experimental results 149
 - 6.3 Impact of adversarial optimization on logits 152
 - 6.4 Softmax for adversarial examples 153
 - 6.5 Identifying adversarial examples with large logits 161
 - 6.5.1 Logit distributions 161
 - 6.5.2 Determining the threshold 163
 - 6.5.3 Experiments 164
 - 6.5.4 Benefits of the proposed defense 166
 - 6.5.5 Limitations of the proposed defense 168
 - 6.5.6 Similar logit-based defenses 169
 - 6.6 Limitations of logit-based defenses 171
 - 6.7 Conclusions and future work 174
- 7 Conclusions and outlook 175**
 - 7.1 Summary 175
 - 7.2 Reflections on adversarial machine learning research . . 177
 - 7.2.1 Representation of adversarial examples 178
 - 7.2.2 Research reproducibility 181
 - 7.2.3 Shortcomings of proposed methods 183
 - 7.3 Opportunities and future perspectives 185
 - 7.4 Epilogue 190
- Bibliography 193**

List of Figures

1.1	A simplified illustration of the human visual system [112].	3
1.2	Illustration of the (left) RGB and (right) CMYK color spaces [90].	4
1.3	Use of a numerical method for image representation, with this numerical method taking the form of a matrix [71].	5
1.4	Electromagnetic spectrum, ranging from gamma waves to long radio waves, including the spectrum visible to the human eye [2].	7
1.5	The Zöllner illusion consists of parallel lines that, when injected with short lines at an acute angle, appear to be divergent.	8
1.6	(left) The Müller-Lyer illusion and (right) a variant of the Müller-Lyer illusion. In both illusions, the horizontal lines have the same length, but the bottom ones appear to be shorter than the top ones.	8
1.7	(left) The Hermann grid, which seems to have grey patches at the intersections of white lines and (right) the corrected Hermann grid, no longer depicting an illusion.	9
1.8	(top) The rotating snakes illusion, evoking illusory motion in the periphery of the visual field, and (bottom) a gray-scaled version of the same illusion.	10
1.9	(left) Picture of a cat and (right) picture of a cat with bounding boxes that allow for localization.	11

1.10 (left) A picture of an apple, (center) a picture of an iPod, and (right) a picture of an apple covered by the text *iPod*. The left picture is correctly classified as an apple by a machine learning model, whereas the right picture is misclassified as an iPod. 12

1.11 A genuine image, initially classified as *cancer* with 0.95 confidence by a deep learning model, is perturbed to become an adversarial example. This adversarial example is then classified by the same model as *healthy* with 0.99 confidence. 13

1.12 A genuine image and its road segmentation are given. With the addition of adversarial perturbation to this genuine image, the predictions obtained for its adversarial example counterpart result in a completely different road segmentation, even no longer having pedestrians. . 14

1.13 Overview of the chapter organization. 16

2.1 (left) An example classification task and (right) an example segmentation task. Labels are provided below the pictures shown. 24

2.2 A visual representation of synapses and neurons in (left) the human brain and (right) neural networks. 26

2.3 Outcome (green for 1 and red for 0) for (left) the OR and (right) XOR operation. The obtained outcomes show that the OR operation is linearly separable, whereas the XOR operation is not. 27

2.4 A visual illustration of a neural network with a single hidden layer. For visual clarity, the bias terms are excluded from this illustration. 29

2.5 Illustration of the input-output relationship for (top) classification and (bottom) segmentation models. 30

2.6 A visual representation of a fully connected layer. 31

2.7 An illustration of a convolutional operation using a single 3×3 kernel, as performed in convolutional layers. . . 33

2.8 A visual representation of feature extraction through the usage of multiple convolutional kernels. 33

2.9 Visualization of a residual layer with a skip-connection. 34

2.10 A recurrent layer is visualized as a directed graph, including its hidden states and the inputs provided at each step. 35

2.11	(top) Activation functions and (bottom) their first order derivatives.	37
2.12	Illustrations of (top) an underfit, (middle) an overfit, and (bottom) a good fit (i.e., a robust model).	43
2.13	Illustration of early stopping with respect to the training duration (i.e., epochs) and the errors obtained for training and testing data.	44
3.1	Threat configurations and their taxonomy for evasion adversarial attacks.	50
3.2	A genuine image and its adversarial counterparts are provided for each threat taxonomy.	51
3.3	Illustration of two unperturbed images and their adversarial counterparts. The $\ell_{\{2,\infty\}}$ norms of the perturbations are provided below each adversarial example.	57
3.4	The prediction likelihood of both the initial class and the target class is displayed throughout 100 iterations of adversarial optimization. The lines represent the mean and the shaded areas represent the 95% confidence interval of the likelihood values that have been obtained for 1,000 adversarial examples. The subspaces defined in Definition 3.8.1 are highlighted as blue, green, and red areas, representing D_1 , D_2 , and D_3 , respectively.	63
3.5	A classification problem consisting of two circular distributions with the same center $(0, 0)$ but different radii. The data are bounded by $(x, y) \in [-1, 1]^2$. An adversarial trajectory is defined as moving a data point from the outer class to the inner class across the decision boundary.	70
3.6	Heat maps characterizing the adversarial optimization are given for the classification problem presented in Figure 3.5, generated using (a) the CE loss, (b) the logit loss, and (c) the CE-sign loss.	71

3.7 (a) The softmax output and (b) the logit prediction for both the initial class and the target class is given for adversarial optimization over 250 iterations, for the experiment “*equal multiplier*”. The values correspond to the mean value of 1,000 samples, taken from the ImageNet validation set. (c) The least amount of total added perturbation required to change the prediction of the data points under consideration (the same data points are presented in (a) and (b)). Best viewed in color. 74

3.8 (a) The softmax output and (b) the logit prediction for both the initial class and the target class is given for adversarial optimization over 250 iterations, for the experiment “*equal perturbation*”. The values correspond to the mean value of 1,000 samples, taken from the ImageNet validation set. (c) The least amount of total added perturbation required to change the prediction of the data points under consideration (the same data points are presented in (a) and (b)). Best viewed in color. 75

3.9 Change in gradient magnitude for the experiment “*equal multiplier*”, for the data points presented in Figure 3.7. . 76

3.10 Adversarial examples generated by regional attacks. . . 78

3.11 The localization masks used the experiments discussed in this chapter. The given percentages correspond to the number of selected pixels compared to the total number of available pixels. 81

3.12 Percentage of adversarial examples with localized perturbation that transfer from the source model (generated from) to the target model (tested against) when 17%, 28%, and 45% of the pixels are selected, respectively (combining all three localization approaches). . . 83

3.13 ℓ_0 , ℓ_2 , and ℓ_∞ distances between the initial images and their adversarial counterparts. The adversarial counterparts originate from the same initial image but were perturbed using different localization methods. All of the adversarial examples successfully transfer to models that they were not created with. 86

3.14 The percentage of adversarial examples with regional perturbation that have less perturbation in terms of ℓ_0 norm (top), ℓ_2 norm (middle), and ℓ_∞ norm (bottom) compared to their counterparts with “global” perturbation. Percentages are calculated based on the adversarial examples with localized perturbation that transfer from the source model to the target model. 87

4.1 (a) An input image for the segmentation model used and (b) its initial (i.e., correct) prediction. When Dense Adversary Generation (DAG) [219] is applied to (a), the prediction (i.e., the segmentation) becomes (c), with the shapes obtained giving away that the input has been tampered with. On the other hand, the proposed technique allows choosing the target mask, making it possible to convert the segmentation prediction to this mask, as shown in (d). 91

4.2 An example optimization of an adversarial example for attacking segmentation models, with the optimization being performed by ASMA. 95

4.3 The input image, the added perturbation compared to the initial image, the prediction for the input image, and the optimization mask used by ASMA are provided for fabricating a new segmentation object. Red areas in the optimization masks represent spaces that ASMA aims at converting to background (i.e., that need to be changed to black) and blue areas denote spaces that ASMA aims at converting to foreground (i.e., that need to be changed to white). 96

4.4 The input image, the added perturbation compared to the initial image, the prediction for the input image, and the optimization mask used by ASMA are provided for reducing the segmentation accuracy. Red areas in the optimization masks represent spaces that ASMA aims at converting to background (i.e., that need to be changed to black) and blue areas denote spaces that ASMA aims at converting to foreground (i.e., that need to be changed to white). 97

4.5 Visualization of the overall architecture of (top) Fcn8s, (middle) SegNet, and (bottom) UNet 99

4.6 Target model transferability and perturbation statistics for the experiment on fabricating new segmentation objects. The given numbers correspond to the mean (standard deviation) for (a1, a2) IoU accuracy between the target masks and the adversarial example predictions, (a3, a4) absolute reduction in the IoU accuracy, (b) ℓ_2 norm of the perturbation, and (c) ℓ_∞ norm of the perturbation. Diagonal entries in (a) represent attacks performed in white-box scenarios; the other entries represent cross-model black-box scenarios. 101

4.7 Target model transferability and perturbation statistics for the experiment on fabricating new segmentation objects and the experiment on reducing the segmentation accuracy. The given numbers correspond to the mean (standard deviation) for (a1, a2) IoU accuracy between the target masks and the adversarial example predictions, (a3, a4) absolute reduction in the IoU accuracy, (b) ℓ_2 norm of the perturbation, and (c) ℓ_∞ norm of the perturbation. Diagonal entries in (a) represent attacks performed in white-box scenarios; the other entries represent cross-model black-box scenarios. 102

4.8 A set of genuine images is provided, together with their adversarial counterparts and the corresponding segmentation predictions. Below each adversarial image, the ℓ_2 and ℓ_∞ norm of the perturbation are given, illustrating the degree of the required changes. Data points are taken from the glaucoma optic disc segmentation dataset and the segmentation predictions are taken from the UNet model. 103

4.9 A set of genuine images is provided, together with their adversarial counterparts and the corresponding segmentation predictions. Below each adversarial image, the ℓ_2 and ℓ_∞ norm of the perturbation are given, illustrating the degree of required changes. Data points are taken from the cell segmentation dataset and the segmentation predictions are taken from the UNet model. . . . 104

4.10 Input image and corresponding adversarial perturbations obtained from deep neural networks designed for (top) classification and (bottom) segmentation. Adversarial perturbations for classification are obtained from a ResNet-50 [75] model trained on ImageNet [175], using the LBFGS [201], IFGS [110], PGD [126], and CW [22] attacks, in that order. Using ASMA, perturbations for segmentation are obtained from Fcn8s, SegNet, UNet, and UNet, in that order. 106

4.11 Mean (standard deviation) absolute reduction in the IoU accuracy between the prediction and the initial (correct) mask for the same 1,000 adversarial examples after applying (a) low-pass Gaussian filtering and (b) JPEG encoding. The data represented here should be compared to table Figure 4.7(a3) and Figure 4.7(a4) for evaluating the effect of the defenses. 107

5.1 (left) Video frames and (right) RD frames for the action *swipe left*. The X -axis and Y -axis of the RD frames, which are omitted for visual clarity, correspond to range and velocity, respectively. 114

5.2 Detailed description for architecture \mathcal{A} 116

5.3 Detailed description for architecture \mathcal{R} 117

5.4 Detailed description for architecture \mathcal{L} 118

5.5 A visual summary of the flow of events in a household when an action is performed, including the entry points for possible adversarial attacks. 122

5.6 Threat configurations and their taxonomy for adversarial examples. Scenarios evaluated in this study are highlighted with their abbreviation. 123

5.7 (1) ℓ_2 distances, (2) original frames, and (3) perturbed frames are given in order to improve the visual understanding of various degrees of perturbation. 129

5.8 A sequence of adversarial padding frames (from frame 36 to 50), as generated by the proposed adversarial padding. The total amount of adversarial perturbation, as calculated for these frames, corresponds to an ℓ_2 distance of 4.75. 132

5.9 Visual representation of the frame replacement operation as explained in Section 5.5. 135

5.10 A boxplot representation of added perturbation, as generated by CW, displayed for individual frames of adversarial examples that transfer from A_{S_+} to A_{S_-} . The amount of added perturbation is plotted against the median frame importance, as calculated by the experiment detailed in Section 5.5. 135

5.11 Mean Grad-CAM magnitude (normalized) is plotted against the mean frame importance, as calculated by the experiment detailed in Section 5.5. 135

5.12 Visualization of a number of consecutive (1) RD frames, (2) video frames, and (3) Grad-CAM heatmaps for the action *swipe left*. The X-axis and Y-axis of the RD frames and the Grad-CAM images, which are omitted for visual clarity, correspond to range and velocity, respectively. 136

5.13 A boxplot representation of added perturbation, as generated by CW, displayed for individual frames of adversarial examples that transfer from A_{S_+} to A_{S_-} . Added perturbation is plotted against the median frame importance, as calculated by the experiment detailed in Section 5.5. 137

5.14 (blue) A line graph representation of Grad-CAM magnitudes (normalized between 0 and 1), displayed for individual frames that belong to genuine data points of adversarial examples that transfer from A_{S_+} to A_{S_-} . This graph is plotted against the median frame importance (red), as calculated by the experiment detailed in Section 5.5. 138

6.1 Genuine images taken from the ImageNet dataset and the added perturbation, illustrated in the form of saliency maps, with the adversarial attacks making use of CE, CE-sign, logit, and M-logit loss. 152

6.2 An illustration of adversarial examples $\alpha_{\{1,2,3\}}$, derived from an input α_0 , for an affine classifier $f(x) = 0$, and their respective distances $\Delta_{\{1,2,3\}} = d(\alpha_{\{1,2,3\}}, f)$ to the decision boundary f 153

- 6.3 (a) Original images, predicted as *apple* (top) and *turtle* (bottom). (b)-(c)-(d) Multi-class optimized adversarial examples that produce higher logit values, but that are predicted with lower confidence by ResNet-50. Logit_1 , Logit_2 , and Logit_3 are the logits of the first, second, and third most likely predictions, respectively. 156
- 6.4 (a) Original image, predicted as *arctic fox* with 0.99 confidence. (b)-(c)-(d) Over-optimized adversarial examples which are predicted with the same confidence but with vastly different logit values by ResNet-50. Logit_1 and Logit_2 represent the logits of the most likely and second most likely predictions, respectively. 157
- 6.5 A linearly separable two-class classification problem, highlighting the saturation of the softmax output, as opposed to the logit values, which keep increasing. . . . 159
- 6.6 Highest logit value and corresponding softmax output as a function of the number of iterations when generating adversarial examples with (a) IFGS and (b) CW. Adversarial examples are tested on ResNet-50 in a white-box setting. 160
- 6.7 Density plots of the logit values associated with the predictions obtained for ten classes, observed for both seen and unseen examples of the (a) MNIST, (b) CIFAR-10, and (c), (d) ImageNet datasets. We used LeNet-5 with ReLU activation [114], Extended LeNet-5 [158, 23], VGG-16 [189], and ResNet-50 [75] to obtain these results. For ImageNet, ten classes were selected randomly. Our models achieved 98%, 80%, 70.5%, and 77% top-1 accuracy on the respective datasets. These results are comparable to the results presented in [23, 75, 158, 189]. 162
- 6.8 Illustration of adversarial subspaces with high logit values for the MNIST (top) and CIFAR-10 (bottom) datasets. Logit value distributions of the predictions for the genuine images from the training set are given in the form of boxplots. The calculated thresholds and the highest logit value of the adversarial examples generated with IFGS and CW are highlighted as red, blue, and black lines, respectively. The adversarial spaces found using our method are indicated with arrows. 165

6.9 Illustration of adversarial subspaces with high logit values for the ImageNet datasets. Logit value distributions of the predictions for the genuine images from the training set are given in the form of boxplots. The calculated thresholds and the highest logit value of the adversarial examples generated with IFGS and CW are highlighted as red, blue, and black lines, respectively. The adversarial spaces found using our method are indicated with arrows. Note that the y -axis is log-scaled for clarity. . . 166

6.10 Two adversarial examples (red and gray circle) and their initial starting point (white circle) are added to the classification problem previously shown in Figure 6.5. Logit and softmax predictions of data points that lie on the cyan ($x_2 = 0$) and magenta ($x_1 = 3$) lines are presented as the second and third graph, respectively. 173

7.1 A number of studies that work with images taken from the ImageNet validation set, grouped based on the number of source images used for creating adversarial examples. 184

7.2 Adversarial examples that have been created with PGD, shown on the left, are misclassified into the categories listed on the right. 186

7.3 An original image and two adversarial examples created through the usage of an adversarial attack and rotation. The original image is correctly predicted as *apple*, while the adversarial examples are predicted as (left) *lab coat* and (right) *plane* by a ResNet-50 model trained on ImageNet. The ℓ_p norm of the perturbations used are provided below both adversarial examples. 188

List of Tables

3.1	Mean and standard deviation, for the experiments “ <i>equal perturbation</i> ” and “ <i>equal multiplier</i> ”, of (1) the least number of iterations of added perturbation needed to change the prediction and (2) the time required to calculate the perturbation (measured on a single Titan-X GPU).	73
3.2	Mean (standard deviation) ℓ_0 distances calculated between genuine images and their adversarial counterparts, and where the adversarial counterparts have been created through localization of the perturbation (see the first column). Adversarial examples are created by the source models listed in the first row and transfer to the target models listed in the second row.	84
3.3	Mean (standard deviation) ℓ_2 distances calculated between genuine images and their adversarial counterparts, and where the adversarial counterparts have been created through localization of the perturbation (see the first column). Adversarial examples are created by the source models listed in the first row and transfer to the target models listed in the second row.	85
3.4	Mean (standard deviation) ℓ_∞ distances calculated between genuine images and their adversarial counterparts, and where the adversarial counterparts have been created through localization of the perturbation (see the first column). Adversarial examples are created by the source models listed in the first row and transfer to the target models listed in the second row.	88

- 4.1 Train and test accuracies of selected models on the glaucoma and cell segmentation datasets. 98

- 5.1 An overview of all activities available in the dataset of [Vandersmissen et al. \[2019\]](#). 115
- 5.2 Validation and testing accuracy of the architectures \mathcal{A} , \mathcal{R} , and \mathcal{L} for each class, as obtained for the respective evaluation splits described in Section 5.2.1. 121
- 5.3 Median value (interquartile range) of the ℓ_2 and ℓ_∞ distances obtained for 1,000 adversarial optimizations, as well as their success rate, for the models and datasets described in Section 5.2.1. For easier comprehension, the threat models are listed from most permissive to least permissive. ℓ_∞ values less than 0.003 are rolled up to 0.003 (this is approximately the smallest amount of perturbation required to change a pixel value by 1, so to make discretization possible). 126
- 5.4 Median value (interquartile range) of the ℓ_2 and ℓ_∞ distances obtained for 1,000 adversarial optimizations, as well as their success rate, for \mathcal{L} and the datasets described in Section 5.2.1. For easier comprehension, the threat models are listed from most permissive to least permissive. ℓ_∞ values less than 0.003 are rolled up to 0.003 (this is approximately the smallest amount of perturbation required to change a pixel value by 1, so to make discretization possible). 128
- 5.5 Median value (interquartile range) of the ℓ_2 and ℓ_∞ distances obtained for 1,000 adversarial optimizations, as well as their success rate, for the models and datasets described in Section 5.2.1, hereby using the padding attack described in Section 5.4.2. 131

- 6.1 Accuracy (genuine, adversarial (overall)) of binary classification between genuine images and adversarial examples, where the adversarial examples have been generated by the adversarial losses listed in the first column for the given models. Each entry represents the outcome of an adversarial retraining. Architectures are ordered from the left to the right in terms of ascending convolutional layer complexity. 150

6.2 Breakdown of the classification accuracy obtained for genuine images and adversarial examples when four types of adversarial losses are incorporated. Each column represents the outcome of an adversarial retraining. Architectures are ordered from the left to the right in terms of ascending convolutional layer complexity. 151

6.3 Percentage of genuine images incorrectly identified as outliers, as obtained for different values of k in the following calculation: $g(\theta, \mathbf{x})_c >? (Q_3 + k\, IQR)_{(c)}$. Thresholds are calculated from correctly classified training examples for each class and tested on both training and test examples. k_{min} is the smallest value of k needed to ensure that none of the examples in the training and the test set are incorrectly identified as outliers. 164

6.4 Approximate proportion of the number of adversarial examples detected in each dataset (calculated using Equation 6.2). 167

1

Introduction

Human anatomy represents highly structured complex machinery that is made up of vastly different organs that, for the most part, cooperate with each other seamlessly [228]. Throughout history, one of the most important goals of scientists was to understand how these organs work. Thanks to advances in technology, we were able to figure out the function of most bodily structures in recent years [51]. For example, we know how veins carry blood, how bones support the body just like columns in a building, providing protection for other organs, and how our digestive system works. Furthermore, thanks to this knowledge, we are able to perform surgeries and find cures to illnesses that seemed fatal in the past. These advances in technology and knowledge resulted in drastic improvements of the life expectancy of humans [145].

Although we currently understand most of the structures in our bodies, the brain, arguably the most important part of the human body, still remains a mystery [87]. We know that the brain is mostly made up of interconnected nerves that receive, forward, and process the information obtained from all parts of the body. We also know that different parts of the brain are responsible for different tasks. We somewhat understand how information from the outside world is received via seeing, hearing, touching, tasting, or smelling, and which part of the brain these signals activate. However, we still do not fully understand how this perceived information from the outside world is experienced, stored, and then made available for future use [87].

As of now, there is a large gap between our understanding of low-level brain activity (e.g., activities of neurons and cells, interactions with different organs) and high-level brain activity (e.g., thinking, understanding, learning, and remembering). To make things more com-

plicated, some studies argue that the modality of learning is not uniform across individuals [168], stating that people learn better when the information is given in their preferred modality (visual, auditory, or kinaesthetic), which is called the *meshing hypothesis* [5]. On the other hand, there are strong claims against this hypothesis, arguing that the evidence for the existence of the meshing hypothesis is weak and inconsistent [159]. As such, despite all the efforts made, up until now, constructing the blueprint of the activity of learning has not been possible.

Even though we are not able to exactly identify how the process of learning occurs, we know that our past experiences have an undeniable impact on the way we make decisions. We learn by experiencing the outside world and then, more often than not, make our decisions based on our past experiences. In this respect, we are able to experience the outside world thanks to a number of sense organs that are equipped with sensory neurons. Although there are multiple sensory organs, for topics related to this dissertation, we are interested in only one of them: vision. As such, in what follows, we will discuss the workings of the human vision system.

1.1 Human vision

Among all the sensory information humans receive from the outside world, we rely the most on visual information in order to perform our day-to-day activities [87]. As a result, we optimize most of the tools and systems commonly used in our lives according to this type of sensory information. Although we often do not think about such topics, we could, for the most part, agree that losing our ability to see would have the most significant impact on our lives, compared to losing another type of sensing (hearing, touching, smelling, tasting). Because of our heavy reliance on vision, a significant effort has been dedicated to understanding the different components of the human visual system as well as the limitations of this system.

As illustrated in Figure 1.1, visual perception starts with light emitted from an object entering our eyes through the cornea, pupil, and lens. Although the binocular human visual field spans about 220 degrees horizontally and 135 degrees vertically, the quality of information in the field of vision is not identical, with the quality depending on where the viewed area falls on the retina. After hitting the retina,

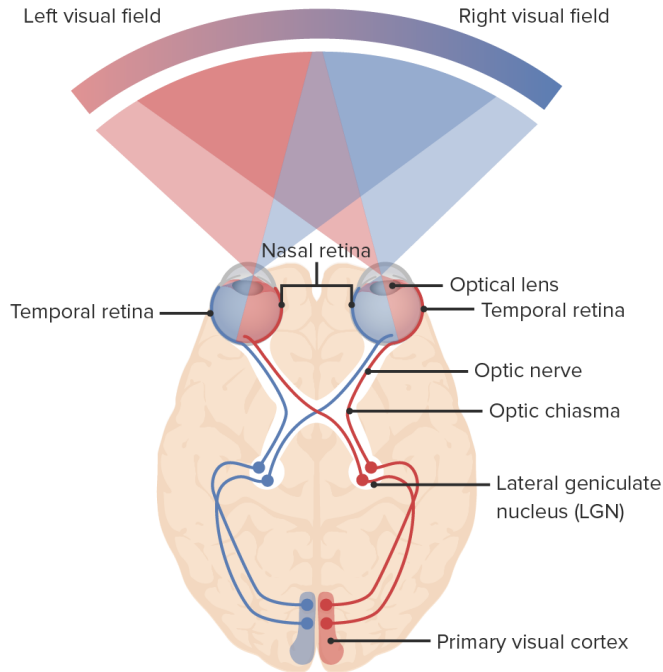


Figure 1.1: A simplified illustration of the human visual system [112].

the wavelength of the light is translated into a biological signal, which is then transferred through neuronal pathways (i.e., optical nerves) to the visual processing areas of the brain. At this point, perception occurs.

We can roughly divide the human visual system into two parts: (1) the mechanical part, where the vision takes place, and (2) the interpretation part, where the perception and understanding of the scene happens. The mechanical part of the human visual system is well understood. Indeed, thanks to our understanding of how light projected from objects is interpreted by our eyes, we were for instance able to create digital cameras that can take more detailed pictures than human eyes. Although this is not a perfect method of evaluation, the human eye is said to be equal to a 576 megapixel camera. To put this in context, recently announced phones have about 10 megapixel cameras and typical digital cameras in stores have a resolution of about 50 megapixels. Moving away from consumer-level products, industrial-level cameras can take pictures with a resolution of up to 3,200 megapixels. There are indeed optical similarities be-

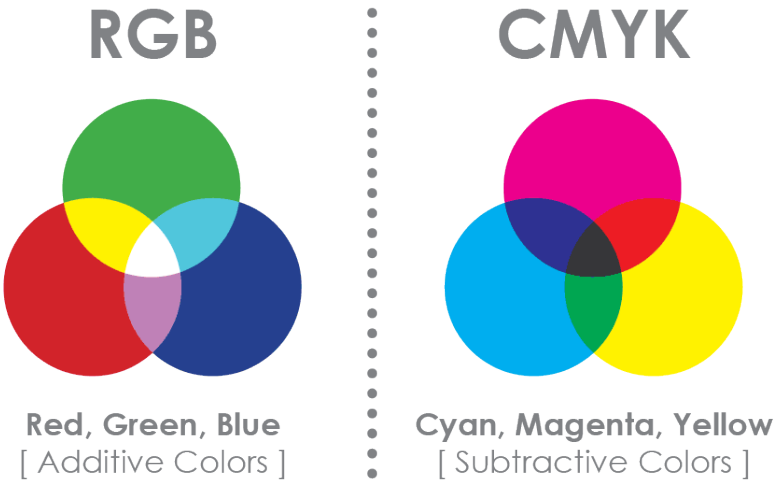


Figure 1.2: Illustration of the (left) RGB and (right) CMYK color spaces [90].

tween a camera and the eye, given the way a camera captures an image of the environment. However, a camera does not have perceptual capabilities or cognitive abilities; it does not know about the world. As a result, the challenge is to come up with a seamless interpretation of the scenery that can be seen in images captured by cameras.

1.2 Computer vision

Computer vision is a field of computer science that aims at enabling computers to understand and interpret the outside world similar to how we humans do. In this context, sensors such as digital cameras act similar to the way our eyes see the outside world and capture pictures. As illustrated in Figure 1.2, individual pixels in pictures captured by a digital camera can be represented using various color spaces, including but not limited to RGB (Red/Green/Blue) and CMYK (Cyan/Magenta/Yellow/Key). In addition, a picture as a whole can be represented using functions, graphs, or matrices, with matrices being the most commonly used representation. An example matrix representation for a grayscale image is presented in Figure 1.3, where the brightest color (white) is mapped to the value of 255 and the darkest color (black) is mapped to the value of 0. The shades of gray are represented as val-

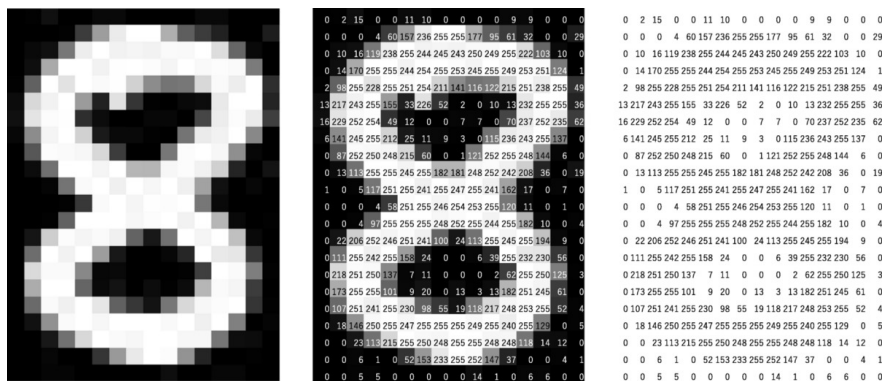


Figure 1.3: Use of a numerical method for image representation, with this numerical method taking the form of a matrix [71].

ues between 0 and 255, depending on their brightness. When working with a colored image, instead of one matrix, three matrices are used, one for each color channel (e.g., Red/Green/Blue).

As described above, humans have invented mechanisms to reliably acquire and represent pictures taken by digital cameras. However, unlike humans, interpretation of such pictures by digital cameras is challenging. For example, tasks like understanding what a picture contains, identifying individuals, or recognizing actions can be seamlessly performed by humans. However, creating computational systems that mimic how humans perform these tasks is proven to be not straightforward.

The search for visual interpretation of images originating from digital cameras is said to have started circa 1966, when Professor Marvin Minsky at MIT asked a group of students to take a number of pictures and make a computer “describe what it saw” by dividing the pictures into sectors and by assigning labels to those sectors according to the objects they contain [82]. Unfortunately, this project did not meet its requirements, since it was discovered that this simple task of object recognition was harder than imagined back then. Although we have found a number of solutions to tasks involving everyday problems such as automatic number-plate recognition, optical character recognition, and red-eye removal, researchers in the field of computer vision still struggle with the problem of visual image interpretation. Alas, visual image interpretation, which humans perform easily and which was devised as a small summer project in 1966, still remains one of the hardest challenges in computer vision.

One of the biggest hurdles of image interpretation is the computational description of the features that are representative for objects. What makes a dog, a dog? And a cat, a cat? Is it the size of their body? Is it the difference in their eyes? Let us assume we have the distinctive properties of cats and dogs. What about other animals/objects? We have to come up with feature descriptions for all known object categories, as well as for objects that lie in their sub categories (e.g., the category of dog and all underlying dog breeds). Unfortunately, this approach is not scalable beyond a couple of categories. For this reason, more-often-than-not, computer vision problems are solved with techniques that involve machine learning approaches. In this context, the term machine learning refers to the study of algorithms that allow learning and decision-making through the usage of data.

Until recently, computer vision problems were solved in two steps: (1) extracting descriptive features for each category with a computer vision algorithm [123] and (2) giving the extracted features as an input to machine learning models in order to facilitate learning (based on the given features). Although this approach achieved a certain degree of success, descriptive features generated with computer vision algorithms failed to capture the essence of images, which the resulting systems failing to achieve human-level performance on complex vision tasks [155].

Nowadays, the approach just described is abandoned in favor of machine learning systems that do not only perform the decision-making step (Step 2), but that also perform the feature extraction step (Step 1). The most prominent of such decision-making systems are artificial neural networks (ANNs) [171], which are inspired by the hypothesized way of working of the human brain. Specifically, to allow learning from data, ANNs were created by taking inspiration from the biological neuron firing process in order to propagate relevant signals, thus having a close resemblance to certain parts of the human vision system. However, the human visual system is not flawless. And, as we will discuss shortly, artificial neural networks, created with inspiration taken from the way our brain works, also seem to have a number of flaws that are similar to the flaws of the human visual system.

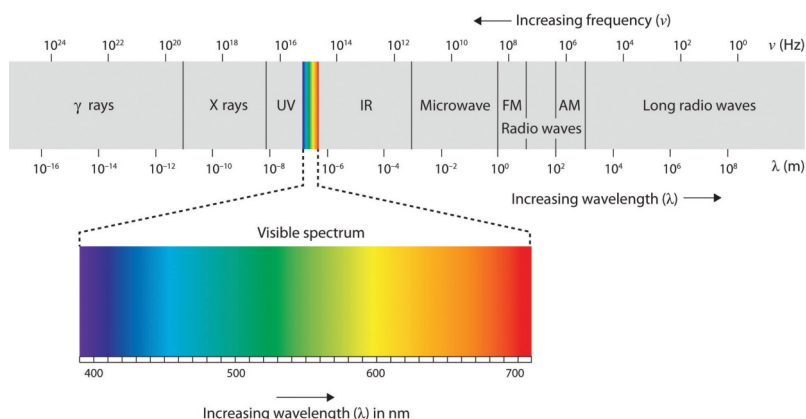


Figure 1.4: Electromagnetic spectrum, ranging from gamma waves to long radio waves, including the spectrum visible to the human eye [2].

1.3 Limitations of human vision

Thus far, we have praised the human vision system for its capabilities. However, this does not mean that the human visual system is impeccable. To start, humans cannot see most of the electromagnetic spectrum. As shown in Figure 1.4, a typical human eye can only respond to wavelengths between 380 and 750 nanometers. The spectrum that is visible to the human eye is often called the *visible spectrum*. Even within the visible spectrum, we cannot identify objects or colors if no decent light source is available in the vicinity. Thus, environmental conditions also have a large impact on our ability to see. Unfortunately, flaws of the human visual system are not limited to the aforementioned ones: the human visual system is also prone to defects, hence the reason for many people like myself wearing glasses in order to see better. Even when the visual system of a person is defect-free at earlier ages, the vision of a person is also known to get worse as aging occurs.

As explained above, although the human visual system has a considerable number of flaws, we will narrow down our interest to a particular limitation, namely the existence of optical illusions. Optical illusions are said to occur when the visual perception of a person differs from the reality in a misleading way. Some illusions are as old as time, such as the waterfall illusion mentioned by Aristotle: after looking at a waterfall for a certain amount of time, surrounding objects

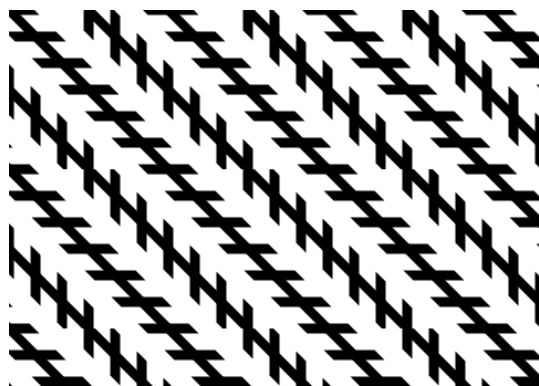


Figure 1.5: The Zöllner illusion consists of parallel lines that, when injected with short lines at an acute angle, appear to be divergent.

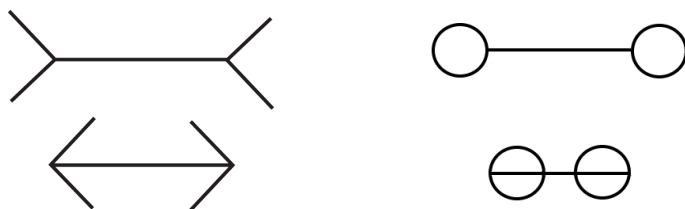


Figure 1.6: (left) The Müller-Lyer illusion and (right) a variant of the Müller-Lyer illusion. In both illusions, the horizontal lines have the same length, but the bottom ones appear to be shorter than the top ones.

appear to move upwards. Other optical illusions have been discovered more recently. The study of optical illusions entered the scientific discourse in the 19th century when the human visual system was being scrutinized from all corners. Since then, we have discovered a vast number of illusions that take advantage of various limitations of the human visual system. In what follows, we will discuss a number of phenomenological groups of optical illusions in more detail.

Geometric optical illusions—This kind of illusion was among the first to be discovered and scrutinized by scientists. Although geometric optical illusions may appear in extremely simple forms, containing just a couple of lines and circles, they are able to deceive our visual system flawlessly. A famous example of this type of illusion is the Zöllner illusion, which can be seen in Figure 1.5. This illusion consists of parallel lines that appear to be divergent when augmented with

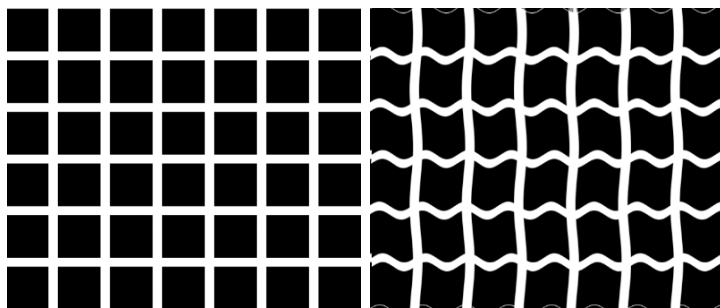


Figure 1.7: (left) The Hermann grid, which seems to have grey patches at the intersections of white lines and (right) the corrected Hermann grid, no longer depicting an illusion.

other, acute-angled short lines [237]. Another example of this type of illusion is the Müller-Lyer illusion that can be seen in Figure 1.6. In this illusion, the horizontal lines at the top appear to be longer than the horizontal lines at the bottom [139], even though they are of the same length.

Geometric optical illusions lay bare the interconnectedness of our eyes to the brain, where, even when the viewer knows that the provided illustration contains an optical illusion, the brain is still fooled by its interpretation. For the examples given above, even though we know that the vertical lines are parallel in the Zöllner illusion, or that the horizontal lines are of equal length in the Müller-Lyer illusion, we still have a different intuition than what we know to be true.

Optical illusions related to luminance and contrast – The Hermann grid provided in Figure 1.7 is a representative example of an optical illusion that involves an interplay between luminance and contrast [80]. When looking at the squares, we see dark, grey spots at the intersections of the white lines, even though no such spots exist. Unlike the geometric optical illusions though, when we directly look at a particular intersection, we can clearly see that the illusion is not real. Thus, what we perceive as grey spots is not a physical object of the external world but just an artefact of our visual system. Different from the two optical illusions discussed before, this type of illusion is more related to the eye and its inability to see clearly, rather than being related to misinterpretation by the brain. For a long time, this type of illusion was explained on the basis of lateral inhibition and the inhibitory processes in the retinal ganglion cells. Recently, however, it was demonstrated that the introduction of simple waves removes the Hermann illusion, as shown in Figure 1.7. As a result, it was argued

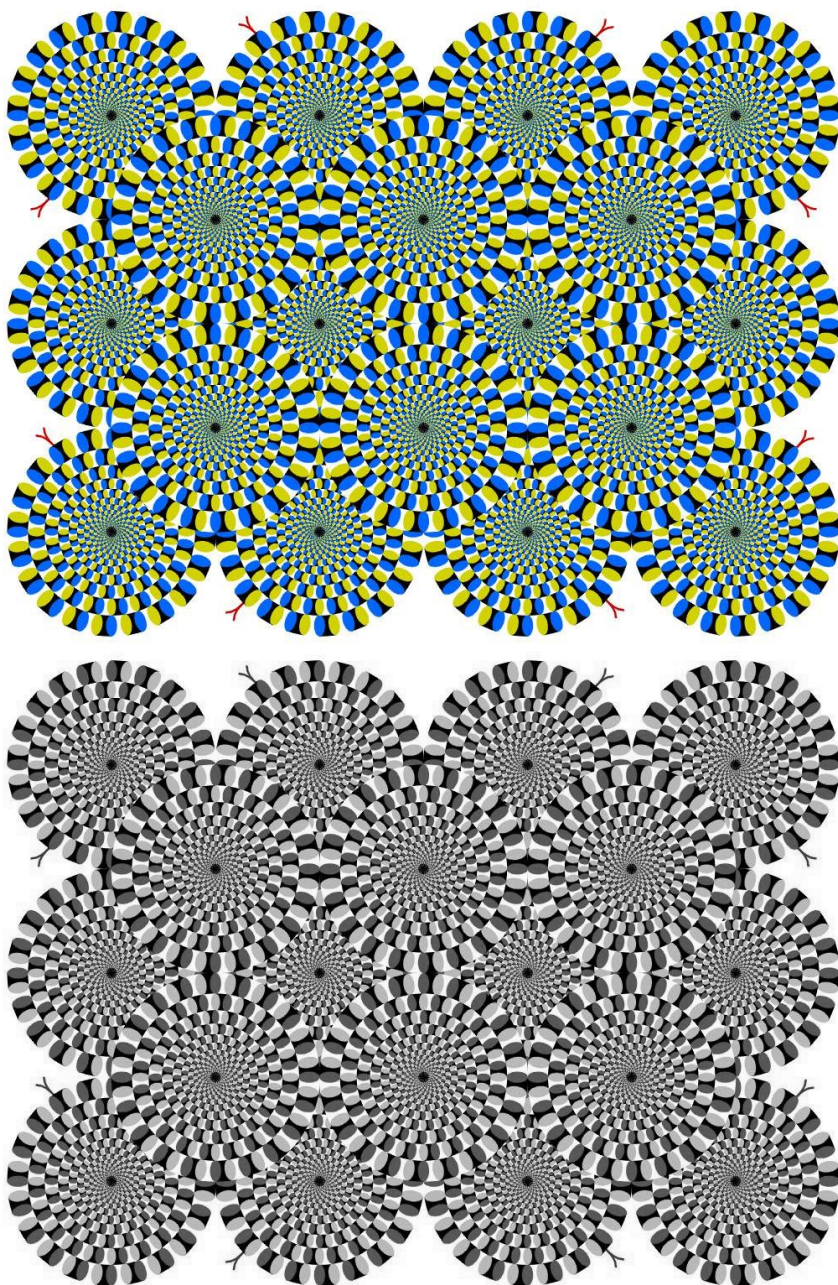


Figure 1.8: (top) The rotating snakes illusion, evoking illusory motion in the periphery of the visual field, and (bottom) a gray-scaled version of the same illusion.

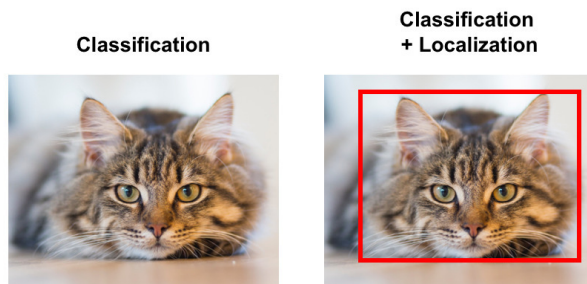


Figure 1.9: (left) Picture of a cat and (right) picture of a cat with bounding boxes that allow for localization.

that an investigation involving the cortical process is also necessary in explaining this type of illusion [54].

Optical illusions with false motion – Proposed by Kitaoka and Ashida [104], the Rotating Snakes illusion provided in Figure 1.8 is a type of optical illusion that appears to have motion, even though it is a motionless picture. Specifically, the Rotating Snakes illusion appears to be motionless at the point of focus, whereas peripheral areas appear to be rotating. Although no consensus is available on why this illusion occurs, one of the convincing hypotheses is related to the speed with which visual signals are processed: it is hypothesized that high-contrast areas are processed faster than low-contrast areas, where contrast is defined over the entire receptive field of an individual retinal neuron [121]. Specifically, for this illusion, the regions with the highest contrast appear in the outermost circles. For this reason, areas that are farther from the focus point appear to be rotating more than the area containing the focus point. Lombrozo [121] also argued that the gray-scaled version of this illusion does not appear to be as severe (i.e., in terms of rotating) as the colored version, which strengthens the aforementioned hypothesis.

1.4 Limitations of computer vision

Similar to the problems associated with the human visual system described above, automated systems involving computer vision techniques also suffer from a number of limitations. One of the biggest problems in the field of computer vision is that, unlike the human visual system, solutions to solve a particular problem do not easily transfer to other problems (i.e., automated systems are not as fluid



Figure 1.10: (left) A picture of an apple, (center) a picture of an iPod, and (right) a picture of an apple covered by the text *iPod*. The left picture is correctly classified as an apple by a machine learning model, whereas the right picture is misclassified as an iPod.

as our visual system). We have ways to design systems to solve a particular problem but expanding the scope of a specialized system to solve another problem is not easy. For example, considering the cat picture shown in Figure 1.9, if we design a system to tell us what a picture contains, we are trying to solve a classification problem. Now, instead of only telling us what the picture contains, if we also ask the system to determine the location of an object of interest, we are trying to solve both a classification and a localization problem. Humans are able to solve such a problem seamlessly, since we can simply show where the object we identify is. However, for computer vision systems, this transformation from classification to both classification and localization is not straightforward.

Another limitation of computer vision methods that heavily involve machine learning approaches (such as deep neural networks) is related to label availability. As briefly mentioned in Section 1.2, most computer vision problems are solved nowadays with approaches involving machine learning techniques. In this respect, when the initial approach to address a computer vision problem has a limited number of labels at its disposal, it is not straightforward to increase the number of labels to cover more objects of interest. As a result, this limitation may result in wrong predictions that are due to a lack of elaborate labeling scenarios [60]. For example, Figure 1.10 (left) contains a picture of an apple that is correctly classified as an apple, but when the text *iPod* is written in front of the apple (as shown in Figure 1.10 (right)), then the picture is misclassified as an iPod. This misclassification is made because (1) most pictures of iPods have this text present, with these pictures thus inherently containing a strong textual clue (see Figure 1.10 (center)) and (2) the lack of the following

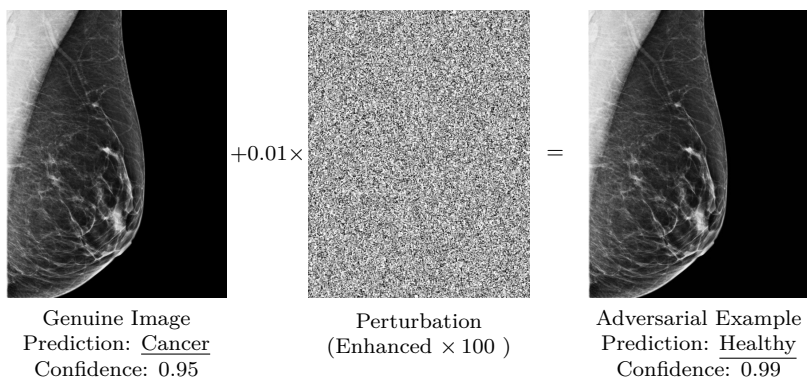


Figure 1.11: A genuine image, initially classified as *cancer* with 0.95 confidence by a deep learning model, is perturbed to become an adversarial example. This adversarial example is then classified by the same model as *healthy* with 0.99 confidence.

elaborate label in the machine learning system: “an apple covered by the text *iPod*” [100].

The aforementioned limitations are problems that the field of computer vision is facing since it started tackling image recognition problems. We are, however, interested in one of the recently discovered security flaws of neural networks that appears to be similar to the phenomenon of optical illusions for human vision: the vulnerability of neural networks to adversarial examples [201].

Adversarial examples are malicious data points that have been created with ill intent, namely to mislead systems for automated prediction [62]. In Figure 1.11 on the left, we provide an image that is correctly classified as belonging to a cancer patient, with this image being turned into an adversarial example through the addition of what is called adversarial perturbation. This adversarial example is then classified as belonging to a healthy patient by the same decision-making system, even though the visual difference between the two images is almost imperceivable.

Such data points are said to exist due to the unique way neural networks learn features from images, which is different from how humans perceive images [89]. Take, for example, the Rotating Snake illusion given in Figure 1.8. Pixel values of this image are not changing. As such, from a computational perspective, it is clearly not moving. However, the human visual system perceives it as moving, even though we know that it is not. Likewise, even though we perceive the vertical lines in the Müller-Lyer illusion to be different, from the perspective

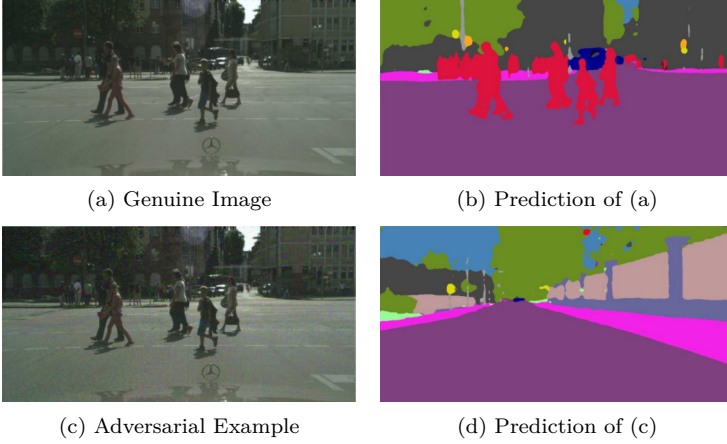


Figure 1.12: A genuine image and its road segmentation are given. With the addition of adversarial perturbation to this genuine image, the predictions obtained for its adversarial example counterpart result in a completely different road segmentation, even no longer having pedestrians.

of a computer, they are the same. On a similar note, the grayish color we observe between the squares in the Hermann grid is, from the perspective of a computer, just white. As such, these optical illusions deceive our vision system perfectly, but they do not have any effect on computational methods. Adversarial examples can be seen as illusions that “trick” computational systems, but not humans (this time, the effects are thus reversed). Specifically, to us, adversarial examples seem no different than the regular images they are created from. However, to decision-making systems, they portray completely different representations. To sum up, these harmful data points take advantage of weaknesses exhibited by automated prediction systems, in a way that is similar to how optical illusions take advantage of flaws in the human visual system. The main difference is that optical illusions that mislead the human visual system are not considered particularly harmful to our day-to-day activities, whereas adversarial examples are considered to be a major security problem of automated prediction systems [22].

Although adversarial examples are reported to be present in all machine learning systems [14, 15], current interest in this phenomenon is due to the super-human results obtained by deep neural networks (DNNs) in recent years for a variety of specifically designed tasks, including, but not limited to, classification [109], object detection [118],

and segmentation [170]. Even though adversarial attacks have been recognized to be a threat for all domains that make use of DNNs, the domain of vision in particular suffers the most from adversarial attacks, since image perturbation is often invisible to the bare eye (see Figure 1.11 and Figure 1.12). Moreover, the increasing deployment of DNNs for mission-critical tasks, such as self-driving cars and medical image diagnosis, further amplifies this threat since adversarial examples cannot be easily detected [28, 49, 131, 214]. Because of their pervasive impact on the predictions made by machine learning models, adversarial examples are now cited as a major reason to advocate against the deployment of fully-automated prediction systems that interact with humans. As a result, substantial research efforts are being dedicated to achieving a better understanding of adversarial examples, with the goal of finding methods that mitigate their negative effects on machine learning models.

1.5 Focus of this dissertation

Given the increasing usage of automated prediction systems in every aspect of our lives and given the increasing adoption of machine learning models in such systems, in this dissertation, we investigate the threat of adversarial examples to machine learning systems, and to DNNs in particular. We perform this investigation from three angles:

- We start our investigation by examining commonly used adversarial attacks and the adversarial examples produced by these attacks. Next, we analyze various properties of the adversarial examples created by the attacks examined, with the goal of finding unique properties that may allow for ease of detection of adversarial examples. Furthermore, we put forward a number of theoretical findings regarding the characteristics of various adversarial loss functions, experimentally confirming our theoretical observations. Moreover, we investigate whether regions of importance exist in the image domain, possibly leading to a different impact of the perturbations exercised.
- Taking a closer look at adversarial attacks, and moving towards more specific application domains, we investigate the threat of adversarial attacks to (1) biomedical segmentation models and (2) radar-based activity detection systems. In doing so, we propose two novel attacks, where the first attack can be used against

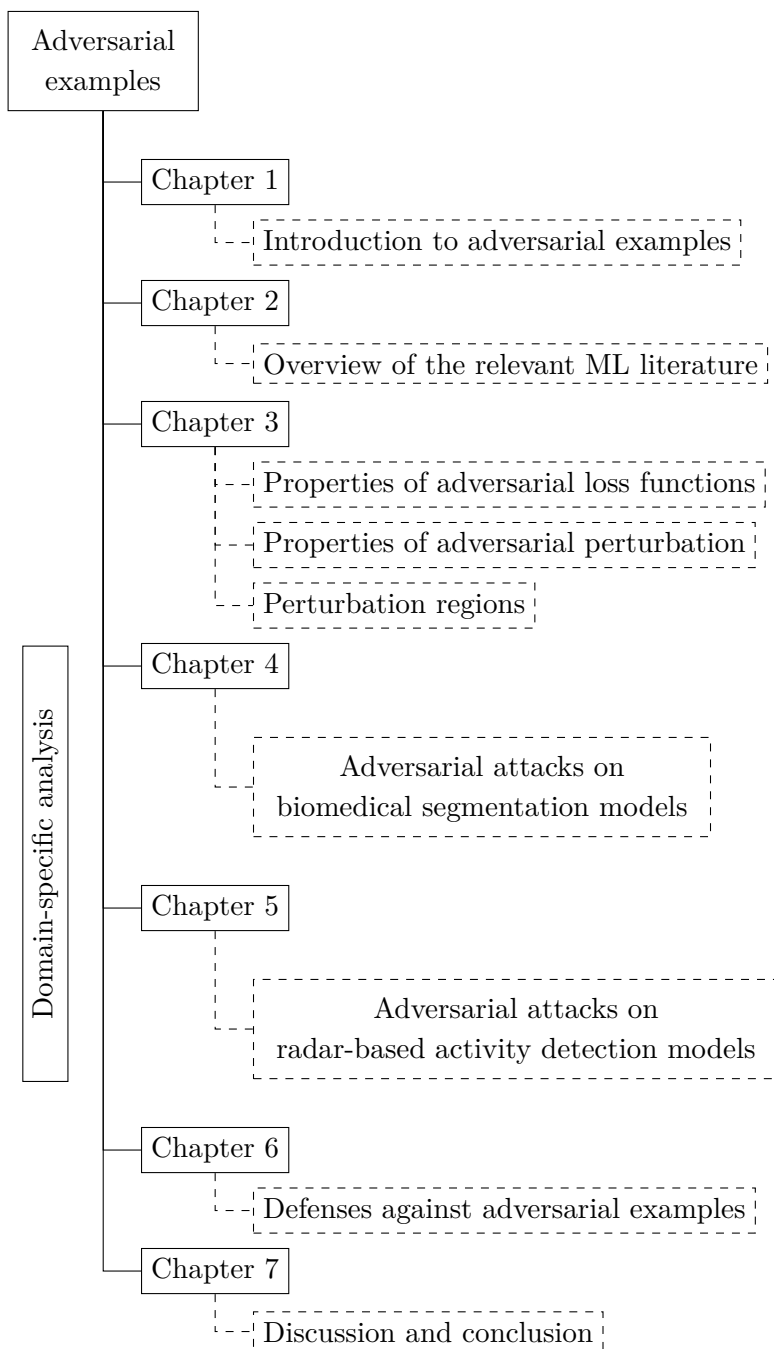


Figure 1.13: Overview of the chapter organization.

segmentation models and where the second attack can be used against systems that make use of padding.

- Lastly, we pay attention to the usage of adversarial defenses, targeting the detection of adversarial examples. We investigate the plausibility of adversarial retraining, as well as a novel adversarial defense that leverages statistical techniques to detect prediction outliers.

1.6 Organization of this dissertation

In what follows, we briefly describe the different topics covered by each chapter. A visual overview of the chapter organization of this dissertation can be found in Figure 1.13.

Chapter 2 - Learning from data

In this chapter, we first briefly go over a number of developments in the field of machine learning. We explain the internal workings of neural networks and the mechanisms that allow them to learn from data. We also go over the mathematical notation used throughout this dissertation, as well as the adopted machine learning techniques.

Chapter 3 - Adversarial attacks on deep neural networks

In this chapter, we first provide a literature review on adversarial examples, shedding light on the research that has thus far been done in this field. We discuss a number of adversarial attacks and we proceed to reveal the properties of a number of popular adversarial attacks, as well as the adversarial examples created by such attacks. Later parts of this chapter contain our findings regarding the impact of adversarial perturbation when such perturbation is only exercised on particular image parts, thus achieving adversariality with so-called regional perturbation. This chapter is based on the research efforts described in [147, 150, 151].

Chapter 4 - Adversarial attacks on biomedical image segmentation models

We explore the adversarial risk for biomedical image segmentation models and we propose a novel adversarial attack, named Adaptive

Segmentation Mask Attack. This novel adversarial attack allows creating adversarial examples for semantic segmentation models, where this attack makes it possible to change the segmentation prediction for an input image. Using two biomedical datasets, we evaluate the performance of the proposed attack, measuring its effectiveness through both qualitative and quantitative analyses. This chapter is based on the research effort described in [148].

Chapter 5 - Adversarial attacks on radar-based activity recognition systems

Adversarial examples are not only a threat for the domain of natural images. Radar-based detection methods have been used increasingly in conjunction with neural networks, which, in turn, also make them vulnerable to adversarial attacks. In this chapter, we investigate the vulnerability of radar-based activity detection methods to typical adversarial attacks, as well as a specially crafted adversarial padding attack that takes advantage of the peculiar nature of the data points used. We provide a detailed investigation of multiple machine learning models that rely on different technologies, finding that such systems are indeed vulnerable to adversarial examples. This chapter is based on the research effort described in [154].

Chapter 6 - Detecting adversarial examples

We discuss a number of defenses that have been proposed to prevent adversarial attacks and then proceed to evaluate adversarial retraining, one of the most successful approaches towards detecting adversarial examples. Examining studies that aim at detecting adversarial examples and leveraging the knowledge obtained through our previous research efforts, we attempt to detect a certain subset of adversarial examples that are called over-optimized adversarial examples. These over-optimized adversarial examples produce unnaturally high logit values, making it possible to differentiate them from genuine data points. Relying on statistical techniques, we propose a novel adversarial defense method that detects such adversarial examples. This chapter is based on the research effort described in [147, 149].

Chapter 7 - Conclusions and outlook

We summarize our research findings regarding the threat posed by adversarial examples to the different decision-making systems discussed.

In later parts of this chapter, I also reflect on my doctoral journey, raising concerns on the way research on adversarial machine learning is done. Lastly, I discuss a number of research questions that have not been answered thus far. In doing so, I provide directions for future research efforts in the field of adversarial machine learning.

1.7 Contributions

During the preparation of this dissertation, a number of research papers and software packages were published. A list of published research papers and software packages can be found below.

Articles published in journals

1. Utku Ozbulak, Wesley De Neve, and Arnout Van Messem. Perturbation Analysis of Gradient-based Adversarial Attacks. *Pattern Recognition Letters, Elsevier*, 2020.
2. Utku Ozbulak, Baptist Vandersmissen, Azarakhsh Jalalvand, Ivo Couckuyt, Arnout Van Messem, and Wesley De Neve. Investigating the Significance of Adversarial Attacks and Their Relation to Interpretability for Radar-based Human Activity Recognition Systems. *Computer Vision and Image Understanding (CVIU), Special Issue on Adversarial Deep Learning in Biometrics & Forensics, Elsevier*, 2020.

Articles published in conferences

1. Utku Ozbulak, Wesley De Neve, and Arnout Van Messem. How the Softmax Output is Misleading for Evaluating the Strength of Adversarial Examples. *Neural Information Processing Systems (NeurIPS), Workshop on Security in Machine Learning (SecML)*, 2018.
2. Utku Ozbulak, Wesley De Neve, and Arnout Van Messem. Not All Adversarial Examples Require a Complex Defense: Identifying Over-optimized Adversarial Examples with IQR-based Logit Thresholding. *International Joint Conference on Neural Networks (IJCNN)*, 2019.

3. Utku Ozbulak, Arnout Van Messem, and Wesley De Neve. Impact of Adversarial Examples on Deep Learning Models for Biomedical Image Segmentation. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2019.
4. Utku Ozbulak, Jonathan Peck, Wesley De Neve, Bart Goossens, Yvan Saeys, and Arnout Van Messem. Regional Image Perturbation Reduces L_p Norms of Adversarial Examples While Maintaining Model-to-model Transferability. *International Conference on Machine Learning (ICML), Workshop on Uncertainty & Robustness in Deep Learning (UDL)*, 2020.

Out-of-scope publications

1. Taewoo Jung, Esla Timothy Anzaku, Utku Ozbulak, Stefan Magez, Arnout Van Messem, and Wesley De Neve. Automatic Detection of Trypanosomosis in Thick Blood Smears Using Image Pre-processing and Deep Learning. *International Conference on Intelligent Human Computer Interaction (IHCI)*, 2020.
2. Utku Ozbulak, Esla Timothy Anzaku, Wesley De Neve, Arnout Van Messem. Selection of Source Images Heavily Influences the Effectiveness of Adversarial Attacks. *British Machine Vision Conference (BMVC)*, 2021.
3. Utku Ozbulak, Maura Pintor, Arnout Van Messem, Wesley De Neve. Evaluating Adversarial Attacks on ImageNet: A Reality Check on Misclassification Classes. *Conference on Neural Information Processing Systems (NeurIPS), Workshop on ImageNet: Past, Present, Future*, 2021.

Software packages

The research efforts described in this dissertation have led to the creation of a number of software packages. These software packages are listed below:

1. github.com/utkuozbulak/pytorch-cnn-visualizations
A widely used convolutional neural network visualization library that is available on GitHub, currently having more than 5,500 stars and 1,500 forks.

2. github.com/utkuozbulak/pytorch-custom-dataset-examples
An introductory repository that contains examples for the practical use cases for the `torch.utils.data.Dataset` class for the PyTorch library.
3. github.com/utkuozbulak/pytorch-cnn-adversarial-attacks
A software package that implements a number of adversarial attacks.
4. github.com/utkuozbulak/adaptive-segmentation-mask-attack
A software package that contains the PyTorch implementation of the Adaptive Segmentation Mask Attack proposed in [148].
5. github.com/utkuozbulak/regional-adversarial-perturbation
A software package that contains the PyTorch implementation of the regional adversarial perturbation approach proposed in [151].

Copyright

A portion of this dissertation contains materials (e.g., text, tables, and figures) that were previously disseminated through various venues and journals. Although the copyright of most of these materials is owned by the authors, we transferred the commercial rights of a number of publications to their respective publishers. For those materials for which we transferred the copyright to a publisher, we have ensured that their usage in this dissertation (both partially or completely) does not cause any copyright issues. The full copyright details can be found on the permissions pages of the relevant journals.

2

Learning from data

The term data refers to a set of quantitative and qualitative information that describes events. Ever since the beginning of humanity, we have been searching for more advanced methods to characterize events that we deem important. In the early ages of humanity, people described events with pictures and illustrations. Later on, written languages made more complex explanations possible, an approach we still use to this day. Although such information was initially preserved on paper, most information generated by humans is nowadays stored in a digital format.

Since digitally stored data can be maintained and analyzed much more efficiently compared to text written on paper, we have expanded our data collection capabilities and started to collect ... pretty much everything. Storage of data became such a central aspect of our lives that, even when we do not explicitly generate data ourselves, the data our actions produce are constantly being collected, either by governments or companies. For example, the GPS data our phones generate are collected by providers, our web activity is collected by ISPs, and our movements outside are tracked by CCTV systems. Clearly, a person does not have to work hard to produce tremendous amounts of data. However, the problem arises when analyzing and monetizing these mountains of data. Indeed, the problem is not with the collection of data, but with the extraction of knowledge.

Due to a tremendous increase in the amount of available data in the 21st century, hand-crafted decision-making systems for extracting knowledge became mostly obsolete. As a result, machine learning as a field saw a soaring interest in recent years in order to analyze these massive amounts of data. Machine learning is a field of computer

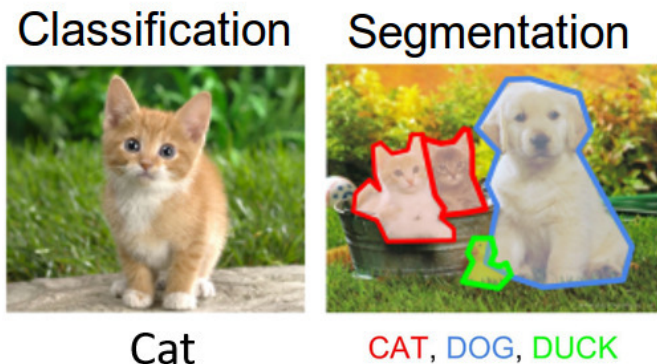


Figure 2.1: (left) An example classification task and (right) an example segmentation task. Labels are provided below the pictures shown.

science that aims at extracting knowledge from data with the help of algorithms. Different from hand-crafted decision-making systems, machine learning approaches aim at “learning” from data, resulting in decision-making systems that are capable of scaling.

Since the representation of tabular data is different from the representation of image data, which is, in turn, different from the representation of audio data, more often than not, the machine learning approach to take depends on the type of data used. Throughout the remainder of this dissertation, we will mostly be concerned with image data. A relevant phenomenon to mention with respect to the type of data used is the curse of dimensionality, which describes the increase in volume as the dimensionality increases. As a result of the increase in dimensionality, the data become sparse and the amount of data required to support results increases dramatically.

When talking about data in tandem with machine learning, we typically deal with two types of data: the data that the model or the algorithm learns from (training data) and the data used to evaluate the correctness of the learning machine (validation and test data). Alas, separation of all available data into training, validation, and testing data is not a simple task, since not all problems are equal in terms of the amounts of data required to address them. Moreover, while training learning machines, having more training data is always desired compared to having less training data. On the other hand, the less validation and testing data we have, the more uncertain we

will be about the effectiveness of our learning machine. As a result, validation and testing data are of utmost importance for measuring the effectiveness of the trained learning machine since machine learning models are known to memorize the training data when the training procedure is not handled correctly. All in all, the task of separating the data into the aforementioned partitions is not a trivial task.

Another important aspect related to data handling is the availability of labels. In this context, the data that come with descriptive information that is useful for the prediction task at hand are said to be labeled data. Unfortunately, labeling data is a costly endeavour, especially for tasks that require expert knowledge, such as tasks related to the medical domain. Based on the availability of labels, machine learning problems can roughly be grouped into two categories: supervised problems and unsupervised problems. In this dissertation, we will be working with supervised machine learning problems related to both natural images and radar images. Specifically, we will be concerned with neural networks that have been trained to solve classification and segmentation problems. As shown in Figure 2.1, the problem of classification is to label the object of interest in the image under consideration, whereas the problem of segmentation is to select the pixels for that object (or multiple objects).

The field of machine learning covers a large number of topics, with a wide variety of techniques proposed to solve unique problems. One such problem that is related to security is the vulnerability of machine learning models, specifically neural networks, to adversarial attacks. We will, in later chapters of this dissertation, explain how adversarial attacks are a threat to systems that have been designed to classify and segment objects. Before doing so, and in what follows, we will describe the neural network concepts that are relevant to the discussions presented in the upcoming chapters. Furthermore, we will establish the mathematical notation used throughout this dissertation.

2.1 Neural networks

Inspired by the way the human brain works, McCulloch and Pitts [127] laid the foundation in 1943 of a system that is comprised of neuron-like structures to solve logical problems (see Figure 2.2). Although the proposed system was, in theory, reliable, a functioning system based on this approach was only invented years later in 1958 by Rosenblatt [171].

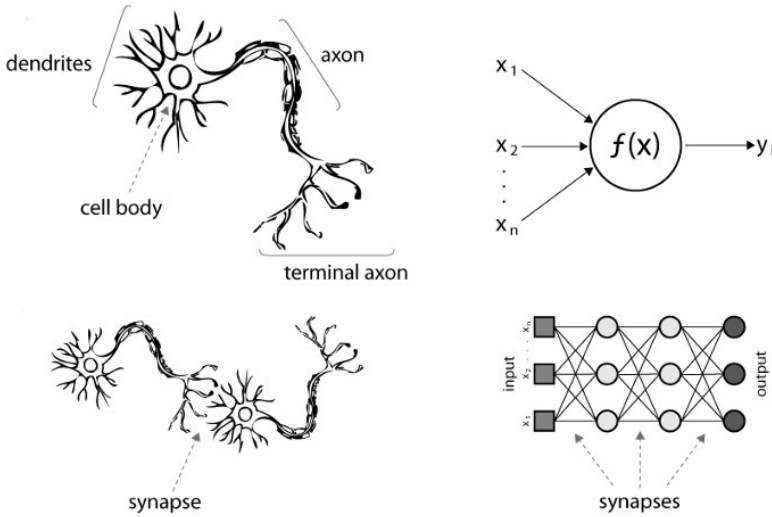


Figure 2.2: A visual representation of synapses and neurons in (left) the human brain and (right) neural networks.

Rosenblatt [171] proposed a learning machine called the perceptron, capable of *learning* from examples. This learning machine can be described as follows:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \mathbf{x} + \mathbf{b} > 1, \\ 0 & \text{otherwise,} \end{cases} \quad (2.1)$$

where the weight \mathbf{w} and the bias \mathbf{b} are tuned (i.e., learned) with the help of data in order to classify the input (\mathbf{x}) into two categories. In modern days, the perceptron is often referred to as a neural network without a hidden layer.

Although the perceptron was proposed as a general method to solve a large number of problems, it failed at doing so for a great number of tasks, with the most famous one being the XOR operation [133]. The reason for this failure is the incapability of the perceptron to make non-linear decisions. Indeed, with the proposed approach, it is only possible to solve linearly separable problems such as the OR problem, as shown in Figure 2.3. In their work, Minsky and Papert [133] demonstrated that, in order to solve non-linear problems, perceptrons should have had more than one trainable weight, as well as a method to introduce non-linearities. At that time, however, no reliable method was available to train such multi-layered perceptrons.

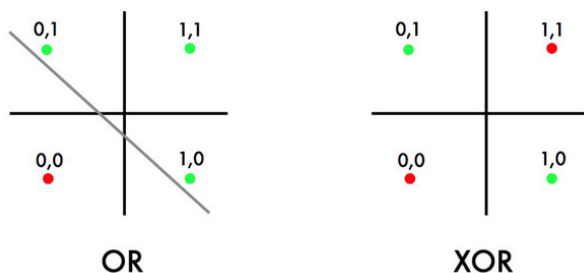


Figure 2.3: Outcome (green for 1 and red for 0) for (left) the OR and (right) XOR operation. The obtained outcomes show that the OR operation is linearly separable, whereas the XOR operation is not.

Approximately between 1974 to 1993, the inability of the perceptron and other algorithms to solve critical problems, combined with an excessive media hype, triggered a mistrust against machine learning research. Due to this mistrust, the funding for projects related to machine learning decreased dramatically. During this period, which witnessed several so-called AI winters, many research programs related to AI were abandoned [34].

Years after the work of Rosenblatt [171], backpropagation was discovered as a method that allows training multi-layered neural networks.¹ This discovery allowed neural networks to realize their true potential and solve many complex problems in the following years. Nevertheless, after the discovery of backpropagation, neural networks did not immediately dominate machine learning research due to a large number of limitations. It was indeed possible to train multi-layered neural networks at the time, but throughout the 1990s, neural networks were seen as data- and compute-hungry methods that could only solve toy problems.

Although the work of LeCun et al. [113] was one of the first studies to reveal the capability of neural networks in solving real-world problems, neural networks did not gain popularity until the seminal work of Krizhevsky et al. [109] in 2012. Arguments against the usage of neural networks (NNs) throughout the 1990s were that:

- NNs require a large number of samples to train,

¹Since it is not exactly clear who invented backpropagation for the first time and since this is a heated discussion in the field of machine learning in recent years, we refer the interested reader to the blog of Schmidhuber [180] for a detailed discussion regarding the invention of backpropagation.

- NN training is computationally expensive, and
- NN optimization is non-convex, making it hard, if not impossible, to prove certain properties regarding learning behavior.

To make things worse for NNs, in the mid 1990s, Support Vector Machines (SVMs) were invented, and their convex optimization approach was seen as superior to NNs on all fronts [33]. In fact, until recent years, techniques that required non-convex optimization were deemed undesired and unreliable.²

As previously mentioned, neural networks, and convolutional neural networks in particular, only gained in research popularity in the field of machine learning after the work of Krizhevsky et al. [109], which outperformed the state-of-the-art on the ImageNet Large-scale Visual Recognition Challenge (ILSVRC) [175] by a large margin in 2012. After 2012, convolutional neural networks dominated ILSVRC with now-famous architectures like VGG [189], Inception-Net [202, 203], and ResNet [75]. Indeed, the rise of the popularity of NNs started within the image domain. Other fields, however, were quick to adopt this technology to solve a wide variety of problems. As it currently stands, deep neural networks have been employed to solve a wide range of prediction problems concerning speech, language, and molecular biology [73, 83, 238].

2.1.1 Overview of neural networks

In Figure 2.4, we provide a schematic outline of a classification neural network, $g(\theta, \mathbf{x}) : \mathbb{R}^Q \rightarrow \mathbb{R}^M$, with a single hidden layer that maps the input $\mathbf{x} \in \mathbb{R}^Q$ to the real-valued output logits $\tilde{\mathbf{y}} \in \mathbb{R}^M$. In this neural network, weights ($\mathbf{w}_{\{1,2\}}$) and biases ($\mathbf{b}_{\{1,2\}}$) are said to be *trainable parameters* of the model. We will explain the different types of trainable parameters available in neural networks in Section 2.1.2.

Given this neural network, a forward-pass (i.e., a prediction) with an input \mathbf{x} can be described as follows:

$$\tilde{\mathbf{y}} = g(\theta, \mathbf{x}) = f(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \quad (2.2)$$

where f represents the selected activation function, employed to introduce non-linearity, and where θ represents the parameters and the

²See the presentation of Yann LeCun at Neural Information Processing Systems (2007) with the title: “Who is afraid of non-convex functions?”

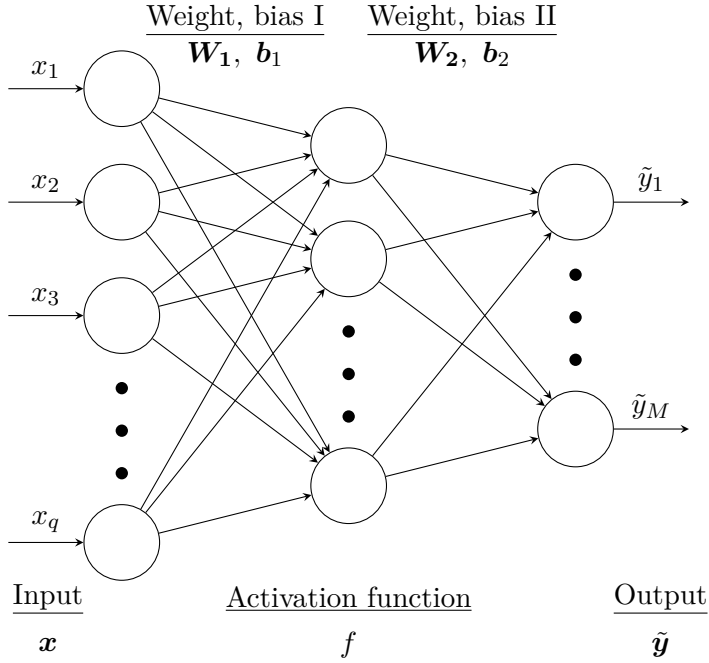


Figure 2.4: A visual illustration of a neural network with a single hidden layer. For visual clarity, the bias terms are excluded from this illustration.

structure of the model. In the absence of f , it is trivial to see the collapse of weights into a single matrix, which results in the whole model acting as a linear transformation. In Section 2.1.3, we will cover a number of activation functions that are relevant to this dissertation.

Backpropagation, as popularized in the work of Rumelhart et al. [174], can be used to train neural networks. This is done by calculating the gradient of the error function with respect to the weights of the model. For example, backpropagation for optimizing the first weight of the neural network given in Equation 2.2 can be written as follows:

$$\frac{\partial L(\mathbf{y}, \tilde{\mathbf{y}})}{\partial \mathbf{W}_1} = \frac{\partial L(\mathbf{y}, \tilde{\mathbf{y}})}{\partial \tilde{\mathbf{y}}} \frac{\partial \tilde{\mathbf{y}}}{\partial f(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)} \frac{\partial f(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)}{\partial \mathbf{W}_1}, \quad (2.3)$$

where $L(\cdot, \cdot)$ represents an arbitrary loss function for measuring the error obtained for the prediction $\tilde{\mathbf{y}}$, as opposed to the correct classification \mathbf{y} represented in a one-hot-encoded way (i.e., $\mathbf{y}_c = 1, \mathbf{y}_{i \neq c} = 0$). Although a large number of loss functions are available for training neural networks, only a few of them are commonly used, as discussed

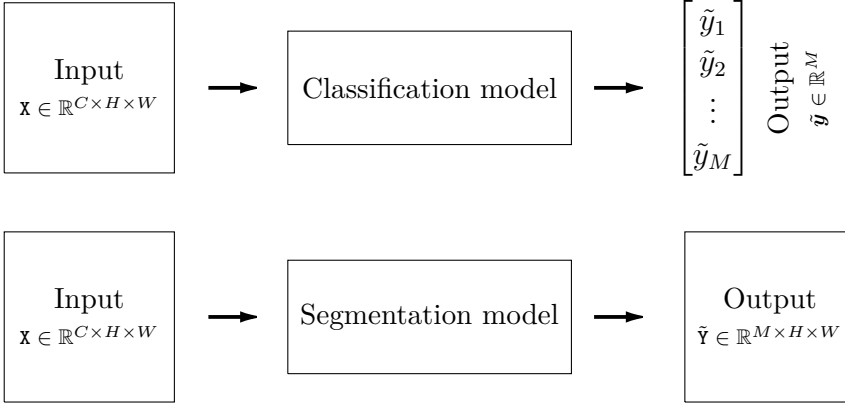


Figure 2.5: Illustration of the input-output relationship for (top) classification and (bottom) segmentation models.

in more detail in Section 2.1.4.

In simple terms, backpropagation is about updating the weights with the gradient of the loss with respect to each weight, hereby leveraging the chain rule. Although the term “backpropagation” refers to the generation of gradients for the purpose of training neural networks, it does not explain how these gradients are used. We will detail a number of optimizers that leverage gradients in different ways in Section 2.1.5.

In the example shown in Figure 2.4, the output of the neural network $\tilde{\mathbf{y}} \in \mathbb{R}^M$ is a real-valued vector corresponding to M classes. Networks of this type are designed to solve classification problems, which aim at assigning the input to one of M categories. Although most of the neural networks that will be discussed in this dissertation are classification models, we will also discuss the prevalence of adversarial examples for segmentation models. Different from classification models, segmentation models aim at producing a classification mask, with classification happening for each individual pixel.³ Specifically, each pixel of the input is classified into one of M classes. Given an input image $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$, we will assume that a segmentation model produces an output tensor $\tilde{\mathbf{Y}} \in \mathbb{R}^{M \times H \times W}$. In Section 2.3, we will further clarify the notation used throughout the dissertation.

In the previous sections, we briefly went over the history and the

³The output produced by some segmentation models can be different from their input, but for the cases studied in this dissertation, the height and the width of the input images for the segmentation models match the height and the width of the prediction masks.

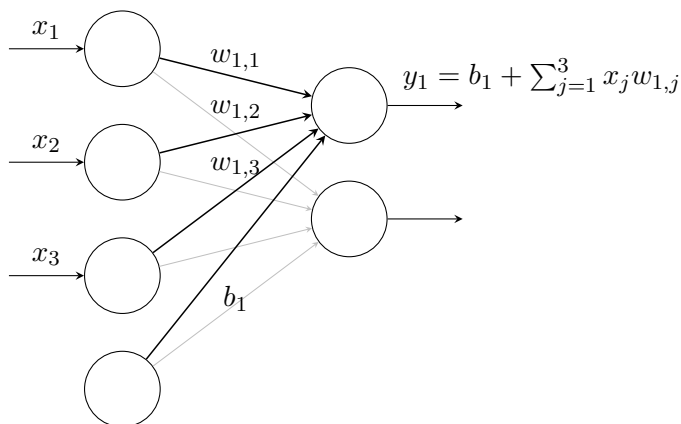


Figure 2.6: A visual representation of a fully connected layer.

structure of neural networks. In what follows, we will cover a number of special but relevant cases of trainable weights, activation functions, losses, and optimizers.

2.1.2 Trainable weights

The trainable weights of Rosenblatt’s perceptron were envisioned as vectors, to which a dot product is applied to solve a binary classification problem, as depicted in Equation 2.1. Modern neural networks were built upon this idea, but use matrices as weights in order to perform a linear transformation [174], allowing the network to solve a larger variety of problems. Such layers that are used to transform every element of an input vector into an output are called fully connected layers:

$$\mathbf{y} = \left[b_k + \sum_{j=1}^J x_j w_{k,j} \right]_{k \in \{1, \dots, K\}}, \quad (2.4)$$

A visual representation of a fully connected layer that maps the vector $\mathbf{x} \in \mathbb{R}^3$ to $\mathbf{y} \in \mathbb{R}^2$ by evaluating $\mathbf{y} = \mathbf{x} \mathbf{W} + \mathbf{b}$ is given in Figure 2.6. When the layer under discussion is not the final layer of the model, after the utilization of a fully connected layer, an activation function is often applied to introduce non-linearity. One of the problems associated with fully connected layers is employing such layers with large inputs. Doing so not only increases the number of trainable parameters, but also the amount of data required to train such layers. Until

now, training large neural networks composed of only fully connected layers as trainable weights was not feasible due to a lack of training data, among other factors. However, recent research efforts suggest that neural networks with such architectures are indeed trainable and that they are able to achieve (beyond) state-of-the-art results, given the massive amounts of data that are available these days [40, 205].

To mitigate the shortcomings of fully connected layers and to extract better features that are invariant (to the extent possible) to shift, scale, and distortion, LeCun et al. [114] proposed the use of convolutional layers, where such layers adopt weight sharing and local-receptive fields. Different from fully connected layers where each neuron is connected to each other, convolutional layers introduce an inductive bias of locality, where regions that are close to each other are assumed to contain more relevant information than regions that are far away. Most of the architectures that employ such layers use convolutional layers to “extract features” from the input and eventually use these extracted features in a fully connected layer to perform classification [109, 189].

Discrete convolution for matrices is defined as follows:

$$(\mathbf{A} * \mathbf{B})_{i,j} = \sum_m \sum_n \mathbf{A}_{i-m,j-n} \mathbf{B}_{m,n}, \quad (2.5)$$

where m denotes the size of the kernel. Although the term “convolution” is used in convolutional layers to denote the operation performed, most of the automatic differentiation libraries that offer support for convolutional layers perform cross-correlation instead of convolution, making it possible to save computational time on flipping the kernel compared to the image. The difference between a convolution and a cross-correlation simply amounts to signature changes:

$$(\mathbf{A} \star \mathbf{B})_{i,j} = \sum_m \sum_n \mathbf{A}_{i+m,j+n} \mathbf{B}_{m,n}. \quad (2.6)$$

A visual representation of a convolutional operation with a 3×3 kernel and a visual representation of feature generation through the usage of multiple kernels is provided in Figure 2.7 and Figure 2.8, respectively.

Even though the idea of convolutional layers was proposed in 1998 by LeCun et al. [114], the widespread use of convolutional layers only started after the pioneering work of Krizhevsky et al. [109], with this work winning the ILSVCR-2012 competition by a large margin. Architectures comprised of convolutional layers have dominated the field

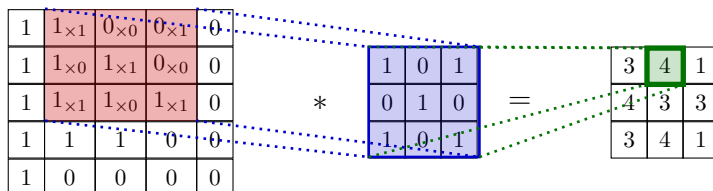


Figure 2.7: An illustration of a convolutional operation using a single 3×3 kernel, as performed in convolutional layers.

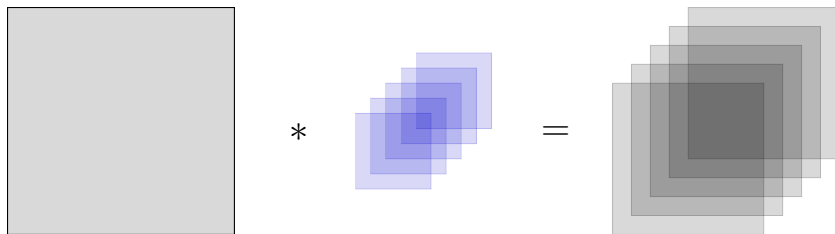


Figure 2.8: A visual representation of feature extraction through the usage of multiple convolutional kernels.

of computer vision ever since, with novel architectures being proposed one after another on a yearly basis [75, 189, 202]. Although the architecture introduced in Krizhevsky et al. [109] was designed to solve a visual classification problem, architectures using convolutional layers have been widely adopted in fields other than computer vision to solve a wide range of problems [73, 164, 238].

At the time of their invention (and even now), convolutional layers were seen as a special (narrower) cases of fully connected layers that are helpful to extract features from data. However, the aforementioned statement is not true. In fact, convolutional layers characterize a broader representation of trainable weights since every fully connected layer can be represented as a convolutional layer, but not vice-versa.⁴

Although convolutional layers and the features produced through the usage of such layers are represented as tensors in automatic differentiation libraries such as PyTorch [160] or Tensorflow [1], the application of a convolutional layer is performed by making use of simple matrix multiplications, which makes convolutional layers no different than fully connected layers when it comes to inference (forward pass) or er-

⁴See the work of Springenberg et al. [193], as well as the following note of Yann LeCun on the topic of fully connected layers in convolutional neural networks: <https://www.facebook.com/yann.lecun/posts/10152820758292143>.

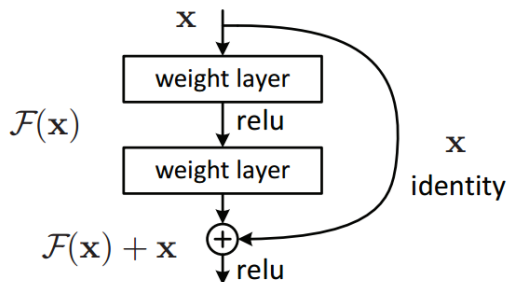


Figure 2.9: Visualization of a residual layer with a skip-connection.

ror backpropagation (backward pass). In order to seamlessly perform both the forward- and backward-pass compatible with other layers, tensor representations are converted to matrix representations with the help of kernel functions.⁵

While the usage of convolutional layers drastically improved the performance of neural networks on many tasks, training very deep neural networks (50+ layers) still remained a challenge to solve. One of the influential ideas that enabled the training of very deep neural networks is the use of skip-connections (shortcut connections) between layers [75]. This type of connection feeds the output of a layer as an input to subsequent layers, instead of only the next one. In particular, adding skip connections avoids the need for gradients to pass through every layer, thus partially mitigating the vanishing gradient problem. The most famous block of trainable weights that employs skip-connections is the residual block introduced in [75]. As shown in Figure 2.9, the output of a residual block with two trainable weights \mathbf{W}_1 and \mathbf{W}_2 can be represented as $\mathbf{Y} = f(f(\mathbf{x}\mathbf{W}_1)\mathbf{W}_2 + \mathbf{x})$, where f represents an activation function that can be selected. Skip-connections, as well as residual blocks, became a staple feature of many deep neural architectures proposed after the successful introduction of ResNets, which are built entirely using residual blocks.

Yet another type of trainable block that will be relevant in the upcoming chapters is a recurrent layer, facilitating temporal dynamic behavior [43]. Preserving information derived from previous inputs (memory), the hidden state of a recurrent layer is determined as fol-

⁵See, for example, *torch.nn.Fold* and *torch.nn.Unfold* in the PyTorch library or *im2col* and *col2im* in MATLAB.

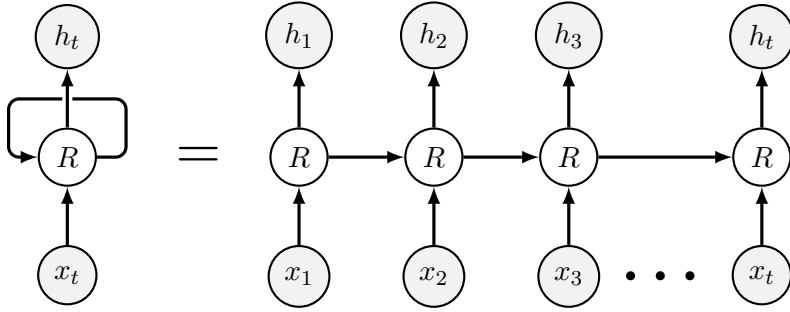


Figure 2.10: A recurrent layer is visualized as a directed graph, including its hidden states and the inputs provided at each step.

lows:

$$\mathbf{h}^{(i)} = \alpha(\mathbf{W}_l^{(i)} \mathbf{x}^{(i)} + \mathbf{W}_h^{(i)} \mathbf{h}^{(i-1)} + \mathbf{b}_l^{(i)} + \mathbf{b}_h^{(i)}). \quad (2.7)$$

The hidden state at the $(i - 1)$ th step is taken as a parameter, next to the actual input $\mathbf{x}^{(i)}$, to determine the state at the i th step. The final output at the i th iteration can be determined through the usage of a fully connected layer $\mathbf{y}^{(i)} = \alpha(\mathbf{W}_f^{(i)} \mathbf{h}^{(i)})$.

Two notorious problems associated with recurrent neural networks (RNNs) are exploding gradients and vanishing gradients, which we will describe in the upcoming section. To alleviate these two problems, up to a certain degree, Hochreiter and Schmidhuber [83] introduced a new type of recurrent layer called long short-term memory (LSTM). Unfortunately, the solution for the two aforementioned gradient-backflow problems does not come for free. Indeed, LSTMs need to perform multiple intermediate calculations before determining the hidden state. The aforementioned calculations can be represented as $\boldsymbol{\iota}_{k \in \{f, u, o, c\}}$, where f , u , o , and c denote the activation vectors of the forget gate, the update gate, the output gate, and the input gate, respectively. These gates are calculated as follows:

$$\boldsymbol{\iota}_{k \in \{f, u, o, c\}}^{(i)} = \alpha(\mathbf{W}_k^{(i)} \mathbf{x}^{(i)} + \mathbf{W}_{hk}^{(i)} \mathbf{h}^{(i-1)} + \mathbf{b}_k^{(i)} + \mathbf{b}_{hk}^{(i)}). \quad (2.8)$$

Based on these activation vectors, the cell state $\tilde{\mathbf{c}}$, which facilitates the concept of long-term memory, and the hidden state \mathbf{h} , which determines the kind of information carried over to the next sequence, are

calculated as follows:

$$\tilde{\mathbf{c}}^{(i)} = \mathbf{f}^{(i)} \odot \tilde{\mathbf{c}}^{(i-1)} + \mathbf{u}^{(i)} \odot \mathbf{c}^{(i)}, \quad (2.9)$$

$$\mathbf{h}^{(i)} = \mathbf{o}^{(i)} \odot \alpha(\tilde{\mathbf{c}}^{(i)}), \quad (2.10)$$

where \odot denotes the Hadamard multiplication. Similar to RNNs, the hidden states of LSTMs are used in conjunction with the next input sequence. An overall visualization of an RNN, using a directed graph representation, can be found in Figure 2.10. Depending on the problem at hand, the configuration of both RNNs and LSTMs can be designed to manage a one-to-one, one-to-many, many-to-one, or many-to-many relationship between the input and the output.

2.1.3 Activation functions

As briefly mentioned in Section 2.1.1, activation functions are employed to introduce non-linearities, thus making it possible to address non-linear problems. These functions, which are applied in an element-wise manner, map a real-valued scalar input $x \in \mathbb{R}$ to a real-valued scalar output. Although a large number of activation functions are available, we will only cover the activation functions shown in Figure 2.11, as these activation functions were used by the different neural networks discussed in this dissertation.

The sigmoid activation function, also called the logistic function, is a monotonically increasing function that is bounded between 0 and 1. It was one of the first activation functions used to train neural networks [174]. The sigmoid function and its partial derivative with respect to x are as follows:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \quad (2.11)$$

$$\frac{\partial}{\partial x} \text{sigmoid}(x) = \text{sigmoid}(x)(1 - \text{sigmoid}(x)). \quad (2.12)$$

The sigmoid function is indeed helpful for introducing non-linearities in neural networks, enabling neural networks to solve non-linear problems. However, the adoption of the sigmoid function by neural networks also created a number of problems, with the most notorious problem being the vanishing gradient problem: the backpropagation signal described in Equation 2.3 becomes extremely small, eventually approximating 0, thus no longer resulting in meaningful weight updates.

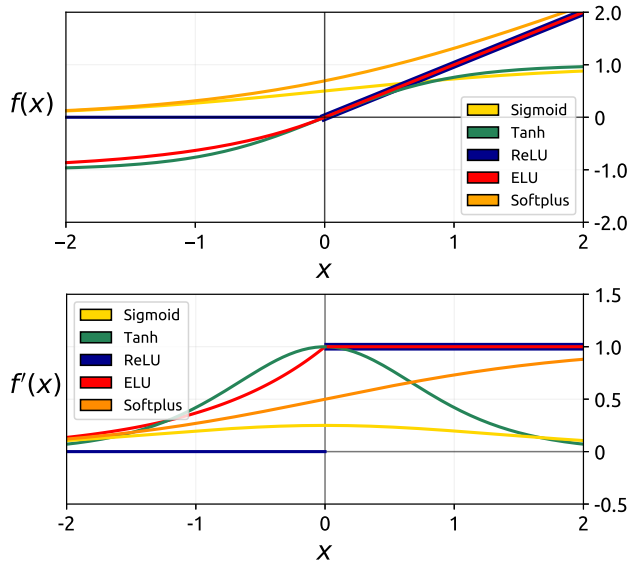


Figure 2.11: (top) Activation functions and (bottom) their first order derivatives.

Research efforts to prevent the vanishing gradient problem from happening led to the usage of another activation function that is similar to the sigmoid: hyperbolic tangent (\tanh). Like the sigmoid function, the hyperbolic tangent function is also a monotonically increasing function, but the function is bounded between -1 and 1 , achieving a steady state at $x = 0$. The hyperbolic tangent function and its partial derivative with respect to x are given as follows:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \frac{\partial}{\partial x} \tanh(x) = 1 - (\tanh(x))^2. \quad (2.13)$$

As shown in Figure 2.11, the backpropagation signal with \tanh is larger than with sigmoid when the input is close to zero. This feature of \tanh is said to alleviate some of the problems associated with vanishing gradients. In reality, \tanh is used together with other approaches, such as unsupervised pre-training [11], in order to mitigate the problem of vanishing gradients.

Sigmoid and \tanh were the two most prominently used activation functions until the mid 2010s, when Glorot et al. [58] proposed the rectifier activation function, with this activation function revolutionizing neural network training. A rectifier linear unit, commonly known as ReLU, is an activation function that is dramatically different from

both sigmoid and tanh. Glorot et al. [58] noted that the biological neuron firing process does not happen until a certain threshold is exceeded (i.e., $f(x) = 0, x < \kappa$), a behavior that is vastly different from the behavior exhibited by the sigmoid and tanh activation functions. Inspired by this observation, they proposed the ReLU activation function, which is defined as follows:

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise,} \end{cases} \quad \frac{\partial}{\partial x} \text{ReLU}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2.14)$$

Two major benefits of using ReLU over the other two activation functions we have described earlier are (1) the reduced likelihood of having vanishing gradients (the gradient is always one for inputs greater than zero) and (2) an increased sparsity (the gradient is zero for inputs less than zero). Unfortunately, ReLU brought its own problem, namely the problem of exploding gradients [165]. Different from the vanishing gradient problem, the exploding gradient problem refers to the phenomenon where the gradient accumulates to large values, resulting in large weight updates that hinder model training.

After the introduction of ReLU by Glorot et al. [58], a large number of rectifier activation functions with minor modifications to the original one have been proposed [59, 105, 125]. Of those, the Exponential Linear Unit (ELU) activation function is the one that will be relevant in later chapters of this dissertation [30]. ELU is defined as follows:

$$\text{ELU}(x, \alpha) = \begin{cases} x & \text{if } x > 0, \\ \alpha(e^x - 1) & \text{otherwise,} \end{cases} \quad (2.15)$$

$$\frac{\partial}{\partial x} \text{ELU}(x, \alpha) = \begin{cases} 1 & \text{if } x > 0, \\ \alpha e^x & \text{otherwise.} \end{cases} \quad (2.16)$$

One advantage of ELU over ReLU is having a gradient signal when $x < 0$, thus pushing the mean unit activation closer to zero. Compared to ReLUs, ELUs were also shown to speed up training [30]. Note that both ReLU and ELU are non-differentiable when $x = 0$.

The last activation function we will discuss is Softplus [42], which can be viewed as a smoothed out version of ReLU. Softplus and its

first order derivative are defined as follows:

$$\text{softplus}(x) = \ln(1 + e^x), \quad \frac{\partial}{\partial x} \text{softplus}(x) = \frac{1}{1 + e^{-x}}. \quad (2.17)$$

2.1.4 Loss functions

In the context of neural networks, loss functions are the type of functions that allow quantifying the prediction error. The aim of training is to minimize this prediction error, which is a scalar value measured using the selected loss function. In what follows, we only discuss the loss functions that are employed to train the classification and segmentation models used by the research efforts presented further in this dissertation.

Before discussing loss functions, a relevant function that also needs to be explained is the softmax function [18], which is often employed to convert predictions made by neural networks (logits) into a probabilistic form. The softmax function, which is a generalized logistic function, is defined as follows:

$$P(\mathbf{z}) = \left[\frac{e^{z_c}}{\sum_{m=1}^M e^{z_m}} \right]_{c \in \{1, \dots, M\}}, \quad (2.18)$$

where $\dim(\mathbf{z}) = \dim(P(\mathbf{z}))$, but where $\mathbf{z} \in \mathbb{R}^M$ and where $P(\mathbf{z}) \in [0, 1]^M$, with $\sum P(\mathbf{z}) = 1$. The aforementioned usage of the softmax function for classification models is slightly modified for segmentation models in order to accommodate for additional dimensions. Since the prediction produced by a segmentation model is given in the form of $\mathbf{Z} \in \mathbb{R}^{M \times H \times W}$, the softmax is applied for each location of H and W over the first dimension as follows:

$$P(\mathbf{Z}) = [P(\mathbf{z}_{i,j})]_{i \in \{1, \dots, H\}, j \in \{1, \dots, W\}}. \quad (2.19)$$

More-often-than-not, classification and segmentation losses employ the softmax function in conjunction with other techniques to measure the uncertainty associated with the prediction.

In statistical analysis, one of the methods to measure the uncertainty associated with a probabilistic prediction is the usage of the negative logarithmic likelihood (NLL) [50]. Combining NLL with the softmax output of a prediction, we obtain the cross-entropy (CE) loss: the measure of the difference between two probability distributions for a given event. Given the prediction logits $\tilde{\mathbf{y}}$ for a data point and the

correct class c associated with it, CE is defined as follows:

$$J(\tilde{\mathbf{y}}, c) = -\log \left(\frac{e^{\tilde{y}_c}}{\sum_{m=1}^M e^{\tilde{y}_m}} \right) = -y_c + \log \left(\sum_{m=1}^M e^{\tilde{y}_m} \right). \quad (2.20)$$

The cross-entropy loss remains the most used loss function for training neural networks [63]. The popularity of cross entropy comes from its preferred behavior when training models. Specifically, it aims at maximizing the prediction likelihood of the correct class while minimizing the prediction likelihood of all other classes. This behavior becomes obvious when taking its partial derivative with respect to the weights of the model:

$$\frac{\partial}{\partial \theta} J(\tilde{\mathbf{y}}, c) = -\frac{\partial}{\partial \theta} \tilde{y}_c + \frac{\sum_{m=1}^M e^{\tilde{y}_m} \frac{\partial}{\partial \theta} \tilde{y}_m}{\sum_{m=1}^M e^{\tilde{y}_m}}. \quad (2.21)$$

As can be seen, the first term aims at maximizing the target class, while the second term aims at minimizing the other classes.⁶

Extending CE loss for segmentation tasks is fairly straightforward. Since each pixel is considered a classification problem on its own, we can formulate CE loss for segmentation problems as follows:

$$J_s(\tilde{\mathbf{Y}}, c) = \frac{1}{H} \frac{1}{W} \sum_{i=1}^H \sum_{j=1}^W J(\tilde{\mathbf{y}}_{i,j}, c_{i,j}). \quad (2.22)$$

Although CE loss is popular among classification problems, training segmentation models with CE loss is known to be challenging when the size of the object of interest is small in training images. In such cases, models trained with CE loss often predict images as the largest class (which often corresponds to the background). One way to mitigate this is to assign class weights to the cross-entropy loss as follows:

$$J_s(\tilde{\mathbf{Y}}, \mathbf{c}, \boldsymbol{\alpha}) = \frac{1}{H} \frac{1}{W} \sum_{i=1}^H \sum_{j=1}^W \alpha_c J(\tilde{\mathbf{y}}_{i,j}, c_{i,j}), \quad (2.23)$$

where the multiplier $\alpha_c \in \mathbb{R}$ is assigned to each class before training in order to overcome class imbalance.

It goes without saying that the types of losses discussed above are

⁶The second term also contains a small minimization for the target class, but with a much smaller multiplier, resulting in an overall maximization of the target class.

only a small portion of what is available in the field and that class weighing is not the only method that can be used to deal with the notorious class imbalance problem [118, 132]. However, such approaches will not be relevant in the upcoming chapters, making a more detailed discussion of these approaches thus out of scope for this dissertation.

2.1.5 Optimizers

In Equation 2.3, we showed how the gradient is generated with back-propagation and stated that the usage of this gradient depends on the selected optimizer. Indeed, given an optimizer, the general idea of training a neural network is to solve the following unconstrained minimization problem:

$$\underset{\theta}{\text{minimize}} \ ||g(\theta, \mathbf{x}) - \mathbf{y}||, \forall (\mathbf{x}, \mathbf{y}) \in \mathfrak{D}_t, \quad (2.24)$$

where θ represents the parameters of the neural network model, $g(\theta, \mathbf{x})$ denotes the prediction made by the model, and \mathfrak{D}_t represents the training data. This non-convex optimization problem can be solved by making use of the well-known method of gradient descent (GD), where the gradient is generated at once for all data points in the training set and used iteratively to train the model as follows:

$$\theta^{(i+1)} = \theta^{(i)} - \beta \nabla_{\theta^{(i)}} L(\tilde{\mathbf{Y}}, \mathbf{Y}), \quad (2.25)$$

$$\tilde{\mathbf{Y}} = \{g(\theta^{(i)}, \mathbf{x}), \forall \mathbf{x} \in \mathfrak{D}_t\}, \quad (2.26)$$

$$\mathbf{Y} = \{\mathbf{y}, \forall \mathbf{y} \in \mathfrak{D}_t\}, \quad (2.27)$$

where $\nabla_{\theta^{(i)}}$ represents the gradient w.r.t the trainable parameters, and where $\tilde{\mathbf{Y}}$ and \mathbf{Y} denote a model prediction and a correct classification, respectively. The gradient update is then multiplied with a scalar β . This approach, although technically possible, is not feasible for training neural networks with large datasets unless one possesses an abundance of memory to store the gradient of all data points. Moreover, it was also discovered that using all the available training data at once may actually not be a great idea, since it increases the likelihood of getting stuck in a local minimum, hindering model training [99, 169].

Stochastic gradient descent (SGD), a slightly modified version of GD, refers to the idea of selecting data points randomly and updating the parameters of the model with the gradient of a fixed amount of data that is smaller than the whole dataset. SGD used with more than a single data point at a time is also referred to as batch (or mini-batch)

gradient descent. We define SGD for a single data point as follows:

$$\theta^{(i+1)} = \theta^{(i)} - \beta \nabla_{\theta^{(i)}} L(\tilde{\mathbf{y}}, \mathbf{y}), \quad (2.28)$$

where this gradient is multiplied with an appropriate scalar β . This approach was shown to perform better than the use of GD for non-convex optimization problems, also requiring less resources [99, 169].

Another important feature for optimizers that will be relevant in this dissertation is momentum, which refers to the idea of accumulating the gradient and using it to accelerate the speed of learning [166]. Gradient updates with momentum can be represented as follows:

$$\theta^{(i+1)} = \theta^{(i)} - \beta o^{(i)}, \quad (2.29)$$

$$o^{(i)} = \tau o^{(i-1)} + \lambda \nabla_{\theta^{(i)}} L(\tilde{\mathbf{y}}, \mathbf{y}). \quad (2.30)$$

In Equation 2.30, the gradient is accumulated into o based on the multipliers τ and λ , with these multipliers determining the ratio of past accumulation (τ) to the gradient at the current step (λ).

A number of optimizers that have been proposed after SGD, for example, AdaGrad, RMSProp, Adam, AdaMax, and Nadam, aimed at addressing a number of weaknesses of SGD, either speeding up the training or improving the generalization [41, 103, 204]. However, SGD still remains one of the most prominently used optimizers when training neural networks, with technical reports suggesting that these novel optimizers often fail to find models that generalize as well as SGD [216]. Indeed, many neural networks that achieve state-of-the-art results are still trained with SGD [75, 189]. Apart from SGD, we also employed the Adam optimizer in later chapters of this dissertation for training a number of neural network models, with Adam using adaptive learning rates for each parameter. Furthermore, Adam stores an exponentially decaying average of past squared gradients, resulting in Adam often converging to the solution faster than SGD. For more details, we refer the interested reader to Kingma and Ba [103].

2.2 Training and error quantification

The main problem with training neural networks, or machine learning models in general, is the problem of generalization: trained models should perform well on unseen data. This task is easier said than done, since the generalization of a model depends on many factors,

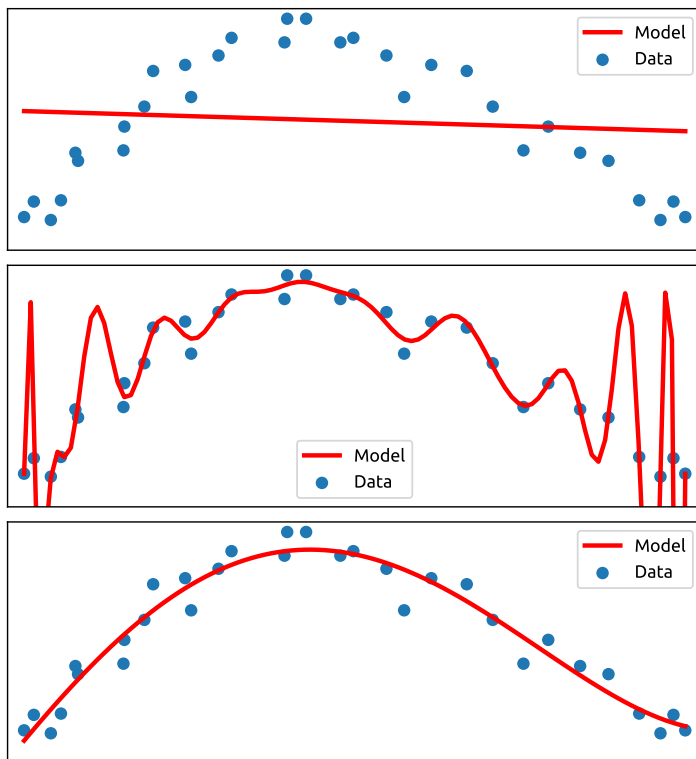


Figure 2.12: Illustrations of (top) an underfit, (middle) an overfit, and (bottom) a good fit (i.e., a robust model).

with the two most important ones being (1) the complexity of the model and (2) the training routine.

Assuming a suitable training routine that is capable of correctly training the model at hand, if we employ a model that is not sophisticated enough to capture the complexity of the data, the training routine may result in what is called an underfitted model. On the other hand, if we employ a model that is too complex for the data at hand, we might end up with a model that is able to exactly fit all the observations without looking into the shape or context, and where such models are called overfitted models. For example, if we attempt to solve the curve-fitting problem given in Figure 2.12 with a linear model, the model fails to capture the essence of the data (i.e., underfitting). On the other hand, if we employ a model that is capable of utilizing a polynomial of the 26th degree, we end up with a model that is too complex for these data (i.e., overfitting). Both of the aforementioned cases are undesirable, since the first one fails to capture

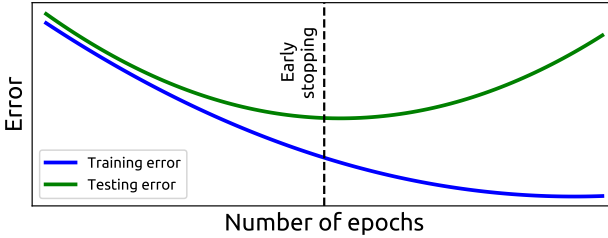


Figure 2.13: Illustration of early stopping with respect to the training duration (i.e., epochs) and the errors obtained for training and testing data.

the essence of the data and the second one fails to generalize. What is desired in machine learning is to produce a model that is capable of learning the context and the shape from the observations without exactly fitting the seen data.

Although the aforementioned problem of underfitting with respect to model selection was a serious problem in the past, with current neural networks, it ceased to exist, since neural networks are often more-than-capable enough to memorize an entire training dataset. This statement even holds true when datasets contain millions of images [75, 109]. As such, the common problem faced when training neural networks is not the problem of underfitting but the problem of overfitting. In fact, overfitting is such a big problem for neural networks that many research efforts were spent on finding methods of regularization to prevent overfitting from occurring [178, 194].

Even with the incorporation of a number of regularization methods, neural networks can still easily memorize all of the training data if training is not stopped at a certain point [63]. The approach of stopping the training before it starts memorizing the training data is called “early stopping”. The duration of training, expressed using epochs and compared to training and testing error, is visualized in Figure 2.13. In the context of training, an epoch refers to a learning period in which the whole training dataset is taken into account (i.e., where every sample in the training dataset is seen by the model once). The error made by the model, on the other hand, can be measured using various error estimation metrics, including accuracy (i.e., the ratio of correct predictions to all predictions). In the upcoming chapters, we will introduce various error quantification methods.

In the next section, we give an overview of the notation that will be used throughout this dissertation.

2.3 Machine learning notation

An image \mathbf{X} is technically represented as a tensor: $\mathbf{X} = [\mathbf{X}_1 \dots \mathbf{X}_C] \in \mathbb{R}^{C \times H \times W}$, where $\mathbf{X}_i \in \mathbb{R}^{H \times W}$ denotes the i th channel, represented as a matrix, C the number of channels, H the height of the input, and W the width of the input. For example, in the case of an RGB image, $C = 3$, with each matrix in \mathbf{X} representing a color channel. However, as explained in Section 2.1, these tensors are converted to vectors or matrices as inputs for the sake of computation. Although we will mostly employ the latter notation to represent inputs, there are a number of cases where the tensor notation simplifies the mathematical representation, allowing for an easier comprehension. As such, in the rest of this dissertation, we will employ the notation described below.

Let $\mathcal{D} = \{\mathbf{x}_n \mid n = 1, \dots, N\}$ be a dataset containing N data points. We denote by $\mathbf{x}_i \in \mathbb{R}^Q$ (with $Q = C \cdot H \cdot W$) and $\mathbf{X}_i \in \mathbb{R}^{C \times H \times W}$ the vector and tensor representation of the i th data point, respectively.

Classification – In the case of an M -class classification problem, the categorical association of the n th data point will be represented as $y_n \in \{1, \dots, M\}$. In this setting, let $g : \mathbb{R}^Q \rightarrow \mathbb{R}^M$ denote the forward-pass performed with a classification neural network, yielding a vector of real-valued logit scores for each category. Logits obtained through the usage of a neural network with parameters θ and a data point $\mathbf{x} \in \mathbb{R}^Q$ are represented by $\tilde{\mathbf{y}} = g(\theta, \mathbf{x})$, with $\tilde{\mathbf{y}} \in \mathbb{R}^M$. Using this notation, the index that contains the largest logit is used as the categorical classification for that data point: $G(\theta, \mathbf{x}) = \arg \max(\tilde{\mathbf{y}})$. If $G(\theta, \mathbf{x}_n) = y_n$, then the data point is correctly classified. In this scenario, the prediction confidence is analyzed with the help of the softmax function $P(g(\theta, \mathbf{x})) = P(\tilde{\mathbf{y}}) \in [0, 1]^M$. This function does not change the dimensions of the input but maps the logit vector onto a vector containing prediction likelihoods (often called probabilities) for each class, with the resulting likelihoods adding up to 1.

Segmentation – For segmentation problems, we will represent the input using the following tensor notation: $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$. In this setting, the categorical association of the n th data point will be represented as $\mathbf{Y}_n \in \{1, \dots, M\}^{H \times W}$. \mathbf{Y} , unless specified otherwise, has the same height and width as the tensor representation of the input image: the pixels of the input are thus matching with the pixels of the output, and where the classes are represented as natural numbers, starting from 1 and ending at M . Furthermore, we make use of $s : \mathbb{R}^{C \times H \times W} \rightarrow \mathbb{R}^{M \times H \times W}$ to denote a forward-pass performed by

a segmentation neural network, yielding a tensor of real-valued logit scores for each category and for each pixel. Logits obtained through the usage of a neural network with parameters θ will be represented by $\tilde{\mathbf{Y}} = s(\theta, \mathbf{X})$, $\tilde{\mathbf{Y}} \in \mathbb{R}^{M \times H \times W}$. Different from classification, in segmentation, the largest value in the first dimension for each location of H and W is assigned as the prediction: $\tilde{\mathbf{Y}} = S(\theta, \mathbf{X}) = \arg \max_M(\tilde{\mathbf{Y}})$. Moreover, the softmax function $P(\tilde{\mathbf{Y}}) \in [0, 1]^{M \times H \times W}$ is also applied for each index of H and W over the first dimension, thus resulting in a tensor of probabilities for each pixel, and where the sum over the first dimension (i.e., over the number of possible categories M) adds up to 1.

3

Adversarial attacks on deep neural networks

Thanks to recent advances in the field of deep neural networks, a wide range of problems that were once thought to be hard challenges found easy-to-adopt solutions [109, 212]. Indeed, many deep learning libraries now come with built-in solutions and pre-trained models, further increasing the adoption rate of such networks in the area of computer vision [1, 96, 160]. In spite of receiving a large amount of research attention, a number of fundamental flaws of DNNs still remain unsolved. One of those flaws is the vulnerability of DNNs to adversarial attacks, where small changes in the inputs may lead to large changes in the predictions [201]. In this chapter, we will analyze a number of popular adversarial attacks and the properties of the adversarial examples created with these attacks. Furthermore, we will investigate the impact of exercising adversarial perturbation on different image regions.

The content of this chapter is based on the following publications:

- Utku Ozbulak, Wesley De Neve, and Arnout Van Messem. Perturbation Analysis of Gradient-based Adversarial Attacks. *Pattern Recognition Letters, Elsevier*, 2020.
- Utku Ozbulak, Jonathan Peck, Wesley De Neve, Bart Goossens, Yvan Saeys, and Arnout Van Messem. Regional Image Perturbation Reduces L_p Norms of Adversarial Examples While Maintaining Model-to-model Transferability. *International Conference on Machine Learning (ICML), Workshop on Uncertainty & Robustness in Deep Learning (UDL)*, 2020.

The research efforts presented in this chapter resulted in the creation of the following software package:

- github.com/utkuozbulak/regional-adversarial-perturbation
A software package that contains the PyTorch implementation of the regional adversarial perturbation approach proposed in [151].

This chapter consists of eleven sections, with the first three sections providing a gentle introduction to various types of attacks against machine learning models, threat scenarios, and commonly used adversarial (evasion) attacks. In Section 3.4, we provide a number of hypotheses that investigate the existence of adversarial examples, following up with ways to measure the amount of perturbation (Section 3.5), constraints in the image domain (Section 3.6), and adversarial transferability (Section 3.7). Next, in Section 3.8, we discuss the properties of perturbations, using content taken from [150]. Finally, in Section 3.9 and Section 3.10, we discuss regions of importance for adversarial perturbations, leveraging content taken from [151]. In Section 3.11, we conclude this chapter with a summary of our contributions and a brief discussion of future work.

The content of Section 3.8, Section 3.9, and Section 3.10 is modified slightly compared to the content of the supporting published papers. In particular, we improved the pacing of the content in the aforementioned sections by adding and moving a number of descriptions. We also slightly changed the mathematical notation, so to be in line with the notation previously introduced in Chapter 2.

3.1 Introduction

Although the definition of adversarial attacks on machine learning systems is not clear, any attempt to mislead, deceive, or steal secrets of an automated system can, simply speaking, be considered an adversarial attack. This broad characterization captures the full scale of adversarial attacks, possibly involving cybersecurity aspects as well. More concisely, adversarial attacks can be understood as sets of techniques that force machine learning models into making mistakes or revealing information that is otherwise confidential. The terms “adversarial attacks” and “adversarial examples”, the deceptive inputs produced by adversarial attacks, gained popularity after the seminal work of Szegedy et al. [201]. However, the attempt to mislead an automated system has been a part of our lives ever since automated systems were deployed for the first time. For example, an attempt to evade an automated spam filter through the usage of particular words can

also be seen as an adversarial attack. Approaching the phenomenon of adversarial attacks from this angle, we can group adversarial attacks according to their purpose:

- **Data poisoning** – Given a machine learning system that is capable of learning from data, data poisoning is the act of contaminating the training data this system is trained with. Such attacks are performed in order to construct blind spots in decision-making systems by disrupting the model training procedure.
- **Model stealing** – Given a machine learning system that is already trained, a model stealing attack attempts to either extract the training data with which this system has been trained or to create a new model that is capable of mimicking the prediction effectiveness of the model under attack. Such attacks are often performed on proprietary systems in order to steal confidential training data or the underlying model.
- **Evasion attacks** – Consider an automated system that is designed to perform a task based on the input it receives. An evasion attack is the attempt to mislead this system in such a way that the outcome is in favor of the attacker. The case we gave above related to spam filter avoidance is an example of an evasion attack. This type of attack is by far the most-commonly analyzed and employed attack in the field of adversarial machine learning. We will also discuss and analyze evasion attacks through the remainder of this dissertation.

Even though the topic of adversarial attacks has been investigated mostly for DNNs, traditional machine learning systems such as decision trees, regression models, and even SVMs were also shown to be vulnerable to such attacks [14]. The reason for the renewed interest in this topic can be attributed to the super-human results obtained by DNNs in recent years [75, 202]. Following this trend, we will also investigate adversarial examples in the context of DNNs, where these malicious data points have been created with the purpose of either misleading classification models or segmentation models.

3.2 Threat scenarios

Although evasion attacks are simply described as misleading an automated system, the difficulty of attacks for each system will not be the

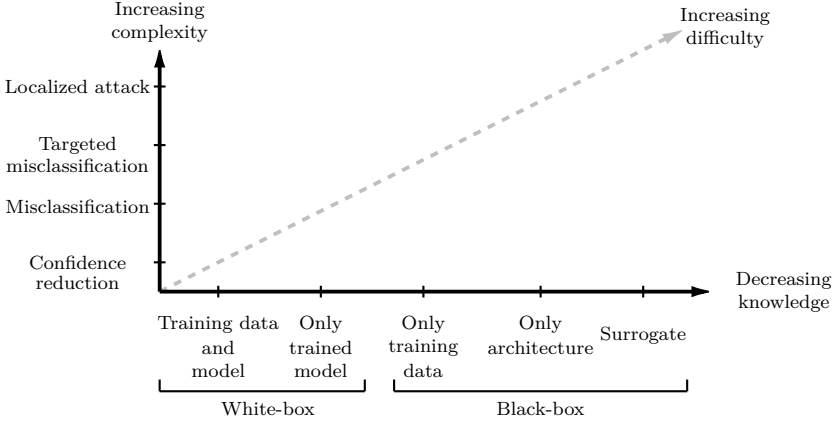


Figure 3.1: Threat configurations and their taxonomy for evasion adversarial attacks.

same. In this context, the two main aspects of a threat model are (1) the knowledge of the adversary about the underlying system and (2) the complexity of the attack. In line with the previous research in the field [156], multiple levels of (1) and (2) are shown in Figure 3.1. In terms of the amount of knowledge the adversary can make use of, we explain the different scenarios, from the most permissive scenario to the scenario that is the most limiting:

- **Training data and model**—The adversary has access to the model that is performing the automated task, as well as the data used to train this model.
- **Only trained model**—The adversary has access to the trained model that is performing the automated task, but not to the underlying training data.
- **Only training data**—The adversary has access to the training data that the underlying model has been trained with, allowing the adversary to leverage knowledge about the underlying distribution of the data.
- **Only architecture**—The adversary has access to the architecture that is performing the classification task, but not to the training data or the trained model. This means that the attack must be implemented using data that have not been seen by the model before.






	<p>Genuine image Prediction: Rooster Confidence: 96%</p>
	<p><u>Confidence reduction</u> Adversarial example Prediction: Rooster Confidence: 42% Perturbation: Global</p>
	<p><u>Untargeted misclassification</u> Adversarial example Attack target: Plane Prediction: Tiger Confidence: 67% Perturbation: Global</p>
	<p><u>Targeted misclassification</u> Adversarial example Attack target: Plane Prediction: Plane Confidence: 67% Perturbation: Global</p>
	<p><u>Localized attack</u> Adversarial example Attack target: Plane Prediction: Plane Confidence: 55% Perturbation: Regional</p>

Figure 3.2: A genuine image and its adversarial counterparts are provided for each threat taxonomy.

- **Surrogate** – The adversary has access to a model (i.e., a surrogate) that has been trained with similar data that the underlying system has been trained with. This allows the adversary to leverage knowledge about a model that has been trained on data that are similar in terms of distribution to the training data of the underlying system.

The attacks originating from the first two cases are usually referred to as *white-box* attacks, which means that the attacker has access to the underlying system, whereas the last three cases are referred to as *black-box* attacks, which means the attacker does not have access to the underlying system. Although the threat configuration seems to be

clear-cut based on the descriptions above, there are also threat scenarios that leverage information partially available about the underlying system, with such scenarios being referred to as *gray-box* attacks.

For the same threat configuration, and based on the knowledge the adversary has (as discussed above), attacks can be listed from easier to harder:

- **Confidence reduction** – Reduce the output confidence of the prediction.
- **Misclassification** – Change the prediction from a correct one to an (unspecified) incorrect one.
- **Targeted misclassification** – Force the prediction to become a specified class that is different from the correct one.
- **Targeted misclassification with a localized attack** – Force the prediction to become a specified class that is different from the correct one and, while doing so, limit the attack (i.e., the perturbation) to selected regions of the input.

In this dissertation, we work with various adversarial threat scenarios, which will be discussed in more detail later on.

3.3 Adversarial examples

Although there is no unified formal definition of adversarial examples, generally, adversarial examples are said to be a threat if they satisfy the following conditions: (i) indistinguishable perturbation, (ii) non-suspicious input, and (iii) forced misclassification. Based on these constraints, for the classification system described in Section 2.3, an ϵ -adversarial example, which is the most commonly used adversarial example, is defined as follows.

DEFINITION 3.1. (*ϵ -adversarial examples*) For a small perturbation Δ measured by an ℓ_p -norm, $\|\Delta\|_p < \epsilon$, $\hat{\mathbf{x}} = \mathbf{x} + \Delta \in \mathbb{R}^Q$ is an ϵ -adversarial example for a neural network with parameters θ , created through the usage of an unperturbed image \mathbf{x} , if $G(\theta, \mathbf{x}) = y$ and $G(\theta, \hat{\mathbf{x}}) \neq y$.

Although other types of adversarial examples with differing properties were proposed in recent literature [93], most of the research

in this field employs ϵ -adversarial examples for experimental procedures [206]. Following this trend, in the rest of this dissertation, we will also use this type of adversarial examples and furthermore, omit the ϵ from the term for simplicity.

Since the inception of adversarial attacks against neural networks, a wide range of attack methods have been proposed in the literature, with each attack method focusing on a different aspect of the adversarial optimization process. When it comes to creating adversarial examples, there are two approaches with regards to the specified category of misclassification. If an attack does not specify which category the produced adversarial example should be misclassified as, then it is called an untargeted attack. Conversely, if the attack targets a specific class for the misclassification, it is called a targeted adversarial attack.

In the seminal works of Tramer et al. [206], Athalye and Carlini [6], and Carlini and Wagner [23], the authors evaluated a large number of defenses proposed to detect or prevent adversarial attacks, observing that targeted adversarial attacks produce so-called *robust* adversarial examples that are harder to defend against. As such, our main focus is also going to be on such attacks. In what follows, we discuss a number of popular adversarial attacks in more detail.

3.3.1 L-BFGS attack

The constrained L-BFGS attack, as introduced by Szegedy et al. [201], is one of the first adversarial attacks to become popular. This attack aims at finding an adversarial example $\hat{\mathbf{x}}$ that is similar to the input \mathbf{x} under the ℓ_2 distance, but that gets assigned a different label by the model. Szegedy et al. [201] expressed this problem as follows:

$$\begin{aligned} & \text{minimize} && \beta \cdot \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 - f(\hat{\mathbf{x}}), \\ & \text{such that} && G(\theta, \mathbf{x}) \neq G(\theta, \hat{\mathbf{x}}). \end{aligned} \tag{3.1}$$

This minimization problem is repeatedly solved for multiple values of β , using bisection search to find optimal perturbations. To determine the perturbation multiplier, the attack under consideration simply uses the Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) [144]. A common choice for the loss function f , and one we will also follow in later chapters, is the cross-entropy loss.

3.3.2 Fast-gradient sign

Fast-Gradient Sign Method (FGS) is a single-step adversarial example generation technique proposed by Goodfellow et al. [62] and defined as follows:

$$\hat{\mathbf{x}} = \mathbf{x} - \alpha \operatorname{sign}(\nabla_{\mathbf{x}} J(g(\theta, \mathbf{x})_c), \quad (3.2)$$

where the perturbation generated with the signature of CE loss (will be referred to as CE-sign loss from this point onward) scaled with the multiplier α . Although FGS was never proposed as a way to generate robust adversarial examples, many past studies that introduced novel adversarial defenses employed this attack to show that the proposed defenses are robust against adversarial examples generated with it. Recent studies do not employ this attack and prefer to make use of the iterative attacks we describe below.

3.3.3 Iterative fast-gradient sign

Iterative fast-gradient sign (IFGS) [110], also referred to as the *basic iterative method* (BIM), finds its origin in FGS. The main objective of both FGS and IFGS is to generate adversarial examples quickly. As such, IFGS too updates an image using the CE-sign loss at each iteration as follows:

$$\hat{\mathbf{x}}^{(i+1)} = \operatorname{Clip}_{\hat{\mathbf{x}}, \epsilon} \left(\hat{\mathbf{x}}^{(i)} - \alpha \operatorname{sign}(\nabla_{\mathbf{x}} J(g(\theta, \hat{\mathbf{x}}^{(i)})_c) \right), \quad (3.3)$$

with $\hat{\mathbf{x}}^{(1)} = \mathbf{x}$. The parameter ϵ controls the maximally allowed amount of perturbation per pixel, α determines the amount of CE-generated perturbation added during each iteration, and the clipping function *Clip* ensures that the resulting adversarial example remains a valid image.

3.3.4 Projected gradient descent

Projected gradient descent (PGD) can be seen as a generalization of FGS and IFGS [126]. In particular, this attack aims at finding an adversarial example $\hat{\mathbf{x}}$ that satisfies $\|\hat{\mathbf{x}} - \mathbf{x}\|_{\infty} < \epsilon$, where the perturbation is defined within an ℓ_{∞} ball centered at \mathbf{x} with a radius ϵ . The

adversarial example is iteratively generated as follows:

$$\hat{\mathbf{x}}^{(n+1)} = \Pi_\epsilon \left(\hat{\mathbf{x}}^{(n)} - \alpha \operatorname{sign}(\nabla_x J(g(\theta, \hat{\mathbf{x}}^{(n)}_c)) \right), \quad (3.4)$$

with $\hat{\mathbf{x}}^{(1)} = \mathbf{x}$, and where the perturbation is calculated using the CE-sign loss, $\operatorname{sign}(\nabla_x J(g(\theta, \hat{\mathbf{x}}_c))$, originating from the target class c . In this setting, α controls the exercised perturbation at each iteration and Π_ϵ is a function that controls the ℓ_∞ limit imposed on the perturbation. When the threat setting is limited within an ℓ_∞ norm perturbation [206], PGD is often used over FGS and IFGS as a method to generate adversarial examples.

3.3.5 Carlini & Wagner’s attack

Carlini and Wagner [22] proposed a heavily optimized attack which produces strong adversarial examples that can bypass defense mechanisms easily. The ℓ_2 version of this attack, which was also used by Carlini and Wagner [23] to test the robustness of several defense mechanisms, is defined as follows:

$$\text{mimize} \quad \|\mathbf{x} - (\mathbf{x} + \Delta)\|_2^2 + f(\mathbf{x} + \Delta), \quad (3.5)$$

where this attack attempts to find a small perturbation Δ that is sufficient to change the prediction made by the model when it is added to the input, while keeping the ℓ_2 distance between the original image \mathbf{x} and the perturbed image $\hat{\mathbf{x}} = \mathbf{x} + \Delta$ minimal.

In their original work, Carlini and Wagner [22] discussed multiple loss functions (i.e., f) for this kind of attack. However, in later work on evaluating multiple defense mechanisms, they preferred to make use of the loss function that is constructed as follows:

$$f(\mathbf{x}) = \max \left(\max\{g(\theta, \mathbf{x})_i : i \neq c\} - g(\theta, \mathbf{x})_c, -\kappa \right), \quad (3.6)$$

where $\max\{g(\theta, \mathbf{x})_i : i \neq c\} - g(\theta, \mathbf{x})_c$ compares the predicted logit value of target class c with that of the next-most-likely class i . The constant κ can be used to adjust the *strength* of the produced adversarial example. Carlini and Wagner [22] refer to the loss function shown in Equation (3.6) as *logit loss*, which is also commonly known as *using the logits* to generate adversarial examples. However, throughout this dissertation, and for the sake of clarity, we will use (1) the term *logit loss* when maximizing a single target class using logits and

(2) the term *multi-target logit loss* (abbreviated as M-logit loss) when we refer to the loss function of CW (the latter focuses on two classes, namely the target class and the next-most-likely class).

3.3.6 Other adversarial attacks

Although the aforementioned adversarial attacks are the most commonly used attacks when it comes to evaluating adversarial defenses, they represent only a fraction of the adversarial attacks proposed in the scientific literature. In what follows, we briefly review a number of influential attacks that inspired future research.

The work of Nguyen et al. [143] was one of the first research efforts to reveal the vulnerability of DNNs against carefully crafted samples. They used evolutionary algorithms to craft adversarial examples that produce high-confidence predictions for pictures that vaguely resemble their prediction categories. Moreover, Nguyen et al. [143] also showed that it is possible to have high-confidence predictions for images that look like random noise, and where these images have been generated using gradient ascent.

Moosavi-Dezfooli et al. [135] devised Deep-Fool, an untargeted adversarial attack that takes advantage of decision boundaries between classes, creating adversarial examples with minimal perturbation that are classified into categories that are semantically similar to the category of their origin. Their later work revealed the existence of universal perturbations that transfer between tasks and fool multiple neural networks [136]. The work of Moosavi-Dezfooli et al. [135] was a cornerstone for the development of untargeted attacks, inspiring a large number of research efforts in this field.

Jacobsen et al. [93] laid out the properties of a subset of adversarial examples called invariance-based adversarial examples. Such adversarial examples have unique properties that make them indistinguishable from genuine inputs when analyzing their predictions. Their research effort is based on the approach proposed by Sabour et al. [176], where this approach is used later on for circumventing a particular type of adversarial defenses [24, 84].

Zhao et al. [234] and Shamsabadi et al. [185] focused on manipulating image colors in order to generate adversarial examples, suggesting unique approaches towards the development of adversarial attacks that are solely based on taking advantage of color channels. Furthermore, their research effort showed that the usage of the ℓ_p norm may not be suitable for the detection of adversarial perturbation.

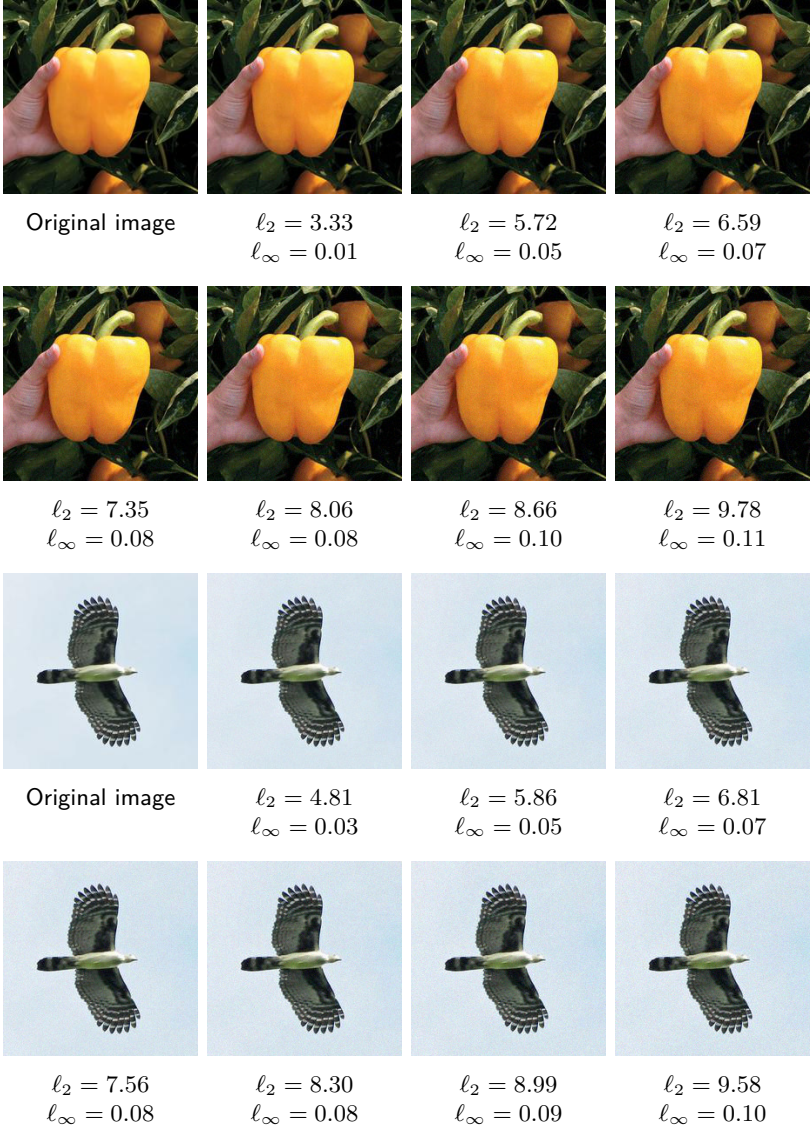


Figure 3.3: Illustration of two unperturbed images and their adversarial counterparts. The $\ell_{\{2,\infty\}}$ norms of the perturbations are provided below each adversarial example.

3.4 Why do adversarial examples exist?

Similar to other areas of research that work with neural networks, research on adversarial examples in the context of neural networks also suffers from being more experimentation-oriented, meaning that most of the results come from some form of experimentation rather than conclusive theoretical work. However, this does not mean that all work is experimental in nature. A number of influential theoretical research efforts in the field of adversarial examples include the work of Bhagoji et al. [13] and Peck et al. [161], analyzing the lower bounds of adversarial robustness. Fawzi et al. [46] took steps towards understanding the precise theoretical bounds on the robustness of classifiers, while Gilmer et al. [57] analyzed the validity of a number of theoretical observations on synthetic datasets. With the aid of previous research efforts, Shafahi et al. [183] tried to answer the question whether or not adversarial examples are inevitable for neural networks.

Perhaps the most interesting research efforts in the field of adversarial examples are the ones that thus far analyzed the reasons of existence for adversarial examples. Although early research on adversariality suggested that the existence of adversarial examples can be attributed to the excessive linearity of neural networks [62] (i.e., the excessive linearity hypothesis), it was later hypothesized that adversarial examples are due to the non-robust features learned by models [89] (i.e., the non-robust feature hypothesis). The latter study attracted a large amount of criticism, with researchers expressing distrust in the results put forward, particularly due to the controversial title of the paper: “Adversarial examples are not bugs, they are features”. However, the distrust in the work of [89] was promptly alleviated thanks to follow-up research efforts that reproduced the results previously presented, thus demonstrating that the observations made by Ilyas et al. [89] were indeed valid [44]. Nevertheless, the work of Nakkiran [141] revealed that a portion of adversarial examples can be just bugs too. This means that there exists a large number of adversarial examples that neither transfer between models nor leak non-robust features, which are argued to exist in Ilyas et al. [89]. Furthermore, a new hypothesis in the field, namely the dimpled manifold hypothesis [184], argues that the existence of adversarial examples can be explained through particularities of high-dimensional decision boundaries. However, the work of [184] was criticized by Kilcher [101] for being not rigorous.

As of writing this dissertation, there is no agreement among researchers on a theory that explains the existence of adversarial examples. Thus far, all research efforts that tried to explain the existence of adversarial examples were criticized and shown to be incomplete by succeeding studies.

3.5 Perturbation measurement

Although they are not a perfect match for how humans perceive noise, ℓ_p norms (with $p \in \{0, 2, \infty\}$) are commonly used methods to quantify image perturbation since the early days of research on adversarial examples [22, 56, 158]. For the settings given in Section 2.3, a perturbation Δ bounded by the ℓ_p ball centered at \mathbf{x} with a radius ϵ ,

$$\mathcal{B}(\mathbf{x})_\epsilon^p := \{\hat{\mathbf{x}} : \|\Delta\|_p := \|\mathbf{x} - \hat{\mathbf{x}}\|_p \leq \epsilon\}, \quad (3.7)$$

is said to be an *adversarial perturbation* if $G(\theta, \mathbf{x}) \neq G(\theta, \hat{\mathbf{x}})$. In this case, $\hat{\mathbf{x}}$ is said to be an adversarial example.

Based on $\Delta \in \mathbb{R}^Q$, we can measure the intensity of the adversarial perturbation according to different ℓ_p norms:

$$\ell_0(\Delta) = \sum \mathbf{1}_{\{\Delta_i \neq 0\}} / Q, \quad (3.8)$$

$$\ell_2(\Delta) = \|\Delta\|_2, \quad (3.9)$$

$$\ell_\infty(\Delta) = \max(\Delta), \quad (3.10)$$

where the ℓ_2 norm measures the Euclidian norm of the perturbation and where the ℓ_∞ norm measures the maximum change. As an example, an ℓ_∞ norm of 1 means that the added perturbation changed a pixel from black to white (e.g., from 0 to 1, depending on the normalization), or vice versa. Throughout the remainder of this dissertation, the ℓ_2 and ℓ_∞ norms are typically used as perturbation measures.

For a qualitative evaluation of the $\ell_{\{2, \infty\}}$ adversarial perturbation, we provide a number of examples in Figure 3.3. As can be seen, even for large perturbations, the adversarial noise is not easily noticeable.

The ℓ_0 norm measures the number of image pixels that have been modified by an adversarial attack. In this respect, an ℓ_0 norm of 1 means that all pixels were modified by the adversarial perturbation. As we will discuss shortly, in this work, we group adversarial attacks into two categories: (1) adversarial attacks that do not put any constraints on where exactly the perturbation is exercised (i.e., “global”

adversarial attacks) and (2) adversarial attacks that exercise perturbation on specific parts of an image (i.e., “local” or “regional” adversarial attacks). As such, the usage of the ℓ_0 norm is only meaningful when discussing regional adversarial attacks since global attacks perturb a large portion of an image (albeit by a small, invisible amount).

3.6 Box constraints

As we have described in Section 3.3, adversarial examples are created through the addition of what is called adversarial perturbations. However, depending on the method employed, the created adversarial example is very likely to end up not respecting the rules of the domain it is created for. To give an example from the image domain, not only does an RGB image have to have pixel values that are natural numbers, those numbers have to be less than or equal to 255 and greater than or equal to 0 in order to have a corresponding color representation (in the case a normalization is employed, the given description can be translated into values after the normalization). As such, an adversarial example created for the image domain has to respect the constraints of the domain. Adversarial examples that satisfy such property are said to satisfy the box constraints of the domain and thus meet the requirements for discretization (also called quantization).

The adversarial examples that we will employ throughout this dissertation indeed satisfy the property explained above for the domain they have been created for.

3.7 Adversarial transferability

Interestingly, depending on the attack used, adversarial examples can be highly *transferable*: an adversarial sample that fools a certain classifier can also fool completely different classifiers trained for the same task [27, 157]. This property is called the transferability of adversarial examples, which is a popular metric for assessing the effectiveness of a particular attack.

Let θ_1 and θ_2 represent the specifications of two DNNs and let \mathbf{x} and $\hat{\mathbf{x}}_1$ be a genuine image and an adversarial example generated from this genuine image through the usage of the DNN represented as θ_1 . If $G(\theta_1, \hat{\mathbf{x}}_1) = G(\theta_2, \hat{\mathbf{x}}_1) \neq G(\theta_{\{1,2\}}, \mathbf{x})$, then (1) the adversarial

example is classified differently than \mathbf{x} and (2) it is predicted into the same category for both θ_1 and θ_2 . In such cases, the adversarial example is said to have achieved targeted adversarial transferability. If $G(\theta_1, \hat{\mathbf{x}}_1) \neq G(\theta_2, \hat{\mathbf{x}}_1)$ and $G(\theta_2, \hat{\mathbf{x}}_1) \neq G(\theta_{\{1,2\}}, \mathbf{x})$, then it means that the adversarial example is classified differently than its unperturbed counterpart. However, the adversarial example in question is classified into a category that is different than the targeted one (i.e., $G(\theta_1, \hat{\mathbf{x}}_1)$). In cases like this, an adversarial example is said to have achieved untargeted adversarial transferability.

Throughout this dissertation, we will regularly use adversarial transferability to measure the effectiveness of attacks and to assess the robustness of the created adversarial examples.

3.8 Investigating the loss functions of adversarial attacks

As more attacks are proposed with each study, hereby showing that the newly introduced methods are *superior* to their predecessors by mostly empirical means, a mathematical explanation as to why the proposed attacks are *better* than others remains lacking. A similar observation can be made regarding the availability of rigorous comparative analyses of the proposed attacks. In what follows, we aim at formally explaining a number of properties of commonly used gradient-based attacks, namely L-BFGS, IFGS, and CW. We investigate (1) why some of these attacks are faster than others when generating adversarial examples [110] and (2) why some of these attacks are creating *stronger* adversarial examples that are more robust against state-of-the-art defense mechanisms [22]. To that end, we first look for ways to generalize the loss functions used by adversarial attacks.

3.8.1 Generalizing loss functions

Although the objective functions presented in Section 3.3 differ in multiple ways and are complex in nature, it is possible to generalize them in terms of how the perturbation is generated. During each step, these methods essentially aim at finding a perturbation that increases the prediction likelihood of the target class. In order to investigate properties of interest of the different methods for adversarial example generation, like speed and robustness, we first rewrite the objective

functions of L-BFGS, IFGS, and CW in a similar format, with this format explicitly displaying the source of the generated perturbation. As mentioned before, to generate the perturbation, L-BFGS uses CE loss, IFGS uses CE-sign loss, and CW uses M-logit loss. In order to display the source of the perturbation, we rewrite the aforementioned attacks as follows:

$$\hat{\mathbf{x}}^{(n+1)} = \zeta_1 \left(\hat{\mathbf{x}}^{(n)} + \gamma_1 (\nabla_x J(g(\theta, \hat{\mathbf{x}}^{(n)})_c)) \right), \quad (3.11)$$

$$\hat{\mathbf{x}}^{(n+1)} = \zeta_2 \left(\hat{\mathbf{x}}^{(n)} + \gamma_2 (\text{sign} (\nabla_x J(g(\theta, \hat{\mathbf{x}}^{(n)})_c))) \right), \quad (3.12)$$

$$\hat{\mathbf{x}}^{(n+1)} = \zeta_3 \left(\hat{\mathbf{x}}^{(n)} + \gamma_3 (\nabla_x g(\theta, \hat{\mathbf{x}}^{(n)})_c) + \gamma_4 (\nabla_x g(\theta, \hat{\mathbf{x}}^{(n)})_{c^*}) \right), \quad (3.13)$$

where Equation (3.11), Equation (3.12), and Equation (3.13) correspond to *adversarial optimizations* performed by L-BFGS, IFGS, and CW, respectively. In the above equations, c refers to the target class. Furthermore, the γ -function, which is unique for each method, is used to satisfy the properties of the generated perturbation and thus encompasses (e.g., the perturbation multiplier and the clipping function). The ζ -function, in its turn, is used to ensure that the generated adversarial examples satisfy various constraints, such as box constraints and ℓ_2 distance minimization. In Equation (3.13), the second-most-likely target class is referred to as c^* .

By writing the adversarial optimizations in this way, it becomes clear that, in order to analyze the properties of adversarial example generation methods, the properties of the loss functions must be investigated. Therefore, in the next section, we will analyze the differences between the cross-entropy loss $\nabla_x J(g(\theta, \mathbf{x}), c)$ as used by L-BFGS, the CE-sign loss $\text{sign}(\nabla_x J(g(\theta, \mathbf{x}), c))$ as used by IFGS, and the logit loss $\nabla_x g(\theta, \mathbf{x})_c$, whose extension M-logit loss is used by CW.

A data point selected for adversarial optimization is assumed to be correctly classified first, and the aim of the attack is to eventually change the prediction, so to force the machine learning model under consideration to incorrectly classify the selected data point (often with high confidence). Consider Figure 3.4, which shows the mean prediction likelihood for both the initial class and the target class. This mean prediction likelihood was obtained by creating 1,000 adversarial examples, applying IFGS to samples taken from the ImageNet dataset. A pretrained ResNet-50 [75] in white-box settings was used for this

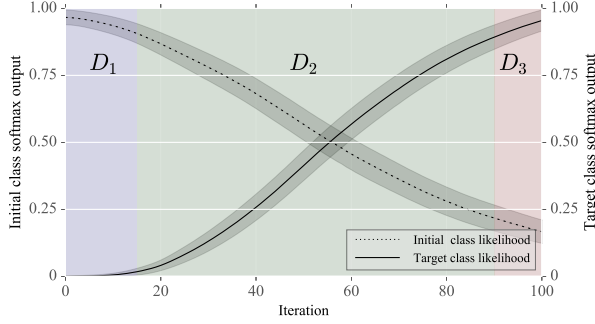


Figure 3.4: The prediction likelihood of both the initial class and the target class is displayed throughout 100 iterations of adversarial optimization. The lines represent the mean and the shaded areas represent the 95% confidence interval of the likelihood values that have been obtained for 1,000 adversarial examples. The subspaces defined in Definition 3.8.1 are highlighted as blue, green, and red areas, representing D_1 , D_2 , and D_3 , respectively.

small experiment.

As can be seen in Figure 3.4, the adversarial optimization causes the prediction likelihood of the initial class to decrease and the prediction likelihood of the target class to increase. Note that the method used for adversarial optimization does not change the overall prediction outcome, but only the rate of change in the likelihoods. Indeed, the ImageNet samples are, in the end, mostly classified as the target class, with confidence values on average higher than 0.9.

Based on observations made for the above experiment, we adopt the definition outlined below.

DEFINITION 3.8.1. Let $\mathbf{y} = g(\theta, \mathbf{x})$ be a neural network that maps an input \mathbf{x} to an output vector $\tilde{\mathbf{y}}$ where the classification is assigned to into categorical label as $G(\theta, \mathbf{x}) = \arg \max(\tilde{\mathbf{y}})$ and let c be the target class. Based on the softmax output $P(g(\theta, \mathbf{x}))$ of the classification result, we define three subspaces $D_{\{1,2,3\}}$:

- D_1 – The input is classified with high confidence as some class r other than the target class c : $\forall \mathbf{x}_1 \in D_1, \max(P(g(\theta, \mathbf{x}_1))) \geq 0.9$ and $G(\theta, \mathbf{x}_1) = r, r \neq c$.
- D_2 – The input is classified with relatively low confidence for any class: $\forall \mathbf{x}_2 \in D_2, \max(P(g(\theta, \mathbf{x}_2))) < 0.9$.

- D_3 – The input is classified with high confidence as the target class c : $\forall \mathbf{x}_3 \in D_3, P(g(\theta, \mathbf{x}_3)) \geq 0.9$ and $G(\theta, \mathbf{x}_3) = c$.

The subspaces listed in Definition 3.8.1 are visualized in Figure 3.4. A data point that is selected for adversarial optimization is assumed to be correctly classified (more-often-than-not with high confidence), and hence resides in D_1 . The aim of the adversarial example generation methods we discussed in Section 3.3 is to move this data point from D_1 to D_3 in an iterative manner, where it is eventually classified as the target class. During this optimization process, the data point under consideration inevitably passes through D_2 .

In what follows, we analyze the objective functions in more detail, with the goal of answering the following question: “*what makes an objective function beneficial or detrimental when generating adversarial examples?*” Previously, we showed that it is possible to distinguish between adversarial example generation methods by their usage of CE, CE-sign, and logits. As a result, we investigate the limitations and advantages of CE loss and CE-sign loss against logit loss in the context of adversarial example generation.

3.8.2 Cross-entropy loss

The cross-entropy loss was, thanks to its desirable properties, one of the first loss functions used to generate adversarial examples [201]. The following theorem describes the behavior of the CE loss for the subspaces $D_{\{1,2,3\}}$.

THEOREM 3.8.2. *Let $g(\theta, \mathbf{x})$ be the prediction made by a neural network described as θ , that maps an input \mathbf{x} to an output vector $\tilde{\mathbf{y}}$, and assume that adversarial examples are generated using the cross-entropy function $J(g(\theta, \mathbf{x}), c)$.*

Based on the subspaces described in Definition 3.8.1, the gradient $\nabla_{\mathbf{x}} J(g(\theta, \mathbf{x}), c)$ of the cross-entropy function can be approximated as follows:

$$\forall \mathbf{x}_1 \in D_1,$$

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_1} \nabla_{\mathbf{x}} J(g(\theta, \mathbf{x}), c) = \nabla_{\mathbf{x}} g(\theta, \mathbf{x}_1)_r - \nabla_{\mathbf{x}} g(\theta, \mathbf{x}_1)_c. \quad (3.14)$$

$$\forall \mathbf{x}_2 \in D_2,$$

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_2} \nabla_x J(g(\theta, \mathbf{x}), c) = \sum_{m \in I \setminus \{c\}} \beta_m \left(\nabla_x g(\theta, \mathbf{x}_2)_m - \nabla_x g(\theta, \mathbf{x}_2)_c \right), \quad (3.15)$$

with constants β_1, \dots, β_M subject to $\sum_{m \in I \setminus \{c\}} |\beta_m| < 1$, where $I = \{1, 2, \dots, M\}$.

$$\forall \mathbf{x}_3 \in D_3,$$

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_3} \nabla_x J(g(\theta, \mathbf{x}), c) = 0. \quad (3.16)$$

Proof of Theorem 5.1. Recall the formulae of the cross-entropy function J :

$$J(g(\theta, \mathbf{x}), c) = -\log \left(\frac{e^{g(\theta, \mathbf{x})_c}}{\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m}} \right), \quad (3.17)$$

where $c \in \{1, \dots, M\}$ is the target label among M classes. We can rewrite the cross-entropy function as

$$J(g(\theta, \mathbf{x}), c) = -g(\theta, \mathbf{x})_c + \log \left(\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m} \right). \quad (3.18)$$

To gain insight into the behavior of the cross-entropy loss, we will, given Equation 3.11, analyze its gradient

$$\nabla_x J(g(\theta, \mathbf{x}), c) = \nabla_x \left(-g(\theta, \mathbf{x})_c + \log \left(\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m} \right) \right), \quad (3.19)$$

$$= -\nabla_x g(\theta, \mathbf{x})_c + \frac{\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m} \nabla_x g(\theta, \mathbf{x})_m}{\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m}}. \quad (3.20)$$

Let us now consider Equation (3.20) for the different subspaces. Recall that $I = \{1, 2, \dots, M\}$ denotes the the set of class indices.

When in subspace D_1 , each data point is classified with high confidence as belonging to some class $r \in I \setminus \{c\}$. Then, $\forall \mathbf{x}_1 \in D_1$, we

have $e^{g(\theta, \mathbf{x}_1)_r} \gg e^{g(\theta, \mathbf{x}_1)_{m \in I \setminus \{r\}}}$. Hence,

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_1} \frac{\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m} \nabla_x g(\theta, \mathbf{x})_m}{\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m}} = \nabla_x g(\theta, \mathbf{x}_1)_r, \quad (3.21)$$

which means that

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_1} \nabla_x J(g(\theta, \mathbf{x}), c) = \nabla_x g(\theta, \mathbf{x}_1)_r - \nabla_x g(\theta, \mathbf{x}_1)_c. \quad (3.22)$$

While in subspace D_3 , each data point is classified, with high confidence, as belonging to the target class: $\forall \mathbf{x}_3 \in D_3, e^{g(\theta, \mathbf{x}_1)_c} \gg e^{g(\theta, \mathbf{x}_1)_{m \in I \setminus \{c\}}}$. Thus,

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_3} \frac{\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m} \nabla_x g(\theta, \mathbf{x})_m}{\sum_{m=1}^M e^{g(\theta, \mathbf{x})_m}} = \nabla_x g(\theta, \mathbf{x}_3)_c. \quad (3.23)$$

Therefore,

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_3} \nabla_x J(g(\theta, \mathbf{x}), c) = 0. \quad (3.24)$$

Finally, for subspace D_2 , we will first continue rewriting Equation (3.20):

$$\nabla_x J(g(\theta, \mathbf{x}), c) = -\nabla_x g(\theta, \mathbf{x})_c + \frac{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m} \nabla_x g(\theta, \mathbf{x})_m}{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m}}, \quad (3.25)$$

$$= -\frac{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m} \nabla_x g(\theta, \mathbf{x})_c}{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m}} + \frac{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m} \nabla_x g(\theta, \mathbf{x})_m}{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m}}, \quad (3.26)$$

$$= \frac{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m} (\nabla_x g(\theta, \mathbf{x})_m - \nabla_x g(\theta, \mathbf{x})_c)}{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m}}. \quad (3.27)$$

For $m = c$, $e^{g(\theta, \mathbf{x})_m} (\nabla_x g(\theta, \mathbf{x})_m - \nabla_x g(\theta, \mathbf{x})_c) = 0$. This means that the case $m = c$ does not contribute to the sum and that we can write Equation (3.27) as:

$$\nabla_x J(g(\theta, \mathbf{x}), c) = \frac{\sum_{m \in I \setminus \{c\}} e^{g(\theta, \mathbf{x})_m} (\nabla_x g(\theta, \mathbf{x})_m - \nabla_x g(\theta, \mathbf{x})_c)}{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m}}. \quad (3.28)$$

Trivially, $\forall m \in I$ and $\forall \mathbf{x}_2 \in D_2$ $\beta_m := \frac{e^{g(\theta, \mathbf{x})_m}}{\sum_{m \in I} e^{g(\theta, \mathbf{x})_m}} < 1$. We can then represent the gradient as

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_2} \nabla_x J(g(\theta, \mathbf{x}), c) = \sum_{m \in I \setminus \{c\}} \beta_m (\nabla_x g(\theta, \mathbf{x}_2)_m - \nabla_x g(\theta, \mathbf{x}_2)_c) \quad (3.29)$$

with $\sum_{m \in I \setminus \{c\}} |\beta_m| < 1$. □

Let us now discuss the outcome of Theorem 3.8.2 for the data points that lie in D_1 , D_2 , and D_3 .

Optimization for data points in D_1 —In this subspace, where the data point under consideration is classified with high confidence as any class other than the targeted one, using the gradient of the CE loss produced by gradient descent, the likelihood of the current class r will be minimized and, at the same time, the likelihood of the target class c will be maximized. Doing so naturally increases the optimization speed compared to only maximizing the target class likelihood, given the fact that the data points in D_1 are classified as r .

CW, even though regarded as the attack that produces the strongest adversarial examples, is also criticized for being substantially slower (i.e., computationally more expensive) than IFGS and L-BFGS [64]. This is not only because the algorithm itself is complex (i.e., incorporating mechanisms like multiple-starting-point gradient descent), but also because CW uses M-logit loss instead of CE loss. As we have stated above, CE loss, compared to logit loss, has a faster speed at the start of the optimization (i.e., for the data points in D_1) and is thus able to change the prediction of the data point at hand faster.

In Section 3.8.4, we will show empirical results on the ImageNet dataset in support of the aforementioned claims.

Optimization for data points in D_2 —A data point located in subspace D_2 is classified with low confidence as one of the available classes. Once here, the optimization speed starts to decrease because $\sum_{m \in I \setminus \{c\}} |\beta_m| < 1$, as can be seen from Equation (3.15). This is different from the optimization in D_1 , where, provided that the equation is written in the same format, the gradient multiplier is 1.

Optimization for data points in D_3 —In subspace D_3 , the data point under consideration is already classified as the target class with high confidence. In this case, the loss generated with the cross-entropy approaches zero, which means that the data point under consideration cannot be further optimized. We will refer to this mathematical constraint of the CE loss in an adversarial setting for data points that lie in D_3 as the mathematical limit of the optimization space of the CE loss.

The biggest difference between logit loss-based optimization methods (e.g., CW) and CE loss-based optimization methods (e.g., FGS, IFGS, and L-BFGS) can be observed for the data points that lie in D_3 . As we have shown in Theorem 3.8.2, CE loss cannot generate any further gradient signal as soon as the data point under consideration is classified with high confidence as the target class. This constraint, as will become clear from the examples that will be given in the upcoming sections, does not apply to logit loss-based optimization methods, explaining why CW is able to create arguably *stronger* adversarial examples.

3.8.3 Cross-entropy sign loss

We also experimented with taking the signature of the gradient of the CE loss, a method that was shown to be successful in generating adversarial examples [62, 110]. Although it is not clear whether or not this method was used in the aforementioned papers to overcome the subspace limitation of the CE loss that we previously laid out, we will show that taking the signature does not alleviate the subspace limitation of the CE loss in adversarial settings, and that it also brings along its own detrimental properties.

Limited optimization space—Since the limit of the gradient generated with the CE loss, for the data points in D_3 , approaches zero, multiplying this gradient with a large number is not sufficient to move the data point under consideration deeper into D_3 . In computational settings with limited decimal precision, when, in absolute value, the largest element of the gradient matrix $\nabla_x J(g(\theta, \mathbf{x}), c)$ for data points

that reside in D_3 becomes smaller than the numerical precision used by adversarial optimization, then $\nabla_x J(g(\theta, \mathbf{x}), c) \equiv 0$. As a result, taking the signature of the gradient generated with CE loss, and using it instead of the gradient itself, will not further optimize the data points that lie in D_3 .

Eliminated rate-of-change among elements – Since both FGS and IFGS take the signature of the gradient, only the direction of the optimization is retained for individual elements of the gradient, and information about the rate of change is lost. As a result, the optimization step size for each element becomes identical, which can hinder the optimization speed. We will show in the upcoming section that, on average, FGS and IFGS perform worse than direct usage of gradients in terms of the total amount of perturbation added.

3.8.4 Experiments

In support of our mathematical observations, we performed two sets of large-scale experiments, with the first set of experiments making use of a controllable 2-D environment and with the second set of experiments leveraging the ImageNet dataset.

3.8.4.1 Use of a controllable 2-D environment

To visually demonstrate how adversarial optimization behaves for different objective functions, we use the setting of adversarial example generation that is illustrated in Figure 3.5. In particular, the corresponding experiment deals with a 2-D two-class classification problem where the data are sampled using the function *make_circles* of the SciPy library [96], obtaining two circular data motifs with the same mean $(0, 0)$, but different radii. To solve this classification problem, we apply a neural network with a single hidden layer that contains 50 neurons followed by a rectifier activation [58]. Our model takes a point with 2-D coordinates (x, y) as input and maps each point to one out of two target classes: (R)ed or (B)lack. The circular decision boundary of this model is drawn in black in Figure 3.5. Under these settings, the adversarial example generation process is defined as moving a point from the outer class (B) to the inner class (R), the latter thus acting as the *target class*.

The strength calculated via the equation given for each loss is visualized in the form of heat maps. Since the direction of optimization is different for different points due to the circularity of the distributions,

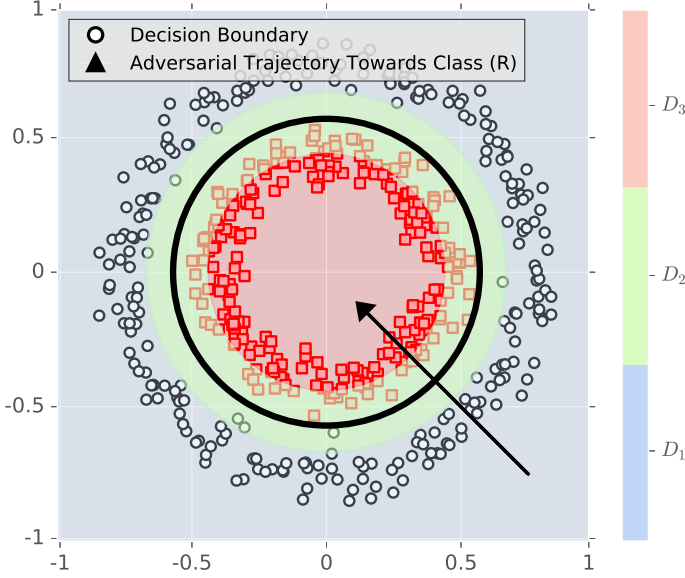


Figure 3.5: A classification problem consisting of two circular distributions with the same center $(0, 0)$ but different radii. The data are bounded by $(x, y) \in [-1, 1]^2$. An adversarial trajectory is defined as moving a data point from the outer class to the inner class across the decision boundary.

we use the absolute magnitude of the gradient in these heat maps. Figure 3.6a, Figure 3.6b, and Figure 3.6c represent the magnitude of the gradients generated with CE loss $\log(\sum |\nabla_x J(g(\theta, \mathbf{x}), R)|)$, CE-sign loss $\text{sign}(|\nabla_x J(g(\theta, \mathbf{x}), R)|)$, and logit loss $\log(\sum |\nabla_x g(\theta, \mathbf{x}, R)|)$ for the points $(x, y) \in [-1, 1]^2$, respectively. For the aforementioned figures, we provide the summarizing observations below:

- Figure 3.6a – The limited optimization space of the CE loss can be clearly observed, with the magnitude of the gradient becoming zero as soon as data points in the target subspace D_3 are selected.
- Figure 3.6b – When the logit loss is used, the gradient exists for all points $(x, y) \in [-1, 1]^2$. Yet, for the logit loss, the magnitude of the gradient for data points in D_1 is less than the magnitude for the same data points when CE loss is selected, which shows the benefit in terms of optimization speed when using CE loss over logit loss for the data points that lie in D_1 .
- Figure 3.6c – The area that contains non-zero gradients in the

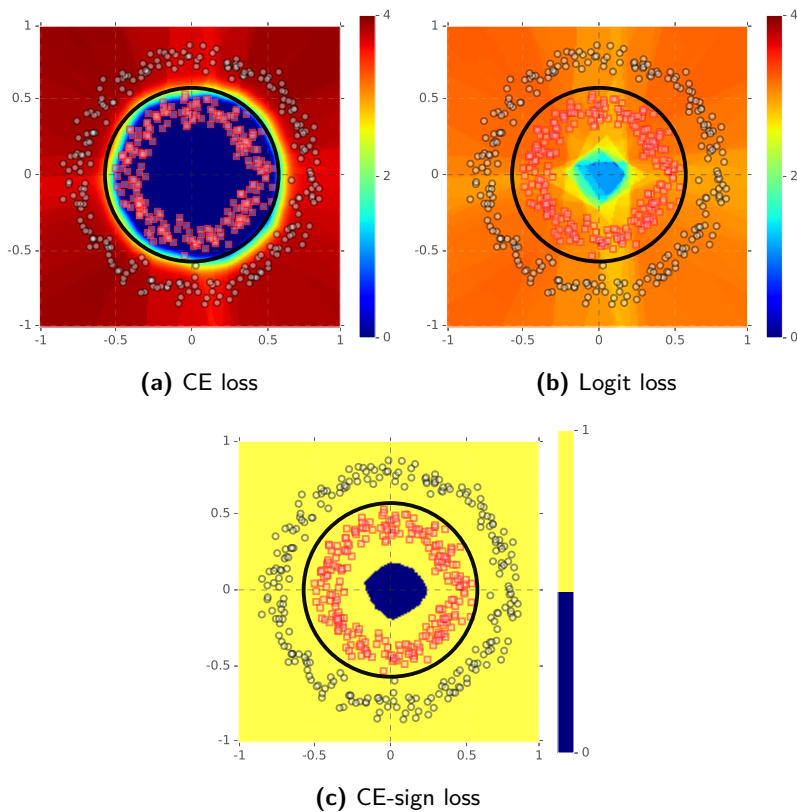


Figure 3.6: Heat maps characterizing the adversarial optimization are given for the classification problem presented in Figure 3.5, generated using (a) the CE loss, (b) the logit loss, and (c) the CE-sign loss.

target subspace D_3 has increased compared to Figure 3.6a. However, there still exists an area at the center of the graph where the magnitude of the gradient is exactly zero.

3.8.4.2 Use of ImageNet

For the experiments conducted on ImageNet, we rely on the following method to generate the adversarial examples:

$$\hat{\mathbf{x}}^{(n+1)} = \zeta\left(\hat{\mathbf{x}}^{(n)} + \alpha \mathbf{p}^{(n)}\right), \quad (3.30)$$

$$CE \text{ loss: } \mathbf{p}^{(n)} = \nabla_x J(g(\theta, \mathbf{x}^{(n)})_c), \quad (3.31)$$

$$CE\text{-sign loss: } \mathbf{p}^{(n)} = \text{sign}(\nabla_x J(g(\theta, \mathbf{x}^{(n)})_c)), \quad (3.32)$$

$$Logit \text{ loss: } \mathbf{p}^{(n)} = \nabla_x g(\theta, \mathbf{x}^{(n)})_c, \quad (3.33)$$

$$M\text{-logit loss: } \mathbf{p}^{(n)} = \alpha' \nabla_x g(\theta, \mathbf{x}^{(n)})_c + \alpha^* \nabla_x g(\theta, \mathbf{x}^{(n)})_{c^*}, \quad (3.34)$$

$$\text{with } g(\theta, \mathbf{x}^{(n)})_c - g(\theta, \mathbf{x}^{(n)})_{c^*} = \kappa, \quad (3.35)$$

where Equation (3.30) represents the general approach to create adversarial examples, with $\mathbf{p}^{(n)}$ the perturbation added during each iteration, and α the perturbation multiplier. In this specific setting, the ζ -function is, again, used to ensure that the generated adversarial examples satisfy box constraints. Equations (3.31) to (3.34) describe the source of perturbation for the different methods. Note that we also include the logit loss in order to continue the comparative analysis between the different types of losses. For the M-logit loss, we follow the work of [22] and select $\kappa = 20$. To ensure that Equation (3.35) holds for $\kappa = 20$, the multipliers α' and α^* are used to adjust the level of perturbation generated by their respective sources.

To investigate the properties of the perturbation generated by the different loss functions discussed in Section 3.8, we conduct two experiments on ImageNet using different settings: (1) keeping the perturbation multiplier constant and (2) keeping the perturbation itself equal. We generate 1,000 adversarial examples for each attack and for each experiment, using a pretrained ResNet-50 network [75] in white-box settings, hereby performing a detailed analysis of the properties of the generated perturbations.

Equal multiplier – In studies that investigate adversarial examples, the perturbation generated using various losses is multiplied with a constant α and then added to the image in order to create an adversarial example. In our first experiment, we follow this common approach, using $\alpha = 5 \times 10^{-4}$ for the equations given in Section 3.8.4 to generate adversarial examples. However, in order to allow for a detailed investigation, instead of only analyzing the prediction, we also study the changes in magnitude of the gradient throughout the opti-

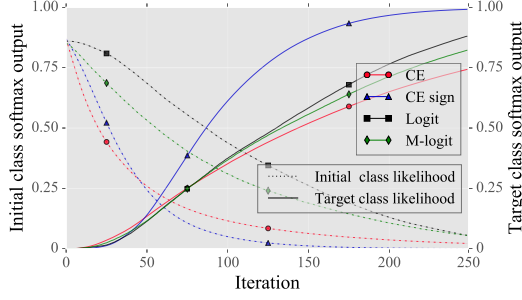
Table 3.1: Mean and standard deviation, for the experiments “equal perturbation” and “equal multiplier”, of (1) the least number of iterations of added perturbation needed to change the prediction and (2) the time required to calculate the perturbation (measured on a single Titan-X GPU).

	Equal Multiplier		Equal Perturbation	
	Iterations	Time (s)	Iterations	Time (s)
CE	114 ± 71	4.25 ± 0.3	55 ± 28	5.6 ± 0.1
CE-sign	80 ± 37	4.05 ± 0.2	116 ± 52	5.6 ± 0.1
Logit	134 ± 72	4.19 ± 0.2	80 ± 47	5.7 ± 0.1
M-logit	126 ± 73	4.30 ± 0.27	95 ± 52	5.8 ± 0.1

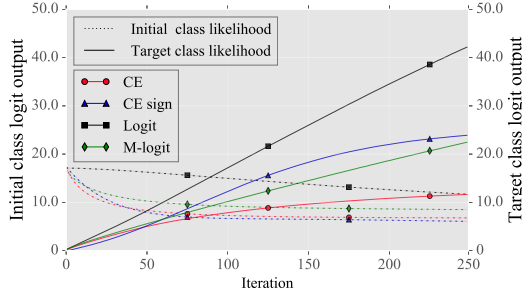
mization, as well as the least amount of perturbation needed to change the prediction.

Equal perturbation—The gradient values produced depend on the loss function used, the subspace the optimized data point is located in, and the number of iterations executed thus far. As a result, the produced gradient values may be different in terms of their total magnitude. In order to allow for a fair comparison of the optimization speed across different types of loss functions, we now diverge from the previous studies and adopt a dynamic multiplier technique adjusted at each iteration n with $\alpha = \beta / \sum_i |\mathbf{p}^{(n)}|$. This method adjusts the perturbation multiplier dynamically, so to make the added perturbation in each step equal (in terms of magnitude) across all methods for all iterations. The purpose of this experiment is to show the effectiveness of each perturbation produced using the aforementioned losses when the added perturbation is held constant. Under these circumstances, we use $\beta = 5$ in order to calculate α dynamically.

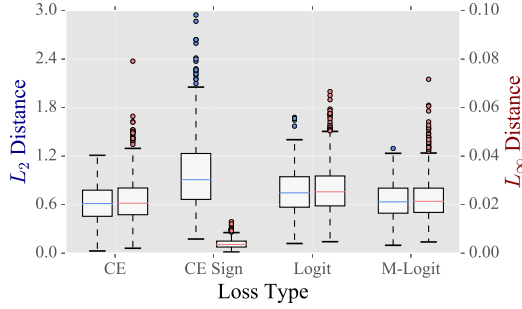
Experimental results—The experimental results obtained are displayed in Figure 3.7, Figure 3.8, Figure 3.9, and Table 3.1. Specifically, in Figure 3.7 and Figure 3.8, (a) shows the softmax output and (b) shows the logit prediction for both the initial class and the target class, for adversarial optimization using 250 iterations, for both the experiment “equal multiplier” and “equal perturbation”. On the other hand, (c) in both Figure 3.7 and Figure 3.8 visualizes the least amount of perturbation required in order to change the prediction to the target class, obtained once for all adversarial examples when their prediction is changed for the first time, with the amount of perturbation represented in the form of box plots for the respective experiments. The average number of iterations required to change the prediction of the data point under consideration and its standard deviation, as well as



(a) Softmax output

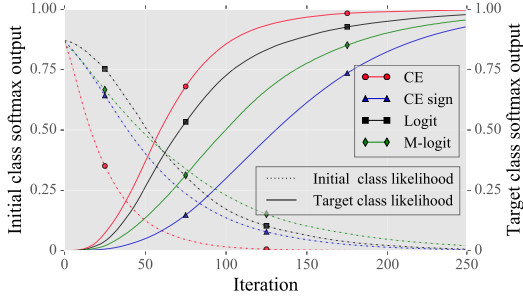


(b) Logit output

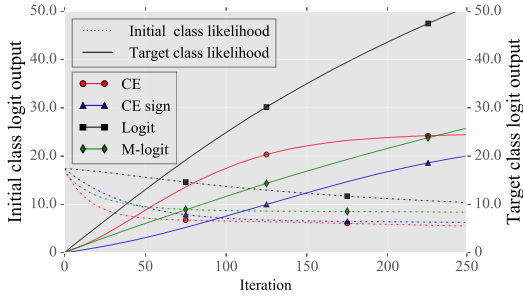


(c) Least amount of required perturbation

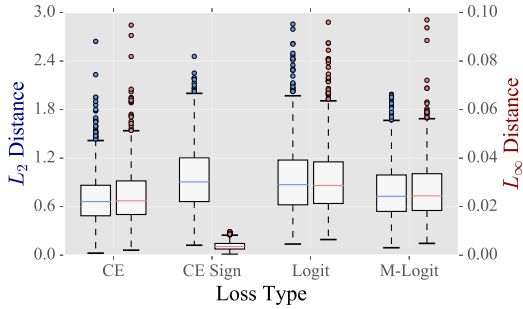
Figure 3.7: (a) The softmax output and (b) the logit prediction for both the initial class and the target class is given for adversarial optimization over 250 iterations, for the experiment “equal multiplier”. The values correspond to the mean value of 1,000 samples, taken from the ImageNet validation set. (c) The least amount of total added perturbation required to change the prediction of the data points under consideration (the same data points are presented in (a) and (b)). Best viewed in color.



(a) Softmax output



(b) Logit output



(c) Least amount of required perturbation

Figure 3.8: (a) The softmax output and (b) the logit prediction for both the initial class and the target class is given for adversarial optimization over 250 iterations, for the experiment “equal perturbation”. The values correspond to the mean value of 1,000 samples, taken from the ImageNet validation set. (c) The least amount of total added perturbation required to change the prediction of the data points under consideration (the same data points are presented in (a) and (b)). Best viewed in color.

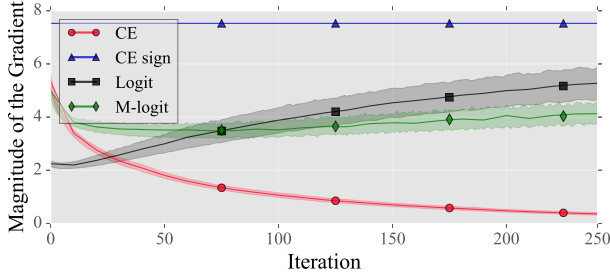


Figure 3.9: Change in gradient magnitude for the experiment “equal multiplier”, for the data points presented in Figure 3.7.

the total amount of time required to calculate the perturbations (for a total of 250 iterations in terms of compute time), are listed in Table 3.1. Lastly, Figure 3.9 shows the change in gradient magnitude for the experiment “equal multiplier”.

Based on these results, we make the following observations:

- When the perturbation multiplier is held the same, the magnitude of the produced gradient becomes substantially different for each method, making this kind of experiments unsuitable for analyzing how *fast* or *slow* a method is.
- The misconception that the CE-sign loss, as used by IFGS and FGS, generates adversarial examples *faster* than the other methods [22] stems from the fact that taking the signature naturally boosts the gradient obtained from the model. This is shown in Figure 3.9, where the perturbation added by the CE-sign loss is significantly larger than the perturbation added by other approaches. However, when the perturbation is held constant, the CE-sign becomes the least effective loss and creates adversarial examples the slowest in terms of the number of iterations needed.
- The CE loss, whose usage is criticized in the works of Carlini and Wagner [22, 23], on average, not only creates adversarial examples the fastest when the perturbation is held constant, but it also does so with the least amount of total ℓ_2 perturbation compared to other losses.
- Even when the perturbation multiplier is held the same across all iterations, the gradient generated by the CE loss decreases as the optimization continues (see Figure 3.9), confirming the practicality of Theorem 3.8.2.

- Both the CE and the CE-sign loss have a limited optimization space, as observed in Figure 3.7b and Figure 3.8b. This explains why the losses that rely on CE are not able to create *strong* adversarial examples from the perspective of fragility against adversarial defenses.
- Although, on average, the M-logit loss creates adversarial examples slower (95 iterations) than the logit loss (80 iterations), we observe that its total added perturbation is less than that of the logit loss. This means that more iterations of adding perturbation does not necessarily equate to a larger total perturbation, further showing the effectiveness of the M-logit loss, as used by Carlini and Wagner [22].
- Even if the M-logit loss indeed seems to be the slowest method, which confirms the criticism of Goodfellow et al. [64], the computational time difference between each method and M-logit loss is not substantial: the M-logit loss is, on average, only 3% slower than the fastest method. What makes CW extremely slow is its extensive search for the most effective perturbation, as described in [22], and not its underlying loss function.
- Even though we observe a correlation between the total amount of added perturbation in terms of ℓ_2 and ℓ_∞ distances, the results for the CE-sign loss are vastly different from others. This is because taking the signature gets rid of changes with a large magnitude, which will especially impact substantial changes. This results in significantly smaller perturbations in terms of ℓ_∞ distance.

The perturbation we thus far employed was applied directly to an image without examining the importance of the regions of the underlying image where it was applied to. Indeed, it is plausible that perturbation, when applied to certain regions of an image, may convert a genuine image to an adversarial example more easily (i.e., with less change), compared to applying this perturbation to other image regions. In what follows, we will perform a more detailed investigation of the relevance of certain image regions when it comes to exercising perturbation and creating adversarial examples.



Figure 3.10: Adversarial examples generated by regional attacks.

3.9 Regional adversarial perturbation

Through different generations of research in the field of computer vision, it was established that certain regions of images are more important for the identification of an object of interest than others [123, 137, 179, 200]. As such, research on localized adversarial attacks also shows that adversarial perturbation applied to these *important* regions may change the prediction faster and with less ℓ_p perturbation than attacks that apply the perturbation to entire images [98, 197, 223, 226].

The idea of regional adversarial examples gained popularity after the work of Su et al. [197], in which the authors showed the possibility of changing the prediction of images by perturbing only a single pixel. They proposed a perturbation generation method based on differential evolution and revealed that approximately 67% of the images in CIFAR-10 [108] and 16% of the images in ImageNet had their predictions changed by modifying a single pixel. As evidenced by the drop in the success rate from CIFAR-10 to ImageNet, this approach loses its potency as the dimensions of the images gets larger. This is due to the fact that, as the dimensions of images get larger, the importance of individual pixels becomes smaller.

The idea of using localized and visible noise (LaVaN) was investigated in the work of Karmon et al. [98], implementing LaVaN through the use of element-wise multiplication. In particular, Karmon et al. [98] propose the usage of two losses, where the first loss minimizes the likelihood of the current prediction and where the second loss maximizes the likelihood of a target class. Furthermore, instead of using the cross-entropy loss, they directly use the logits of the model.

Similar to the usage of LaVaN, Brown et al. [19] proposed the usage

of adversarial patches to sway the prediction of DNNs. Different from LaVaN, they envisioned a printable patch (hence the name, adversarial patch) and used it to occlude certain image parts. Their experiments revealed that printable adversarial perturbations (in this case, patches) can be used to mislead automated systems when making predictions.

Although most of the research efforts in the area of regional adversarial perturbation focused on creating visible distortions such as the patches of adversarial perturbation, Xu et al. [223] showed that systematic application of adversarial perturbation may lead to adversarial examples with invisible perturbations. In that regard, they devised a complex method to apply perturbation in image regions with high color variation, calling their approach a structured adversarial attack.

Example adversarial examples created with the aforementioned regional approaches are provided in Figure 3.10. As can be seen, the structured adversarial attack of Xu et al. [223] is able to create adversarial examples that come with high ℓ_p norm perturbations, but where these perturbations are not visible to the bare eye.

3.10 Global to regional conversion

Although regional attacks are shown to be as effective as global attacks, analyses to prevent adversarial examples often do not evaluate robustness against such regional attacks. Adversarial defenses are often studied exclusively against well-understood global attacks such as IFGS, PGD, and CW, with these attacks applying perturbations to entire images, hereby taking into account the magnitude of the loss gradient for each pixel and the ℓ_p norm constraints set.

We believe the lack of evaluation of defenses against regional attacks is because (1) regional attacks are often studied in permissive white-box settings that do not represent real-world scenarios and (2) the proposed attacks usually come with a completely new and complicated way of generating adversarial examples, thus making it not straightforward to apply these attacks to different datasets, especially not locally, as opposed to the well-understood global attacks. However, employing regional attacks does not necessarily mean having to deal with complicated approaches. Indeed, we will show in the following sections that a simple, general method for localizing adversarial perturbation can be used to exercise perturbation in selected regions.

The main question we are trying to answer when performing the

upcoming experiments is: “are some regions of an image more important than other regions when creating adversarial examples?” Based on the experimentation discussed below, we want to show that it is indeed possible to create adversarial examples by adding smaller amounts of perturbation to certain image regions, even when those image regions are pre-determined.

3.10.1 Framework

As we described in the previous chapter, it was the success of AlexNet in 2012 that popularized DNN architectures [109]. Recent research in the field of adversarial robustness also revealed AlexNet to be one of the more robust architectures [196]. Following the success of AlexNet, VGG [189] architectures were proposed with smaller convolutional kernel sizes. Thanks to their simple architecture, VGG architectures are still frequently used to address computer vision problems. In order to overcome issues with vanishing gradients in deep architectures, He et al. [75] proposed ResNet architectures, introducing the usage of residual layers. These residual architectures were later expanded upon, currently being some of the most popular architectures for solving a variety of problems in the field of deep learning [73, 222]. Given the history of the aforementioned architectures in the field of adversarial machine learning, as well as in other deep learning areas, we opted for the use of AlexNet, VGG-16, and ResNet-50 in our experiments, in conjunction with the use of the ImageNet dataset.

3.10.2 Experimental setup

Carlini and Wagner [23] demonstrated the fragility of adversarial examples generated by single-step attacks and argued that iterative attacks should be used for evaluating novel defenses. Iterative attacks calculate and add perturbation to the input in an iterative way, according to the following rule:

$$\hat{\mathbf{x}}^{(n+1)} = \hat{\mathbf{x}}^{(n)} + \mathbf{p}^{(n)}, \quad (3.36)$$

where $\hat{\mathbf{x}}^{(n)}$ represent the adversarial example created with the perturbation at the n th iteration. We generate the perturbation as follows:

$$\mathbf{p}^{(n)} = \alpha \operatorname{sign}(\nabla_x J(g(\theta, \hat{\mathbf{x}}^{(n)}), c)), \quad (3.37)$$

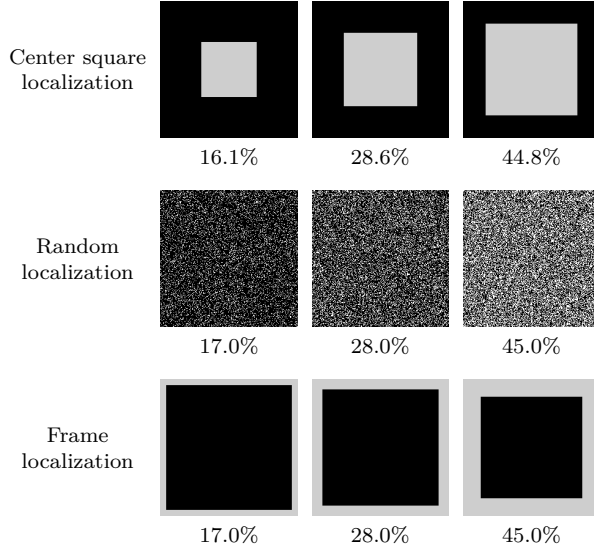


Figure 3.11: The localization masks used the experiments discussed in this chapter. The given percentages correspond to the number of selected pixels compared to the total number of available pixels.

where $\nabla_x J(g(\theta, \mathbf{x}^{(n)}), c)$ represents the gradient with respect to \mathbf{x} , obtained with the CE loss J when targeting the class c . We use $\alpha = 0.004$ as the perturbation multiplier. This corresponds approximately to changing the pixel values of images by $1/255$ at each iteration, performing this attack for 250 iterations. Typically, adversarial attacks such as FGS, IFGS, and PGD enforce a constraint on the magnitude $\|\mathbf{x} - \hat{\mathbf{x}}\|_p$ of the perturbation. However, in order to make a valid comparison between adversarial examples in terms of ℓ_0 , ℓ_2 , and ℓ_∞ norms, we only enforce a discretization constraint, thus ensuring that the produced adversarial examples can be represented as valid images (i.e., the values of $\hat{\mathbf{x}}$ lie within the range $[0, 1]$ so that it is convertible to regular image by denormalization).

In order to localize the perturbation to selected regions, we employ a similar approach to [98], thus making use of

$$\hat{\mathbf{x}}^{(n+1)} = \hat{\mathbf{x}}^{(n)} + \mathbf{p}^{(n)} \odot \mathbf{l}, \quad (3.38)$$

where \mathbf{l} is a *localization mask* (that is, a binary mask of the same shape as the input). In this mask, regions where the perturbation needs to be applied are set to 1, while the remainder is set to 0.

We evaluate the use of three different perturbation regions, each with three different settings. These regions are (1) randomly selected pixels, (2) center square pixels, and (3) outer frame pixels. For (1), we randomly select $\{45\%, 28\%, 17\%\}$ of all pixels, where these percentages approximately correspond to a center square with a side length of $\{90, 120, 150\}$ pixels and an outer frame with a width of $\{20, 34, 58\}$ pixels, respectively. Thus, the number of selected pixels for all regions in each of the three different settings is virtually the same. Visual examples of the localization masks are provided in Figure 3.11. Note that for the perturbation localized on the image frame, unlike Zajac et al. [226], we do not expand the size of the image. We simply exercise the perturbation on the selected outermost pixels.

3.10.3 Experimental results

We first analyze model-to-model transferability (also called black-box transferability) for adversarial examples with localized perturbation. For each model-to-model pair, we generate 2,000 adversarial examples that transfer from the source model to the target model. Using the same initial images as these adversarial examples do, we now apply perturbation to nine different regions (i.e., three regions, with each region coming with three different settings). In Figure 3.12, we present the percentage of adversarial examples that transfer from model to model when localized perturbation is applied, as opposed to performing the adversarial attack without any localization constraints. We see that a large portion of adversarial examples maintains model-to-model transferability when perturbation is applied to local regions.

For the adversarial examples that maintain model-to-model transferability, Table 3.2, Table 3.3, and Table 3.4 provide exhaustive details on the mean and standard deviation of the ℓ_0 , ℓ_2 , and ℓ_∞ properties of the produced adversarial examples, respectively.

As can be seen from the aforementioned results, adversarial perturbation applied to the center square of an image reduces the mean ℓ_2 norm while it increases the mean ℓ_∞ norm. However, through additional experiments, we could observe that 43% of the individual adversarial examples with localized perturbation have lower ℓ_∞ distances than their non-locally perturbed counterparts, showing that localized perturbation nevertheless reduces the ℓ_∞ norm for a large number of cases. Since the perturbation region is what is controlled in this experiment, the resulting perturbations have much less ℓ_0 deviation compared to ℓ_2 or ℓ_∞ .

		Target Model		
		AlexN.	VGG	ResN.
Source Model	ResN.	—	73%	66%
	VGG	60%	—	56%
	AlexN.	52%	59%	—

(a) 17% of pixels selected

		Target Model		
		AlexN.	VGG	ResN.
Source Model	ResN.	—	76%	67%
	VGG	70%	—	63%
	AlexN.	67%	65%	—

(b) 28% of pixels selected

		Target Model		
		AlexN.	VGG	ResN.
Source Model	ResN.	—	78%	75%
	VGG	83%	—	75%
	AlexN.	78%	77%	—

(c) 45% of pixels selected

Figure 3.12: Percentage of adversarial examples with localized perturbation that transfer from the source model (generated from) to the target model (tested against) when 17%, 28%, and 45% of the pixels are selected, respectively (combining all three localization approaches).

Another important observation we could make is the difference in perturbation for different regions. As can be seen, not all regions are equally important when it comes to manipulating the prediction of a DNN with adversarial perturbation. We can clearly observe that adversarial perturbation applied to the center square is more influential than perturbation applied to other regions. Surprisingly, applying perturbation to randomly selected pixels requires less distortion than applying it to the frame of an image, further highlighting the differences between *important* and *unimportant* regions. Allowing perturbation in a more condensed area versus a more expanded area provides different results for the center square region and the other two regions. Increasing the number of selected pixels in the center square region also increases the ℓ_2 norm of the perturbation, while doing so for frame and random pixels reduces the aforementioned norm.

In Figure 3.13, we provide a number of qualitative examples, showing the ℓ_0 , ℓ_2 , and ℓ_∞ norms of adversarial perturbation generated using various localization settings. All of the examples presented in Figure 3.13 are generated using AlexNet and transfer to ResNet-50.

For the experiments discussed, Figure 3.14 provides the percentage of adversarial examples that have lower ℓ_p norm than their counterparts generated with “global” perturbation. Our experiments show that regional perturbation almost always leads to lower ℓ_0 norms compared to non-regional perturbation, whereas in the case of ℓ_2 and ℓ_∞ norms, this depends on the initial image-target class combination.

Table 3.2: Mean (standard deviation) ℓ_0 distances calculated between genuine images and their adversarial counterparts, and where the adversarial counterparts have been created through localization of the perturbation (see the first column). Adversarial examples are created by the source models listed in the first row and transfer to the target models listed in the second row.

Source:		AlexNet		VGG-16		ResNet-50	
Target:		VGG-16	ResNet-50	AlexNet	ResNet-50	AlexNet	VGG-16
Norm:		ℓ_0		ℓ_0		ℓ_0	
No Localization		0.93 (0.08)	0.94 (0.07)	0.90 (0.07)	0.84 (0.11)	0.89 (0.10)	0.83 (0.10)
Center	90px	0.15 (0.01)	0.15 (0.01)	0.15 (0.01)	0.15 (0.01)	0.15 (0.01)	0.14 (0.01)
	120px	0.27 (0.01)	0.28 (0.01)	0.27 (0.01)	0.26 (0.03)	0.27 (0.01)	0.26 (0.03)
	150px	0.43 (0.02)	0.43 (0.02)	0.43 (0.02)	0.40 (0.04)	0.43 (0.02)	0.41 (0.04)
Frame	20px	0.16 (0.01)	0.16 (0.01)	0.16 (0.01)	0.16 (0.01)	0.16 (0.01)	0.16 (0.01)
	34px	0.27 (0.01)	0.27 (0.01)	0.27 (0.01)	0.26 (0.02)	0.27 (0.01)	0.26 (0.02)
	58px	0.43 (0.02)	0.43 (0.02)	0.43 (0.02)	0.42 (0.04)	0.43 (0.04)	0.42 (0.03)
Random	17%	0.16 (0.01)	0.16 (0.01)	0.16 (0.01)	0.16 (0.01)	0.16 (0.01)	0.15 (0.01)
	28%	0.28 (0.01)	0.28 (0.01)	0.28 (0.01)	0.27 (0.02)	0.28 (0.01)	0.27 (0.02)
	45%	0.44 (4.80)	0.43 (0.11)	0.43 (4.20)	0.41 (0.09)	0.43 (3.98)	0.41 (0.09)

3.11 Conclusions and future work

In this chapter, we explained the concept of adversarial attacks against DNNs, reviewing a number of popular attacks that have been described in the literature and discussing a number of examples that illustrate how these attacks can be employed. We then generalized a number of popular methods for adversarial example generation, making their objective functions mathematically comparable in terms of the way perturbation is generated. By doing we, we were able to establish that each of these objective functions falls into one out of two categories: the ones that use cross-entropy loss and the ones that use logit loss. Next, we demonstrated that the gradient of the cross-entropy loss can be approximated differently for three prediction subspaces. We established both mathematically and empirically that using the CE loss — or any other loss function that, at its core, relies on CE — has

Table 3.3: Mean (standard deviation) ℓ_2 distances calculated between genuine images and their adversarial counterparts, and where the adversarial counterparts have been created through localization of the perturbation (see the first column). Adversarial examples are created by the source models listed in the first row and transfer to the target models listed in the second row.

Source:		AlexNet		VGG-16		ResNet-50	
Target:		VGG-16	ResNet-50	AlexNet	ResNet-50	AlexNet	VGG-16
No Localization		7.35 (5.37)	6.39 (4.50)	6.91 (4.17)	3.62 (3.16)	6.79 (4.31)	3.76 (2.20)
Center	90px	6.55 (4.36)	5.33 (3.52)	4.01 (2.64)	3.41 (2.77)	3.54 (2.54)	2.79 (2.23)
	120px	6.47 (4.48)	6.30 (4.45)	5.01 (2.99)	3.68 (3.05)	4.50 (2.93)	3.70 (3.09)
	150px	6.80 (4.48)	6.46 (4.33)	6.71 (3.79)	3.92 (3.07)	6.64 (3.90)	4.65 (3.54)
Frame	20px	9.86 (8.37)	10.1 (7.94)	6.07 (3.74)	4.64 (3.61)	4.77 (2.88)	4.32 (2.90)
	34px	8.92 (6.60)	8.63 (6.52)	6.71 (4.04)	4.50 (2.96)	5.68 (3.25)	4.85 (3.44)
	58px	8.44 (5.72)	7.23 (4.94)	7.79 (4.17)	5.44 (3.89)	7.02 (3.90)	5.78 (3.79)
Random	17%	8.11 (5.63)	7.41 (4.63)	5.20 (3.14)	4.43 (3.30)	4.59 (2.65)	3.81 (2.92)
	28%	6.82 (4.99)	7.51 (4.54)	5.97 (3.55)	4.29 (2.91)	5.50 (3.14)	4.35 (3.02)
	45%	7.42 (4.80)	6.76 (4.20)	7.21 (3.98)	4.61 (3.41)	7.39 (4.03)	5.04 (3.53)

only a limited target subspace when creating adversarial examples, as compared to the usage of the logit loss.

Our results show that, when the cross-entropy loss is used as a baseline loss function to generate adversarial examples, it is not possible to create adversarial examples whose logit output exceeds a certain limit, due to mathematical constraints, whereas it is possible to do so when using the logit loss. However, this does not mean that the use of CE is always detrimental to the process of adversarial example generation. On the contrary, compared to all other methods studied in this chapter, the use of CE creates adversarial examples the fastest and with the least amount of ℓ_2 perturbation.

With the experiments discussed in this chapter, we aim at guiding future research efforts that focus on designing novel methods for adversarial example generation, showing the impact of the selected baseline loss function on the behavior of adversarial optimization. We also aim at guiding future research efforts that focus on constructing novel defenses against adversarial examples by exposing the optimiza-

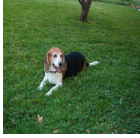
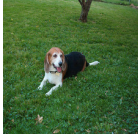
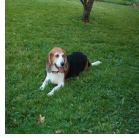
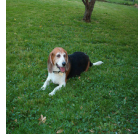
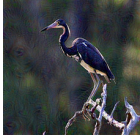

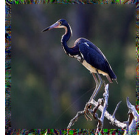
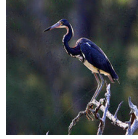












No localization	Center	Frame	Random
			
$\ell_0 = 0.96$ $\ell_2 = 3.25$ $\ell_\infty = 0.02$	$\ell_0 = 0.15$ $\ell_2 = 0.83$ $\ell_\infty = 0.01$	$\ell_0 = 0.17$ $\ell_2 = 3.72$ $\ell_\infty = 0.03$	$\ell_0 = 0.16$ $\ell_2 = 1.14$ $\ell_\infty = 0.01$
			
$\ell_0 = 0.99$ $\ell_2 = 10.5$ $\ell_\infty = 0.12$	$\ell_0 = 0.16$ $\ell_2 = 3.51$ $\ell_\infty = 0.05$	$\ell_0 = 0.17$ $\ell_2 = 25.4$ $\ell_\infty = 0.55$	$\ell_0 = 0.17$ $\ell_2 = 7.02$ $\ell_\infty = 0.09$
			
$\ell_0 = 0.99$ $\ell_2 = 8.80$ $\ell_\infty = 0.10$	$\ell_0 = 0.16$ $\ell_2 = 2.91$ $\ell_\infty = 0.03$	$\ell_0 = 0.17$ $\ell_2 = 5.97$ $\ell_\infty = 0.07$	$\ell_0 = 0.13$ $\ell_2 = 0.44$ $\ell_\infty = 0.01$
			
$\ell_0 = 0.99$ $\ell_2 = 11.9$ $\ell_\infty = 0.16$	$\ell_0 = 0.28$ $\ell_2 = 6.98$ $\ell_\infty = 0.12$	$\ell_0 = 0.25$ $\ell_2 = 4.82$ $\ell_\infty = 0.05$	$\ell_0 = 0.27$ $\ell_2 = 4.39$ $\ell_\infty = 0.04$
			
$\ell_0 = 0.99$ $\ell_2 = 7.06$ $\ell_\infty = 0.06$	$\ell_0 = 0.28$ $\ell_2 = 4.39$ $\ell_\infty = 0.06$	$\ell_0 = 0.28$ $\ell_2 = 6.08$ $\ell_\infty = 0.08$	$\ell_0 = 0.28$ $\ell_2 = 4.51$ $\ell_\infty = 0.04$

Figure 3.13: ℓ_0 , ℓ_2 , and ℓ_∞ distances between the initial images and their adversarial counterparts. The adversarial counterparts originate from the same initial image but were perturbed using different localization methods. All of the adversarial examples successfully transfer to models that they were not created with.

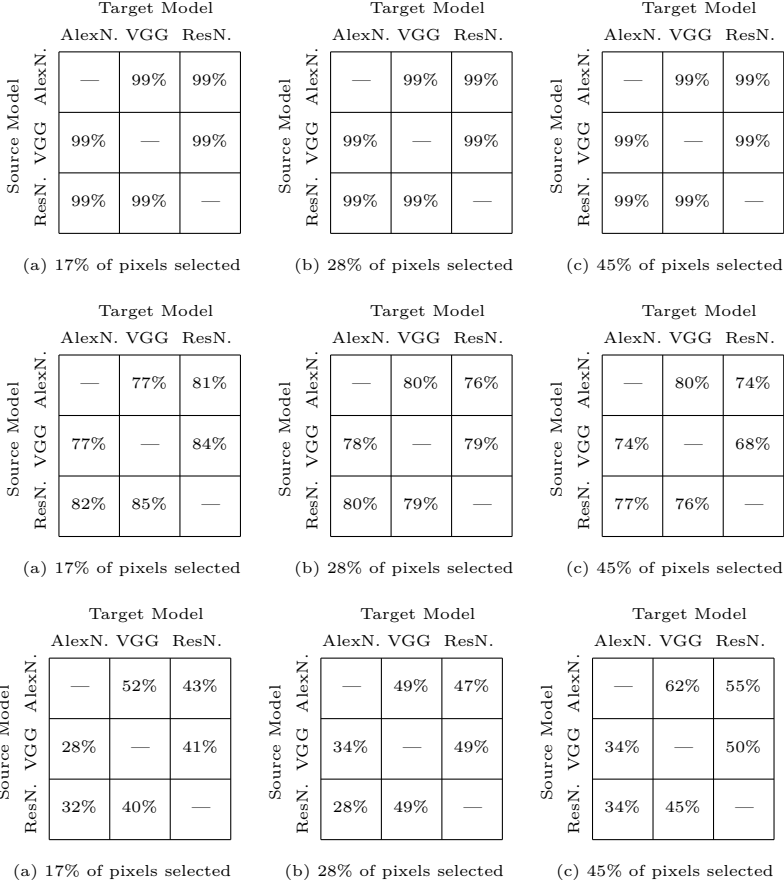


Figure 3.14: The percentage of adversarial examples with regional perturbation that have less perturbation in terms of ℓ_0 norm (top), ℓ_2 norm (middle), and ℓ_∞ norm (bottom) compared to their counterparts with “global” perturbation. Percentages are calculated based on the adversarial examples with localized perturbation that transfer from the source model to the target model.

Table 3.4: Mean (standard deviation) ℓ_∞ distances calculated between genuine images and their adversarial counterparts, and where the adversarial counterparts have been created through localization of the perturbation (see the first column). Adversarial examples are created by the source models listed in the first row and transfer to the target models listed in the second row.

Source:		AlexNet		VGG-16		ResNet-50	
Target:		VGG-16	ResNet-50	AlexNet	ResNet-50	AlexNet	VGG-16
No Localization		0.07 (0.08)	0.05 (0.06)	0.07 (0.05)	0.02 (0.04)	0.07 (0.06)	0.02 (0.02)
Center	90px	0.15 (0.14)	0.11 (0.10)	0.10 (0.09)	0.09 (0.10)	0.09 (0.09)	0.06 (0.07)
	120px	0.11 (0.11)	0.10 (0.11)	0.10 (0.08)	0.06 (0.07)	0.09 (0.08)	0.06 (0.08)
	150px	0.10 (0.10)	0.09 (0.09)	0.11 (0.08)	0.05 (0.06)	0.11 (0.08)	0.07 (0.07)
Frame	20px	0.18 (0.20)	0.20 (0.21)	0.16 (0.14)	0.12 (0.15)	0.13 (0.12)	0.11 (0.11)
	34px	0.13 (0.13)	0.13 (0.14)	0.15 (0.12)	0.08 (0.08)	0.12 (0.09)	0.10 (0.10)
	58px	0.12 (0.13)	0.09 (0.11)	0.14 (0.09)	0.08 (0.09)	0.12 (0.09)	0.09 (0.08)
Random	17%	0.15 (0.16)	0.13 (0.14)	0.13 (0.11)	0.10 (0.12)	0.10 (0.09)	0.08 (0.09)
	28%	0.10 (0.12)	0.11 (0.11)	0.12 (0.10)	0.07 (0.07)	0.11 (0.09)	0.07 (0.08)
	45%	0.10 (0.11)	0.09 (0.09)	0.12 (0.09)	0.06 (0.07)	0.12 (0.08)	0.07 (0.07)

tion space limitation of the cross-entropy loss, which in turn enables defenses that focus on the different subspaces we identified.

In this chapter, we also proposed a simple and general method for localizing perturbations generated by existing adversarial attacks to specific image regions. Our method is experimentally confirmed to be effective, maintaining high black-box transferability at distortion levels significantly lower than the distortion levels required by existing attacks. The reduction in the amount of perturbation achieved by our method raises the concern that existing adversarial defenses may be undermined, since these are usually designed to be effective only against non-local attacks requiring larger perturbation budgets.

A number of future research topics when it comes to regional perturbation are (1) to investigate to what extent our localization method can fool state-of-the-art adversarial defenses and (2) to more precisely identify regions of importance where localized perturbations are highly effective, thus linking the observations made in this study to the interpretability of DNNs.

4

Adversarial attacks on biomedical image segmentation models

In the previous chapter, we investigated the adversarial vulnerability of DNNs designed to solve classification problems. Although such classification models are employed to solve a large variety of problems, segmentation models are another type of DNNs that received a large interest in recent years, especially from the biomedical domain. In this chapter, we will investigate the vulnerability of DNNs used to implement biomedical image segmentation models.

The content of this chapter is based on the following publication:

- Utku Ozbulak, Arnout Van Messem, and Wesley De Neve.
Impact of Adversarial Examples on Deep Learning Models for Biomedical Image Segmentation. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2019.

The research efforts presented in this chapter resulted in the creation of the following software package:

- github.com/utkuozbulak/adaptive-segmentation-mask-attack.
A software package that contains the PyTorch implementation of the Adaptive Segmentation Mask Attack proposed in [148].

This chapter is divided into six sections, the first of which discusses the relevance of adversarial examples in the context of biomedical image segmentation. In Section 4.2, we provide details for the adversarial attack proposed in [148], followed by a discussion of experimental results in Section 4.3 and Section 4.4. Next, we describe the impact of adversarial examples on the medical domain in Section 4.5. Finally, we conclude this chapter in Section 4.6.

Compared to the content of the supporting published paper, the content of this chapter contains a number of changes. Specifically, we have improved the mathematical notation throughout this chapter, adopting the conventions previously outlined in Chapter 2. Our experimental setup also includes an additional dataset and two additional deep learning models, thus making it possible to extend the range of our experiments. Furthermore, we incorporate a large number of additional qualitative results in order to provide more insight into the proposed attack.

4.1 Introduction

In order to solve complex problems related to localization, segmentation, and classification, the medical domain is deploying deep learning models at a quick pace. One reason for the fast adoption of these models is the recent superhuman results obtained by deep neural networks [75, 109, 189]. Although these models come with challenges such as being data hungry and a lack of interpretability [55, 146], DNNs substantially outperform other approaches on many image-related problems [202, 203]. Another reason for the fast adoption of deep learning models is the increasing cost of labor in the medical field (i.e., salaries of nurses, doctors, and other relevant personnel) [49], triggering a strong interest in the augmentation of manual labor with so-called artificial intelligence, and more controversially, in the eventual replacement of the human-in-the-loop with completely automated systems [128, 218]. The latter scenario naturally raises concerns related to cybersecurity and a possible failure of the machine learning models deployed [49, 138]. Although the effects of adversarial examples are largely studied for non-medical datasets, it was also shown that classification problems in medical imaging datasets are of no exception to this exploit [49]. An adversarial example in the context of breast cancer classification was already given in Figure 1.11 in Chapter 1.

In the field of non-medical imaging, pixel by pixel detail is most of the time not task critical. As a result, segmentation problems are often expressed as detection or localization problems [45]. However, in medical imaging, precision is of utmost importance. Therefore, instead of detection or localization, segmentation covers a large portion of medical imaging problems [78].

Many studies are currently being conducted in order to increase

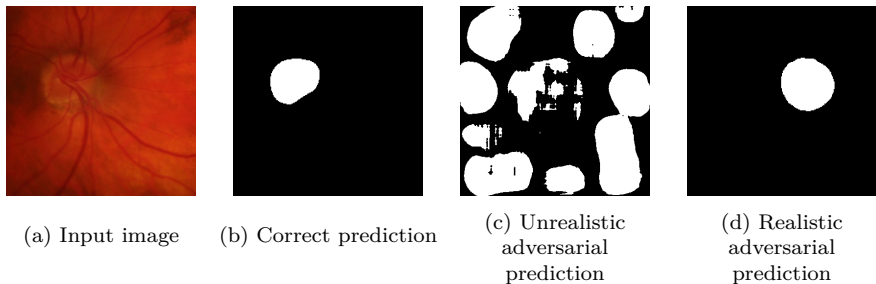


Figure 4.1: (a) An input image for the segmentation model used and (b) its initial (i.e., correct) prediction. When Dense Adversary Generation (DAG) [219] is applied to (a), the prediction (i.e., the segmentation) becomes (c), with the shapes obtained giving away that the input has been tampered with. On the other hand, the proposed technique allows choosing the target mask, making it possible to convert the segmentation prediction to this mask, as shown in (d).

the robustness of machine learning models against adversarial attacks, whilst retaining a high accuracy in solving image-related problems. Most of these studies analyze adversarial attacks in the context of classification problems. However, because many medical problems are effectively segmentation problems [66, 78, 164], we analyze the threat of adversarial examples in the area of medical image segmentation.

In what follows, we first discuss a method that is able to generate targeted adversarial examples in the context of semantic image segmentation, making it possible to change the prediction of an input as we see fit. In Section 4.3, we detail the two datasets and the three DNN architectures used by the study presented in this chapter. In Section 4.4, we lay out our experiments, covering adversarial attacks in both white-box and black-box settings. In addition, we evaluate plausible defenses that incorporate techniques for input modification. Next, in Section 4.5, we discuss the validity of the claim that adversarial examples are a threat in the domain of medical image segmentation. Finally, in Section 4.6, we provide our conclusions and some guidance for future research.

4.2 Adaptive segmentation mask attack

Adversarial attacks on image segmentation models are mostly studied in the context of self-driving cars, focusing on the segmentation of cars, pedestrians, and road signs. In this context, an adversarial attack that was proposed for image segmentation models is Dense Adversary Generation [219]. This attack aims at changing the prediction of a segmentation model in such a way that every pixel is incorrectly classified. Even though this attack was shown to produce adversarial examples reliably, the predictions produced for the generated adversarial examples give away that inputs have been tampered with, especially in the context of medical applications where the segmentation predictions often contain fewer classes and consist of distinguishable shapes. Another approach for generating adversarial examples for segmentation models consists of the usage of the Fast Gradient Sign attack [4]. However, this attack also fails to produce adversarial examples with realistic segmentation predictions.

An example prediction for an adversarial example generated by DAG is given in Figure 4.1(c), for an input image taken from the glaucoma optic disc segmentation dataset used in our research [163], showing that the segmentation predictions look unrealistic. A realistic (adversarial) segmentation prediction is also provided in Figure 4.1(d), where the segmentation prediction is taken from another valid data point in the aforementioned dataset. In this case, it is not immediately clear from the prediction itself whether or not the input has been tampered with. In order to obtain an adversarial attack that is a threat to medical segmentation models, the prediction mask has to be realistic for the type of data that is used. Specifically, whether or not a prediction is real world in nature depends on the dataset. For example, when we talk about a realistic prediction in the context of optic disc segmentation, we expect to see a single, circular segmentation object, rather than seeing multiple objects having arbitrary shapes. We found that the easiest way to ensure that a prediction is realistic is to leverage the prediction obtained for another data point. That way, an adversarial example will have a misleading prediction, but where this misleading prediction is still valid in terms of its expected outcome.

When developing our attack, we placed the aforementioned principle of having a realistic prediction outcomes centrally in our methodology. As a result, our attack allows targeting a pre-specified prediction (i.e., a target mask), and rather than having a random or non-targeted

segmentation prediction, our attack makes it possible to iteratively adjust the initial prediction (which is correct) to this pre-specified segmentation mask.

Our attack applies the standard way of generating an adversarial example by minimizing the ℓ_2 distance between the created adversarial example and the original image in order to ensure minimal (visual) perturbation. This is done as follows:

$$\hat{\mathbf{X}} \in \arg \min ||\mathbf{X} - \hat{\mathbf{X}}||_2^2, \quad (4.1)$$

$$\text{with } \hat{\mathbf{X}} \in [0, 1]^{C \times H \times W}, \quad (4.2)$$

$$\text{such that } \arg \max_M (s(\theta, \hat{\mathbf{X}})) = \mathbf{Y}^A. \quad (4.3)$$

where \mathbf{X} is the (unperturbed) input image, represented as a tensor of size $C \times H \times W$ and where $\hat{\mathbf{X}}$ is the adversarial example created by $\hat{\mathbf{X}} = \mathbf{X} + \mathbf{P}$, with \mathbf{P} representing the generated perturbation. $s(\theta, \hat{\mathbf{X}})$ then denotes a forward pass with the input $\hat{\mathbf{X}}$ (as performed by a segmentation neural network model with parameters θ), resulting in a prediction over M classes, and $\mathbf{Y}^A \in \{0, \dots, M\}^{H \times W}$ is the target (adversarial) mask. Using this approach, the perturbation is then iteratively generated as follows:

$$\mathbf{O}_c^{(n)} = \mathbb{1}_{\{\mathbf{Y}^A = c\}} \odot \mathbb{1}_{\{\arg \max_M (s(\theta, \hat{\mathbf{X}}^{(n)})) \neq c\}}, \quad (4.4)$$

$$\mathbf{P}^{(n)} = \sum_{c=0}^M \nabla_x (s(\theta, \mathbf{X}^{(n)})_c \odot \mathbf{O}_c^{(n)}), \quad (4.5)$$

where $\mathbf{O}_c^{(n)} \in \mathbb{R}^{H \times W}$ represents the optimization mask at the n th iteration for c th class, $\mathbb{1}$ the indicator function, and \odot the Hadamard product. The optimization mask is calculated each iteration, taking into account the target adversarial mask as well as the prediction at n th iteration. Specifically, the first term in the calculation of $\mathbf{O}_c^{(n)}$, $\mathbb{1}_{\{\mathbf{Y}^A = c\}}$ selects the output pixels to target for the c th class chosen with \mathbf{Y}^A . The second term $\mathbb{1}_{\{\arg \max_M (s(\theta, \hat{\mathbf{X}}^{(n)})) \neq c\}}$ identifies the predictions where these predictions are different than c .

This perturbation is calculated by utilizing the prediction obtained for each class c among all M classes and is then added to the image at every iteration:

$$\hat{\mathbf{X}}^{(n+1)} = \hat{\mathbf{X}}^{(n)} + \alpha \mathbf{P}^{(n)}, \quad (4.6)$$

where α denotes the perturbation multiplier.

When α is determined in a static manner (i.e., a fixed value is used), we discovered that the optimization halts at later iterations (that is, when the prediction gets closer to the target mask). This behavior can be attributed to a reduction in the number of pixels that can act as a source for the optimization. Therefore, in order to overcome this problem, we introduce a dynamic perturbation multiplier strategy that increases the numerical value of α as the prediction of the adversarial example gets closer to \mathbf{Y}^A . Specifically, we use $\alpha^{(n)} = \beta \times \text{IoU}(\mathbf{Y}^A, S(\theta, \hat{\mathbf{X}}^{(n)})) + \tau$, where β and τ are parameters used to calculate the final perturbation multiplier, making it possible to tune the perturbation multiplier to the selected model and dataset. This perturbation multiplier also takes into account the intersection-over-union score of the prediction $S(\theta, \hat{\mathbf{X}}^{(n)})$ at the n th iteration, with the IoU score being calculated as:

$$\text{IoU}(\mathbf{Y}^1, \mathbf{Y}^2) = \sum_{i,j} A_{i,j} \cap B_{i,j} / \sum_{i,j} B_{i,j}, \quad (4.7)$$

for $(i, j) \in (\{1, \dots, H\}, \{1, \dots, W\})$, and where $A_{i,j} = \mathbb{1}_{\{\mathbf{Y}_{i,j}^2 = \mathbf{Y}_{i,j}^1\}}$ and $B_{i,j} = \mathbb{1}_{\{\mathbf{Y}_{i,j}^2 + \mathbf{Y}_{i,j}^1 \neq 0\}}$.

We have named the resulting attack Adaptive Segmentation Mask Attack (ASMA), given its adaptive mask targeting feature. Indeed, our attack dynamically changes the optimization mask (i.e., $\mathbf{O}_c^{(n)}$ in Figure 4.3 and in Figure 4.4) from which the gradient is sourced, based on the prediction at that particular iteration ($s(\theta, \hat{\mathbf{X}}^{(n)})$), hereby ensuring that the gradients target correct pixel/label combinations.

In order to clarify the optimization procedure performed by our attack, we provide Figure 4.2, showing the target adversarial mask, the adaptive optimization masks, and the generated adversarial example.

The foundation of ASMA is similar to Carlini and Wagner’s attack [22], where an ℓ_2 norm minimization is incorporated instead of an ℓ_∞ limit. The latter is the approach used in several variations of the Gradient Sign attack [62, 110, 126]. The degree of the perturbation needs to be determined experimentally by tuning both τ and β . Although the proposed attack has the possibility of incorporating the signature of the gradient and an ℓ_∞ limit, we opted not to do so, given our recent findings regarding the detrimental effects of using the sign-function and the cross-entropy loss [150].

Thanks to the flexibility of ASMA, the proposed attack can be used for both (1) fabricating a new segmentation object that keeps the prediction realistic and (2) misleading the initial segmentation

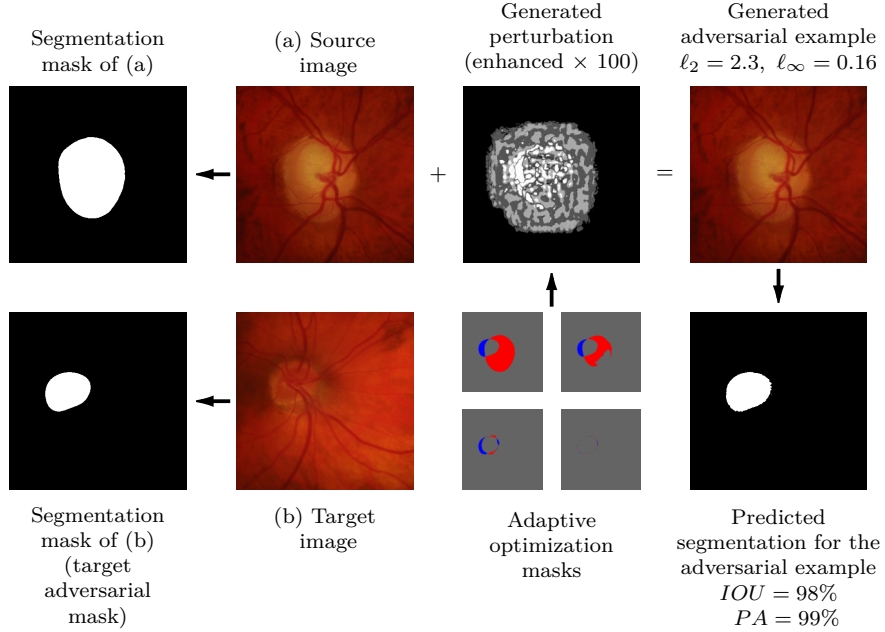


Figure 4.2: An example optimization of an adversarial example for attacking segmentation models, with the optimization being performed by ASMA.

target to minimize the segmentation accuracy. We now detail these two options for the generation of adversarial examples using ASMA.

4.2.1 Fabricating a new segmentation object

In this first approach, the aim is to create adversarial examples that have somewhat realistic segmentation predictions. An illustration for this case is provided in Figure 4.1, where the segmentation prediction in (c) immediately reveals that the input is tampered with, whereas in (d), input tampering is not obvious by only looking at the prediction.

For this approach, we can simply use a target mask (\mathbf{Y}^A) that is taken from a similar, but different, data point. For example, we can use a target mask from another sample in the selected dataset which will result in a visually valid prediction. When such a sample segmentation target is not available, it is also possible to just draw a shape similar to that of the object of interest. For the specific case of the glaucoma dataset, this would mean that we could draw a random circle-like target segmentation mask. Naturally, for more complex segmentation

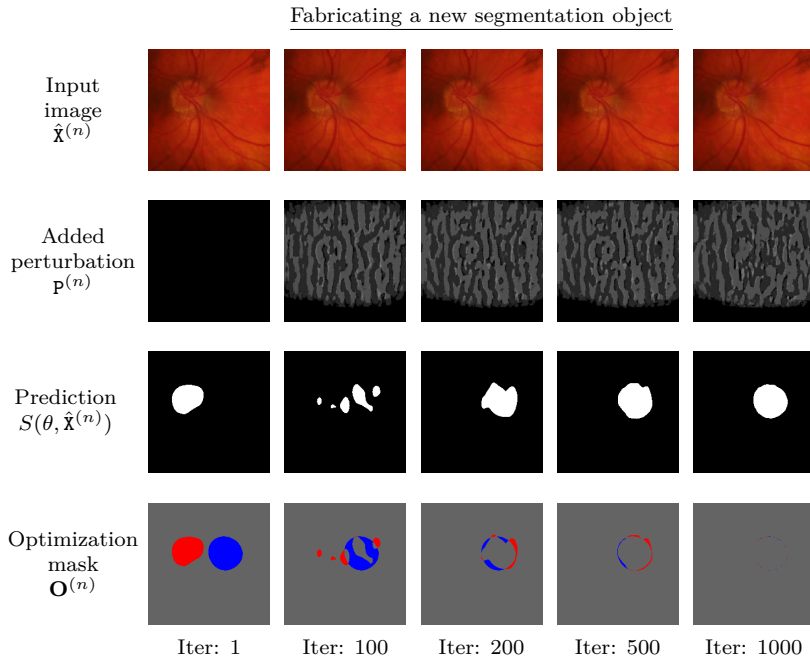


Figure 4.3: The input image, the added perturbation compared to the initial image, the prediction for the input image, and the optimization mask used by ASMA are provided for fabricating a new segmentation object. Red areas in the optimization masks represent spaces that ASMA aims at converting to background (i.e., that need to be changed to black) and blue areas denote spaces that ASMA aims at converting to foreground (i.e., that need to be changed to white).

problems, generating a target segmentation mask also becomes more challenging, in which case the next approach can be employed.

4.2.2 Reducing the segmentation accuracy

Although the previous approach results in a reduction of the segmentation accuracy, its purpose is to create a prediction that is looking realistic. However, we can also formulate ASMA in such a way that it directly reduces the segmentation accuracy, by forcing the model to incorrectly predict every single pixel. This can be achieved by simply using the prediction of the initial sample and selecting the target mask as follows: $\mathbf{Y}^A = 1 - \arg \max_M (s(\theta, \mathbf{X}))$. That way, the attack will create an adversarial example with pixels that are all misclassified.

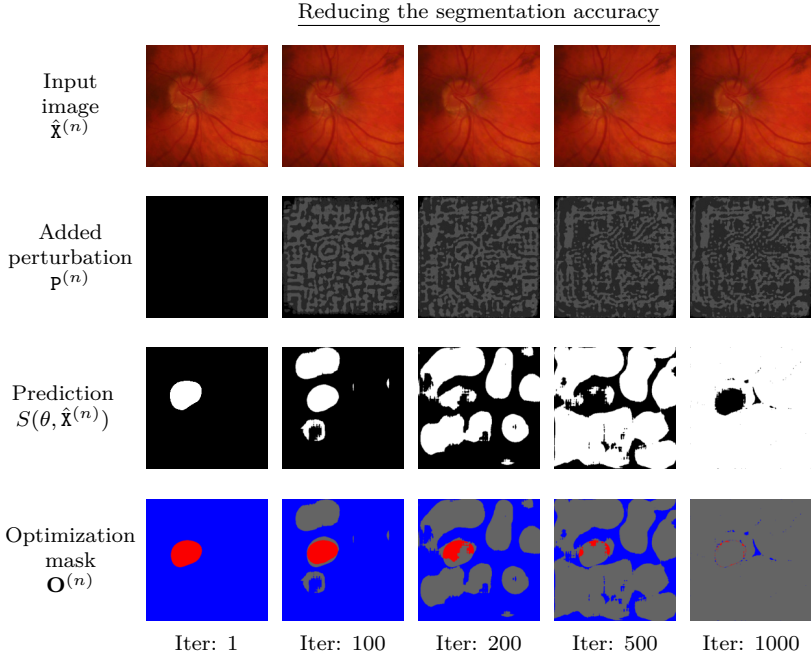


Figure 4.4: The input image, the added perturbation compared to the initial image, the prediction for the input image, and the optimization mask used by ASMA are provided for reducing the segmentation accuracy. Red areas in the optimization masks represent spaces that ASMA aims at converting to background (i.e., that need to be changed to black) and blue areas denote spaces that ASMA aims at converting to foreground (i.e., that need to be changed to white).

Figure 4.3 and Figure 4.4 visualize the proposed optimization procedure for both of the aforementioned approaches, showing the change in the optimization mask related to the change in the prediction across several iterations.

4.3 Data and deep learning models

For the research effort presented in this chapter, the first dataset we opted to make use of is the glaucoma optic disc segmentation dataset published in [163]. This dataset, which contains RGB images of eyes, aims at improving optic disc segmentation in order to help detect associated disease(s) [177]. In particular, we chose this dataset because

its images contain a single, often circular shape that facilitates a clean demonstration of the properties of our attack, while at the same time making it possible to illustrate the outcomes obtained with clear visual examples.

Table 4.1: Train and test accuracies of selected models on the glaucoma and cell segmentation datasets.

Model	Glaucoma dataset		Cell dataset	
	Train	Test	Train	Test
Fcn8s	96%	93%	85%	79%
SegNet	97%	92%	87%	80%
UNet	99%	95%	89%	82%

The second dataset we use is the cellular structure segmentation dataset published in [124], targeting the detection of mitochondria in electron microscopy images. This segmentation task is not only more challenging than the optic disc segmentation task, the images in this dataset are also grayscale in nature. As a result, to the best of our knowledge, this is the first time adversarial threats in the area of segmentation are evaluated for a dataset that only contains grayscale images, given that most research efforts work with datasets of color images related to self-driving cars [4, 29, 131, 219].

We use three popular encoder-decoder style segmentation networks, namely Fcn8s [122], SegNet [8], and UNet [170]. Although all of these networks are fully convolutional neural networks, the underlying architectures and the number of trainable parameters are vastly different (134M for Fcn8s, 29M for SegNet, and 7M for UNet). A visual summary of the selected networks can be found in Figure 4.5. We scale the input image size to 224×224 in order to be able to use Fcn8s and SegNet, given that the encoder part of both networks is derived from the VGG-16 model [189], and where the encoder part has also been pretrained on the ImageNet dataset [175]. We train these models on the glaucoma dataset for 100 epochs and on the cell dataset for 250 epochs, using Adam as the optimizer [103] with a learning rate of 0.0001. The accuracies of the models on the training and testing sets of the selected datasets are given in Table 4.1. Although the accuracy of the UNet model is slightly superior to the accuracy of the other two models, all three models achieve comparable results, thus making them suitable for experimentation with adversarial attacks.

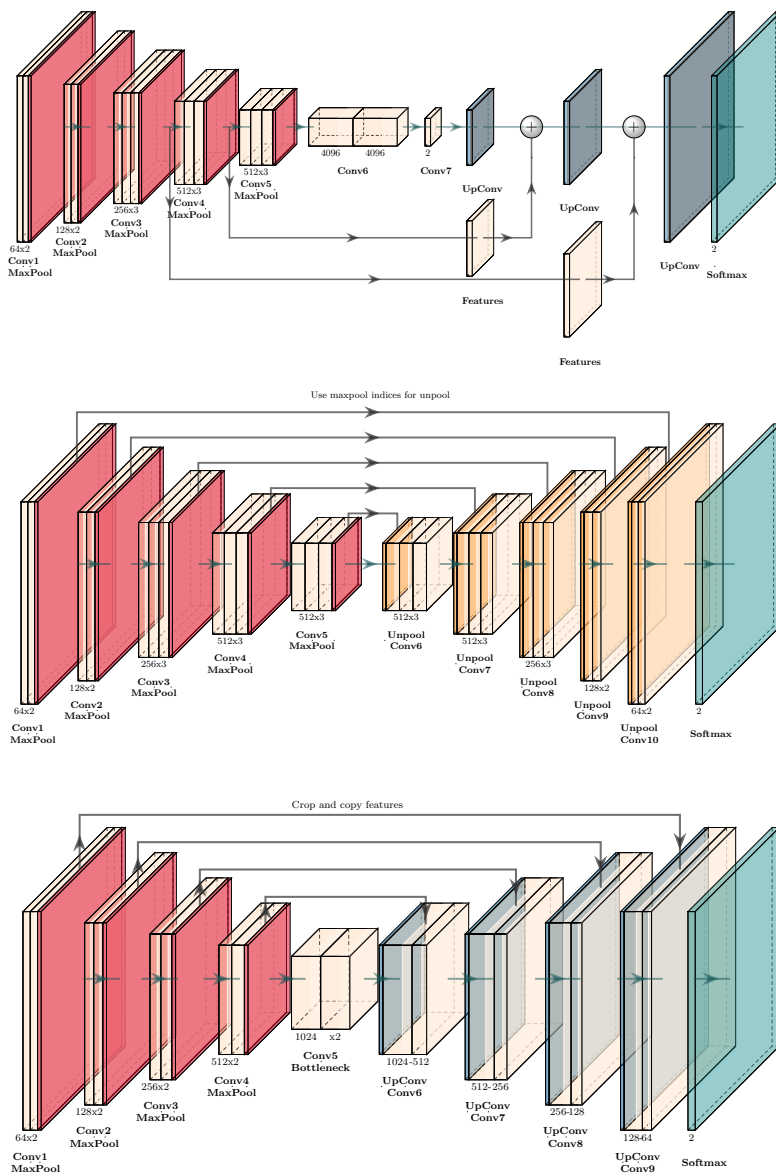


Figure 4.5: Visualization of the overall architecture of (top) Fcn8s, (middle) SegNet, and (bottom) UNet .

4.4 Experiments

4.4.1 Quantifying the threat of adversarial attacks

In order to quantify the threat of adversarial attacks, we employ ASMA to produce two types of adversarial examples, as described in Section 4.2, for both the glaucoma optic disc dataset and the cellular structure dataset.

Fabricating a new segmentation object—In order to generate adversarial examples with predictions that look realistic, we use segmentation masks taken from other data points in the same dataset as the target mask. This approach makes it possible to obtain authentic predictions for adversarial examples since the prediction is taken from a valid data point. For this experiment, we quantify the IoU accuracy between the target masks and the predictions generated for our adversarial examples ($IoU(\mathbf{Y}^A, \arg \max_M(s(\theta, \hat{\mathbf{X}})))$), while also measuring the ℓ_2 and ℓ_∞ norm of the perturbation introduced. Our experimental results are presented in Figure 4.6.

Minimizing the segmentation accuracy—In order to generate this type of adversarial examples, we first perform a forward pass with the data point at hand in order to obtain its prediction. We then use the inverse of this prediction as the target mask: $\mathbf{Y}^A = 1 - \arg \max_M(s(\theta, \mathbf{X}))$. This approach effectively aims at minimizing the segmentation accuracy since it simultaneously tries to convert areas that contain segmented objects to background and vice versa. For this experiment, we quantify the absolute reduction in IoU between the original segmentation masks and the predictions generated for the adversarial examples created. Similarly, we quantify the amount of added noise by measuring the ℓ_2 and ℓ_∞ norm of the perturbation introduced. Experiments for this case are presented in Figure 4.7.

For the two experiments described above, we generate 1,000 adversarial examples for each of the three segmentation models, using $\tau = 10^{-7}$ and $\beta = 10^{-5}$. After the creation of these adversarial examples, we use them as input for both the model they were generated from and the two other models, in order to achieve an in-depth understanding of adversarial threats in both white-box and black-box settings. The black-box approach we use is the cross-model adversarial threat [213], where the attacker is assumed to have knowledge about the data used to train a surrogate model and produce adversar-

Reducing the segmentation accuracy

Target model transferability					Perturbation statistics									
Target model														
FcN8s SegNet UNet														
Glaucoma dataset	Source model	UNet	SegNet	FcN8s	Source model	UNet	SegNet	FcN8s	Source model	UNet	SegNet	FcN8s		
	0.64 (0.17)	0.35 (0.14)	0.15 (0.19)		5.34 (1.34)					0.10 (0.07)				
	0.32 (0.18)	0.59 (0.23)	0.17 (0.21)		4.96 (1.52)					0.09 (0.08)				
	0.57 (0.39)	0.45 (0.40)	0.71 (0.22)		5.12 (2.72)					0.10 (0.09)				
(a3) Reduction in IoU accuracy					(b3) ℓ_2 norm					(c3) ℓ_∞ norm				

Target model														
FcN8s SegNet UNet														
Cell dataset	Source model	UNet	SegNet	FcN8s	Source model	UNet	SegNet	FcN8s	Source model	UNet	SegNet	FcN8s		
	0.43 (0.27)	0.27 (0.11)	0.24 (0.09)		7.52 (7.79)					0.25 (0.22)				
	0.19 (0.09)	0.42 (0.18)	0.18 (0.11)		7.69 (5.93)					0.26 (0.21)				
	0.29 (0.29)	0.21 (0.21)	0.45 (0.23)		9.93 (6.77)					0.27 (0.24)				
(a4) Reduction in IoU accuracy					(b4) ℓ_2 norm					(c4) ℓ_∞ norm				

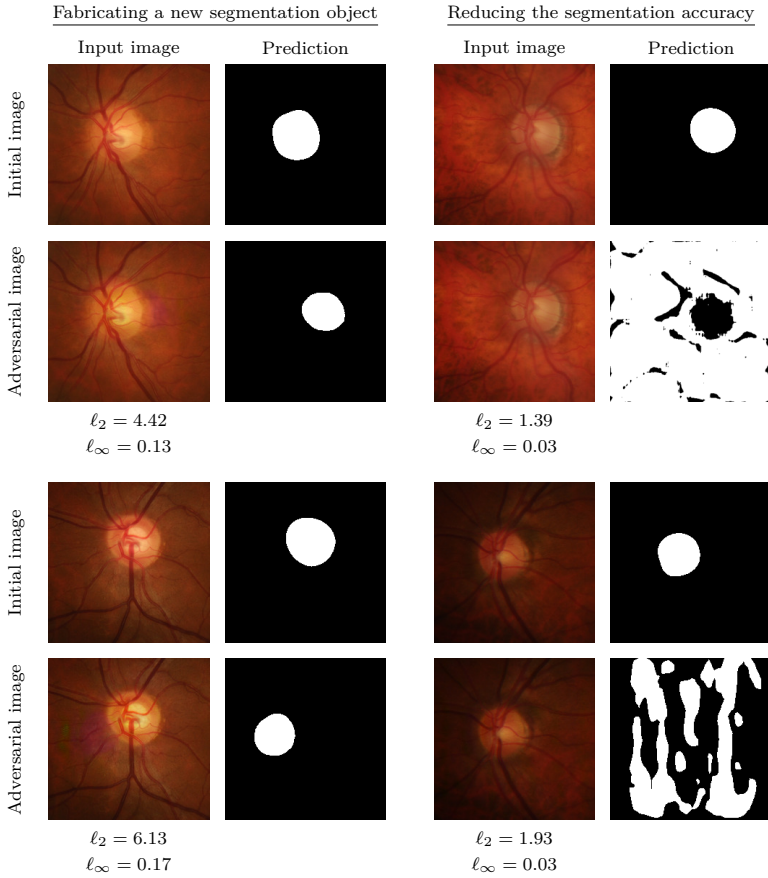


Figure 4.8: A set of genuine images is provided, together with their adversarial counterparts and the corresponding segmentation predictions. Below each adversarial image, the ℓ_2 and ℓ_∞ norm of the perturbation are given, illustrating the degree of the required changes. Data points are taken from the glaucoma optic disc segmentation dataset and the segmentation predictions are taken from the UNet model.

ability for perturbation generation. Since data points taken from the glaucoma dataset (RGB images containing $3 \times 224 \times 224$ individual points) contain three times more perturbation space than samples taken from the cell dataset (grayscale images containing $1 \times 224 \times 224$ individual points), generating adversarial examples for the former dataset is easier. This observation was also made in [188], where the authors found that an increase in input dimension also increases the adversarial vulnerability.

- It is not only harder to generate adversarial examples for the

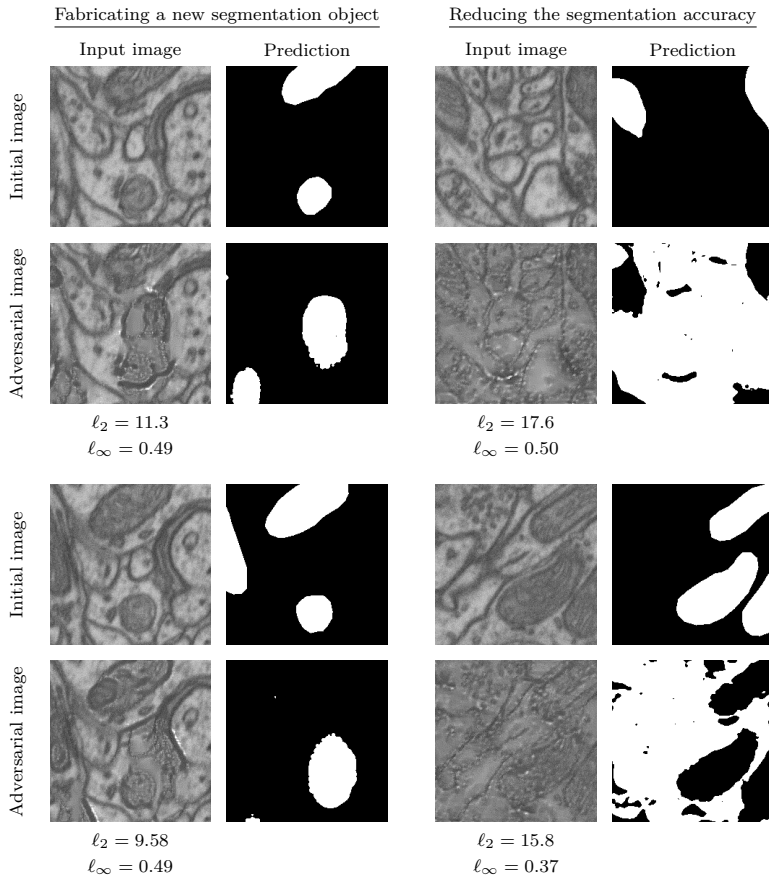


Figure 4.9: A set of genuine images is provided, together with their adversarial counterparts and the corresponding segmentation predictions. Below each adversarial image, the ℓ_2 and ℓ_∞ norm of the perturbation are given, illustrating the degree of required changes. Data points are taken from the cell segmentation dataset and the segmentation predictions are taken from the UNet model.

data points taken from the cell dataset, in order to have results comparable to the glaucoma dataset in terms of IoU, the required perturbation also becomes larger for both the ℓ_2 and ℓ_∞ norms. The amount of added perturbation sometimes makes the noise visible to the bare eye, as illustrated in Figure 4.9. However, an interesting observation to make is how some of the adversarial examples resemble a natural image that comes with a completely new cell structure. As such, we believe it may be possible to incorporate these adversarial examples into training

to further improve the accuracy of the model, an item left for future research.

- Even in white-box settings, some models seem to be more vulnerable to adversarial attacks than others. Indeed, we observe that the skip connections of UNet make it easier to generate adversarial examples, given that these connections make it possible for backpropagation to reach the initial layers, facilitating modification of an input image without having vanishing gradients [217]. On the other hand, both SegNet and Fcn8s have features that restrict such access, with gradients having to travel completely through the entire architecture before reaching the input image. As a result, the skip connection feature of UNet, which makes this model generally more effective than other models in terms of segmentation accuracy, also makes it surprisingly more vulnerable to adversarial attacks.

4.4.2 Robustness of the generated perturbations

A peculiar observation we made while conducting the experiments above is how the perturbations generated by segmentation models look qualitatively different from the ones generated by classification models. Apart from the perturbation patterns provided in Figure 4.2, Figure 4.3, and Figure 4.4, we also provide a comparison of the adversarial perturbations obtained from classification models and segmentation models in Figure 4.10. As can be seen, the adversarial noise introduced by the segmentation models follows a pattern that seems to depend on the input, which is different from the salt-and-pepper style adversarial noise produced by the classification models, with the latter adversarial noise looking more random-like.

Given the aforementioned observations, we decided to investigate the robustness of the adversarial perturbations generated with ASMA against adversarial perturbation removal techniques that leverage input transformation. The effectiveness of input transformation methods as a defense against FGS-based adversarial examples in segmentation was already experimented with in the work of Arnab et al. [4]. Here, the authors found JPEG encoding and Gaussian blur to be effective methods for noise removal when defenses are not incorporated in adversarial perturbation generation (i.e., complete white-box approaches). As such, we analyze the effectiveness of (1) JPEG encoding [120] and (2) low-pass filtering [116] against adversarial examples

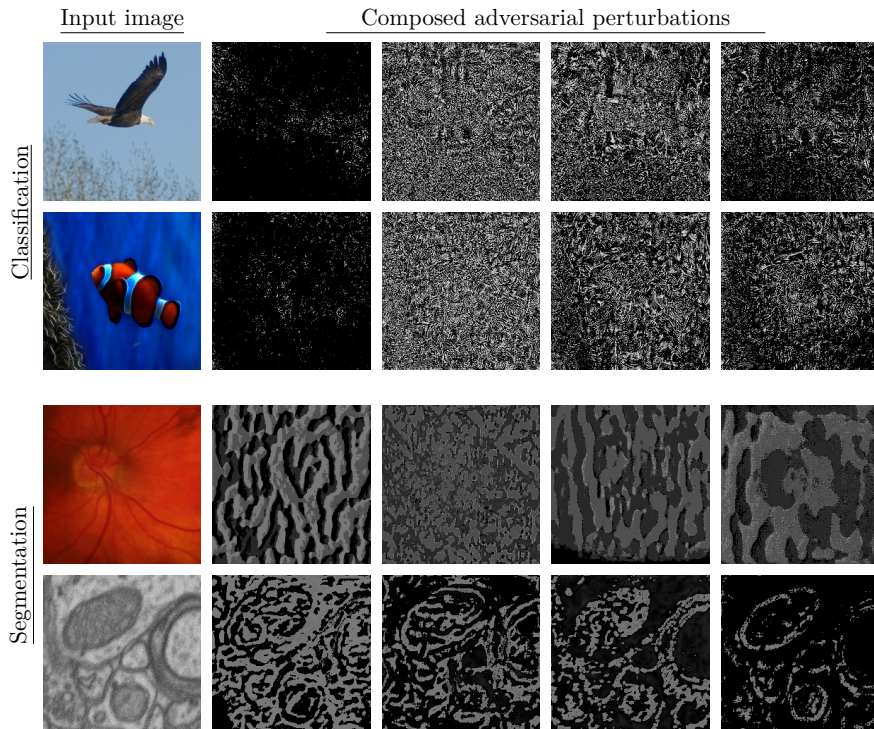


Figure 4.10: Input image and corresponding adversarial perturbations obtained from deep neural networks designed for (top) classification and (bottom) segmentation. Adversarial perturbations for classification are obtained from a ResNet-50 [75] model trained on ImageNet [175], using the LBFGS [201], IFGS [110], PGD [126], and CW [22] attacks, in that order. Using ASMA, perturbations for segmentation are obtained from Fcn8s, SegNet, UNet, and UNet, in that order.

produced by ASMA. We adopt similar settings as employed in [4], using JPEG encoding with 25% compression instead of randomly drawing it between 0% and 50%. For the low-pass Gaussian filtering, we use a kernel size of 3 and a σ -value of 2. These settings ensure that an image is still recognizable after applying an input transformation.

We provide the mean and standard deviation of the absolute reduction in IoU accuracy for low-pass Gaussian filtering and JPEG encoding in Figure 4.11(a) and Figure 4.11(b), respectively. The results presented in this table are directly comparable to Figure 4.7(a3) and Figure 4.7(a4). Based on these results, we observe a significant improvement in segmentation accuracy when input modification tech-

Glaucoma dataset	Source model				Target model		
	Source model			Fcns8s	SegNet	UNet	
	UNet	0.22 (0.10)	0.13 (0.07)	0.12 (0.07)			
	SegNet	0.12 (0.04)	0.17 (0.12)	0.09 (0.07)			
	Fcns8s	0.11 (0.07)	0.12 (0.05)	0.19 (0.11)			
	(a1) Gaussian filter						
	Source model			Fcns8s	SegNet	UNet	
	UNet	0.20 (0.12)	0.10 (0.04)	0.13 (0.08)			
SegNet	0.14 (0.04)	0.19 (0.15)	0.11 (0.09)				
Fcns8s	0.12 (0.05)	0.11 (0.07)	0.21 (0.13)				
(b1) JPEG							

Cell dataset	Source model				Target model		
	Source model			Fcns8s	SegNet	UNet	
	UNet	0.32 (0.21)	0.23 (0.13)	0.21 (0.14)			
	SegNet	0.21 (0.12)	0.38 (0.21)	0.16 (0.13)			
	Fcns8s	0.27 (0.23)	0.23 (0.19)	0.38 (0.23)			
	(a2) Gaussian filter						
	Source model			Fcns8s	SegNet	UNet	
	UNet	0.29 (0.14)	0.18 (0.08)	0.21 (0.07)			
SegNet	0.14 (0.09)	0.27 (0.15)	0.15 (0.07)				
Fcns8s	0.18 (0.16)	0.19 (0.15)	0.31 (0.25)				
(b2) JPEG							

Figure 4.11: Mean (standard deviation) absolute reduction in the IoU accuracy between the prediction and the initial (correct) mask for the same 1,000 adversarial examples after applying (a) low-pass Gaussian filtering and (b) JPEG encoding. The data represented here should be compared to table Figure 4.7(a3) and Figure 4.7(a4) for evaluating the effect of the defenses.

niques are employed to remove adversarial noise, especially in white-box scenarios. However, the average reduction in segmentation accuracy still remains relatively high.

4.5 Discussion

Are adversarial examples a threat to medical image segmentation models? Unlike self-driving cars where the input for machine learning models is not supervised (i.e., the input comes directly from the outside world), for machine learning models in the medical domain, the input comes from other medical devices (e.g., a computer tomography scanner). In this context, for the input to be tampered with, either (1) the medical device that produces the input data has to be compromised or (2) there has to be a mischievous individual present (i.e., an adversary) with security clearance to the underlying system (e.g., medical devices and servers), who is willing to tamper with the input data. In the case of the former, an attack requires a breach of multiple layers of

cybersecurity defenses, such as the network security, the medical device security, and eventually the security of the server that performs the prediction task. In the case of the latter, there are more dangerous acts the adversary within the system could perform compared to tampering the input. In both cases, the vulnerability of the underlying machine learning model to adversarial examples is only a minor security concern compared to other possible threats, thus weakening the argument of adversarial attacks being the most significant threat to machine learning models performing medical image analysis.

Also, given the current use of human-in-the-loop approaches towards medical decision-making, when an adversarial attack is performed on an image segmentation model, bypassing its defense mechanisms, it can be expected that the resulting forced misclassification can still be identified by medical experts, for instance through the unrealistic nature of the segmentation (see Figure 4.1(c)). Although the adversarial attacks for segmentation models in the literature focus on forced misclassification, we have shown both qualitatively and quantitatively that it is possible to create adversarial examples with predictions that look realistic on multiple medical datasets. This implies that, even with human-in-the-loop approaches, it may not always be trivial to detect these adversarial examples.

Moreover, falsifying a medical prediction is not the only use case for adversarial attacks. As pointed out in [49], adversarial examples have a potential to have large implications for individuals living in countries where medical systems heavily rely on insurance policies. Without the coverage of these insurance policies, individuals may pay a large cost for potential medical operations or may even not be able to afford them. As such, pursuing additional studies that cover the legal aspects of these mischievous data points is a necessity.

Even though the threat of adversarial attacks to automated medical diagnosis systems might seem unrealistic in present-day settings, thanks to human-in-the-loop approaches and other systems in place that guarantee input and prediction integrity, in the not-so-distant future, when these diagnosis systems become fully autonomous, such attacks have the potential to be threatening attack vectors. As such, the dangers possibly emanating from these attacks should not be ignored just because they seem to be unrealistic in the current climate.

4.6 Conclusions and future work

In this chapter, we investigated ASMA, a novel gradient-based adversarial attack that can generate targeted adversarial examples for semantic segmentation of medical images, laying out experiments using three popular image segmentation models, namely Fcn8s, SegNet, and UNet. With our experiments, we illustrated that it is not only possible to considerably reduce the prediction accuracy in both white-box and black-box settings, but that it is also feasible to produce adversarial examples that come with realistically looking segmentation predictions. Such adversarial examples pose a significant threat to automated systems for medical image analysis, even when a human-in-the-loop approach is incorporated. Although some of the adversarial examples, as created by ASMA, are shown to be fragile against noise-removal techniques that involve input modification, we demonstrated that, on average, it is possible to reduce the accuracy of the selected model, especially for large perturbation budgets.

Although the simple noise removal techniques we employed in this study were partly effective in eliminating adversarial noise, they were not able to remove all of the perturbation. We believe promising future work in this field should focus on finding a definitive solution for removing adversarial noise from adversarial examples generated by and for image segmentation models. In that regard, we believe advanced noise removal techniques such as Laplacian smoothing [81, 191], total variation denoising [69, 173], and other recent research efforts in the domain of discretization [231] are promising directions of future research.

Another area of future research is the usage of adversarial examples for training. In particular, the adversarial examples, as produced with the approach for fabricating segmentation objects, resemble artificial data points created by generative models, having the potential to improve model accuracy.

5

Adversarial attacks on radar-based activity recognition systems

Although the image domain is recognized as the domain that suffers most from adversarial examples [22] (since the perturbation is often invisible to the bare eye), these malicious inputs are known to be prevalent in other domains. One such domain is the domain of radar signals, where one way to obtain data is through the usage of frequency-modulated continuous-waves [210]. Similar to other fields, researchers that work with radar data were also quick to adopt DNNs to solve a wide variety of problems [95, 215]. One of those problems is radar-based human activity recognition [211]. In this chapter, we will investigate the adversarial vulnerability of a variety of DNNs employed to recognize human activities using radar frames.

The content of this chapter is taken from the following publication:

- Utku Ozbulak, Baptist Vandersmissen, Azarakhsh Jalalvand, Ivo Couckuyt, Arnout Van Messem, and Wesley De Neve. Investigating the Significance of Adversarial Attacks and Their Relation to Interpretability for Radar-based Human Activity Recognition Systems. *Computer Vision and Image Understanding, Special Issue on Adversarial Deep Learning in Biometrics & Forensics*, Elsevier, 2020.

This chapter is organized into seven sections, the first of which explains our rationale for analyzing the threat posed by adversarial attacks to radar-based activity recognition systems. In Section 5.2 and Section 5.3, we pay attention to the framework and the threat model used, followed by a discussion of extensive experimental results in Section 5.4. In Section 5.5, we explore connections between adversarial perturbation and model interpretability. Finally, in Section 5.6,

we review a number of recent adversarial attacks on radar-based deep neural networks, subsequently concluding this chapter in Section 5.7.

The content presented in this chapter is slightly modified compared to the content of the supporting published paper. First, we changed the mathematical notation, adhering to the mathematical conventions used by the other chapters in this dissertation. Second, we moved a number of sections to Chapter 2 and Chapter 3, making it possible to obtain a smoother reading experience. Lastly, we added a section that gives an overview of recent developments in the area of adversarial attacks on radar-based neural networks (Section 5.6).

5.1 Introduction

Recent advancements in the field of computer vision, natural language processing, and audio analysis enabled the deployment of intelligent systems in homes in the form of assistive technologies. These so-called *smart homes* come with a wide range of functionality such as voice- and gesture-controlled appliances, security systems, and health-related applications. Naturally, multiple sensors are needed in these smart homes to capture the actions performed by household residents and to act upon them.

Microphones and video cameras are currently two of the most commonly used sensors in smart homes. The research in the domain of video-oriented computer vision is extensive, and the combined usage of a video camera and computer vision enables a wide range of assistive technologies, including applications related to security (e.g., intruder detection) and applications incorporating gesture-controlled functionalities [230]. However, one of the major drawbacks of using video cameras is their privacy intrusiveness [167]. These privacy-related concerns are, at an increasing rate, being covered by media articles [195]. Furthermore, a largely overlooked aspect of video-assisted technologies in smart homes is that video cameras are able to capture both smart home residents and visitors. Therefore, residents of smart homes need to be aware of the statutory restrictions on privacy invasion.

Low-power radar devices, as complementary sensors, are capable of alleviating the privacy concerns raised over the usage of video cameras. In that regard, the main advantages of radar devices over video cameras are as follows: (1) better privacy preservation, (2) a higher effectiveness in poor capturing conditions (e.g., low light, presence of

smoke), and (3) through-the-wall sensing [232].

Frequency-modulated continuous-wave (FMCW) radars capture the environment by transmitting an electromagnetic signal over a certain line-of-sight. The reflections of this transmitted signal are then picked up by one or more receiving antennas and converted into range-Doppler (RD) and micro-Doppler (MD) frames [26]. These frames contain velocity and range information about all the objects in the line-of-sight (for the duration of the recording). Recent studies show that with the help of (deep) neural networks, it is possible to leverage these RD and MD frames to recognize multiple individuals [94, 210] or to detect human activities with high precision [215]. The aforementioned studies represent these RD and MD frames in the form of a sequence of mono-color images which are supplied as an input to deep CNNs. Three example RD frames and their corresponding video frames for the gesture *swiping left* are given in Figure 5.1.

Given the extensive research in the machine learning community on techniques to prevent adversarial attacks, we analyze the vulnerability of radar-based CNNs to adversarial examples, with the goal of assessing their significance as a security threat in smart homes. For this analysis, we consider the human activity recognition task presented in Vandersmissen et al. [211], in which the proposed models were able to identify human gestures with high precision using a low-power FMCW radar. Our analysis of adversarial attacks covers a wide range of scenarios, from white-box attacks, in which the adversary is assumed to have all the knowledge about the underlying system, to localized attacks on radar frames under strict conditions, in which the adversary is assumed to have limited knowledge about the underlying system. Furthermore, we also attempt to analyze the connection between adversarial attacks and neural network interpretability by investigating the connection between prediction, perturbation amount, and Grad-CAM [181], a popular deep neural network (DNN) interpretability technique.

5.2 Framework

5.2.1 Data

Our experiments are conducted on a dataset of human gestures, containing six different hand-based actions performed in an indoor environment, as first presented in [211]. The different gestures, along

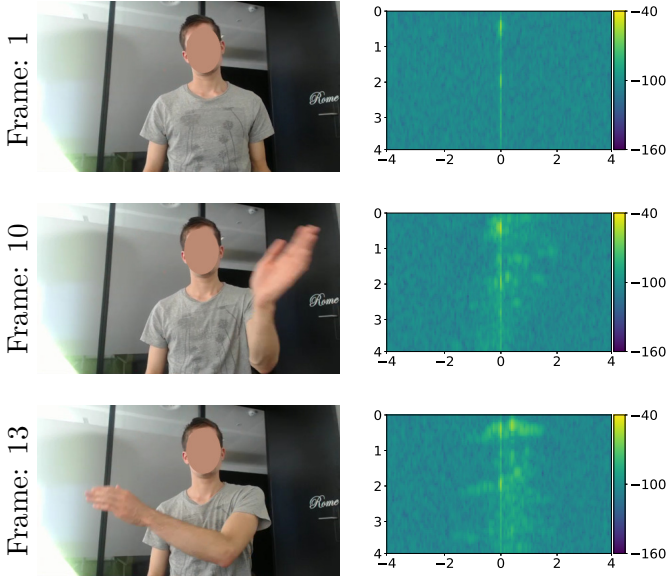


Figure 5.1: (left) Video frames and (right) RD frames for the action *swipe left*. The X -axis and Y -axis of the RD frames, which are omitted for visual clarity, correspond to range and velocity, respectively.

with the number of samples per gesture and their average duration, are listed in Table 5.1. These activities vary from dynamic and clear movements (e.g., *swiping left*) to static (e.g., *thumbs up*) and subtle (e.g., *drumming*) motions.

The samples are recorded using nine different subjects, with each subject repeating each activity several times, and with each subject performing different activities at different speeds and with different pause intervals. This recording approach results in less generic and more diverse activities, given that the length of the activities is not predetermined, nor is their order. The gestures are performed in front of both a radar sensor and an RGB camera, with both devices recording in a synchronized manner. As shown in Table 5.1, the extent of time in which each activity is performed differs significantly per activity class. The dataset contains 2,347 activities in total, with an average duration of 2.16 s per activity, thus making it one of the larger radar datasets concerning human actions [95, 102, 182].

In order to implement a number of threat scenarios, which are discussed in more detail in Section 5.3, and in order to work with a scenario that better reflects real-world settings, we apply a dataset

Table 5.1: An overview of all activities available in the dataset of Vandersmissen et al. [2019].

(Class ID) Activity	Samples	Avg. duration (Std.)
(0) Drumming	390	2.92s (± 0.94)
(1) Shaking	360	3.03s (± 0.97)
(2) Swiping Left	436	1.60s (± 0.27)
(3) Swiping Right	384	1.71s (± 0.31)
(4) Thumb Up	409	1.85s (± 0.37)
(5) Thumb Down	368	2.06s (± 0.42)

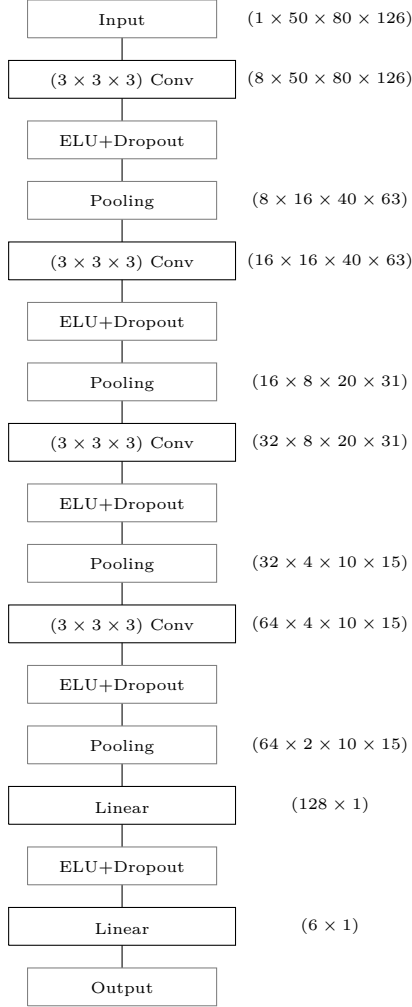
split different from the random stratified split used in [211]. Instead, we use a subject-specific split, ensuring that the data of a single subject are only present in either the training, testing, or validation set.

- (S_+): A subject-specific split, with the training set consisting of samples obtained from subjects 2, 6, 8, and 9. Samples originating from subjects 1 and 7 are used for the validation set and samples obtained from subjects 3, 4, and 5 are used for the test set. This approach leads to a total of 1050, 572, and 725 samples for the training, testing, and validation set, respectively.
- (S_-): This subject-specific split is the opposite of (S_+), which means that the training set contains samples obtained from subjects 1, 3, 4, 5, and 7. Likewise, samples obtained from the subjects 2 and 9 are used for the testing set, while subjects 6 and 8 provide samples for the validation set. This approach leads to a total of 1297, 404, and 646 samples for the training, testing, and validation set, respectively.

In line with the study of Vandersmissen et al. [211], we consider a fixed sample length of 50 frames, which matches the average length of the majority of the activity samples. Samples that are shorter than 50 frames are padded with the median RD frame. This median frame is calculated by using all of the samples in the dataset in order to acquire a padding frame that does not disturb the prediction (i.e., that is not an out-of-distribution sample). For the samples that possess more than 50 frames, only the middle 50 frames are considered.

5.2.2 Models

In this study, we use three substantially different architectures in order to solve the multiclass classification problem of human activity



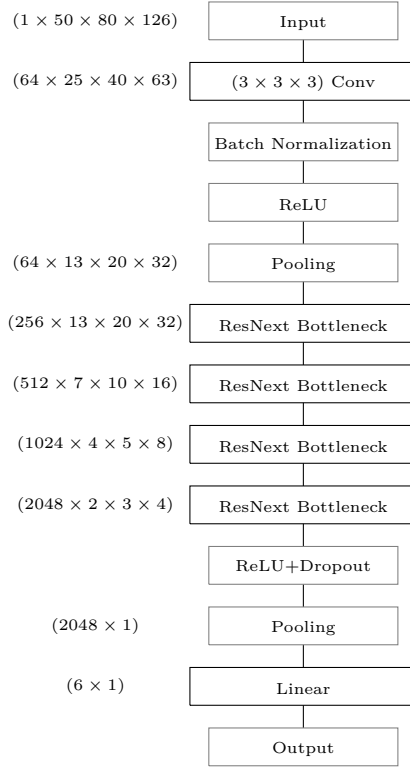
Architecture: \mathcal{A}

Trainable parameters: $\sim 6.4 \times 10^5$

Size: $\sim 2.5\text{MB}$

Figure 5.2: Detailed description for architecture \mathcal{A} .

recognition. The first architecture is the 3D-CNN architecture used in Vandersmissen et al. [211], which we will refer to as \mathcal{A} . Figure 5.2 shows a detailed description of the neural network layers of \mathcal{A} , our 3D-CNN model, as well as the evolving size of the input as it is processed with a forward pass. This architecture is introduced in [211]



Architecture: \mathcal{R}

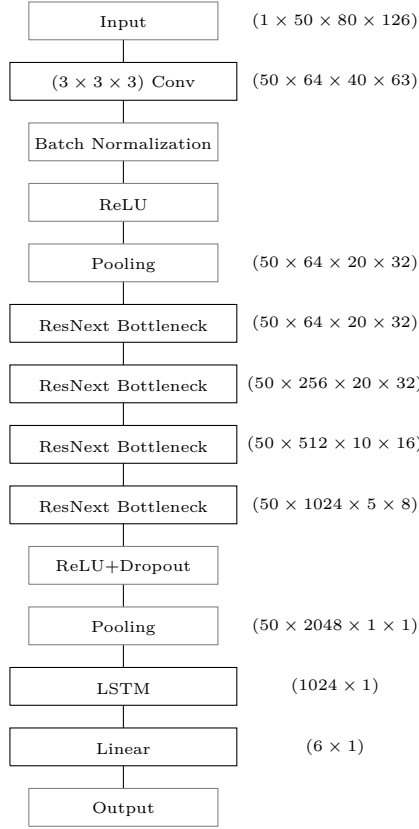
Trainable parameters: $\sim 8.1 \times 10^6$

Size: $\sim 32.9\text{MB}$

Figure 5.3: Detailed description for architecture \mathcal{R} .

as a lightweight model, only taking up a space of about 2.5MB and coming with approximately 6.4×10^5 trainable parameters. The main purpose of this model is to be deployed in household environments. Thanks to its lightweight nature, a prediction can be performed in an efficient way, while reducing the cost of the required hardware. On the dataset splits explained in Section 5.2.1, we train \mathcal{A} for 100 epochs, using Adam [103] with a learning rate of 10^{-3} .

The second architecture is a variant of ResNeXt [222], a relatively new architecture that achieved the second place in ILSVCR 2017 [175]. The design of this architecture is heavily inspired by VGG [189] and ResNet [75]. To handle data with a temporal dimension (e.g., radar or video data), we use a modified version of this architecture (adopted



Architecture: \mathcal{L}

Trainable parameters: $\sim 17.8 \times 10^6$

Size: $\sim 214\text{MB}$

Figure 5.4: Detailed description for architecture \mathcal{L} .

from Hara et al. [73]). In the remainder of this chapter, we will refer to the ResNeXt architecture used as \mathcal{R} . Figure 5.3 shows a detailed description of the second model used in our study (\mathcal{R}). The architecture shown in Figure 5.3, which contains four ResNeXt Bottleneck layers (more detailed information about such layers can be found in Xie et al. [222]), is significantly more complex compared to the model presented in Figure 5.2. Indeed, this model contains approximately 8.1×10^6 trainable parameters (roughly 12 times more than \mathcal{A}). Moreover, this model is also larger in terms of size, taking up a space of about 32.9MB. To add, a single prediction made by the ResNeXt model takes about 9

times longer than a single prediction made by the 3D-CNN model, thus possibly introducing significant time delays when it is deployed on similar hardware. The aforementioned considerations make it challenging for the ResNeXt model to be deployed in household environments with cheap hardware, not only because of its size, but also because of the time required to make a prediction. Nevertheless, we selected this model in order to be able to make a comparison, in terms of adversarial vulnerability, between a simple model that can be easily deployed and a larger model that is more capable. On the dataset splits explained in Section 5.2.1, we train \mathcal{R} for 50 epochs, using Adam [103] with a learning rate of 10^{-5} .

Furthermore, a recent trend in the field of activity recognition is the usage of CNN-LSTM architectures [140, 225, 233], leveraging the underlying CNN as the feature extractor and employing a Long Short-Term Memory (LSTM) [83] layer in order to discover temporal relations. Apart from the usage of the previously explained fully convolutional architectures, we also employ a similarly capable CNN-LSTM architecture (\mathcal{L}) in order to discover differences between fully convolutional and CNN-LSTM architectures in terms of adversarial robustness. Figure 5.4 shows a detailed description of the third and the last model \mathcal{L} used in our study. The architecture of \mathcal{L} is similar to that of \mathcal{R} . However, \mathcal{L} comes with a simple but crucial difference: it is a CNN-LSTM architecture that uses convolutions as feature extractors and that leverages an LSTM layer to discover the underlying relations along the temporal dimension. As such, the convolution operations are only performed on individual frames and not along the temporal dimension. This allows the employed LSTM to discover temporal relations and make judgements based on the information stored over an extended period of time. The down side of this model is its greater complexity in terms of storage and inference time. Due to the addition of an LSTM layer, the size of the model significantly increases compared to \mathcal{R} , containing $\sim 17.8 \times 10^6$ trainable parameters and taking a space of about 214MB. As a result, both a forward pass (prediction) and a backward pass (training) take significantly longer than in the case of \mathcal{A} and \mathcal{R} . These properties pose an important challenge when employing such models using low-cost and low-power equipment in smart homes. The best performing model for this architecture is the one we trained on the dataset splits explained in Section 5.2.1, for 30 epochs, hereby using Adam [103] with a learning rate of 10^{-5} .

Containing 8.1 million trainable parameters, the employed ResNeXt model (\mathcal{R}) is significantly more complex than the 3D-CNN model

(\mathcal{A}), which contains approximately 647 thousand trainable parameters. Consequently, the size of the two models is also considerably different: ResNeXt occupies about 32.9MB of memory, whereas 3D-CNN only takes about 2.9MB. Although the space occupied by each of the models does not make a significant difference for many of the current commercial products, on the same hardware, the 3D-CNN model is up to 7 times faster than the ResNeXt model in terms of inference speed. When considering edge-computing and real-time applications in the context of smart homes, this means that deploying the ResNeXt model will naturally cost more than deploying the 3D-CNN model.

By employing convolutional models that are significantly different in terms of both architecture and the number of trainable parameters, we are able to study the impact of adversarial examples generated by an *advanced* model on a *simpler* model, and vice versa. Moreover, by evaluating the adversarial examples generated by these convolutional models on the CNN-LSTM model, we are able to analyze the effectiveness of non-LSTM adversarial examples on LSTM architectures.

The accuracies of \mathcal{A} , \mathcal{R} , and \mathcal{L} are provided in Table 5.2, as obtained for the evaluation splits. As can be observed from this table, although the number of trainable parameters is significantly different for each architecture, they achieve comparable results. Note that the models trained on the evaluation splits $S_{\{-,+\}}$ achieve slightly lower test and validation accuracies than the models presented in the work of Vandersmissen et al. [211]. This can be attributed to the reduced amount of training data we intentionally assigned to these splits, with the goal of covering a wide range of threat models (see Section 5.3). Throughout this chapter, we adopt the notation $\mathcal{A}_{\text{Dataset split}}$ in order to describe a trained model. For instance, \mathcal{A}_{S_+} means that the model is of architecture \mathcal{A} and that this model has been trained on the training set of S_+ . As will be described in the next section, our approach towards selecting models and creating evaluation splits makes it possible to evaluate a wide range of white- and black-box attack scenarios.

5.3 Threat model

We now discuss the threat models evaluated in this chapter. To that end, recall that activities performed in smart homes cause either a global response, meaning that the assistance of a third party is required

Table 5.2: Validation and testing accuracy of the architectures \mathcal{A} , \mathcal{R} , and \mathcal{L} for each class, as obtained for the respective evaluation splits described in Section 5.2.1.

Class	\mathcal{A}		\mathcal{R}		\mathcal{L}	
	Validation	Test	Validation	Test	Validation	Test
(0)	57%	67%	60%	67%	75%	79%
(1)	88%	86%	80%	81%	58%	72%
(2)	83%	79%	77%	78%	85%	77%
(3)	76%	79%	76%	71%	62%	55%
(4)	55%	68%	54%	69%	55%	52%
(5)	70%	69%	59%	63%	60%	65%
Total	72%	74%	67%	72%	66%	67%

(e.g., calling the police to prevent an intruder from entering a home or calling an ambulance for a health-related emergency situation), or a local, in-house response, meaning that the request of a household resident is related to a functionality confined to the house (e.g., turning on the lights). In this work, we evaluate threat scenarios concerning an adversarial attack to the neural network that is part of the decision making mechanism, which may affect both in-house and out-house functionalities. Naturally, there are also other types of security-related topics that need to be analyzed when a smart home system requests aid from outside the house. However, such topics, which are mainly related to home network security, are deemed out of scope for this work.

Given the context described above, an activity and a corresponding flow of events taking place in a smart home environment are visualized in Figure 5.5. To that end, when any action is performed in a household environment, a radar sensor (in this case, an FMCW radar device) is able to detect this movement, with further processing leading to a sequence of range-Doppler frames. These frames are then sent to an on-site server (small and portable hardware) that contains a CNN, with the CNN performing a prediction for the frames received. Given the prediction made by the CNN, either a functionality inside the house (e.g., lights) or outside the house (e.g., calling emergency services) may be triggered. We conjecture three possible entry points for adversarial attacks: (1) when the radar frames are generated, (2) when the radar frames are being transferred from the detector to the on-site server, and (3) right before the prediction. Given this set-

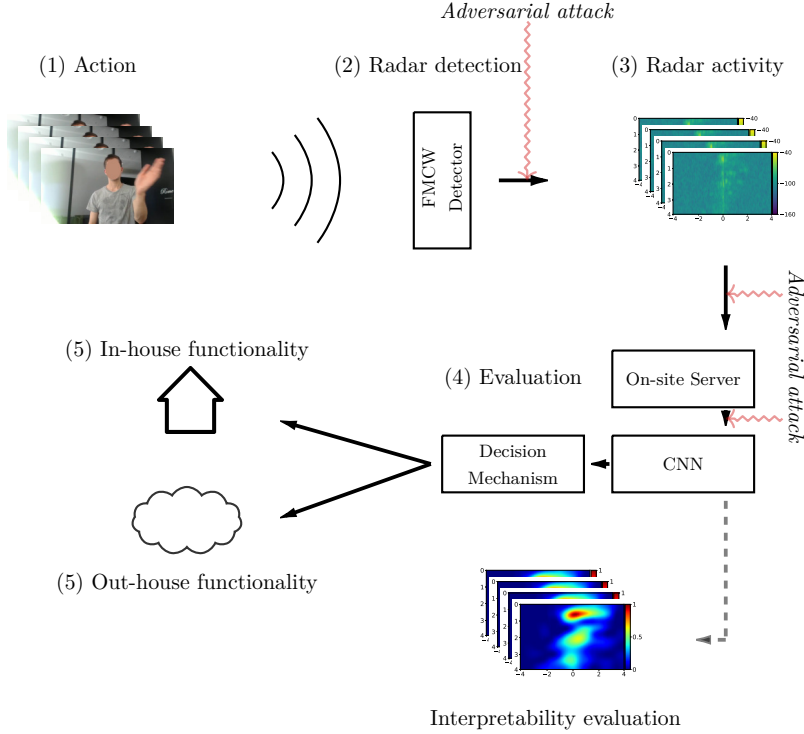


Figure 5.5: A visual summary of the flow of events in a household when an action is performed, including the entry points for possible adversarial attacks.

ting, we evaluate the vulnerability of predictive models that may be deployed in households and perform an investigation of the relationship between adversarial examples and the interpretability of neural networks.

In order to be consistent with past research that studied adversarial examples, we use a taxonomy similar to the one previously outlined in Chapter 2. To that end, among all possible combinations of the attacks listed in Figure 5.6, we only evaluate the two most restrictive cases, which are (1) targeted misclassification and (2) targeted misclassification with a localized attack. Based on the different types of attacks and the level of knowledge of the adversary about the underlying system, we evaluate the following scenarios as threat models:

- **White-box threat model (WB):** The adversary has access to the underlying trained model (including the trained weights) that performs the classification when a radar activity is per-

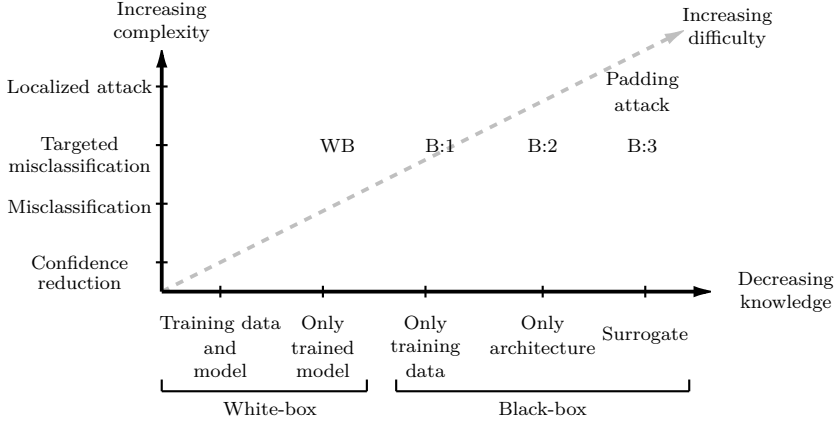


Figure 5.6: Threat configurations and their taxonomy for adversarial examples. Scenarios evaluated in this study are highlighted with their abbreviation.

formed (e.g., adversarial examples generated by \mathcal{A}_{S_+} and tested on \mathcal{A}_{S_+}).

- **Black-box threat models:** The adversary does not have access to the underlying trained model (this includes the trained weights), but the adversary does have access to the following specifications of the underlying decision-making system:
 - Scenario 1 (B:1): The adversary has access to (1) the architecture of the underlying model that performs the classification (without the trained weights) and (2) data from a similar distribution that the underlying model was trained with (e.g., adversarial examples generated by \mathcal{A}_{S_+} and tested on \mathcal{A}_{S_-}).
 - Scenario 2 (B:2): The adversary has access to the training data used to train the underlying model, but not to the exact specifics of this model such as the weights, layers, and nodes (e.g., adversarial examples generated by \mathcal{A}_{S_+} and tested on \mathcal{R}_{S_+}).
 - Scenario 3 (B:3): The adversary neither has access to the underlying model nor the training data used to train this model. However, the adversary has access to data obtained from a similar distribution and a model that is akin to the underlying model (e.g., adversarial examples generated by \mathcal{A}_{S_+} and tested on \mathcal{R}_{S_-}).

5.4 Threat model evaluation

In this section, we analyze the robustness of radar-based CNNs against adversarial examples using the threat models described in Section 5.3. To that end, we investigate the significance of commonly used adversarial attacks, as well as additional attacks that are only possible because of the existence of the temporal domain in the employed input.

During the generation of adversarial examples, when not considering constraints for the generated adversarial examples, (1) the optimization may result in an adversarial example that does not represent a valid input for the targeted neural network or (2) the attack may not be representative of a real-world scenario. In order to avoid such scenarios, we impose a box constraint, a time constraint, and a discretization constraint on the way adversarial examples are generated. A detailed description of these constraints is as follows:

- **Box constraint** – In order to ensure that the generated adversarial example is a valid image, its values are constrained as follows: $\hat{\mathbf{X}} \in [0, 1]$, with 0 denoting black and 1 denoting white. However, different from the image domain, the radar data we use in this study always contain a portion of noise, which limits the values even further when the radar signal is converted to a sequence of RD frames. Thus, for the radar signal, we select the box constraint as $\hat{\mathbf{X}} \in [0.31, 0.83]$, with 0.31 and 0.83 representing the smallest and the largest value present in our dataset, respectively.
- **Time constraint** – The threat scenarios that are tackled in this study consider data obtained from sensors manipulated by an adversary. However, we only assume the adversary to be capable of manipulating the frames (i.e., adversarial attacks). In doing so, we assume there is no delay between the capturing of frames and the transfer of these frames to the underlying model. As a result, in order to work with a realistic attack scenario, we assume there is a limited amount of time available to implement perturbations. In particular, we restrict the amount of time available for adversarial optimization to one second. This limitation approximately corresponds to 200 optimization iterations for IFGS and 72 iterations for the CW attack on a single Titan-X GPU for model \mathcal{A} .

- **Discretization**—As we have described above, the input data are bounded between 0.31 and 0.83. However, when the input is represented as a grayscale image, these values must be represented as integers between 0 and 255. Thus, if a value does not have a direct integer correspondence, it is rounded to the closest integer. Studies that investigate adversarial examples often disregard the discretization property of the produced adversarial examples, hereby providing results for images that are impossible to represent in reality. This topic is discussed in more detail in Carlini and Wagner [22]. In this study, we make sure that the generated adversarial examples can be represented as valid grayscale images.

5.4.1 Evaluating common adversarial attacks

Recently published studies typically aim at improving the *strength* of the produced adversarial examples, making it easier for these adversarial examples to evade deployed defense systems [126]. In the literature, IFGS and CW are often selected as methods for evaluating newly proposed defense mechanisms against adversarial examples. As such, in the study presented in this chapter, we use IFGS and CW to investigate the vulnerability of radar-based CNNs.

For IFGS, we use $\alpha = 15 \times 10^{-4}$, meaning that a single iteration of perturbation will change values by half a pixel value (i.e., $0.5/255$), reporting results based on the use of this perturbation multiplier. For CW, we adhere to the study of Carlini and Wagner [22] and set $\kappa = 20$.

In Table 5.3 and Table 5.4, we present the experimental results obtained in terms of white-box and black-box transferability success, for the adversarial examples created with IFGS and CW. Specifically, Table 5.3 and Table 5.4 details the success rate obtained for 1,000 adversarial examples that originate from unseen data points by their respective models during training time, as well as median ℓ_2 and ℓ_∞ distances between adversarial examples and their initial data points, giving an idea of the minimum amount of perturbation necessary to change the prediction of a model by both attacks. Since we use the ℓ_2 and ℓ_∞ distances between adversarial examples when they transfer successfully from a source model to a target model, we also provide the interquartile range in order to gain insight into the spread of the ℓ_2 and ℓ_∞ distances. We use median and interquartile range over mean and standard deviation in order to mitigate the influence of outliers when the success rate of the attacks is low. Based on this experiment, we make the following observations:

Table 5.3: Median value (interquartile range) of the ℓ_2 and ℓ_∞ distances obtained for 1,000 adversarial optimizations, as well as their success rate, for the models and datasets described in Section 5.2.1. For easier comprehension, the threat models are listed from most permissive to least permissive. ℓ_∞ values less than 0.003 are rolled up to 0.003 (this is approximately the smallest amount of perturbation required to change a pixel value by 1, so to make discretization possible).

Threat model	Source model	Target model	IFGS			CW		
			ℓ_2	ℓ_∞	Success %	ℓ_2	ℓ_∞	Success %
WB	$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{A}_{\{S_+, S_-\}}$	0.79 (0.72)	0.003 (0.003)	100%	0.50 (0.42)	0.02 (0.02)	97%
	$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{R}_{\{S_+, S_-\}}$	0.19 (0.11)	0.003 (0.003)	100%	0.17 (0.11)	0.01 (0.02)	99%
BB:1	$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{A}_{\{S_-, S_+\}}$	2.27 (3.17)	0.006 (0.011)	79%	0.72 (1.17)	0.02 (0.09)	54%
	$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{R}_{\{S_-, S_+\}}$	0.24 (0.08)	0.006 (0.003)	35%	0.20 (0.17)	0.01 (0.03)	31%
BB:2	$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{R}_{\{S_+, S_-\}}$	2.24 (4.77)	0.009 (0.012)	35%	0.92 (0.97)	0.03 (0.11)	30%
	$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{A}_{\{S_+, S_-\}}$	0.24 (0.12)	0.006 (0.003)	28%	0.22 (0.18)	0.03 (0.02)	27%
BB:3	$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{R}_{\{S_-, S_+\}}$	3.12 (5.14)	0.012 (0.07)	30%	0.87 (1.03)	0.03 (0.13)	21%
	$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{A}_{\{S_-, S_+\}}$	0.22 (0.10)	0.006 (0.005)	19%	0.21 (0.27)	0.02 (0.04)	18%

- Unsurprisingly, the success rate of the adversarial attacks decreases as the knowledge of the adversary on the underlying system decreases. In opposition to this trend, the minimal required perturbation to change the prediction of the target model often increases as the knowledge of the adversary decreases.
- More often than not, adversarial attacks with IFGS are more successful than the ones with CW, even though the latter is considered a more advanced attack. Using additional experiments, we observed this can be primarily attributed to the time constraint imposed on the optimization. Since CW is computationally more expensive than IFGS, given its extensive search for a minimum amount of perturbation, generating an adversarial example with CW within the imposed time limit becomes challenging.
- Although IFGS is more successful in generating adversarial ex-

amples than CW, the adversarial examples generated by IFGS come with much stronger perturbations in terms of ℓ_2 distance than those generated by CW. On the other hand, thanks to the $\text{sign}(\cdot)$ function flattening the gradients to an equal level, adversarial examples created with IFGS come with much less perturbation in terms of ℓ_∞ distance. This finding for radar data is also in line with the observations we previously made in the image domain [150].

- Even though ResNeXt models are able to find adversarial examples with less perturbation, the time limit set on the generation of adversarial examples also affects ResNeXt models more than 3D-CNN models, since it takes them longer to perform a prediction, as well as to calculate the gradients for adversarial example generation, ultimately resulting in lower success rates.
- For all black-box cases, the ResNeXt architecture is able to find adversarial examples with much less perturbation than 3D-CNN. Our initial interpretation of this finding was that the adversarial examples generated from *stronger* models transfer with less perturbation when attacking similar or *weaker* models. However, recent results in the area of adversarial research suggest that residual models that contain skip-connections allow generating adversarial examples with much less perturbation [217]. Our experiments also confirm this observation.
- The first three observations we made for the models \mathcal{A} and \mathcal{R} regarding the properties of the selected attacks also hold true for attacks against the CNN-LSTM model \mathcal{L} . We also observe that adversarial examples originating from fully convolutional architectures are able to deceive the CNN-LSTM model without requiring an extra effort (i.e., another specialized attack).
- Experimental results obtained for the CNN-LSTM architecture show that adversarial examples generated by fully convolutional architectures are capable of adversarial transferability. Moreover, we observed that the CNN-LSTM architecture employed in this study provides no additional security compared to non-LSTM models.
- Different from the results presented in Table 5.3, the results provided in Table 5.4 show a low success rate for attacks in the

Table 5.4: Median value (interquartile range) of the ℓ_2 and ℓ_∞ distances obtained for 1,000 adversarial optimizations, as well as their success rate, for \mathcal{L} and the datasets described in Section 5.2.1. For easier comprehension, the threat models are listed from most permissive to least permissive. ℓ_∞ values less than 0.003 are rolled up to 0.003 (this is approximately the smallest amount of perturbation required to change a pixel value by 1, so to make discretization possible).

Threat model	Source model	Target model	IFGS			CW		
			ℓ_2	ℓ_∞	Success %	ℓ_2	ℓ_∞	Success %
WB	$\mathcal{L}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_+, S_-\}}$	0.26 (0.29)	0.003 (0.003)	100%	0.30 (0.30)	0.01 (0.01)	84%
BB:1	$\mathcal{L}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_-, S_+\}}$	0.32 (0.16)	0.003 (0.003)	13%	0.33 (0.47)	0.01 (0.02)	11%
BB:2	$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_+, S_-\}}$	1.96 (2.15)	0.009 (0.006)	35%	0.71 (1.09)	0.02 (0.001)	32%
	$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_+, S_-\}}$	0.28 (0.31)	0.006 (0.003)	38%	0.27 (0.23)	0.03 (0.02)	34%
BB:3	$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_-, S_+\}}$	2.07 (1.45)	0.012 (0.011)	26%	0.84 (0.93)	0.03 (0.07)	23%
	$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_-, S_+\}}$	0.36 (0.52)	0.009 (0.007)	24%	0.28 (0.17)	0.02 (0.03)	19%

context of threat scenario BB:1. The reason for this low success rate is the number of parameters in the model \mathcal{L} , which significantly increases the time to generate adversarial examples. When using this model for generating adversarial examples under limited time settings such as the ones we employ, the success rate drops significantly.

Detailed visual examples of the degree of perturbation needed and the perturbation visibility can be found in Figure 5.7.

5.4.2 Adversarial padding for radar data

Our experiments show that, in the most restrictive case, the success rate of the adversarial attacks falls as low as 18%. The reason behind this low success rate can again be mainly attributed to the time constraint imposed on the attacks. However, an attacker may already be in possession of a pattern of adversariality that is ready to be deployed without needing any additional computation, thus nullifying the time constraint set on generating an adversarial example. In the literature,

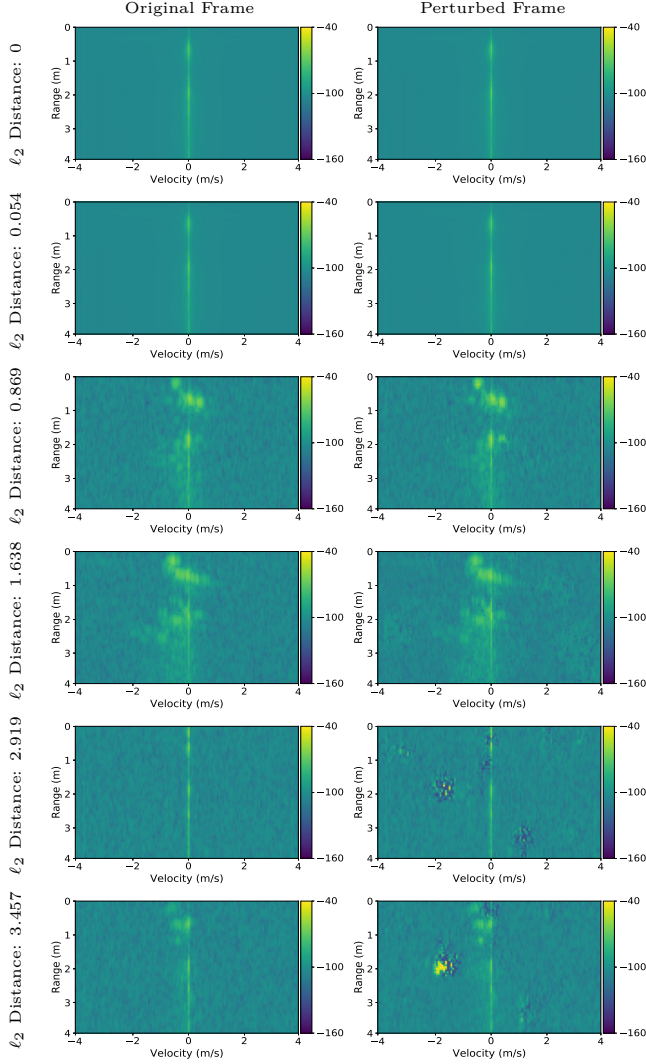


Figure 5.7: (1) ℓ_2 distances, (2) original frames, and (3) perturbed frames are given in order to improve the visual understanding of various degrees of perturbation.

such patterns of adversariality are also known as *universal perturbations* [136]. The main idea behind universal perturbations is to have these patterns generated in advance and to use them during inference time.

Moosavi-Dezfooli et al. [136] demonstrated that universal perturbations exist for DNNs and that these universal perturbations may even be diverse in nature (i.e., more than one universal perturbation

may be available). Instead of evaluating the techniques proposed in Moosavi-Dezfooli et al. [136], which allow for finding a universal perturbation eventually, we experiment with the idea of generating adversarial padding. This approach can then be employed for samples in the dataset that contain less than 50 frames, facilitating prediction without requiring additional optimization. In other words, adversarial padding allows an adversary to change the prediction without even having to modify the frames in which the activity of interest takes place. Inspired by the work of Moosavi-Dezfooli et al. [136], we use the approach described below to generate the adversarial padding.

Let $\mathbf{P} \in [0.31, 0.83]^{80 \times 126}$ be the median padding used in this study,¹ as described in Section 5.2.1, and let $\mathbf{X} \in [0.31, 0.83]^{d \times 80 \times 126}$ be a d -frame-long activity for some $d \in \{1, \dots, 50\}$. We define $\mathbf{M}_1 = [\mathbf{P} \dots \mathbf{P}] \in [0.31, 0.83]^{50 \times 80 \times 126}$ as the initial adversarial padding pattern made up of multiple \mathbf{P} s and denote by \mathbf{M}_i , $i \in \{1, \dots, 50\}$, the i -th frame of \mathbf{M} . In order to generate the adversarial padding, we use the following approach:

$$\tilde{\mathbf{X}} = [\mathbf{X} \underbrace{\mathbf{P} \dots \mathbf{P}}_{50-d}], \quad \hat{\mathbf{X}} = [\mathbf{X} \underbrace{\mathbf{M}_{d+1} \dots \mathbf{M}_{50}}_{50-d}], \quad (5.1)$$

$$\text{minimize } \|\tilde{\mathbf{X}} - \hat{\mathbf{X}}\|_2, \quad (5.2)$$

$$\text{such that } \arg \max (g(\theta, \hat{\mathbf{X}})) = c, \quad \arg \max (g(\theta, \tilde{\mathbf{X}})) \neq c \quad (5.3)$$

where c is the target class. We then calculate \mathbf{M} in an iterative manner as follows:

$$\mathbf{M}^{(n+1)} = \mathbf{M}^{(n)} + (\alpha \nabla_x (g(\theta, [\mathbf{X} \mathbf{M}_{d+1}^{(n)} \dots \mathbf{M}_{50}^{(n)}]))_c \odot \mathbf{K}), \quad (5.4)$$

$$\mathbf{K}_i = \begin{cases} \mathbf{1}^{80 \times 126}, & \text{if } i \in \{i_1, i_2, i_3\} \\ \mathbf{0}^{80 \times 126}, & \text{otherwise} \end{cases}, \quad (5.5)$$

$$\text{with } \|\mathbf{M}_{\{i_1, i_2, i_3\}}^{(n)} - \mathbf{P}\|_2 < \|\mathbf{M}_{i_u}^{(n)} - \mathbf{P}\|_2, \quad (5.6)$$

$$\forall i_u \in \{1, \dots, 50\} \setminus \{i_1, i_2, i_3\}, \quad (5.7)$$

where \odot is the element-wise tensor multiplication. The idea behind

¹The padding used as a starting point is the median frame of all data points. This, strictly speaking, suggests a mild form of data leakage. After the publication of our study, we additionally experimented with a padding frame calculated from only the unseen data points, observing that the newly obtained results were very close to the previously obtained results. As such, we refrained from changing the experimental results provided in this section, thus keeping the experimental results that can be found in the published paper.

Table 5.5: Median value (interquartile range) of the ℓ_2 and ℓ_∞ distances obtained for 1,000 adversarial optimizations, as well as their success rate, for the models and datasets described in Section 5.2.1, hereby using the padding attack described in Section 5.4.2.

Source model	Target model	Padding attack		
		ℓ_2	ℓ_∞	Success %
$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{A}_{\{S_+, S_-\}}$	3.34 (0.56)	0.012 —	84%
$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{R}_{\{S_+, S_-\}}$	3.07 (0.28)	0.012 —	74%
$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{A}_{\{S_-, S_+\}}$	3.47 (0.82)	0.011 —	68%
$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{R}_{\{S_-, S_+\}}$	4.19 (1.07)	0.010 —	51%
$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{R}_{\{S_+, S_-\}}$	5.19 (0.95)	0.011 —	48%
$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{A}_{\{S_+, S_-\}}$	5.56 (0.87)	0.012 —	28%
$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{R}_{\{S_-, S_+\}}$	4.67 (1.21)	0.010 —	41%
$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{A}_{\{S_-, S_+\}}$	5.47 (0.80)	0.012 —	25%
$\mathcal{L}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_+, S_-\}}$	4.47 (0.72)	0.015 —	78%
$\mathcal{L}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_-, S_+\}}$	3.94 (0.91)	0.015 —	73%
$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_+, S_-\}}$	3.58 (1.16)	0.012 —	57%
$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_+, S_-\}}$	4.11 (1.23)	0.012 —	63%
$\mathcal{A}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_-, S_+\}}$	4.81 (1.46)	0.012 —	37%
$\mathcal{R}_{\{S_+, S_-\}}$	$\mathcal{L}_{\{S_-, S_+\}}$	4.99 (0.93)	0.012 —	33%

using K is to force the optimization to alter the least modified padding frames (in this case, the three frames i_1 , i_2 , and i_3). This leads to a more uniform distribution of the perturbation over the padding frames, rather than having perturbation that is concentrated in just a few frames. Indeed, we observed that the optimization focuses on just a few frames rather than all frames when K is not incorporated, resulting in an adversarial example that is not able to reliably change the prediction. We also experimented with selecting more than three frames: although results were comparable, selecting three frames produced the

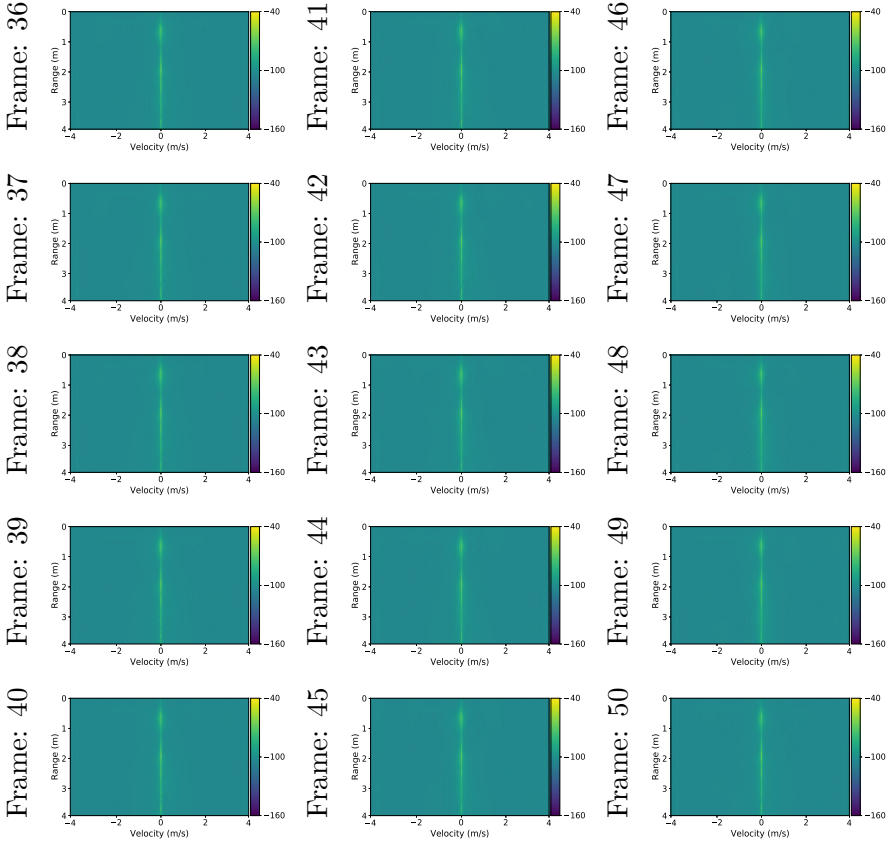


Figure 5.8: A sequence of adversarial padding frames (from frame 36 to 50), as generated by the proposed adversarial padding. The total amount of adversarial perturbation, as calculated for these frames, corresponds to an ℓ_2 distance of 4.75.

best results in terms of adversarial transferability.

By following the aforementioned procedure, we are able to create padding sequences that convert model predictions to the targeted class, without even having to change the frames in which the activity occurs. In order to demonstrate the effectiveness of this attack, we provide Table 5.5, containing details about the success rate of the padding attack, obtained under the same conditions as the results presented in Table 5.3. Note that the ℓ_∞ interquartile range is not presented in Table 5.5 because the padding attack aims at spreading the adversarial perturbation equally over all padding frames, thus keeping the ℓ_∞ distance between genuine data points and their adversarial counterparts

for multiple data points approximately the same.

Given Table 5.5, we can observe that the padding attack is not able to achieve a success rate upwards of 80% for white-box cases. This is because certain samples, which usually belong to activities 0 and 1, are either not padded or padded with very few frames (see Table 5.1 for average activity duration). Thus, it is very challenging, or downright impossible, for the padding attack to change the prediction. Trivially, the shorter activities are more affected by the padding attack. Furthermore, for most black-box cases, we can observe that the padding attack achieves higher success rates than the attacks presented in Table 5.3, albeit by incorporating stronger perturbations.

Our experiments show that it is indeed possible to exploit the structure of datasets that contain a temporal dimension with special attacks similar to the above-described padding attack. In this case, we demonstrated the possibility of changing a model prediction by only perturbing the frames where the activity does not take place. Moreover, the adversarial padding generated by our padding attack only needs to be computed once and can then be used multiple times, thus allowing it to be incorporated in scenarios where the attacker has limited time for performing a malicious attack. We provide a detailed illustration of adversarial padding in Figure 5.8, showing that it is remarkably hard to spot adversarial padding using the bare eye.

The proposed adversarial padding attack is also able to create adversarial examples that transfer to CNN-LSTM models. In particular, even though the attack success rate is on average slightly lower for these models compared to the fully convolutional models, this difference is not large, considering the difference in the initial accuracy of the models. Hence, we observe that in this particular case, employing a CNN-LSTM model does not significantly improve (or reduce) adversarial robustness compared to fully convolutional models.

In the next section, we discuss a number of interesting observations related to model interpretability, as made during our analysis of adversarial attacks on radar-based CNNs.

5.5 Relation of adversarial attacks to interpretability

A major criticism regarding DNNs is their lack of interpretability; it is often challenging (if not impossible) to understand the reasoning

behind the decisions made by a neural network-based model. In order to overcome this issue and to increase the trustworthiness of DNNs, several techniques have been proposed. These techniques can broadly be divided into the following two groups: (1) perturbation-based forward propagation methods [187, 227] and (2) back-propagation-based approaches [181, 190, 235]. The main goal of these techniques is to highlight those parts of the input that are *important* for the prediction made by a neural network. For example, assuming a simple classification problem that aims at making a distinction between cats and dogs, we would expect these techniques to highlight the features that separate these animals from one another, such as the ears, the eyes, or the fur. When the input consists of a natural image, this analysis is often done subjectively, unless the evaluated data comes with, for example, weakly-supervised localization labels, which can then be used for evaluating the correctness of the selected interpretability technique. In our case, different from natural images, the input consists of a sequence of RD frames that are significantly harder to interpret by humans. However, different from prediction using a single image, radar data also bring useful features, such as allowing for a frame-by-frame analysis.

A first peculiar observation we made during the experiments presented in Section 5.4 is that CW focuses on only introducing perturbation in certain frames, rather than spreading out the perturbation equally. In Figure 5.10, we present the amount of perturbation added by CW to each frame in the form of boxplots. Note that padding frames at the end receive considerably less perturbation than the frames containing the action. We hypothesize that frames that are the recipient of stronger perturbations are *important* frames, making it possible to distinguish actions from one another.

To confirm this hypothesis, we perform an exhaustive experiment on measuring the importance of a frame. As illustrated in Figure 5.9, we replace individual frames, one at a time, by the median frame we described in Section 5.2.1, subsequently performing a forward pass. Since this median frame is used throughout the training procedure to pad the data, it is not an out-of-distribution sample, thus not favoring one class over another. By doing so, for each data point, we measure the change in the prediction logit for the correct class 50 times (for each frame individually) and plot the median difference in Figure 5.10, showing the relation between the perturbation amount per frame and the logit change when those frames are replaced. Specifically, the red line represents the median logit change and the shaded area represents the interquartile range. As can be observed, the frames favored

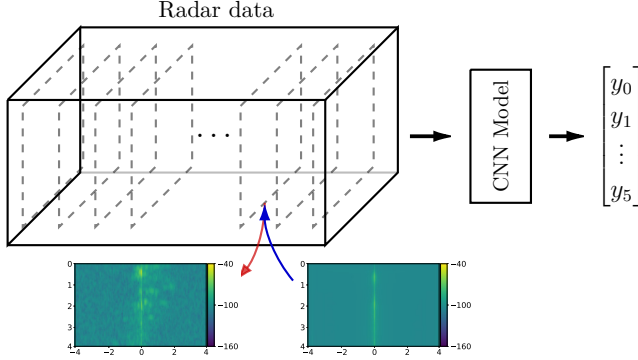


Figure 5.9: Visual representation of the frame replacement operation as explained in Section 5.5.

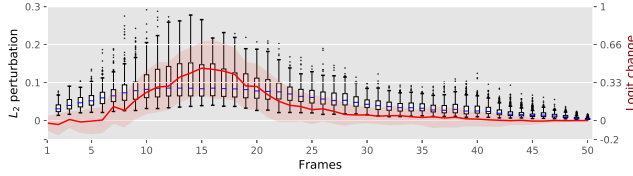


Figure 5.10: A boxplot representation of added perturbation, as generated by CW, displayed for individual frames of adversarial examples that transfer from A_{S+} to A_{S-} . The amount of added perturbation is plotted against the median frame importance, as calculated by the experiment detailed in Section 5.5.

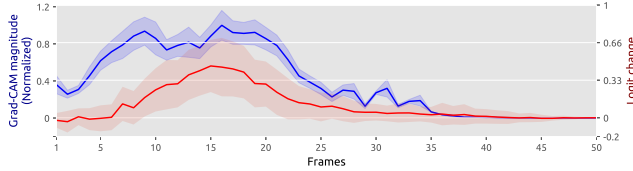


Figure 5.11: Mean Grad-CAM magnitude (normalized) is plotted against the mean frame importance, as calculated by the experiment detailed in Section 5.5.

by adversarial attacks in terms of added perturbation are also the ones that contribute more to the prediction, confirming our hypothesis. An extended version of this experiment, conducted on each class individually, can be found in Figure 5.13: the sub-figure labeled with (a) shows this graph for all of the adversarial examples in an aggregated manner, whereas the following sub-figures are class-specific.

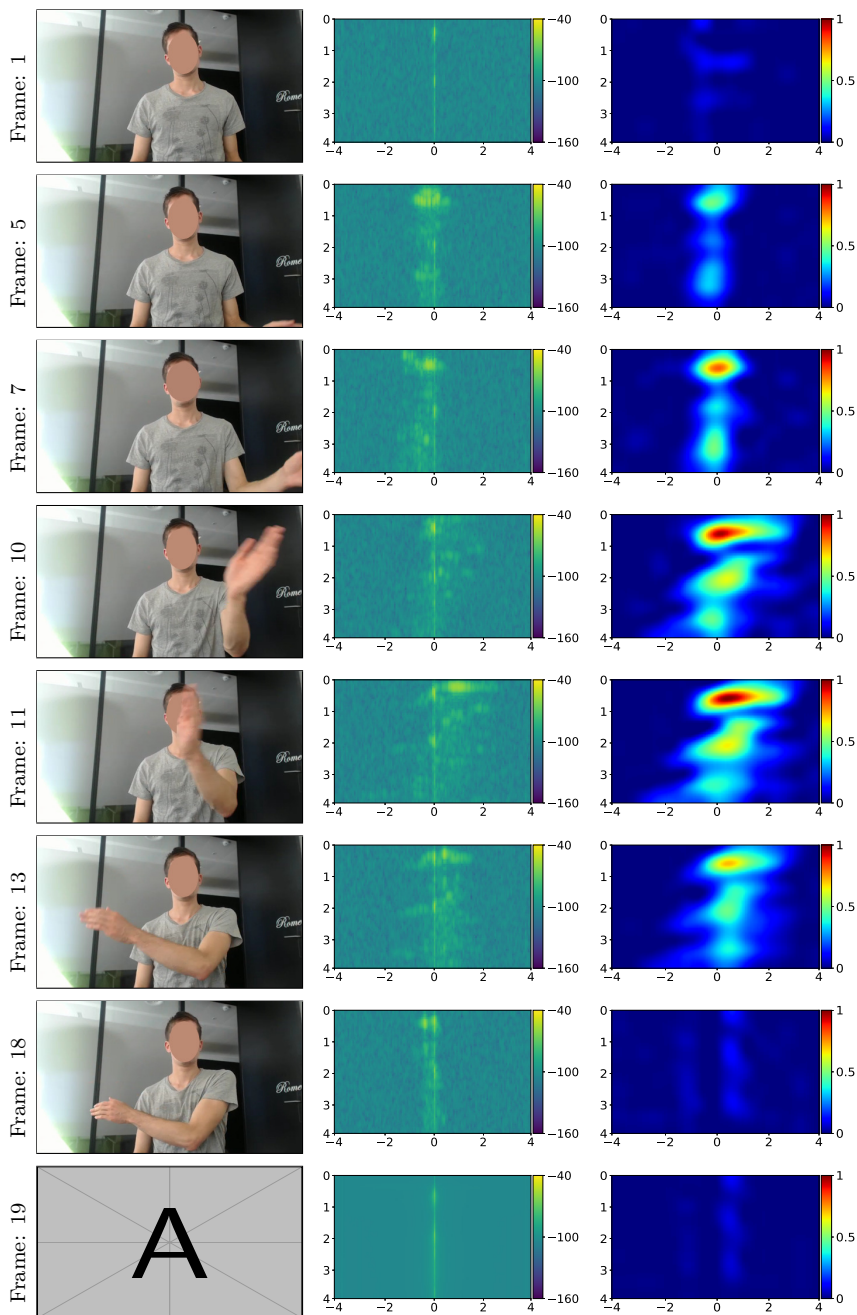


Figure 5.12: Visualization of a number of consecutive (1) RD frames, (2) video frames, and (3) Grad-CAM heatmaps for the action *swipe left*. The X -axis and Y -axis of the RD frames and the Grad-CAM images, which are omitted for visual clarity, correspond to range and velocity, respectively.

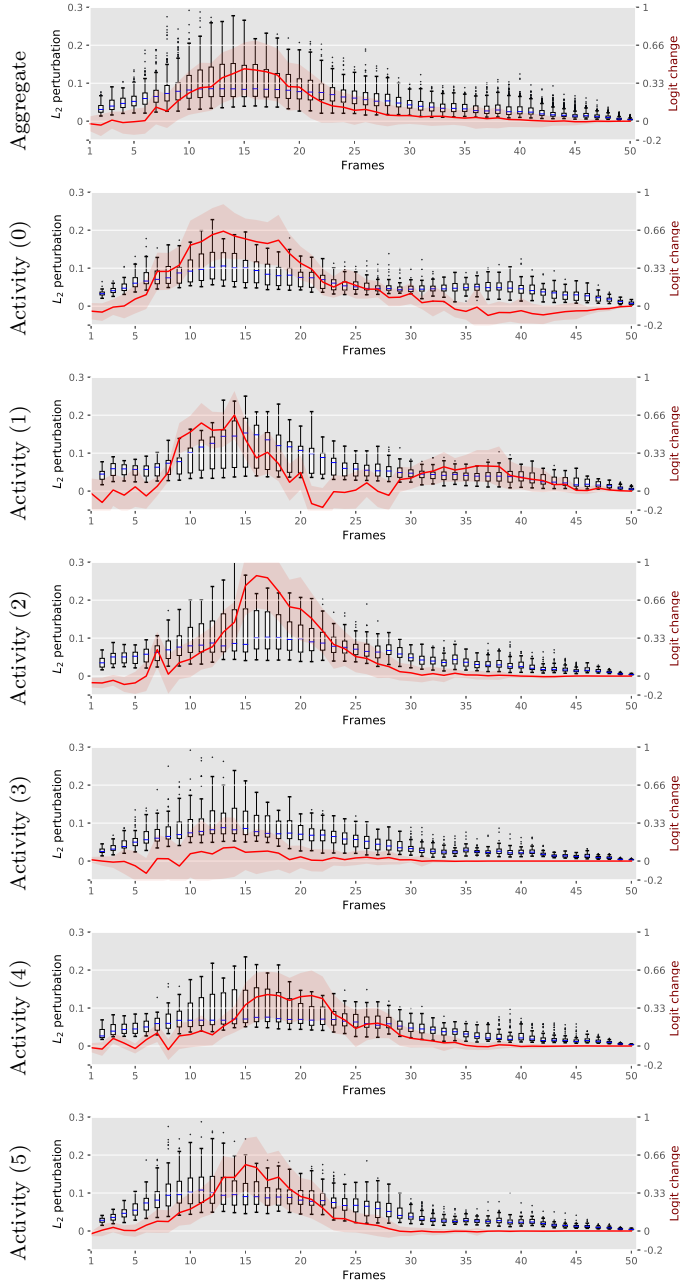


Figure 5.13: A boxplot representation of added perturbation, as generated by CW, displayed for individual frames of adversarial examples that transfer from \mathcal{A}_{S+} to \mathcal{A}_{S-} . Added perturbation is plotted against the median frame importance, as calculated by the experiment detailed in Section 5.5.

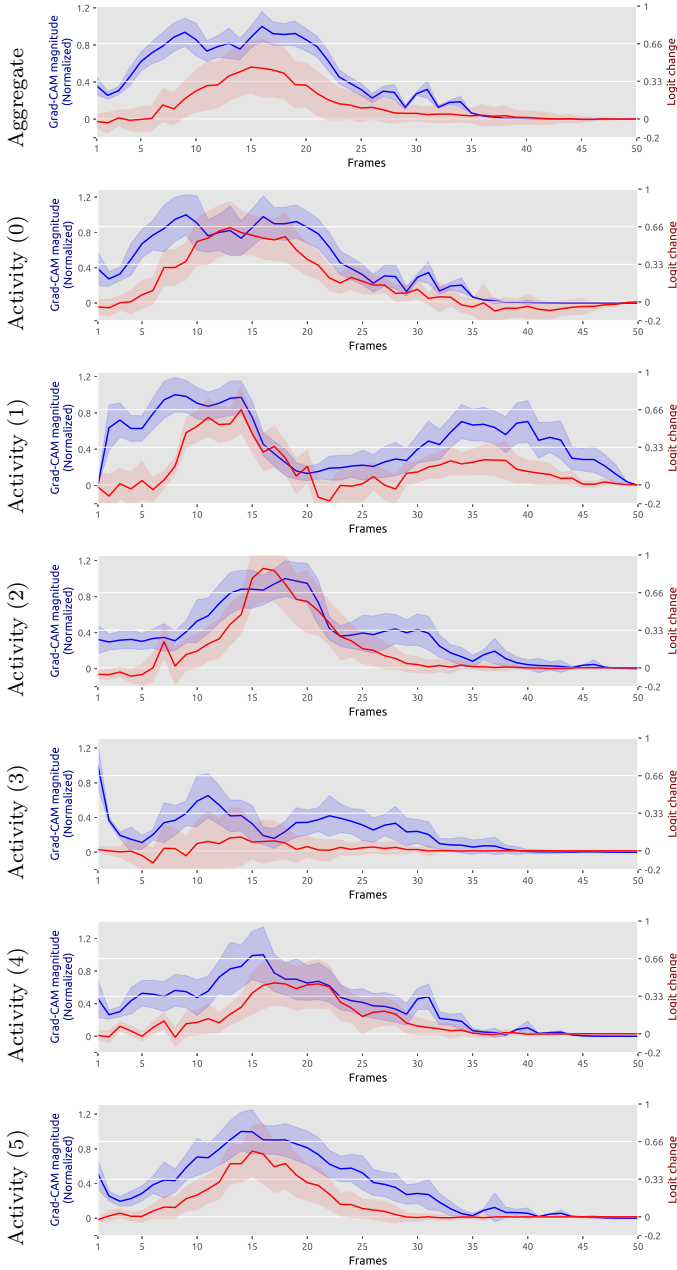


Figure 5.14: (blue) A line graph representation of Grad-CAM magnitudes (normalized between 0 and 1), displayed for individual frames that belong to genuine data points of adversarial examples that transfer from A_{S+} to A_{S-} . This graph is plotted against the median frame importance (red), as calculated by the experiment detailed in Section 5.5.

Following this experiment, we investigate the applicability of CNN interpretability techniques to radar data. Among different interpretability techniques, Grad-CAM [181] stands out, thanks to its superior weakly-supervised localization results obtained on ImageNet. Another reason for selecting this method is that the underlying approach is based on backpropagation, meaning that the input is not perturbed. We especially want to avoid methods based on input perturbation because, unlike natural images, small changes in RD frames may lead to large changes in terms of correctness of the data (i.e., being a valid data point). In our setting, Grad-CAM is defined as follows:

$$\text{Grad-CAM} = \sum_k \left(\text{ReLU} \left(\sum_i \sum_j \nabla_x \mathbf{L}_{i,j}^p \right) \mathbf{L}_k^p \right), \quad (5.8)$$

where \mathbf{L}^p denotes the output of the forward pass after the p -th layer (i.e., discriminative features) and $\nabla_x \mathbf{L}^p$ denotes the gradient obtained with a backward pass from the same layer with respect to the input (i.e., weighted gradient). Different from adversarial attacks, as well as vanilla and guided backpropagation, Grad-CAM does not use the gradients of the first layer, thus arguably allowing for a more robust explanatory approach. Because the input is not a single image but a sequence of frames, Grad-CAM produces class activation maps for each frame individually. An example set of video frames, their corresponding radar frames, and the obtained Grad-CAM heatmaps can be found in Figure 5.12. These qualitative results show that the heatmaps usually highlight (1) those frames where the *most important* part of the activity occurs and (2) those locations where the radar activity is the largest.

Apart from the qualitative results provided in Figure 5.12, which are heavily criticized in Lipton [119] and Ghorbani et al. [55], we now aim at performing a quantitative evaluation of the correctness of the produced Grad-CAM activation frames. We calculate the median magnitude of the produced Grad-CAM frames, which are normalized between 0 and 1, and compare it to the previously presented frame importance data in Figure 5.11, where the blue line represents the median Grad-CAM magnitude and the red line the median frame importance. The bands around the lines correspond to the respective interquartile ranges. The same type of illustration on a per-class basis is provided in Figure 5.14: the sub-figure labeled with (a) shows this graph for all of the adversarial examples in an aggregated manner, whereas the following sub-figures are class-specific.

Given Figure 5.11, we again observe a correlation between the importance of frames and their corresponding Grad-CAM activations. Both experiments, as presented in Figure 5.10 and Figure 5.11, show strong correlation between their respective data. In particular, the higher the magnitude of the positive Grad-CAM heatmap, the larger the change in the prediction will be when replacing the underlying frame with the padding frame. Consequently, our experiments confirm that the output of Grad-CAM can effectively be used to assess the relative importance of each radar frame for the prediction made. Indeed, the frames that contribute the most to a prediction are also the ones that are naturally perturbed more than the others during an adversarial optimization, pointing to a strong connection between adversarial optimization and model interpretability.

5.6 Recent developments

When it comes to the intersection of adversarial attacks and radar-based decision-making systems, most research efforts investigate sensors that make use of LIDAR (Laser Imaging, Detection, And Ranging), given the recent focus on self-driving cars and the importance of the aforementioned type of sensors in that field [77, 3]. In this respect, Sun et al. [198], Cao et al. [20], and [21] evaluated sensory attacks on LIDAR-based systems, identifying their flaws and suggesting a number of methods to fix those flaws. Tu et al. [208] demonstrated that physically realizable objects can act like adversarial examples in the real world, fooling LIDAR-based decision-making systems. Perhaps the most intriguing effort in the field is the work recently presented in [74], with the authors of [74] demonstrating the possibility of removing an object completely through the use of adversarial attacks.

Only a limited number of research efforts make use of FMCW sensors. Nonetheless, moving away from LIDAR, the recent work discussed in Komissarov and Wool [106] is a research effort that is close to the work described in this chapter. Specifically, in [106], the authors investigate multiple threat vectors to FMCW radars used in automotive systems, designing a novel approach to attack these systems. This approach builds on top of research results previously presented in [142, 134], in which a distance-based spoofing attack is discussed.

All in all, the threat of adversarial attacks on radar-based decision-making systems that make use of neural networks appears to be a field

that is growing in importance, with recent research efforts mostly focusing on physically realizable attacks and practical attack vectors that can potentially have a real-world impact. Most of the work discussed in this chapter has been published in the last two years, with very limited comparative analysis available, thus making it difficult to identify the most state-of-the-art attack strategy. In this context, I believe a study that systematically reviews these research efforts, for instance consolidating their validity similar to what was done in [31], would be an immensely helpful contribution to the field.

5.7 Conclusions and future work

We evaluated multiple scenarios in which adversarial attacks are performed on CNNs trained with a sequence of range-Doppler images obtained from a low-power FMCW radar sensor, with the goal of performing gesture recognition. Our analysis showed that these models are vulnerable, not only to commonly used attacks, but also to unique attacks that take advantage of how the dataset is crafted. In order to demonstrate a unique attack that leverages knowledge about the dataset, we proposed a padding attack that creates a padding sequence that changes the predictions made by CNNs.

An often mentioned drawback of CNNs is their lack of interpretability. By taking advantage of the data selected for this study, we were able to show the connection between the perturbation exercised by adversarial attacks and the importance of individual frames. Moreover, we were also able to demonstrate that it is possible to identify *important* frames using Grad-CAM, thus showing (1) the relation between adversarial optimization and interpretability, and (2) a quantitative method to evaluate interpretability techniques.

We believe one of the future research directions in this topic is the adversarial threat analysis of radar sensors against so-called real-world adversarial examples [110, 199], as well as black-box attacks that do not use surrogate models [27, 65, 207]. In the case of activity detection, real-world adversarial examples may occur when radar sensors are employed in environments exhibiting poor recording conditions, such as environments that contain reflective materials (e.g., metal objects), or similarly, when the subject itself carries any reflective material. Moreover, it would be of interest to investigate the influence of multiple moving subjects in the same recording environment on adversariality.

6

Detecting adversarial examples

We have thus far investigated adversarial attacks employed to deceive a variety of decision-making systems, as well as the properties of the adversarial examples created by such attacks. In this chapter, we will address a topic we did not discuss in detail yet, namely methods for adversarial example detection (i.e., adversarial defenses).

The content of this chapter is based on the following publications:

- Utku Ozbulak, Wesley De Neve, and Arnout Van Messem. How the Softmax Output is Misleading for Evaluating the Strength of Adversarial Examples. *Neural Information Processing Systems (NeurIPS), Workshop on Security in Machine Learning (SecML)*, 2018.
- Utku Ozbulak, Wesley De Neve, and Arnout Van Messem. Not All Adversarial Examples Require a Complex Defense: Identifying Over-optimized Adversarial Examples with IQR-based Logit Thresholding. *International Joint Conference on Neural Networks (IJCNN)*, 2019.
- Utku Ozbulak, Wesley De Neve, and Arnout Van Messem. Perturbation Analysis of Gradient-based Adversarial Attacks. *Pattern Recognition Letters, Elsevier*, 2020.

This chapter is composed of six sections, the first of which discusses a number of defenses described in the literature to detect adversarial examples. In Section 6.2, we re-evaluate the feasibility of adversarial retraining on ImageNet. In Section 6.3 and Section 6.4, we discuss our findings on the impact of adversarial optimization on logits and the softmax output. In Section 6.5, we demonstrate how

the aforementioned findings can be used to easily detect a large subset of (over-optimized) adversarial examples. Finally, in Section 6.7, we conclude this chapter on adversarial defenses, also providing directions for future research.

The content presented in this chapter is slightly different from the content of the supporting published papers. We changed the mathematical notation, bringing this notation in line with the mathematical conventions used by other chapters. We also introduce more qualitative results, facilitating a better understanding of adversarial defenses. The discussion provided in Section 6.4 is more elaborate compared to the published version, introducing additional experiments in a controlled 2-D setting. Finally, we included Section 6.5.6 and Section 6.6, first outlining the benefits and limitations of adversarial defenses that are similar to the proposed adversarial defense, followed by an explanation of the limitations of logit-based defenses.

6.1 Introduction

After the discovery of adversarial examples and their pervasive effects on DNNs, a plethora of defenses were proposed [116, 79, 162, 7]. These defenses either aim at getting rid of the adversarial perturbation while keeping the relevant features of the input intact, or they try to identify adversarial examples, thus categorizing inputs into two categories: adversarial or genuine. Although the proposed defenses rely on a variety of different techniques, we can roughly group them into the following categories:

- **Input modification** – This approach was one of the first methods tested in order to defend against adversarial examples, since the idea of removing adversarial perturbation is quite intuitive. The defenses that use this approach modify the input with various methods in order to get rid of the adversarial perturbation, while preserving the class-defining properties of an image. Below, we briefly explain a number of defenses that employ this approach.
 - Li and Li [116] proposed the usage of a 3×3 filter to blur an input image, resulting in the removal of adversarial perturbation.

- Liu et al. [120] used JPEG encoding in order to remove adversarial perturbation, leveraging the underlying compression algorithm.
- Guo et al. [69] used total variation [173] and denoising in order to reconstruct a perturbation-free input image.
- Hendrycks and Gimpel [79] proposed the usage of PCA in order to investigate the properties of the input.
- Bhagoji et al. [12] aimed at reducing the dimensionality of the input, resulting in the training of neural networks with inputs that have a lower dimension than the original inputs.
- **Architectural investigation** – Instead of analyzing the input, it is also possible to investigate the properties of the neural network building blocks in order to detect adversarial examples. Defenses employing this approach either augment the underlying architecture or use the outputs of intermediate layers for distinguishing genuine inputs from adversarial ones. A number of defenses involving this type of approach are given below.
 - Metzen et al. [130] proposed the investigation of convolutional layers and the augmentation of architectures with special layers designed to detect adversarial examples.
 - Li and Li [116] applied PCA after inner convolutional layers and built a robust classifier that detects adversarial examples.
 - Feinman et al. [47] argued that the usage of a dropout layer [194] during inference time could be used as a method to detect adversarial examples.
- **Prediction analysis** – One of the most prominent techniques is also the analysis of outputs (i.e., the logits or the probabilities produced by softmax). Defenses that use this approach often apply a statistical test to the predictions obtained for the data points at hand in order to differentiate adversarial examples from regular inputs. Examples of this type of defense include the following works:
 - Koo et al. [107] proposed to train a secondary two-class (adversarial versus non-adversarial) neural network on the predictions that originate from an underlying model.

- Roth et al. [172] proposed the usage of a statistical test on the predictions obtained for adversarial inputs, comparing them to genuine inputs when both types of inputs are modified with a small amount of added noise.
- Grosse et al. [67] used Maximum Mean Discrepancy, a statistical hypothesis test that tries to find out whether two sets of data are coming from different distributions, to detect whether predictions obtained for adversarial examples are different from the predictions obtained for genuine examples.
- Sotgiu et al. [192] empirically tried to identify anomalous features of adversarial examples in different layers, thus separating them from real data based on such features.
- **Defenses with a modified training routine**—Often called adversarial retraining, this approach trains a neural network using adversarial examples in order to increase its robustness. Methods using this approach either use adversarial examples as an additional category or use the categories of source images the adversarial examples originate from while training a neural network [61, 111]. Even though this approach was one of the early approaches used for detecting adversarial examples, it is still considered to be the most effective one.

A recent direction in the field of adversarial defenses is the development of so-called certified adversarial defenses, guaranteeing the robustness of the proposed defenses within an ℓ_p norm perturbation [36]. However, Xu et al. [223] and Sharif et al. [186] showed that defenses claiming robustness within an ℓ_p norm do not necessarily provide the envisioned protection since norm measurement itself is not a good indicator of the detectability of adversarial examples by humans.

The different adversarial defenses briefly discussed above are only a small subset of what was proposed since the discovery of adversarial examples. As more defenses were introduced, stronger attacks able to bypass these defenses were also discovered. This cat-and-mouse game naturally created an arms race between attacks and defenses, resulting in stronger adversarial attacks being proposed with each study [22, 126, 136], and, in response, stronger defense mechanisms to prevent these attacks from taking place [158].

The seminal works of Athalye and Carlini [6], Athalye et al. [7], and Carlini and Wagner [23] investigated the validity of the claims made

by studies on adversarial defenses and found them to be vulnerable to adversarial examples within the boundaries for which they claimed robustness. In their latest work, Tramer et al. [206] employed an adaptive attack technique, thus showing that, even when a defense is able to detect adversarial example generated by one type of attack, they may still be vulnerable to adversarial examples generated by other attacks. As a result, it is currently widely accepted that no defenses are available that offer protection against all types of adversarial examples. This observation perhaps calls for the usage of multiple defenses to protect against adversarial attacks, making it possible to leverage the strength of various techniques at the same time.

The rate in which adversarial examples can be detected has been the primary focus of research efforts dedicated to the development and evaluation of adversarial defenses. However, it is possible to achieve a perfect detection rate by simply labeling all inputs as adversarial examples. Therefore, it is crucial to achieve a balance between having a high detection rate for adversarial examples while having a low false positive rate (i.e., mislabeling genuine inputs as adversarial examples). Naturally, an ideal adversarial defense would be the one that identifies as many adversarial examples as possible while not mislabeling any genuine data point as an adversarial one. Based on the experience we have accumulated through the research efforts presented in the previous chapters, as well as the observations made above, in what follows, we will briefly investigate the feasibility of adversarial retraining and the effectiveness of an unobtrusive adversarial defense that also leverages prediction analysis.

6.2 Adversarial retraining

As mentioned above, one of the first methods tested to defend against adversarial attacks was the use of adversarial retraining [67]. This method retrains the model under consideration with adversarial examples, while adding an additional $((M + 1)th)$ adversarial class to the set of original classes. Adversarial retraining was later extended by Gong et al. [61], with the authors training a model in a binary fashion in order to obtain a discriminator that differentiates genuine images from adversarial ones. Because this experiment was performed with a limited number of images and on datasets that are not fit to study adversarial attacks (due to limitations in terms of (1) color chan-

nels (MNIST [114]), (2) image sizes (CIFAR [108]), or (3) the total number of images available), the method of adversarial retraining was later labeled as an ineffective defense mechanism [23] when sufficient data are not provided.

We now revisit the idea of adversarial retraining. Recall from Chapter 3 that we have unified the objective functions of a number of commonly used adversarial attacks. As such, instead of generating adversarial examples with the attacks themselves, we generate them based on the loss functions and the settings discussed in Chapter 3, while keeping the other conditions the same across the board. In that regard, we use CE-loss (as used by the LBFGS attack), CE-sign loss (as used by IFGS), Logit loss, and M-logit loss (as used by CW). With the help of each of these objective functions, we generate 50,000 adversarial examples using a pretrained ResNet-50 network, taking initial images from the validation set of the ImageNet dataset. This artificial dataset of 200,000 adversarial images is then utilized to train AlexNet [109], VGG-16 [189], and ResNet-18/34/152 [75], with the goal of analyzing whether these neural networks can distinguish genuine images from adversarial ones.

Pretrained weights for the models are taken from the PyTorch library [160]. After loading the models, we replace the final linear layer with another newly initialized layer that has two class outputs (one for genuine images and one for adversarial images). The weights for this layer are obtained using the initialization method provided in [75].

In the first part of this experiment, we analyze the detectability of adversarial examples for each loss separately (i.e., models trained with genuine images and adversarial examples, and where the latter are all generated by one of the adversarial losses given above). In the second part, we incorporate adversarial examples generated by all four types of adversarial losses, analyzing whether this approach is effective as a defense mechanism and which types of adversarial examples are harder to detect.

Training on a single type of adversarial examples – The results provided in Table 6.1 were obtained by training the aforementioned architectures in a binary fashion, where the set of adversarial examples only contains those adversarial examples generated by a single loss (provided in the first column of Table 6.1). For training purposes, we use 47,500 genuine images available in the ImageNet validation set and their 47,500 adversarial counterparts. The remaining 5,000 images (half genuine, half adversarial) are used as a test set.

Training on multiple types of adversarial examples – The results provided in Table 6.2 were obtained by training the aforementioned architectures in a binary fashion, where all adversarial examples generated by all losses are labeled as the adversarial class. We use 50,000 genuine images available in the ImageNet validation set and 200,000 adversarial examples, with the latter generated from the aforementioned genuine images using four types of losses. Due to the imbalance between both classes, we incorporate 150,000 extra genuine images taken from the extended dataset ImageNet-10K [38]. The reason for not using any images from the training set of ImageNet is that the models applied have already been pretrained on this training set. The resulting dataset of 400,000 images is then split, in a stratified way, into 380,000 and 20,000 images for training and testing, respectively.

Training methodology – For each model, we use the same training approach and parameter values (e.g., learning rate, weight decay, and annealing), as provided in the respective papers. In addition to that, we also experimented with training these models using Adam [103], with learning rates of $10^{\{-5, -4, -3\}}$ and with a weight decay of 10^{-3} . Unless specified otherwise in the respective papers, we train each model for 100 epochs, presenting results for the experiment that achieved the highest overall accuracy on the test sets described above.

6.2.1 Experimental results

Experimental results obtained for adversarial retraining can be found in Table 6.1 and Table 6.2, with the former representing the classification accuracy when models are retrained with genuine images and only one type of adversarial examples, and with the latter showing the classification accuracy when the adversarial retraining is done with both genuine images and all types of adversarial examples. Furthermore, in Table 6.2, next to providing the classification accuracy of adversarial examples for each type of loss, we also provide aggregate results in order to present the detectability per loss type. Given the experimental results shown in Table 6.1 and Table 6.2, our findings can be summarized as follows:

- Adversarial retraining requires massive amounts of data. The reason why we had to generate such a large artificial dataset was that any adversarial retraining effort conducted with less

Table 6.1: Accuracy (genuine, adversarial (overall)) of binary classification between genuine images and adversarial examples, where the adversarial examples have been generated by the adversarial losses listed in the first column for the given models. Each entry represents the outcome of an adversarial retraining. Architectures are ordered from the left to the right in terms of ascending convolutional layer complexity.

Adv. Loss	AlexNet	ResNet-18	VGG-16	ResNet-34	ResNet-152
CE	88% 30% (59%)	80% 79% (79%)	94% 90% (92%)	92% 90% (91%)	96% 94% (95%)
CE-sign	90% 94% (92%)	97% 96% (96%)	98% 99% (98%)	99% 99% (99%)	99% 99% (99%)
Logit	94% 94% (94%)	95% 96% (95%)	98% 97% (97%)	98% 99% (98%)	99% 99% (99%)
M-logit	74% 75% (74%)	90% 91% (90%)	95% 96% (95%)	96% 96% (96%)	98% 97% (97%)

than this amount of data produced inconsistent results. This observation is also made by Gong et al. [61], where the classifier trained achieved an accuracy of either 0% or 100%.

- Different from the study of Carlini and Wagner [23], we find adversarial retraining to be an effective method for detecting adversarial examples that have been generated using ImageNet, for which the perturbation patterns obtained are more distinctive than the perturbation patterns obtained for datasets with smaller image sizes (see Figure 6.1 for perturbation examples).
- We observe a clear correlation between the learning capacity of a model (in terms of convolutional layer complexity) and its ability to discriminate genuine images from adversarial ones, an observation also made by Madry et al. [126].
- Confirming [61], we also find that adversarial examples, as generated with CE-sign, are the easiest to detect in comparison to other types of adversarial examples. In addition, we find that adversarial examples generated with CE and M-logit are harder to detect when training incorporates all adversarial examples.
- It might seem like CE is a decent baseline approach to generate adversarial examples. However, due to its constrained optimiza-

Table 6.2: Breakdown of the classification accuracy obtained for genuine images and adversarial examples when four types of adversarial losses are incorporated. Each column represents the outcome of an adversarial retraining. Architectures are ordered from the left to the right in terms of ascending convolutional layer complexity.

Adv. Loss	AlexNet	ResNet-18	VGG-16	ResNet-34	ResNet-152
CE	53%	83%	95%	92%	93%
CE-sign	76%	97%	99%	98%	99%
Logit	71%	95%	98%	97%	99%
M-logit	52%	84%	96%	93%	94%
Adversarial Examples	64%	90%	97%	95%	97%
Genuine Images	81%	94%	93%	95%	99%
Overall Accuracy	72%	92%	95%	95%	98%

tion space and limited perturbation generation capacity (as previously discussed in Chapter 3), adversarial examples generated with CE are less likely to transfer to other models, and are more susceptible to defense mechanisms that use input transformations such as blurring or total variation [23].

We observe that neural networks are indeed able to identify adversarial perturbation patterns as generated by individual losses and that adversarial retraining can be used to differentiate genuine images from their adversarial counterparts on ImageNet, given that the perturbation patterns obtained for this dataset are more distinct than the perturbation patterns obtained for datasets with smaller image sizes.

Moving away from the topic of adversarial retraining, in the next section, we will discuss the impact of adversarial optimization on prediction and how this impact can be leveraged as a defense.

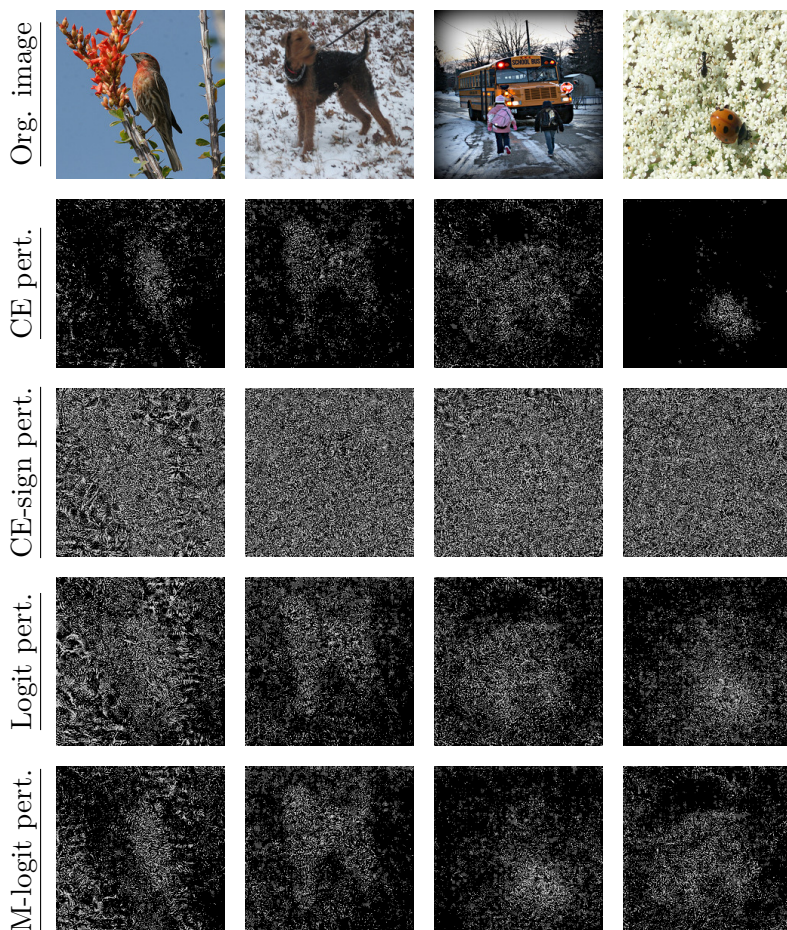


Figure 6.1: Genuine images taken from the ImageNet dataset and the added perturbation, illustrated in the form of saliency maps, with the adversarial attacks making use of CE, CE-sign, logit, and M-logit loss.

6.3 Impact of adversarial optimization on logits

Methods for adversarial example generation use specific optimization techniques to increase the chance that a given input is labeled as the targeted class. Most of the proposed optimization techniques are iterative in nature. Indeed, it has been shown that these techniques produce more effective adversarial examples with less perturbation,

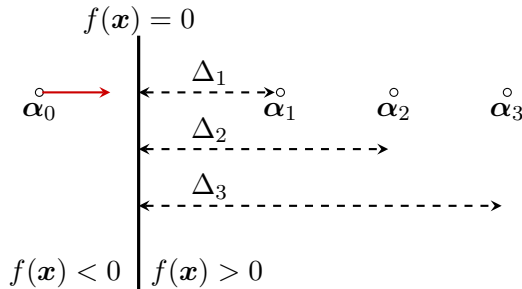


Figure 6.2: An illustration of adversarial examples $\alpha_{\{1,2,3\}}$, derived from an input α_0 , for an affine classifier $f(\mathbf{x}) = 0$, and their respective distances $\Delta_{\{1,2,3\}} = d(\alpha_{\{1,2,3\}}, f)$ to the decision boundary f .

compared to single-step optimization techniques or techniques that facilitate untargeted attacks [7].

We can simplify the behavior of targeted iterative optimization techniques as follows: assume that we have a classifier in a two-dimensional setting, $f(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}$, separating the input space into two subspaces $f(\mathbf{x}) > 0$ and $f(\mathbf{x}) < 0$, with the line $f(\mathbf{x}) = 0$ denoting the decision boundary. This setting is shown in Figure 6.2. Under these circumstances, when the given example α_0 is iteratively optimized towards $f(\mathbf{x}) > 0$, the distances $\Delta_{\{1,2,3\}}$ between the generated adversarial points $\alpha_{\{1,2,3\}}$ and the decision boundary are increased at each step, with the goal of increasing the likelihood that the data point under consideration (that is, α_0) is classified as $f(\mathbf{x}) > 0$. To that end, it does not matter whether the attack uses the logits of the model (like CW) or not, given that the overall impact of the optimization techniques used on the logits remains the same (recall the observations made in Chapter 3).

6.4 Softmax for adversarial examples

When the prediction of a neural network is analyzed, the output is often represented in terms of probabilities. Such a probability is usually referred to as the confidence of the prediction made. To convert logit values into probabilities, a normalized exponential function called the softmax function is used [16, 18, 63]. As we have detailed in Chapter 3, the softmax function uses the exponential function to squeeze the in-

put values between zero and one in such a way that the output values add up to one. This property makes the softmax function helpful in more easily interpreting the predictions of a neural network, instead of having to rely on the logits, which are more difficult to interpret. The first usage of the softmax function in convolutional neural networks dates back to 1998 [114], with this function soon thereafter becoming a common tool to convert logits into probabilistic values.

In reality, the softmax function has two drawbacks for correctly interpreting the predictions of a neural network when adversarial examples are at stake and when this function is used in settings with limited decimal precision. The first one is its lack of a unique input-to-output mapping (in other words, the function is not injective [53]); the second one is its sensitivity to high-magnitude inputs, which is due to its reliance on the exponential function. As we will show momentarily, these limitations can mask certain characteristics of adversarial examples, and could in some cases even be *abused* by certain techniques for adversarial example generation.

In order to generate more robust adversarial examples, multi-class optimization was proposed by Carlini and Wagner [22]. This method aims at producing an adversarial example that is not only predicted as the targeted class with high confidence, but this approach also optimizes the second most likely class so that the adversarial example can be easily transferred between models, almost surely being predicted as one of these two classes. In simple terms, this attack implicitly adds perturbations from two sources: the target class and the second most likely class. In this case, when the optimization is performed multiple times and the logits of these two targeted classes become much larger than all others, then the confidence of the prediction will only depend on these two classes. This attack effectively takes advantage of how the softmax function maps inputs to outputs to disguise a *strong* adversarial example as a *weak* one. Using this attack, or any other multi-class optimization technique, it is therefore possible to generate adversarial examples that produce extremely high logit outputs for the selected classes and that are still disguised as *low-confidence* adversarial examples.

DEFINITION 6.4.1. (*Computation under limited decimal precision*)

Given a function $f : \mathbb{R} \rightarrow \mathbb{R}$ and an input $u \in \mathbb{R}$. If the operation $y = f(u)$ is performed under a limited decimal precision, hereby representing the smallest positive number as δ , then a change in the input that has a magnitude less than δ does not cause a change in the output.

As such, $\forall p$, with $0 < p < \delta$, it holds that $u \equiv u + p \equiv u - p$ and $y = f(u) = f(u + p) = f(u - p)$.

LEMMA 6.4.2. *When the softmax function P is used in settings with limited decimal precision, increasing the input that corresponds to the highest output of the softmax function does not result in an increase in that output when the input of any other non-zero output has a larger increase.*

Proof. Take $\mathbf{u} \in \mathbb{R}^M$ such that $P(u_1, u_2, \dots, u_M)^T = (p_1, p_2, \dots, p_M)^T$, with $p_1 > p_i$, $\forall i \in \{2, \dots, M\}$. Furthermore, assume that the calculation is performed under a limited decimal precision as described in Definition 6.4.1, where the smallest positive number that can be represented is δ , with $\delta \in (0, 0.1)$. If $p_n < \delta$, $n \in \{1, 2, \dots, M\}$, then $p_n \equiv 0$, and if $p_n > 1 - \delta$, then $p_n \equiv 1$. For any $t \in \{2, \dots, M\}$ with $p_t > 0$, take $\mathbf{v} \in \mathbb{R}^M$ such that $v_t > v_1 > 0$, $u_1 + v_1 > u_t + v_t$, and $v_k = 0$ for $k \notin \{1, t\}$, then $P(\mathbf{u} + \mathbf{v})^T = (l_1, l_2, \dots, l_M)^T$, with $l_1 > l_t$, but $l_1 < p_1$. \square

Practical examples for this lemma can be found in Figure 6.3, showing adversarial examples that are predicted with lower confidence than their predecessors, although the logits of the corresponding input have increased. The prediction confidence of these adversarial examples almost entirely depends on two out of a thousand classes, given the vast difference among the magnitudes of the different predictions made. Naturally, since the softmax outputs only depend on two entries, the lowest confidence that can be achieved for this case is slightly higher than 0.50. However, it is possible to extend this two-class attack to a larger multi-class attack, producing confidence values that are even lower, further disguising the adversarial examples when the softmax output is measured. This again shows that the output of the softmax function may give rise to misleading results when evaluating the strength of adversarial examples. Note that the results of Lemma 6.4.2 can easily be extended to larger multi-class schemes in which more than two classes are optimized.

Most of the adversarial attacks reviewed in Chapter 3 use an iterative approach for creating adversarial examples [22, 110, 143, 156, 201], making it possible to further optimize an adversarial example (in terms of the logit values obtained), even after reaching full confidence. However, once the prediction confidence is mapped to one, it is impossible

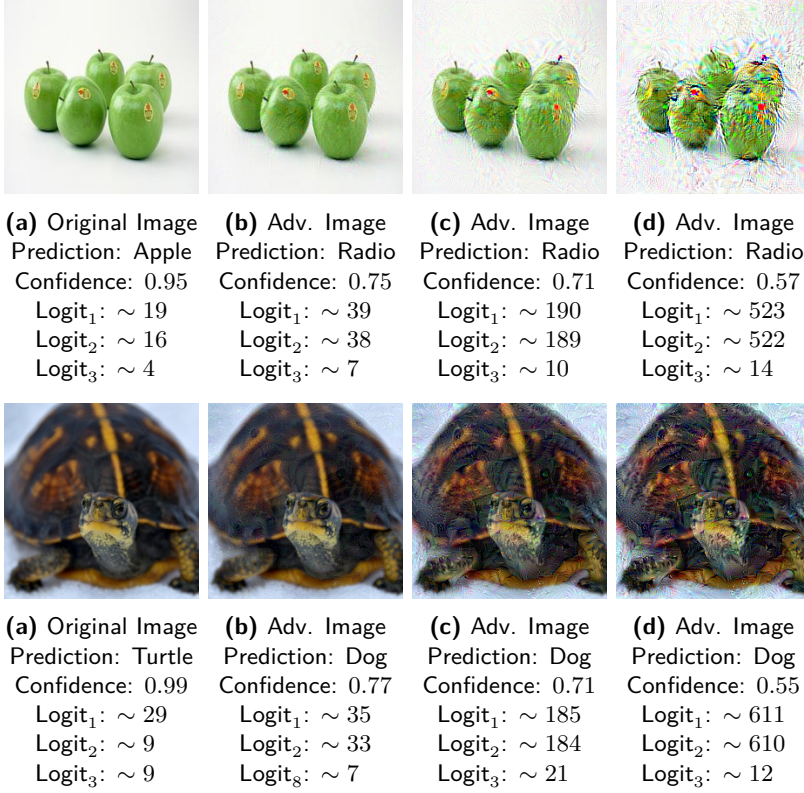


Figure 6.3: (a) Original images, predicted as *apple* (top) and *turtle* (bottom). (b)-(c)-(d) Multi-class optimized adversarial examples that produce higher logit values, but that are predicted with lower confidence by ResNet-50. Logit₁, Logit₂, and Logit₃ are the logits of the first, second, and third most likely predictions, respectively.

to differentiate between the next iterations of the adversarial example based on the softmax output. Indeed, as the corresponding input (logit) keeps increasing, the softmax output will remain the same, as shown in Lemma 6.4.3.

LEMMA 6.4.3. *When the softmax function is used in settings with limited decimal precision, it is no longer sensitive to positive changes in the magnitude of the largest input once the corresponding output has been mapped to one.*

Proof. Take $\mathbf{u} = (u_1, u_2, \dots, u_M)^T \in \mathbb{R}^M$, with $u_1 \in \mathbb{R}^+$, such that $P(\mathbf{u}) = (p_1, p_2, \dots, p_M)^T$, $p_1 > p_i, \forall i \in \{2, \dots, M\}$. Furthermore,

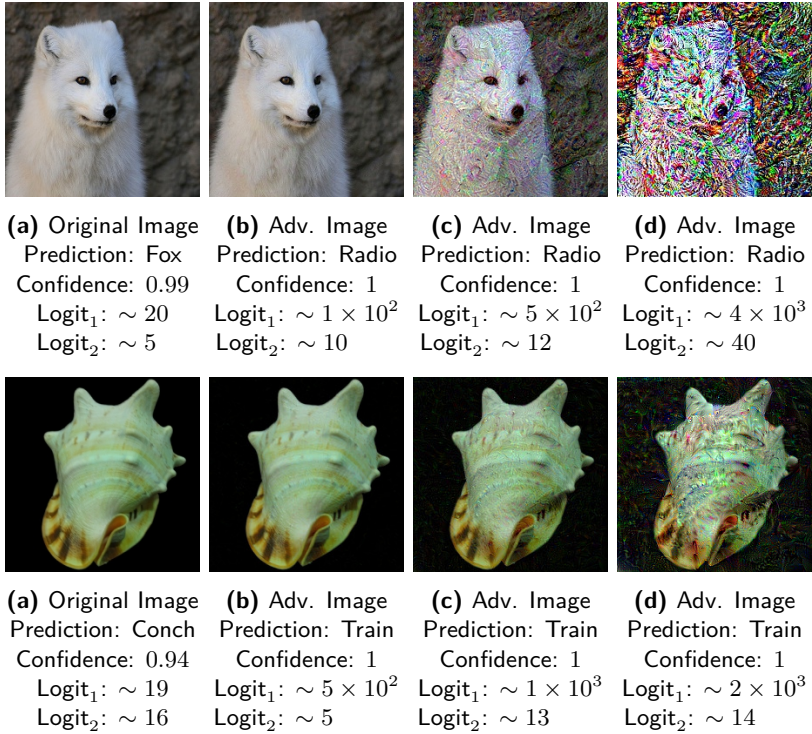


Figure 6.4: (a) Original image, predicted as *arctic fox* with 0.99 confidence. (b)-(c)-(d) Over-optimized adversarial examples which are predicted with the same confidence but with vastly different logit values by ResNet-50. Logit₁ and Logit₂ represent the logits of the most likely and second most likely predictions, respectively.

assume that the calculation is performed under a limited decimal precision as described in Definition 6.4.1., where the smallest positive number that can be represented is δ , with $\delta \in (0, 0.1)$. Under these conditions, if $p_1 > 1 - \delta$, then $p_1 \equiv 1$, and $\forall v_1 \in \mathbb{R}^+$ with $v_1 > u_1$, $P(v_1, u_2, \dots, u_M)^T = (1, 0, \dots, 0)^T$. \square

To show a practical outcome of Lemma 6.4.3 for neural networks, we provide two genuine images in Figure 6.4, classified as *arctic fox* and *conch* with high confidence, and three adversarial counterparts generated from these genuine images, all of which are misclassified with a confidence value of 1 by a pretrained ResNet-50 [75]. All of the adversarial examples have been *over-optimized* to produce logits that are beyond the reach of any natural image, with logit values approx-

imately achieving values as large as 4×10^3 . We will call adversarial examples of this type *over-optimized adversarial examples*. As a comparison, the highest logit value achieved by a genuine image in the whole ImageNet validation dataset [175] for the pretrained ResNet-50 model we use in this experiment is ~ 52 , and the highest logit value for the target class *radio* is only ~ 23 . As can be observed in Figure 6.4, when an adversarial example is referred to as a *high-confidence* adversarial example, based on the output of the softmax function, both the logit values and how far it is optimized are not clear, given the masking effect of the softmax function.

Although we were able to notice an increase in logits in Figure 6.4, as opposed to the softmax output, such an observation is only possible for neural networks that contain an activation function β that has the property of infinite scaling (i.e., $\lim_{t \rightarrow \infty} \beta(t) = \infty$). In order to put this claim in plain view, we present a straightforward two-class classification problem in Figure 6.5. This problem is solved using a neural network with a single hidden layer containing ten neurons. This neural network can be formulated as

$$[y_1, y_2] = g(\theta, (x_1, x_2)) = \beta([x_1 \ x_2] \mathbf{W}_1 + \mathbf{b}_1) \mathbf{W}_2 + \mathbf{b}_2, \quad (6.1)$$

with $(\mathbf{W}_1, \mathbf{W}_2)$, $(\mathbf{b}_1, \mathbf{b}_2)$, and β denoting weights ($\mathbf{W}_1 \in \mathbb{R}^{[2 \times 10]}$, $\mathbf{W}_2 \in \mathbb{R}^{[10 \times 2]}$), biases ($\mathbf{b}_1 \in \mathbb{R}^{[1 \times 10]}$, $\mathbf{b}_2 \in \mathbb{R}^{[1 \times 2]}$), and the activation function, respectively. The upper graph in Figure 6.5 shows the approximate decision boundary ($x_1 = 0$) obtained (highlighted by the dashed line) when this problem is solved by the aforementioned neural network. In particular, to solve this classification problem, we apply the neural network at hand using four different activation functions: $\beta \in \{\text{sigmoid}, \text{tanh}, \text{ReLU}, \text{softplus}\}$. This leads to the lower four graphs in Figure 6.5, each displaying the logit and softmax predictions when following the cyan line for the different choices of β .

From Figure 6.5, it can be observed that, as the distance between a data point and the decision boundary increases, the corresponding logit prediction $\max(g(\theta, (x_1, x_2)))$ for that point will also increase when $\lim_{t \rightarrow \infty} \beta(t) = \infty$. Adversarial attacks that continuously increase the prediction likelihood of a targeted class may then create easy-to-detect outliers (i.e., adversarial examples) when activation functions of a similar type are used. However, when $\lim_{t \rightarrow \infty} \beta(t) = k < \infty$, as is the case for sigmoid and tanh, then the aforementioned statement does not hold true, therefore making the detection of outliers using logit information impossible. Moreover, for all cases, we

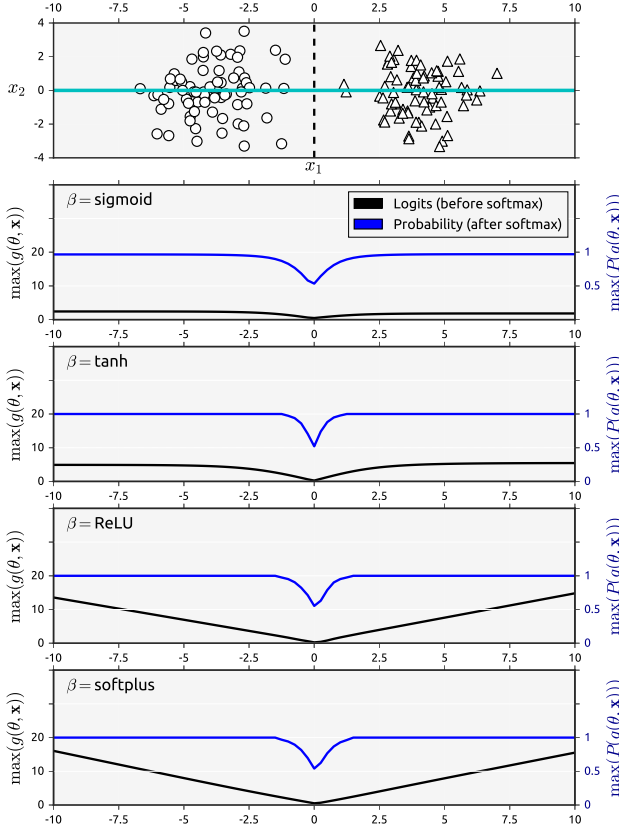


Figure 6.5: A linearly separable two-class classification problem, highlighting the saturation of the softmax output, as opposed to the logit values, which keep increasing.

can observe that the softmax output, after a certain point, remains the same and bounded.

In Figure 6.6, we present observations for multiple adversarial examples, showing the mean and the confidence interval of the highest logit value of 1,000 adversarial examples and their corresponding softmax output. The 1,000 adversarial examples were generated by making use of IFGS and the ℓ_2 version of CW, using 100 optimization steps in a white-box setting. For IFGS, we perturb the image one pixel at a time, and for CW, we use $\kappa = 40$. The increase in the logit values for IFGS is less pronounced than for CW since IFGS relies on the signature of the gradient, which removes the precision in the perturbation between pixels. On the other hand, the spike in the confidence for IFGS appears faster than for CW because the latter comes with

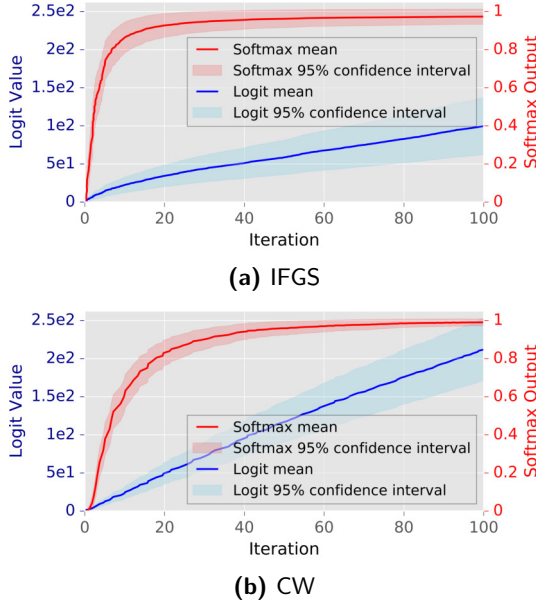


Figure 6.6: Highest logit value and corresponding softmax output as a function of the number of iterations when generating adversarial examples with (a) IFGS and (b) CW. Adversarial examples are tested on ResNet-50 in a white-box setting.

a multi-class optimization nature and implements an extensive search process. Figure 6.6 clearly shows the masking effect of softmax, given that the confidence almost immediately jumps to 100% after only a couple of iterations, making it from this point onward impossible to differentiate between consecutive adversarial examples, whereas logit values keep increasing over the course of the optimization.

The results provided in Figure 6.6 can be generalized to other methods for generating adversarial examples in an iterative and targeted way [22, 110]. These iterative methods make it possible to further optimize an adversarial example (in terms of logit values), even after obtaining full confidence. However, once the prediction confidence is equal to one, it is impossible to differentiate between the next iterations of the adversarial example based on the softmax output. Indeed, as the corresponding softmax input (i.e., logit values) keeps increasing, the softmax output will remain the same.

6.5 Identifying adversarial examples with large logits

Targeted attacks, in every form, focus on maximizing an activation; for neural networks, this activation is the logit value for a particular class. Unless this optimization is stuck in a local maximum, we can say that $g(\theta, \mathbf{x}^{(i+1)})_c \geq g(\theta, \mathbf{x}^{(i)})_c$, where c is the targeted class. If \mathbf{x} is a data point that is the subject of box constraints, then this activation can only be maximized up to a certain point. Let $\mathbf{x}_{(r)}$ be a hypothetical genuine image that achieves the highest logit value for its category (i.e., class r) in supervised settings. We are interested in finding the numerical value of $g(\theta, \mathbf{x}_{(r)})_r$, as this will allow us to label any data point that produces a higher logit value than this value as an adversarial example without any further evaluation. To that end, in order to come up with an effective method to estimate $g(\theta, \mathbf{x}_{(r)})_r$, we first analyze the prediction distributions of genuine images in terms of logit values.

6.5.1 Logit distributions

To come up with a robust method for countering adversarial examples (that is, to find a method that works for all models across all datasets), we first analyze the logit distribution of genuine images. A reoccurring problem in the field of adversarial examples is the usage of low-resolution images to show the effectiveness of a defense mechanism. Indeed, not all of the defense techniques for low-resolution images transfer to high-resolution images [23]. For this reason, we investigate the effectiveness of our method for MNIST, as well as for CIFAR-10 and ImageNet. In this context, we present Figure 6.7, which shows the densities of predicted logit values for all correctly classified samples of ten classes taken from MNIST, CIFAR-10, and ImageNet, for both training and test datasets. In the case of MNIST and CIFAR-10, we use LeNet and Extended LeNet [158, 23] architectures and in the case of ImageNet, we present the distribution for both VGG-16 and ResNet-50. We can observe that the distributions of the logit values change between different datasets and between different classes/labels. On top of that, the distributions are vastly different for different architectures (VGG-16 and ResNet-50), even when the same dataset is used.

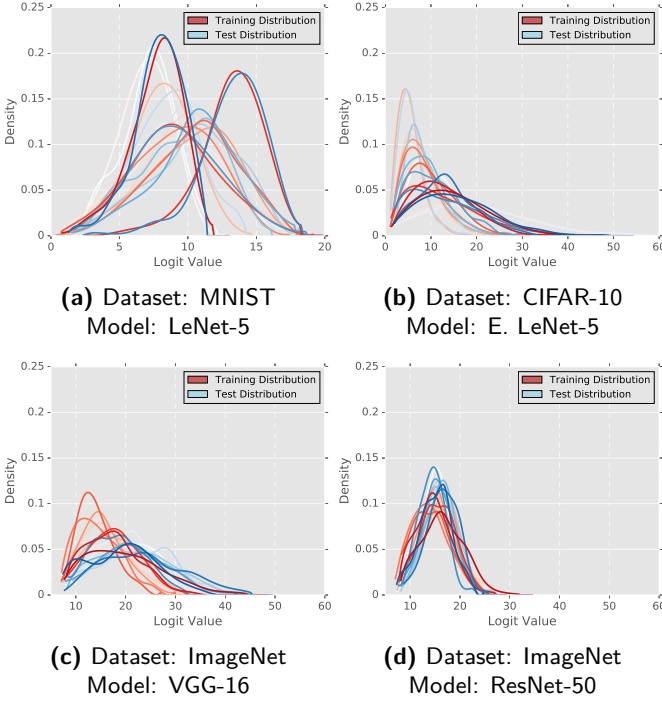


Figure 6.7: Density plots of the logit values associated with the predictions obtained for ten classes, observed for both seen and unseen examples of the (a) MNIST, (b) CIFAR-10, and (c), (d) ImageNet datasets. We used LeNet-5 with ReLU activation [114], Extended LeNet-5 [158, 23], VGG-16 [189], and ResNet-50 [75] to obtain these results. For ImageNet, ten classes were selected randomly. Our models achieved 98%, 80%, 70.5%, and 77% top-1 accuracy on the respective datasets. These results are comparable to the results presented in [23, 75, 158, 189].

Keeping the aforementioned observations in mind, we arrive at the following conclusions: (a) the proposed method should be distribution-free so that it can generalize across multiple datasets and multiple models, and (b) since the distribution of logit values also changes between different classes, the computational complexity of the proposed method should be low.

6.5.2 Determining the threshold

Considering that datasets only contain a limited number of representations (per class), it is highly unlikely that the hypothetical genuine image that attains the highest logit value is present in the dataset. However, we can estimate a threshold based on the logits of the observations at hand. A critical point in determining this threshold is to make sure that none of the existing images in the dataset are labeled as adversarial, as it is more important not to cast doubt on good observations than to miss an outlier [52, 72]. The same point is also highlighted by Carlini and Wagner [23] when evaluating defense mechanisms.

As we showed previously, the logit value distributions are vastly different. Hence, when identifying outliers, we want to avoid any method that (implicitly or explicitly) makes assumptions on the data distribution. Therefore, based on the idea of boxplots, we propose using the interquartile range (IQR) to determine the threshold for identifying outliers (i.e., identifying adversarial examples). The IQR is defined as the difference between the 75th percentile (Q_3) and the 25th percentile (Q_1): $IQR = Q_3 - Q_1$. In basic statistical analysis, outliers are generally defined as those points that lie beyond the whiskers of the boxplot (i.e., below $Q_1 - k \text{ IQR}$ or above $Q_3 + k \text{ IQR}$). Traditionally, k is set to 1.5 [209], in which case the outliers are referred to as *mild*. We are, of course, only interested in large positive outliers. In this context, the authors of [52], [72], and [209] recommend using $k = 3$ for determining *extreme* outliers (i.e., highly unusual data points). Since we do not exactly know the underlying distribution of the logits, we experiment with different k -values.

For each class, we calculate $Q_3 + k \text{ IQR}$ as the threshold from the training sets of the aforementioned datasets and we present the percentage of misidentified images for the test datasets. Specifically, in Table 6.3, we show the percentage of genuine images that are misidentified as outliers (i.e., adversarial examples) by the calculation $g(\theta, \mathbf{x})_c \stackrel{?}{>} (Q_3 + k \text{ IQR})_{(c)}$, for different values of k . Unlike MNIST and CIFAR-10, for which we only make use of one architecture, in the case of ImageNet, we present results for three different architectures, namely, VGG-16 [189], ResNet-50 [75], and Inception-V3 [203]. Based on these results, we can observe that $k = 3$ is mostly sufficient for nearly all architectures, as the percentage of misidentified genuine images is, at most, as low as 0.07% (this corresponds to only 35 images in the test dataset of ImageNet). Nevertheless, it is also reasonable to

Dataset (Model)	1.5 IQR	2 IQR	3 IQR	4 IQR	5 IQR	k_{min}
MNIST (LeNet-5)	0.1%	0.004%	0%	0%	0%	2.2
CIFAR-10 (Extended LeNet-5)	1.7%	0.6%	0.07%	0.01%	0%	4.9
ImageNet (VGG-16)	1.1%	0.3%	0.03%	0.004%	0%	4.6
ImageNet (ResNet-50)	0.7%	0.1%	0.009%	0%	0%	3.9
ImageNet (Inception-v3)	1.2%	0.4%	0.07%	0.001%	0%	4.9

Table 6.3: Percentage of genuine images incorrectly identified as outliers, as obtained for different values of k in the following calculation: $g(\theta, \mathbf{x})_c >? (Q_3 + k \text{ IQR})_{(c)}$. Thresholds are calculated from correctly classified training examples for each class and tested on both training and test examples. k_{min} is the smallest value of k needed to ensure that none of the examples in the training and the test set are incorrectly identified as outliers.

prefer a threshold that does not reject *any* genuine image at all. In that regard, we can easily find k_{min} for different datasets and models, given that the proposed method is non-parametric in nature, which makes it straightforward to adopt this method for different problems. Additionally, by using the IQR instead of parameters such as the mean or standard deviation, which are sensitive to outliers and which make implicit assumptions about the underlying distributions, the proposed method is also more robust (i.e., not attracted by outliers). In the next section, we present experiments that show how the proposed method can be effectively used to detect over-optimized adversarial examples.

6.5.3 Experiments

Using $T_{(c)} = (Q_3 + k \text{ IQR})_{(c)}$, we calculated logit thresholds that do not leave out any genuine images for MNIST, CIFAR-10, and ImageNet, using the k_{min} values given in Table 6.3. For each of the selected classes in those datasets (all classes in the case of MNIST and CIFAR-10; ten randomly selected classes in the case of ImageNet), we generate 500 adversarial examples using IFGS and the ℓ_2 version of CW, totalling up to 5,000 adversarial examples for each dataset, with the aim of finding the adversarial example that generates the highest logit value for that class. By doing so, we want to determine the space for which we can identify *all* adversarial examples that lie between the proposed threshold and the produced logit limit for the adversarial example generation methods.

Figure 6.8 and Figure 6.9 show the results obtained for MNIST, CIFAR-10, and ImageNet (using ResNet-50), with the predicted logit

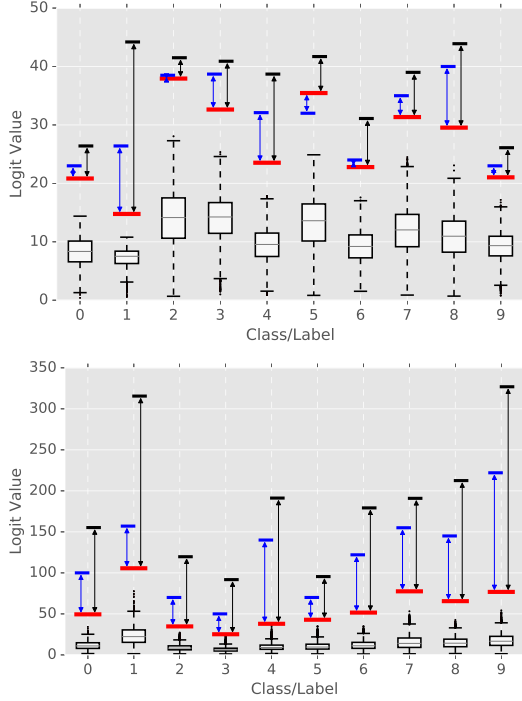


Figure 6.8: Illustration of adversarial subspaces with high logit values for the MNIST (top) and CIFAR-10 (bottom) datasets. Logit value distributions of the predictions for the genuine images from the training set are given in the form of boxplots. The calculated thresholds and the highest logit value of the adversarial examples generated with IFGS and CW are highlighted as red, blue, and black lines, respectively. The adversarial spaces found using our method are indicated with arrows.

values given in the form of boxplots and with the calculated thresholds highlighted using red lines. The adversarial examples that generate the highest logit prediction values are highlighted separately for the two adversarial example generation methods, using a blue line for IFGS and a black line for CW. We annotate the space between the proposed threshold (red) and the maximum logit value for each adversarial example generation method $g(\theta, \mathbf{x})_c > T_c$ to show that we can immediately identify numerous adversarial examples within this space, no matter which method is used to generate these adversarial examples. As expected, for all three datasets, CW generates *stronger* adversarial examples that produce higher logit values than IFGS.

Note that, as the image resolution increases, the upper limit for

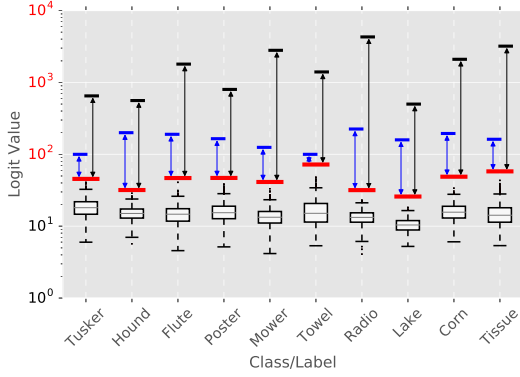


Figure 6.9: Illustration of adversarial subspaces with high logit values for the ImageNet datasets. Logit value distributions of the predictions for the genuine images from the training set are given in the form of boxplots. The calculated thresholds and the highest logit value of the adversarial examples generated with IFGS and CW are highlighted as red, blue, and black lines, respectively. The adversarial spaces found using our method are indicated with arrows. Note that the y -axis is log-scaled for clarity.

the logit values that can be produced by an adversarial example also increases. From MNIST to CIFAR-10, this limit increases by a factor of 7, and from MNIST to ImageNet, this limit increases by a factor of 200. In the case of the ImageNet dataset, the increase is so high that we present the y -axis of the graph in the base ten logscale for the sake of readability.

6.5.4 Benefits of the proposed defense

Feasibility for higher resolution images – As shown by Figure 6.8 and Figure 6.9, when the resolution of an image increases, the space in which adversarial examples can be generated also increases in a significant way. This property is also highlighted by other studies [188].

Let us define, for a given model, the effectiveness of our method as

$$D_{Adv} = \frac{1}{M} \sum_{c=1}^M \frac{g(\theta, \hat{\mathbf{x}}_{(c)})_c - T_{(c)}}{g(\theta, \hat{\mathbf{x}}_{(c)})_c}, \quad (6.2)$$

where M is the number of classes in the dataset, c is the current tar-

Dataset (Model)	Image Resolution	k_{min}	Space Covered $D_{Adv} \in [0, 1]$
MNIST (LeNet-5)	$1 \times 28 \times 28$	2.2	26%
CIFAR-10 (Ext. LeNet-5)	$3 \times 32 \times 32$	4.9	68%
ImageNet (VGG-16)	$3 \times 224 \times 224$	4.6	75%
ImageNet (ResNet-50)	$3 \times 224 \times 224$	3.9	79%
ImageNet (Inception-v3)	$3 \times 299 \times 299$	4.9	89%

Table 6.4: Approximate proportion of the number of adversarial examples detected in each dataset (calculated using Equation 6.2).

get class, $\hat{\mathbf{x}}_{(c)}$ is the adversarial example that produces the highest logit value when targeting class c , and $T_{(c)} = (Q_3 + k_{min} IQR)_{(c)}$ is the calculated threshold for the targeted class. For each class, we calculate the space between the threshold $T_{(c)}$ and the logit value of $\hat{\mathbf{x}}_{(c)}$, and we subsequently normalize this space by dividing it by the length of the total space. This value corresponds to the proportion of the adversarial space for class c with respect to the size of the total target class that we can detect by our method. We then take the average over all classes to get an approximate idea of the proportion of the adversarial space we can detect in function of the total dataset. Thanks to the normalization factor, D_{Adv} will allow us to compare different dataset/model combinations, where a higher number indicates a higher effectiveness of detecting adversarial examples.

Applying this formula allows us to construct Table 6.4, which shows the proportion of potential adversarial examples detected. Based on the results presented in Table 6.4, we observe that the proposed method is able to detect more adversarial examples when the resolution of an image is higher.

Scalability and computational cost – In Section 6.5.2, we noted that the thresholds must be calculated individually for each class. Naturally, when the number of classes increases, the number of thresholds that must be calculated also increases. However, since these thresholds only need to be calculated and stored once, the computational cost of the proposed method is small; it only requires one full forward pass over the training dataset, after which the thresholds can be used indefinitely. For the ImageNet dataset, which consists of almost 1.3 million samples and 1,000 classes, calculating a threshold with ResNet-50 for all classes takes only 14 minutes on a single Titan-X GPU complemented by an Intel Xeon CPU E5-2620v4.

Another strength of the proposed method is the easiness with which thresholds can be updated. When the size of a training set increases (thanks to an influx of additional data), the thresholds can simply be re-calculated.

6.5.5 Limitations of the proposed defense

In previous sections, we put the emphasis on the number of potential adversarial examples countered using the proposed method. However, only using the proposed method is not a viable option to identify all adversarial examples. This is especially true in white-box settings, for which it is easy to come up with an attack that would bypass our method. Indeed, one can simply introduce a logit constraint when generating adversarial examples, making sure the generated adversarial examples produce lower logit values than the proposed thresholds. A generalized implementation of this approach is given below,

$$\hat{\mathbf{x}}^{(n+1)} = \hat{\mathbf{x}}^{(n)} - f_1(\hat{\mathbf{x}}^{(n)}) + f_2(\hat{\mathbf{x}}^{(n)}) \quad (6.3)$$

$$f_2(\mathbf{x}) = \sum_{m=1}^M \mathbf{1}_{\{g(\theta, \mathbf{x}^{(n)})_m > \tau\}} \nabla_{\mathbf{x}} g(\theta, \mathbf{x})_m, \quad (6.4)$$

where f_1 indicates a selected adversarial loss employed for the creation of the adversarial example and where f_2 indicates the loss for avoiding the proposed defense, and where the latter loss aims at reducing the logits of the classes if they exceed the chosen threshold of τ .

Nonetheless, our intention in proposing this method is not to use it as a universal method for detecting all adversarial examples, but rather to add it to a particular defense workflow as a first (and computationally inexpensive) line of defense for identifying adversarial examples, with these adversarial examples producing unnatural logit values that are beyond the reach of any genuine images. Applying the proposed defense ensures that adversarial example generation methods will be limited not only by the amount of perturbation and the discretization constraint, but also by the maximum logit value produced by an adversarial example. This will limit the search space extensively, especially when the resolution of the images under consideration is large.

6.5.6 Similar logit-based defenses

Since the discovery of adversarial examples, a large number of defenses were proposed. As discussed in Section 6.1, these approaches can be categorized as follows: (1) input modification, (2) architectural investigation, (3) prediction analysis, and (4) modified training routine. Each of these methods comes with its own unique problems. Input modification may for instance remove important features of an image, thus decreasing the effectiveness of the model. To avoid a breakdown of the model under consideration, architecture modification requires meticulous planning. Adversarial retraining requires an enormous number of adversarial examples and increases the computational complexity of training, especially for large models and datasets. As such, alternative ways of defending against adversarial examples were investigated. Prediction analysis, and analysis of logits in particular, is just one of these alternatives [67, 107, 129, 172, 192]. In this section, we will highlight two approaches that are similar to the defense we proposed.

Hawkeye — Koo et al. [107] proposed a novel method for adversarial retraining, arguing that training a second neural network with a different architecture (i.e., with different θ_2) could be used in order to identify anomalies in logit predictions, thus making it possible to detect adversarial examples. This new neural network is trained using $g(\theta_1, \mathbf{x}) - g(\theta_1, q(\mathbf{x}))$ as an input, where $g(\theta_1, \mathbf{x})$ and $g(\theta_1, q(\mathbf{x}))$ refer to the logit predictions made by the first neural network for the original image and the quantized version of the original image, respectively. Details on the quantization function used can be found in Koo et al. [107]. The underlying assumption is that the difference between $g(\theta_1, \mathbf{x})$ and $g(\theta_1, q(\mathbf{x}))$ is significant when \mathbf{x} is an adversarial image rather than a genuine image, thus making it possible for the second neural network to recognize any anomalies. The authors claim that this method is able to achieve a white-box detection rate of almost 100% for adversarial examples generated with FGS and IFGS, for genuine images taken from MNIST and ImageNet [107, Figure 10].

Upon investigating why a detectable difference exists for $[g(\theta_1, \mathbf{x}) - g(\theta_1, q(\mathbf{x}))]$ when \mathbf{x} is an adversarial example, we found that, when FGS or IFGS is used in the settings adopted by Koo et al. [107], the logit prediction of the target class becomes too large or too small, depending on the class used, compared to the logit prediction of the genuine data points and the quantized input. The difference between the logits then allows the second neural network to identify the aforementioned logit anomalies. However, different from their work, we

observed that the usage of the second neural network does not provide any benefit to this detection method. Furthermore, we noticed that any applicable statistical test is also able to detect the adversarial examples created within the given conditions.

In order to further scrutinize this defense, we generated adversarial examples using the following approach:

$$\begin{aligned} \text{mimize} \quad & \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}\|_2^2 + \zeta(\tilde{\mathbf{x}}, \hat{\mathbf{x}}), \\ \zeta(\tilde{\mathbf{x}}, \hat{\mathbf{x}}) = & \|g(\theta, \tilde{\mathbf{x}}) - g(\theta, \hat{\mathbf{x}})\|_2^2, \end{aligned} \tag{6.5}$$

where $\tilde{\mathbf{x}}$ is a genuine data point that has been taken from the training set, having a different class prediction than \mathbf{x} (i.e., $\arg \max(g(\theta, \mathbf{x})) \neq \arg \max(g(\theta, \tilde{\mathbf{x}}))$). This simple approach aims at minimizing the ℓ_2 distance between the prediction logits of the adversarial example and the prediction logits of another genuine data point, with the goal of disguising logit predictions as if they are from a genuine data point. At the same time, the aforementioned approach also adopts ℓ_2 distance minimization between the produced adversarial example and its genuine counterpart, making it possible to satisfy the requirement of having visually unnoticeable perturbations.

Using a ResNet-50 model, hereby generating adversarial examples through Equation 6.5 and targeting logit distributions where the prediction class is among the top-5 classes of the natural image, we observe that, even when quantization is applied to an adversarial example, the difference between logits for each class remains low. Even for the case where $\arg \max(g(\theta_1, q(\mathbf{x}))) \neq \arg \max(g(\theta_1, \mathbf{x}))$, the difference for the largest prediction becomes insignificant, with the second neural network not being able to detect any anomalies. As a result, we are able to circumvent the defense at hand with a success rate of 98% in white-box settings.

Log-odds Statistical Test — Roth et al. [172] proposed the use of perturbed log-odds (odds of logits; not to be confused with the logarithm), a statistical test to estimate the true class of adversarial examples. Assume that

$$g_{c_1, c_2}(\theta, \mathbf{x}) = g(\theta, \mathbf{x})_{c_2} - g(\theta, \mathbf{x})_{c_1} \tag{6.6}$$

is the pairwise log-odds between classes c_1 and c_2 for an input \mathbf{x} . For each fixed class pair $(c_1, c_2) \in (\{1, 2, \dots, M\} \times \{1, 2, \dots, M\})$,

$$h_{c_1, c_2}(\theta, \mathbf{x}, \delta) = g_{c_1, c_2}(\theta, (\mathbf{x} + \delta)) - g_{c_1, c_2}(\theta, \mathbf{x}) \tag{6.7}$$

is calculated and subsequently standardized using Z-score standardization, an outcome that will be further referred to as $\bar{h}_{c_1, c_2}(\theta, \mathbf{x}, \delta)$. The authors put forward the use of the expected log-odds $\bar{h}_{c_1, c_2}(\theta, \mathbf{x}) = \mathbb{E}_\delta[\bar{h}_{c_1, c_2}(\theta, \mathbf{x}, \delta)]$ as a statistical test in order to detect adversarial examples, hereby calculating

$$\max_{c_1 \neq c_2} \{\bar{h}_{c_1, c_2}(\theta, \mathbf{x}) - \tau_{c_1, c_2}\} \geq 0, \quad (6.8)$$

where τ_{c_1, c_2} is a threshold for class pairs, selected to guarantee a maximum false positive rate of 1% for genuine images.

Recreating this defense in adversarial settings, we were indeed able to achieve similar results as claimed in Roth et al. [172]. A strength of this approach is its ability to detect adversarial examples when the logit prediction for the target class $g(\theta, \mathbf{x})_c$ is originally low. As such, during the adversarial optimization, the increase in the logits for the other classes shows large logit differences compared to its unperturbed counterpart, which enables the log-odds test to detect irregularities in the logit differences.

In order to circumvent this defense, we first select a data point \mathbf{x} from the testing set and then another data point $\bar{\mathbf{x}}$ from the training set such that $\sum_{i=1}^M g(\theta, \mathbf{x})_i - g(\theta, \bar{\mathbf{x}})_i$ is smallest among all other data points, while satisfying $\arg \max(g(\theta, \mathbf{x})) \neq \arg \max(g(\theta, \bar{\mathbf{x}}))$ (M represents the number of classes). Using these data points, we apply the adversarial attack described by Equation 6.5 and evaluate the defense under consideration against the produced adversarial examples, for a ResNet-50 model using the ImageNet dataset. We find that we are able to bypass this defense with 96% accuracy.

Upon closer inspection, we can observe that for certain images in the testing dataset with initially low-confidence predictions, even when adversarial examples are generated with the method described above, the added perturbation in the log-odds test may sway the prediction in favor of unpredictable classes. Picking up from this behavior allows the log-odds test to identify the adversarial nature of the input, leading to the 4% failure rate.

6.6 Limitations of logit-based defenses

Based on our experience with logit-based defenses, both the proposed defense as well as other defenses discussed in the literature, we found

that such approaches contain two major flaws in detecting certain portions of adversarial examples. In what follows, we will describe the identified limitations in more detail.

Choice of activation functions — The reason logits became a target for adversarial defense studies, instead of the softmax output [18], is because of the vanishing property of the softmax function, as described in Section 6.4. However, most, if not all, of the studies that propose a logit-based defense mechanism present experimental results obtained for DNN architectures that use some form of rectifier [58] as an activation function.

Recall Figure 6.5 where we presented a two-class classification problem that was solved with a number of neural networks that contain various activation functions. In that experiment, we illustrated that, when an activation function with a bounded positive limit is used to solve that problem, after a certain point, the highest logit prediction does not change. For such neural networks that incorporate activation functions similar to sigmoid or tanh, it is impossible to detect adversarial examples based on logit differences under certain conditions.

Illusion of identifiable logit predictions — Even if we assume that the underlying neural networks contain activation functions that facilitate logit scaling, thus allowing for the identification of outlying data points using logits, another limitation of logit-based defenses is their inability to detect adversarial examples in certain regions. In order to clearly visualize this limitation, Figure 6.10 shows an extension of the classification problem presented in Figure 6.5. Specifically, the upper graph in Figure 6.10 now contains two adversarial examples, one at $(x_1, x_2) = (12.5, 0)$ and another one at $(x_1, x_2) = (3, 10)$. Both adversarial examples originated from the point $(x_1, x_2) = (-4.5, -4.2)$, but with the former generated through adversarial logit manipulation (using Equation 6.5) and the latter generated through IFGS. The movement from the original point to the generated adversarial example, through the decision boundary and beyond, is illustrated using a red line for adversarial logit manipulation and a black line for IFGS.

The second and third graph of Figure 6.10 provide logit and softmax predictions of points that lie along the cyan and magenta line, respectively, when classification is performed with the neural network that makes use of the ReLU activation function, as previously used to obtain the corresponding results shown in Figure 6.5. As can be seen, adversarial examples that lie along the direction of the cyan line are trivial to identify based on their increasing logit predictions. However,

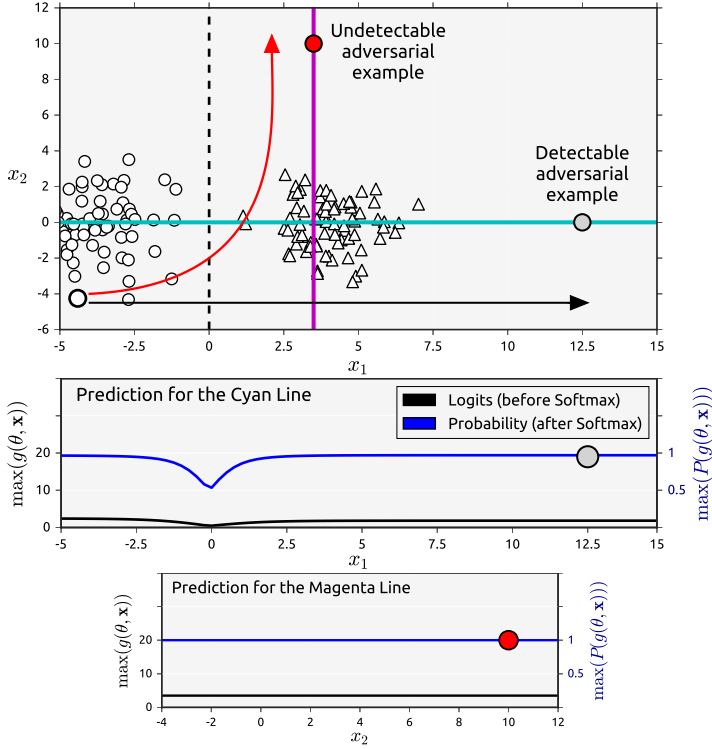


Figure 6.10: Two adversarial examples (red and gray circle) and their initial starting point (white circle) are added to the classification problem previously shown in Figure 6.5. Logit and softmax predictions of data points that lie on the cyan ($x_2 = 0$) and magenta ($x_1 = 3$) lines are presented as the second and third graph, respectively.

all data points that lie along the magenta line have exactly the same logit prediction, thus making it impossible to identify an adversarial example in this direction based on logits.

The entire range of directions (which we referred to as *regions*) with undetectable outliers based on logit predictions has not been determined but we conjecture these directions to be (approximately) parallel to the decision boundary in our simple case. However, methods to identify these regions for more complex situations are currently missing.

6.7 Conclusions and future work

In this chapter, we briefly discussed a number of defenses employed to prevent adversarial examples. We then examined the feasibility of adversarial retraining, observing that neural networks are indeed able to identify adversarial perturbation patterns as generated by individual attacks and that adversarial retraining can be used to differentiate genuine images from their adversarial counterparts on ImageNet, given that the perturbation patterns obtained for this dataset are more distinct than the perturbation patterns obtained for datasets with smaller image sizes.

Next, we investigated the feasibility of another approach for detecting adversarial examples, establishing the fundamentals for the use of logit values as an indicator of adversariality. To that end, we first discussed the masking effect of the softmax function, showing that the logit values keep increasing, even when the softmax output has already achieved its maximum value of one during the generation of adversarial examples.

Finally, we presented logit distributions from multiple datasets, demonstrating the need for a distribution-free method to identify adversarial examples. This observation led to the introduction of a non-parametric and computationally cheap technique for detecting over-optimized adversarial examples. Throughout this chapter, we presented experimental results for MNIST, CIFAR-10, and ImageNet, also using different neural network architectures.

The main purpose of the newly introduced defense technique is to further limit the expressiveness of methods for adversarial example generation, not only in terms of perturbation amount, but also in terms of possible logit values. This makes it possible for our technique to be used as a first line of defense, before triggering a well-rounded defense that is more complex in nature. Therefore, future research may focus on investigating the compatibility of our technique with other defense mechanisms that are complimentary in nature.

7

Conclusions and outlook

Automated decision-making systems that leverage DNNs are used to solve a wide variety of everyday problems. Such models are, at an increasing pace, being employed to directly interact with humans. In this dissertation, we have investigated adversarial examples, one of several security threats to automated systems relying on DNNs. Our investigation started with the exploration of the properties of adversarial examples in Chapter 3, continued with a threat model evaluation in the context of adversarial attacks, first on biomedical segmentation models in Chapter 4 and then on radar-based activity recognition systems in Chapter 5, and ended by examining a number of plausible defenses in Chapter 6.

7.1 Summary

The research efforts presented in the preceding chapters can be summarized as follows:

Chapter 3

- Generating adversarial examples is a challenging topic. To do so, researchers can either make use of a simple approach, such as FGS, or a complex approach, such as CW. In any case, what is the difference between adversarial examples produced by different methods? In order to find an answer to this question, we investigated a number of popular loss functions used for the generation of adversarial examples, revealing several properties of interest of these loss functions.

- Is the adversarial perturbation equally effective in all regions of an image? We found that certain parts of an image allow genuine images to become adversarial examples with much less perturbation when perturbation levels are measured with ℓ_p norms.

Chapter 4

- Adversarial examples were, and still are, investigated mostly in the context of classification problems. In order to analyze the vulnerability of DNNs that are used to solve other types of machine learning problems, we investigated adversarial threats to biomedical image segmentation models. We proposed the Adaptive Segmentation Mask Attack, a novel adversarial attack that can be employed to produce both targeted and untargeted adversarial examples for image segmentation models.
- Observations made in this chapter demonstrated that biomedical image segmentation models are, indeed, vulnerable to adversarial attacks, possibly leading to the exploitation of automated systems for misdiagnosis, insurance fraud, and manipulating public opinion.

Chapter 5

- Given the recent developments in consumer-level radio detection and ranging devices (radar), radar-based automated decision-making systems are actively explored for augmenting video cameras, with the goal of mitigating the downsides of visual data (e.g., privacy concerns). We investigated the vulnerability of such systems against adversarial attacks, finding that commonly used adversarial attacks are effective in deceiving them.
- Leveraging the intrinsic properties of the data at hand, we proposed a novel adversarial attack that is able to mislead predictive models by only perturbing padding, without modifying the radar frames in which a human activity of interest takes place.
- We aimed at linking the adversarial vulnerability of each radar frame depicting a human activity to interpretability, finding strong correlations between the radar frames deemed to be important for adversarial perturbation and the radar frames identified as important by the Grad-CAM interpretability technique.

Chapter 6

- Adversarial retraining is argued to be a method that can effectively detect or mitigate the impact of adversarial examples. We experimented with the use of adversarial retraining on ImageNet, paying attention to the effectiveness of retraining with a diverse set of adversarial examples that were generated using the loss functions investigated in Chapter 3. We found that adversarial retraining is indeed effective in detecting adversarial examples. However, it requires a tremendous amount of data in order to achieve a high effectiveness.
- Thanks to the lessons learned from the research efforts presented in the previous chapters, we proposed a novel adversarial defense that is based on a statistical analysis of the predictions made by neural networks. This defense was designed as a first-line defense against adversarial attacks, targeting the detection of what we called over-optimized adversarial examples.

7.2 Reflections on adversarial machine learning research

In previous chapters, I have provided technical details about the contributions made during my doctoral study on adversarial examples, attacks, and defenses. These contributions, which were published as peer-reviewed scientific articles, were made possible thanks to the efforts of many people, with the co-authors of the aforementioned articles being the most important ones. However, a doctoral study is much more than the articles published and the technicality of the discoveries made. In this section, I would like to share a few concerns I have regarding the state of research in the field of adversarial machine learning, and where these concerns were accumulated over the course of my doctoral study, actively pursuing research in this field. I feel burdened to share them because such topics are often excluded from published articles. I will narrate these concerns using three topics: (1) representation of adversarial examples, (2) reproducibility of research, and (3) shortcomings of proposed methods.

7.2.1 Representation of adversarial examples

After the discovery of adversarial examples and their pervasive effects on DNNs, researchers labeled these malicious data points as one of the biggest security threats to predictive models [22, 62]. In doing so, research that followed the initial discovery of adversarial examples worked in the spirit of “making machine learning more secure” [157, 158, 206]. However, while discussing this security issue, most papers seem to disregard a simple yet important topic: correctly representing adversarial examples for the application domain they are discussed in.

Let me give a number of examples from the image domain, given that images were the primary type of input used by my doctoral research. Although numerical pixel values of an image are taken from $\{0, 1, \dots, 255\}$, and although an image is represented as $\mathbf{X} \in \{0, 1, \dots, 255\}^{C \times H \times W}$, more often than not, these values are normalized in order to be able to make use of a smaller range. In that case, an input is represented as $\mathbf{X} \in \{f(0), f(1), \dots, f(255)\}^{C \times H \times W}$, where f is the adopted normalization method. A simple yet popular example of such a normalization method in the image domain is $f(x) = x/255$, which enforces inputs to take a value between 0 and 1. In this case, an input is represented as $\mathbf{X} \in \{0/255, 1/255, \dots, 255/255\}^{C \times H \times W}$. The aforementioned normalization method does not change the number of unique values the input can take (i.e., 256 unique values), a statement that also holds true for another commonly-used normalization method, which is employed when training popular DNNs such as VGGs [189], ResNets [75], and DenseNets [86]: per-channel (i.e., RGB) mean/standard deviation normalization.

Given the above observations, let us assume that a neural network was trained with images, and where these images contain pixels that can only take 256 unique values. In this case, what happens if we provide this neural network with an image input that contains values outside the ones the neural network was trained with? As this input image contains abnormalities (compared to the values the model was trained with), is it not expected that the output will also contain abnormalities?

Related to the hypothetical questions above, let me narrate the fascinating yet embarrassing insight I gained during the first year of my doctoral study. The notation I will employ for explaining this insight is as follows: let $\mathbf{X}_G \in \{0, 1, \dots, 255\}^{C \times H \times W}$ denote a genuine image in which each pixel has an RGB value between 0 and 255, and let f and f^{-1} denote the normalization and denormalization functions,

respectively. This means that $f(\mathbf{X}_G) = \mathbf{X}_N$ and $f^{-1}(\mathbf{X}_N) = \mathbf{X}_G$, where \mathbf{X}_N represents the normalized image. Similar to the previous chapters, I will represent the vector output, obtained through a forward pass with a classification neural network that takes as input the normalized image, as $g(\theta, \mathbf{X}_N)$.

For one of the articles we were working on at that time, I employed IFGS to generate adversarial examples ($\hat{\mathbf{X}}$). Following the guidance of previous work, I perturbed images iteratively with one third of a pixel value at a time (corresponding to $f(1/3)$ after normalization) until an adversarial example was created. When an adversarial example was produced, I saved the prediction vector $g(\theta, \hat{\mathbf{X}}_N)$, as well as the denormalized (discretized) adversarial example $\hat{\mathbf{X}}_G = f^{-1}(\hat{\mathbf{X}}_N)$ in the PNG image format. Doing so, I created about 1,000 adversarial examples. In the following days, just to convince myself my code was free from bugs, I normalized all of the produced adversarial examples $f(\hat{\mathbf{X}}_G)$ and inspected their output with the same model. To my surprise, the predicted values obtained for all adversarial examples were different from the predicted values I had saved before (i.e., $g(\theta, \hat{\mathbf{X}}_N) \neq g(\theta, f(\hat{\mathbf{X}}_G))$). Not only that, about one third of the adversarial examples were not even adversarial examples. Their predictions were the same as the source images they were created from! After this observation, I spent hours debugging my code. My first suspicion was the implementation of the normalization and denormalization functions, which I quickly realized did not any have bugs, convincing myself $f^{-1}(f(a)) = a$, $a \in \{0, \dots, 255\}$ holds. I then investigated the external libraries I employed to save images, then the model, then the images themselves, but I could not find any bugs.

It was only a couple of days later that I realized what had happened. Because the adversarial attack performed the perturbation with a multiplier of $f(1/3)$, any perturbation that remained in the range of $(f(\kappa), f(\kappa + 1))$, $\kappa \in \{0, 1, \dots, 254\}$ got mapped to either κ or $\kappa + 1$ during discretization (i.e., denormalization), because genuine images must have values in \mathbb{N} . Furthermore, during the adversarial attack, some pixels had their values changed to values outside the box constraints of real images (i.e., values that are less than 0 or greater than 255, which do not have a corresponding color). As a result, the adversarial examples as they were could not be discretized as real images. Indeed, those adversarial examples were anomalous inputs with pixel values the model had never seen before (during the training procedure). Moreover, saving those adversarial examples as real images would lose a portion of the adversarial perturbation (i.e.,

$f(f^{-1}(\mathbf{x}_N)) \neq \mathbf{x}_N$, lessening or sometimes completely removing the adversariality of the input.

Note that the box constraints, as well as the values the input can take, depend on the application domain used. Recall from Chapter 5 that the box constraints for the radar inputs we examined were in between 0.31 and 0.83 after normalization, instead of in $[0, 1]$. As such, domain knowledge is something that must be incorporated into the generation of adversarial examples for a faithful analysis.

As far as I am aware of, the work of Carlini and Wagner [22] is the only paper that mentioned the above-described phenomenon. Specifically, acknowledging the condition, Carlini and Wagner [22] noted that the potency of adversarial perturbation may lessen during discretization. However, it is not unusual to come across research papers that denote adversarial examples as $\hat{\mathbf{x}} \in \mathbb{R}^{C \times H \times W}$, refraining from mentioning both discretization and box constraints.

One may question the significance of the topic discussed above, something that, I believe, has to be discussed from two different viewpoints: its relevance to academic research and its consequences for industrial usage.

The first viewpoint touches upon the trustability of the observations made when relying on “adversarial examples” that come with non-discretizable values, which are technically incorrect inputs. From an academic perspective, do the observations made for such “adversarial examples” also hold true for real-world adversarial examples (i.e., adversarial examples that are valid for the given domain)? Would there be a difference? Should we disregard the prior and only analyze the “real-world” adversarial examples or should the adversarial examples be analyzed separately? I do not know.

The second viewpoint is related to the industrial usage of DNNs. Since adversarial examples are deemed to be “a threat” to the in-production usage of DNNs, many academic studies employ a tone of computer security while discussing the relevance of the proposed methods. However, if such a study employs adversarial examples that are not valid for the particular application domain they are discussed in, their claims regarding the threatening nature of adversarial examples are insignificant because it ignores the basics of the life cycle of secure software development [37]. Specifically, it ignores one of the most important practices in the area of secure application development: input validation (also called input control). DNNs deployed to take input directly from the outside world can easily be augmented, or perhaps are already augmented, with simple input validation methods to detect

so-called “adversarial examples” that do not respect the basic rules of the application domain at hand. Given an input \mathbf{x} in the image domain, a simple set of rules for validating this input could look as follows:

$$\mathbf{y} = \begin{cases} \text{Reject input,} & \text{if } x_i \notin \{0, 1, \dots, 255\} \ \forall x_i \in \mathbf{x}, \\ g(\theta, \mathbf{x}), & \text{otherwise.} \end{cases} \quad (7.1)$$

Moreover, the set of rules outlined above can be expanded based on the expected input specifications, such as the color scheme, the resolution of the input, and the scenery.

Reflecting on the research papers we have published, even though I found ways to make sure that the problem I described above does not occur again and that the adversarial examples I created can be represented as valid images, in our published articles, we also continued to describe adversarial examples as, for example, $\hat{\mathbf{X}} \in \mathbb{R}^{C \times H \times W}$. This is because of unpleasant experiences I had with the academic review process after meticulously describing adversarial examples as, for example, $\hat{\mathbf{X}} \in \{f(0), \dots, f(255)\}^{C \times H \times W}$. Unfortunately, because the domain \mathbb{R} is almost exclusively used in this field to describe adversarial examples, using any other domain, even if it is more precise, alienates reviewers, creates unnecessary questions, and leads to additional hurdles. Given that this “less precise” description is employed *en masse*, do other researchers who work with adversarial examples ensure that their adversarial examples are discretizable and that they satisfy box constraints? Or do they discuss outcomes based on unrealistic adversarial examples that are anomalies? I do not know.

7.2.2 Research reproducibility

Unfortunately, not all peer-reviewed scientific articles are reproducible [92]. Although the reproducibility crisis is well-known and documented in other fields such as psychology [32] and medicine [91], in recent years, discussions about reproducibility problems also appeared in the machine learning community [76, 236], with recent news revealing that a large number of AI-related papers is suffering from reproducibility issues [10]. Different from, for example, research conducted in the field of medicine, research conducted in the field of computer science and its different branches enables reproducibility in a straightforward way: by releasing source code and data. However, Benaich and Hogarth

[10] found that about 80% of papers published in the area of computer science does not release the source code of underlying implementations.

Since the field of adversarial machine learning is closely aligned with the field of security, reproducibility is of utmost importance. Methods proposed to defend against adversarial examples are often accompanied by bold claims, with success rates upwards of 90% in preventing adversarial examples [206]. However, only in rare cases, articles that contain such claims are reproducible so that their results can be verified through (independent) follow-up studies [84]. Articles often do not provide enough information in order to allow for a perfect replication of the proposed methods, and as a result, the verification of such methods remains lacking [6].

On the other side of the story, one problem that pops up when making source code available is the time spent on cleaning and commenting this source code before it can be released. In three of our articles, we have proposed a novel adversarial attack: ASMA in [148], the padding attack in [154], and localized perturbation generation in [151]. I have made the source code publicly available for two of them: ASMA and localized perturbation generation.¹ For both research efforts, I had to spend a considerable amount of time on cleaning the source code so that others would have an easier time understanding. I decided not to release the source code for the padding attack because of the sheer amount of source code we produced for that research effort, combined with the legacy source code we inherited from Vandersmissen et al. [211]. I believe cleaning that source code to bring it to the quality level of the source code that was released for the two aforementioned research efforts would take weeks, if not more. Moreover, when releasing source code, cleaning is not the only effort that takes time. In my personal experience, making source code publicly available also brings additional hurdles, such as personal inquiries from people who want to run the source code but who either do not possess the technical skills to do so or who do not want to invest time in reading the documentation.

Although releasing the source code underpinning a scientific paper is seen as the honorable thing to do, the truth of the matter is that authors are not incentivized to do so. For example, given that doctoral students are often expected to publish as many papers as possible in order to guarantee their graduation, spending time on cleaning source code, releasing an implementation, and replying to inquiries effectively

¹See the repositories `adaptive-segmentation-mask-attack` and `regional-adversarial-perturbation` available at <https://github.com/utkuozbulak>.

consumes time that could otherwise be spent on producing more papers. Although I also believe that sharing an implementation in the form of source code is the right thing to do, I came to realize that the systems in place actively discourage me from releasing source code by not rewarding it the slightest. As such, it is indeed better for me in the long run to publish more papers than publish less papers that are easily reproducible. As a result, I came to believe that the push for reproducible science should come from institutional managers, conference organizers, funding agencies, and journal editors. Is that going to happen? I do not know.

7.2.3 Shortcomings of proposed methods

As discussed in Chapter 6, after the discovery of adversarial examples, a variety of methods to defend DNNs against adversarial attacks were investigated. Instead of taking these defense methods at face value, Carlini and Wagner [23] evaluated several of them and found that the claims made in the corresponding papers were incorrect, often in the settings those papers claimed robustness in. The authors of [23] continued evaluating a large number of defenses in later work, demonstrating that most studies proposing novel defenses against adversarial attacks lack rigor [6, 7, 206].

Upon the discovery of the aforementioned lack of rigor, Gilmer et al. [56] proposed a set of rules that can be used to guide adversarial machine learning research, encouraging researchers to act as true adversaries when evaluating defenses against adversarial attacks, so that shortcomings of these defenses can be investigated correctly and truthfully. However, revealing the shortcomings of novel methods often leaves the work exposed to easy criticism from reviewers, thus leading to rejections. Given the typical push to publish as many “high-quality” papers as possible, why would anyone expose their own work in such a way?

Before narrating the experience I had while publishing our study on a new defense, let me clarify a concept related to adversarial examples. Although the term “adversarial example” encompasses all types of adversarial examples, after the initial work of Szegedy et al. [201], it was discovered that different types of adversarial attacks lead to different types of adversarial examples [206]. Recall from Chapter 3, that we analyzed the properties of adversarial examples that were created through the usage of different loss functions. We also discussed adversarial examples with regional perturbation that are unique in

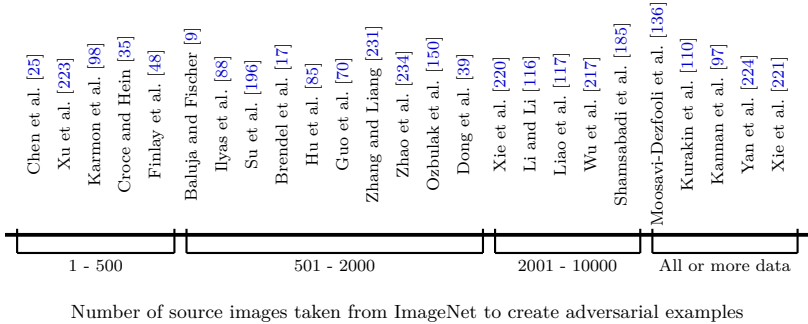


Figure 7.1: A number of studies that work with images taken from the ImageNet validation set, grouped based on the number of source images used for creating adversarial examples.

their own way. Given the aforementioned observations, the novel defense we proposed in [149] (explained in Chapter 6) was designed to detect a particular type of adversarial examples, namely those adversarial examples that produce anomalous outputs with logit values not attainable by genuine data points. We showed, through extensive analysis on multiple datasets, that our defense is indeed effective against such adversarial examples. However, our defense has a particular weak point: adversaries can simply use adversarial examples that do not produce such anomalous outputs and bypass it. In our draft article, we acknowledged that the proposed defense is vulnerable to such adversarial examples. However, due to this explicit acknowledgement, we made it possible for reviewers to easily criticize this particular part of our work. As a result, this weakness of our defense was pointed out by several reviewers, leading to rejections of our draft article before it was finally accepted for publication.

Although there are strong voices in the field of adversarial machine learning advocating for fairly criticizing new work [56, 206], it is my experience that explicitly pointing out weaknesses of new work exposes this work to effortless criticism by reviewers, with this criticism often bringing no added value to the table. Similar to the topic discussed in the previous section, I am actively discouraged to reveal the shortcomings of my own work because I need to have a certain number of papers published to secure my graduation. Although I believe it is indeed better for science, and the field of adversarial machine learning in particular, to be open about shortcomings in personal work, doing so just seems to be counterproductive. How can researchers be incentivized to reveal the downsides of their own work? I do not know.

7.3 Opportunities and future perspectives

From Chapter 3 to Chapter 6, we discussed a number of promising future work items that are relevant to the topics discussed in the respective chapters. In what follows, I will outline a number of future perspectives that I believe to be crucial to the field of adversarial machine learning, regardless of the focus of the investigation (e.g., on attacks, defenses, or properties of adversarial examples).

Data and research comparability – As discussed in Chapter 6, in recent years, numerous adversarial defenses were proposed in order to prevent or detect adversarial examples [67, 107, 116, 172]. The proposed defenses often claim a certain level of robustness against adversarial examples that have an amount of perturbation less than a selected norm [36]. Since the topic of adversariality is closely linked with security, reproducibility is of utmost importance. As a result, there have been a number of influential studies that analyze the correctness and reliability of newly proposed adversarial defenses [6, 7, 23, 206]. In this context, Carlini and Wagner [23], for instance, demonstrated that most of the defenses proposed for MNIST [114] do not even generalize to CIFAR [108]. This observation prompted research on the suitability of datasets for adversarial research [115], with Carlini and Wagner further suggesting that the usage of larger datasets such as ImageNet [175] may be necessary, given the lack of generalization of defenses proposed for smaller datasets [23].

Even though results obtained with ImageNet are more convincing, working with ImageNet is much more challenging than, for example, working with MNIST or CIFAR. Indeed, not only does ImageNet contain more images than the other two datasets, the images themselves are also larger. In addition, DNNs that achieve state-of-the-art results for ImageNet are also much bigger than their counterparts that achieve state-of-the-art results for MNIST or CIFAR, thus posing a challenge in terms of computational power needed. As a result, most of the studies that work with ImageNet only use a subset of images in order to create adversarial examples, unless that research is performed by a large industry lab that can afford the computational power. This fact can be observed from Figure 7.1, where we group a number of studies based on the number of images used from ImageNet.

Based on preliminary experiments with ImageNet, as discussed

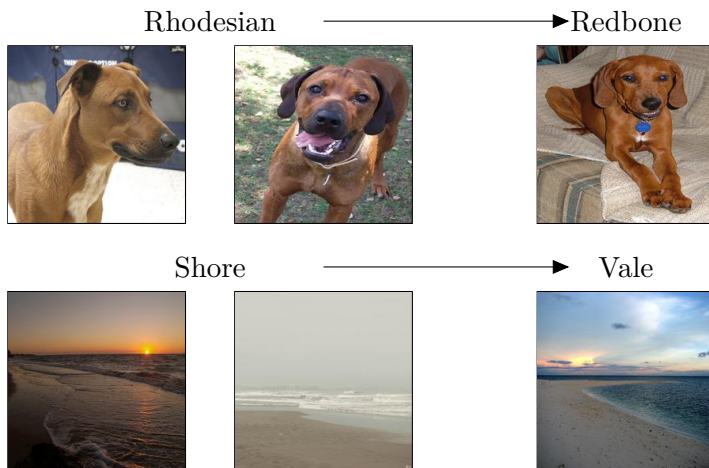


Figure 7.2: Adversarial examples that have been created with PGD, shown on the left, are misclassified into the categories listed on the right.

in more detail in [152, 153], we found that it is possible to have a difference of up to 12.5% in model-to-model adversarial transferability success, 1.01 in average ℓ_2 perturbation, and 0.03 (8/225) in average ℓ_∞ perturbation (when 1,000 source images are sampled randomly among all suitable candidates). We can thus confirm that it is not unlikely to arrive at somewhat misleading conclusions regarding the proficiency of attacks, as well as the robustness of defenses, when such methods are tested using a limited number of data points.

Given the typical lack of generalization power when working with datasets that contain a limited number of images, such as MNIST or CIFAR, with these images also coming with a small resolution, ImageNet stands out as the dataset that is most suitable for experimentation, as also suggested in [23, 206]. However, it appears that random sampling of images from ImageNet may yield inconclusive results, especially when a study claims marginal improvement over a preceding one [152]. As such, I believe the curation of a dataset containing 5,000 to 10,000 robust images from the ImageNet validation set and/or the ImageNet-10K set [38] for the purpose of benchmarking adversarial attacks and defenses would immediately allow for comparability between different studies. Such an approach would also democratize the research on adversarial machine learning, thus creating a level playing field between industry-funded labs and academic labs.

Forced misclassification and conceptual similarity – One of

the commonly used metrics when measuring the proficiency of adversarial attacks, the one we also used in various parts of this dissertation, is the adversarial model-to-model transferability of adversarial examples. The research efforts presented in Gilmer et al. [56] and Su et al. [196] hint that untargeted misclassification should not be considered an adversarial success, especially for large datasets that contain similar categories such as ImageNet [175]. Indeed, while performing experiments in the context of the topics discussed in previous chapters, we have encountered many scenarios where adversarial examples are classified into categories that are similar to the categories of genuine images they are created from. In order to demonstrate this property, we provide Figure 7.2, where the adversarial examples on the left are misclassified into the categories on the right. Note that both categories are highly similar and hard to distinguish. As such, I also believe targeted misclassification to be a better metric to measure the proficiency of adversarial attacks than untargeted misclassification. However, it is also known that the targeted misclassification success of adversarial examples is extremely low compared to their untargeted misclassification success [196]. Instead of making use of one of the aforementioned metrics, I believe a potential solution to this predicament would be the evaluation of, for example, top-5 classes and their categorical similarities to the category of the genuine image the adversarial example is created from. This type of approach would yield outputs rich with information, especially in comparison to doing a binary True/False assessment of model-to-model transferability.

Perturbation and conceptual similarity—During the earlier stages of research on adversariality, ℓ_p norms for measuring perturbation were employed in order to satisfy the requirement of having visually unnoticeable perturbation (i.e., of having non-suspicious input). However, it was recently demonstrated that measuring perturbation with ℓ_p norms is in fact not suitable for conceptual similarity [186]. In this context, a high conceptual similarity between two images means that those images contain objects with similar semantic content, thus evoking similar mental representations. Later, Zhao et al. [234] showed the possibility of having large yet invisible ℓ_p norm perturbations, thus suggesting that the adversarial risk associated with a classifier is much larger than initially thought. Perhaps the simplest example to demonstrate the lack of correlation between ℓ_p norm measurement and conceptual similarity is the usage of simple image transformations. In Figure 7.3, we provide two adversarial examples where the first one is created with PGD and the second one is created through the use

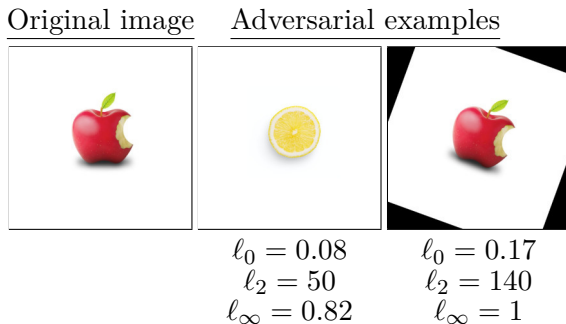


Figure 7.3: An original image and two adversarial examples created through the usage of an adversarial attack and rotation. The original image is correctly predicted as *apple*, while the adversarial examples are predicted as (left) *lab coat* and (right) *plane* by a ResNet-50 model trained on ImageNet. The ℓ_p norm of the perturbations used are provided below both adversarial examples.

of a rotation. Note that the second adversarial example is conceptually the same as the original image, but has larger ℓ_p norm perturbations compared to the first adversarial example, thus highlighting the weakness of using ℓ_p norms for measuring perturbation and conceptual similarity. Given the aforementioned shortcomings of ℓ_p norms for measuring conceptual similarity, future research on metrics that facilitate the measurement of such similarities is of high interest to the field of adversarial machine learning.

Need for re-evaluation—Some of the most influential research efforts in the field of adversarial machine learning have been studies that evaluate claims made in already published papers [6, 7, 206]. I have praised those works in Chapter 6, discussing their impact on the field of adversarial defenses. Such studies that analyze the integrity of claims made in previously published papers are a necessity to keep research efforts honest, not only for the field of adversarial machine learning, but for all fields. In the future, I expect to see more and more efforts that evaluate previously published studies, revealing the (intentional or unintentional) shortcomings of those studies, and where those shortcomings were not discussed before. Specifically, for the field of adversarial machine learning, I have no doubt that there will be many more efforts that evaluate previously published papers, identifying weaknesses and shortcomings of both attacks and defenses.

Adversarial defenses and robust decision-making systems –

In recent years, there has been an increase in the adoption of automated decision-making systems for solving all kinds of problems. As such, the potential impact of non-robust predictions on humans came to a point where they cannot be overlooked any longer. In the upcoming years, I expect regulatory bodies to question the trustability of predictions made by neural networks, leading to attempts to put systems in place that enable secure and trustable predictions. However, I do not know how successful these attempts will be, since a number of large corporations that leverage “AI” systems strongly lobby against such regulations. Despite that, I believe we will see regulations in at least two fields: self-driving cars and computer-assisted medical diagnosis. The former is due to the large media coverage of self-driving cars, and any problems associated with these cars in particular, which I believe will force the relevant regulatory bodies to act. The latter is because of potential insurance problems, especially in countries like the United States, where healthcare is expensive and where medical insurance coverage, or a lack thereof, may have a large impact. As such, I believe the implementation of (safety) regulations will give researchers a great opportunity in turning mostly theoretical work into practice. In particular, I believe that inventors of certified adversarial defenses, who patent their work, may gain a significant amount of income in the future, either by selling or renting their work.

Fully automated decision-making systems – Research on adversarial examples gained traction in recent years because deep learning models started to achieve results that were thought to be out of reach for methods that were then available. Since deep learning models themselves were proven to be capable in solving complex problems, researchers were able to investigate other aspects of these models, such as interpretability [181], calibration [68], and, of course, trustworthiness [62]. As of now, most of the work in the space of automated decision making aims at using models in conjunction with humans, namely through the adoption of human-in-the-loop approaches. However, in the far (or perhaps not-so-far) future, when we start to use these systems in a fully automated way, adversarial examples may pose a significant threat to the widespread adoption of such systems. It can be thought that the phenomenon of adversarial examples is only a threat to neural networks, but other machine learning methods that do not involve gradient-based training have also been found to suffer from similar shortcomings [15, 229]. As such, even if we replace deep neural networks with other types of decision-making systems, it is very

likely that those systems will have similar vulnerabilities. As a result, I believe that the field of trustworthy AI will witness a significant rise in opportunities, both from an academic and an industry perspective.

Explanation for the existence of adversarial examples – In line with the overall theme of this dissertation, the final and most important topic I expect to see major future research and development on is achieving a better understanding of adversarial examples. As explained in Chapter 3.4, the non-robust feature hypothesis is currently the most popular hypothesis for explaining the existence of adversarial examples. But, as of now, there is no consensus on a conclusive hypothesis that explains their existence. In the upcoming years, I have no doubt that many more hypotheses will be put forward for explaining the existence of adversarial examples. However, I doubt unanimous agreement on any hypothesis will be reached as long as the proposed explanations remain mostly experimental and not theoretical. With that said, I must clarify that previous work on this topic does not deliberately rule out theoretical explanations. Rather, when it comes to neural networks and non-convex optimization, establishing conclusive theoretical work remains highly challenging.

7.4 Epilogue

All in all, the topic of adversarial examples, as well as the vulnerability of neural networks to such data points, remains a controversial topic, characterized by incompatible research findings [62, 89, 184], claims that are demonstrated to be incorrect [206], and findings that cannot be generalized, only working for a particular problem [23]. In this dissertation, I detailed my work on the adversarial vulnerability of DNNs, also sharing my experience with doing doctoral research in this field. I sincerely hope that my contributions will be useful in enabling the development of safer neural networks. That being said, having read numerous research articles written by brilliant researchers, having trained a large number of neural network models myself, having produced countless adversarial examples to abuse those models, and having spent four intense years in the field, I am concluding this dissertation on a rather pessimistic note: having done what I have done and knowing what I know about adversarial examples, I have absolutely no trust in automated decision-making systems that employ neural networks.

Bibliography

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL <https://www.tensorflow.org/>.
- [2] Saylor Academy. Seeing. https://saylordotorg.github.io/text_introduction-to-psychology/s08-02-seeing.html, 2021.
- [3] Evan Ackerman. Lidar That Will Make Self-driving Cars Affordable. *IEEE Spectrum*, 2016.
- [4] Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On The Robustness Of Semantic Segmentation Models To Adversarial Attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [5] Karoline Aslaksen and Håvard Lorås. The Modality-specific Learning Style Hypothesis: A Mini-review. *Frontiers in Psychology*, 2018.

- [6] Anish Athalye and Nicholas Carlini. On the Robustness of the CVPR 2018 White-box Adversarial Example Defenses. *CoRR*, abs/1804.03286, 2018.
- [7] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give A False Sense Of Security: Circumventing Defenses To Adversarial Examples. *International Conference on Machine Learning*, 2018.
- [8] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A Deep Convolutional Encoder-Decoder Architecture For Image Segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2017.
- [9] Shumeet Baluja and Ian Fischer. Adversarial Transformation Networks: Learning to Generate Adversarial Examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.
- [10] Nathan Benaich and Ian Hogarth. State of AI. https://docs.google.com/presentation/d/1ZUimafgXCBSLsgbacd6-a-dq07yLyzIl1ZJbiCBUUT4/edit#slide=id.g557254d430_0_0, 2014.
- [11] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-wise Training of Deep Networks. In *Advances in Neural Information Processing Systems*, 2007.
- [12] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality Reduction As A Defense Against Evasion Attacks On Machine Learning Classifiers. *CoRR*, abs/1704.02654, 2017.
- [13] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Lower Bounds on Adversarial Robustness from Optimal Transport. In *Advances in Neural Information Processing Systems*, 2019.
- [14] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning Attacks Against Support Vector Machines. *International Conference on Machine Learning*, 2012.
- [15] Battista Biggio, Iginio Corona, Davide Maiorca, Blaine Nelson, Nedim Šrđić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks Against Machine Learning At Test Time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

- [16] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [17] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, Reliable and Fast Robustness Evaluation. In *Advances in Neural Information Processing Systems*, 2019.
- [18] John S Bridle. Probabilistic Interpretation Of Feedforward Classification Network Outputs, With Relationships To Statistical Pattern Recognition. In *Neurocomputing*. Springer, 1990.
- [19] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial Patch. *CoRR*, abs/1712.09665, 2017.
- [20] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial Sensor Attack on Lidar-based Perception in Autonomous Driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019.
- [21] Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. Adversarial Objects Against Lidar-based Autonomous Driving Systems. *arXiv preprint arXiv:1907.05418*, 2019.
- [22] Nicholas Carlini and David A. Wagner. Towards Evaluating The Robustness of Neural Networks. *2017 IEEE Symposium on Security and Privacy*, 2017.
- [23] Nicholas Carlini and David A. Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.
- [24] A. Chaturvedi and U. Garain. Mimic and Fool: A Task-Agnostic Adversarial Attack. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [25] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

- [26] Victor C. Chen, David Tahmoush, and William J. Miceli. *Radar Micro-Doppler Signatures: Processing And Applications*. Radar, Sonar & Navigation. Institution of Engineering and Technology, 2014.
- [27] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving Black-box Adversarial Attacks with a Transfer-based Prior. In *Advances in Neural Information Processing Systems*, 2019.
- [28] Alesia Chernikova, Alina Oprea, Cristina Nita-Rotaru, and BaekGyu Kim. Are Self-Driving Cars Secure? Evasion Attacks Against Deep Neural Networks For Steering Angle Prediction. *IEEE Security and Privacy Workshops*, 2019.
- [29] Seung Ju Cho, Tae Joon Jun, Byungsoo Oh, and Daeyoung Kim. Dapas: Denoising Autoencoder to Prevent Adversarial Attack in Semantic Segmentation. *CoRR*, abs/1908.05195, 2019.
- [30] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast And Accurate Deep Network Learning By Exponential Linear Units (Elus). *CoRR*, abs/1511.07289, 2015.
- [31] Shai Cohen, Tomer Gluck, Yuval Elovici, and Asaf Shabtai. Security Analysis of Radar Systems. In *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, 2019.
- [32] Open Science Collaboration et al. Estimating the Reproducibility of Psychological Science. *Science*, 2015.
- [33] Corinna Cortes and Vladimir Vapnik. Support-vector Networks. *Machine Learning*, 1995.
- [34] Daniel Crevier. *AI: The Tumultuous History of the Search for Artificial Intelligence*. Basic Books, Inc., 1993.
- [35] Francesco Croce and Matthias Hein. Sparse and Imperceivable Adversarial Attacks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [36] Francesco Croce and Matthias Hein. Provable Robustness Against All Adversarial l_p -perturbations for $p \geq 1$. In *International Conference on Learning Representations*, 2020.

- [37] Noopur Davis. Secure Software Development Life Cycle Processes: A Technology Scouting Report. Technical report, Carnegie-Mellon University, 2005.
- [38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [39] Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking Adversarial Robustness on Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [40] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.
- [41] Timothy Dozat. Incorporating Nesterov Momentum into ADAM. *Workshop Track, International Conference on Learning Representations*, 2016.
- [42] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating Second-Order Functional Knowledge For Better Option Pricing. In *Advances in Neural Information Processing Systems*, 2001.
- [43] Jeffrey L Elman. Finding Structure in Time. *Cognitive science*, 1990.
- [44] Logan Engstrom, Justin Gilmer, Gabriel Goh, Dan Hendrycks, Andrew Ilyas, Aleksander Madry, Reiichiro Nakano, Preetum Nakkiran, Shibani Santurkar, Brandon Tran, Dimitris Tsipras, and Eric Wallace. A discussion of 'adversarial examples are not bugs, they are features'. *Distill*, 2019. doi: 10.23915/distill.00019. <https://distill.pub/2019/advex-bugs-discussion>.
- [45] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 2015.

- [46] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of Classifiers: from Adversarial to Random Noise. *CoRR*, abs/1608.08967, 2016.
- [47] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting Adversarial Samples From Artifacts. *CoRR*, abs/1703.00410, 2017.
- [48] Chris Finlay, Aram-Alexandre Pooladian, and Adam Oberman. The LogBarrier Adversarial Attack: Making Effective use of Decision Boundary Information. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [49] Samuel G Finlayson, Isaac S Kohane, and Andrew L Beam. Adversarial Attacks Against Medical Deep Learning Systems. *Science*, 2019.
- [50] Ronald Aylmer Fisher. Statistical Methods for Research Workers. In *Breakthroughs in statistics*, 1992.
- [51] Richard SJ Frackowiak. *Human Brain Function*. Elsevier, 2004.
- [52] Michael Frigge, David C. Hoaglin, and Boris Iglewicz. Some Implementations Of The Boxplot. *The American Statistician*, 1989.
- [53] Bolin Gao and Lacra Pavel. On The Properties Of The Softmax Function With Application In Game Theory And Reinforcement Learning. *CoRR*, abs/1704.00805, 2017.
- [54] János Geier, László Bernáth, Mariann Hudák, and László Séra. Straightness as the Main Factor of the Hermann Grid Illusion. *Perception*, 2008.
- [55] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation Of Neural Networks Is Fragile. *CoRR*, abs/1710.10547, 2017.
- [56] Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating The Rules Of The Game For Adversarial Example Research. *CoRR*, abs/1807.06732, 2018.
- [57] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial Spheres. *CoRR*, abs/1801.02774, 2018.

- [58] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 2011.
- [59] Frederic Godin, Jonas Degraeve, Joni Dambre, and Wesley De Neve. Dual Rectified Linear Units (DReLU): A Replacement for Tanh Activation Functions in Quasi-recurrent Neural Networks. *Pattern Recognition Letters*, 2018.
- [60] Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal Neurons in Artificial Neural Networks. *Distill*, 6, 2021.
- [61] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial And Clean Data Are Not Twins. *CoRR*, abs/1704.04960, 2017.
- [62] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *International Conference on Learning Representations*, 2015.
- [63] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [64] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. Making Machine Learning Robust Against Adversarial Inputs. *Communications of the ACM*, 2018.
- [65] Diego Gragnaniello, Francesco Marra, Giovanni Poggi, and Luisa Verdoliva. Perceptual Quality-preserving Black-Box Attack Against Deep Learning Image Classifiers. *CoRR*, abs/1902.07776, 2019.
- [66] Vicente Grau, AUJ Mewes, M Alcaniz, Ron Kikinis, and Simon K Warfield. Improved Watershed Transform For Medical Image Segmentation Using Prior Information. *IEEE Transactions on Medical Imaging*, 2004.
- [67] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On The (Statistical) Detection Of Adversarial Examples. *CoRR*, abs/1702.06280, 2017.
- [68] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On Calibration of Modern Neural Networks. In *International Conference on Machine Learning*, 2017.

- [69] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering Adversarial Images Using Input Transformations. *CoRR*, abs/1711.00117, 2017.
- [70] Chuan Guo, Jacob R Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q Weinberger. Simple Black-box Adversarial Attacks. *International Conference on Machine Learning*, 2019.
- [71] Sanchit Gupta. Convert .csv file to Images. <https://medium.com/lifeandtech/convert-csv-file-to-images-309b6fdb8c49>, 2020.
- [72] Frank R. Hampel. The Breakdown Points Of The Mean Combined With Some Rejection Rules. *Technometrics*, 1985.
- [73] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [74] Zhongyuan Hau, Kenneth T Co, Soteris Demetriou, and Emil C Lupu. Object Removal Attacks on Lidar-based 3D Object Detectors. *arXiv preprint arXiv:2102.03722*, 2021.
- [75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning For Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [76] Will Douglas Heaven. AI is Wrestling with a Replication Crisis. <https://www.technologyreview.com/>, 2014.
- [77] Jeff Hecht. Lidar for Self-driving Cars. *Optics and Photonics News*, 2018.
- [78] Tobias Heimann and Hans-Peter Meinzer. Statistical Shape Models For 3D Medical Image Segmentation: A Review. *Medical Image Analysis*, 2009.
- [79] Dan Hendrycks and Kevin Gimpel. Early Methods For Detecting Adversarial Images. *CoRR*, abs/1608.00530, 2016.
- [80] Ludimar Hermann. Eine Erscheinung Simultanen Contrastes. *Archiv für die gesamte Physiologie des Menschen und der Tiere*, 1870.

- [81] Leonard R Herrmann. Laplacian-Isoparametric Grid Generation Scheme. *Journal of the Engineering Mechanics Division*, 1976.
- [82] Lyndon Hill. Urban Legends in Computer Vision. <http://www.lyndonhill.com/opinion-cvlegends.html>, 2021.
- [83] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.
- [84] Hossein Hosseini, Sreeram Kannan, and Radha Poovendran. Are Odds Really Odd? Bypassing Statistical Detection Of Adversarial Examples. *CoRR*, abs/1907.12138, 2019.
- [85] Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q Weinberger. A New Defense Against Adversarial Images: Turning a Weakness Into a Strength. In *Advances in Neural Information Processing Systems*, 2019.
- [86] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [87] David H Hubel. *Eye, Brain, and Vision*. Scientific American Library/Scientific American Books, 1995.
- [88] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box Adversarial Attacks with Limited Queries and Information. *International Conference on Machine Learning*, 2018.
- [89] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial Examples Are Not Bugs, They Are Features. In *Advances in Neural Information Processing Systems*, 2019.
- [90] Clean PNG Images. <https://www.cleanpng.com>, 2021.
- [91] John PA Ioannidis. Contradicted and Initially Stronger Effects in Highly Cited Clinical Research. *The Journal of the American Medical Association*, 2005.
- [92] John PA Ioannidis. Why Most Published Research Findings are False. *PLoS Medicine*, 2005.

- [93] Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive Invariance Causes Adversarial Vulnerability. In *International Conference on Learning Representations*, 2019.
- [94] Azarakhsh Jalalvand, Baptist Vandersmissen, Wesley De Neve, and Erik Mannens. Radar Signal Processing For Human Identification By Means Of Reservoir Computing Networks. In *IEEE Radar Conference*, 2019.
- [95] Branka Jokanovic, Moeness Amin, and Fauzia Ahmad. Radar Fall Motion Detection Using Deep Learning. In *2016 IEEE radar conference (RadarConf)*. IEEE, 2016.
- [96] Eric Jones, Travis Oliphant, and Pearu Peterson. Scipy: Open Source Scientific Tools For Python, 2001.
- [97] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial Logit Pairing. *CoRR*, abs/1803.06373, 2018.
- [98] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and Visible Adversarial Noise. *International Conference on Machine Learning*, 2018.
- [99] Jack Kiefer and Jacob Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 1952.
- [100] Yannic Kilcher. Apple or iPod??? Easy Fix for Adversarial Textual Attacks on OpenAI’s CLIP Model. <https://www.youtube.com/watch?v=Rk3MBx20z24>, 2021.
- [101] Yannic Kilcher. The Dimpled Manifold Model of Adversarial Examples in Machine Learning (Research Paper Explained). https://www.youtube.com/watch?v=k_hUdZJNzkU, 2021.
- [102] Youngwook Kim and Taesup Moon. Human Detection and Activity Classification Based on Micro-Doppler Signatures Using Deep Convolutional Neural Networks. *IEEE Geoscience and Remote Sensing Letters*, 2015.
- [103] Diederik P. Kingma and Jimmy Ba. Adam: A Method For Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

- [104] Akiyoshi Kitaoka and Hiroshi Ashida. Phenomenal Characteristics of the Peripheral Drift Illusion. *Vision*, 2003.
- [105] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing Neural Networks. In *Advances in Neural Information Processing Systems*, 2017.
- [106] Rony Komissarov and Avishai Wool. Spoofing Attacks Against Vehicular FMCW Radar. *arXiv preprint arXiv:2104.13318*, 2021.
- [107] Jinkyu Koo, Michael Roth, and Saurabh Bagchi. Hawkeye: Adversarial Example Detector For Deep Neural Networks. *CoRR*, abs/1909.09938, 2019.
- [108] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers Of Features From Tiny Images. Technical report, Citeseer, 2009.
- [109] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 2012.
- [110] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Examples In The Physical World. *Workshop Track, International Conference on Learning Representations*, 2016.
- [111] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Machine Learning At Scale. *International Conference on Learning Representations*, 2017.
- [112] Lecturio. The Visual Pathway and Related Disorders. <https://www.lecturio.com/concepts/the-visual-pathway-and-related-disorders/>, 2021.
- [113] 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*, 1989.
- [114] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied To Document Recognition. *Proceedings of the IEEE*, 1998.
- [115] Eden Levy, Yael Mathov, Ziv Katzir, Asaf Shabtai, and Yuval Elovici. Not all datasets are born equal: On heterogeneous data and adversarial examples. *CoRR*, abs/2010.03180, 2020.

- [116] Xin Li and Fuxin Li. Adversarial Examples Detection In Deep Networks With Convolutional Filter Statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [117] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense Against Adversarial Attacks Using High-level Representation Guided Denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [118] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [119] Zachary C Lipton. The Mythos Of Model Interpretability. *CoRR*, abs/1606.03490, 2016.
- [120] Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. Feature Distillation: Dnn-Oriented Jpeg Compression Against Adversarial Examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2019.
- [121] Tania Lombrozo. The Rotating Snakes Are All In Your Mind. <https://www.npr.org/sections/13.7/2014/03/24/293740555/the-rotating-snakes-are-all-in-your-mind>, 2014.
- [122] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks For Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [123] David G Lowe. Distinctive Image Features from Scale-invariant Keypoints. *International journal of computer vision*, 2004.
- [124] Aurélien Lucchi, Pablo Márquez-Neila, Carlos Becker, Yunpeng Li, Kevin Smith, Graham Knott, and Pascal Fua. Learning Structured Models for Segmentation of 2-D and 3-D Imagery. *IEEE Transactions on Medical Imaging*, 2014.
- [125] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *International Conference on Machine Learning*, 2013.

- [126] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant To Adversarial Attacks. *International Conference on Learning Representations*, 2018.
- [127] Warren S McCulloch and Walter Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*, 1943.
- [128] Scott McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg Corrado, Ara Darzi, Mozziyar Etemadi, Florencia Garcia-Vicente, Fiona Gilbert, Mark Halling-Brown, Demis Hassabis, Sunny Jansen, Alan Karthikesalingam, Christopher Kelly, Dominic King, and Shravya Shetty. International Evaluation of an AI System for Breast Cancer Screening. *Nature*, 2020.
- [129] Marco Melis, Ambra Demontis, Battista Biggio, Gavin Brown, Giorgio Fumera, and Fabio Roli. Is Deep Learning Safe For Robot Vision? Adversarial Examples Against The Icub Humanoid. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [130] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations. *CoRR*, abs/1702.04267, 2017.
- [131] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal Adversarial Perturbations Against Semantic Image Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [132] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks For Volumetric Medical Image Segmentation. In *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016.
- [133] Marvin Minsky and Seymour A Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 2017.
- [134] Noriyuki Miura, Tatsuya Machida, Kohei Matsuda, Makoto Nagata, Shoei Nashimoto, and Daisuke Suzuki. A low-cost Replica-based Distance-spoofing Attack on mmWave FMCW Radar. In

- Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, 2019.
- [135] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A Simple And Accurate Method To Fool Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
 - [136] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal Adversarial Perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
 - [137] Hans P. Moravec. Rover Visual Obstacle Avoidance. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, 1981.
 - [138] Mehran Mozaffari-Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K Jha. Systematic Poisoning Attacks On And Defenses For Machine Learning In Healthcare. *IEEE journal of biomedical and health informatics*, 2014.
 - [139] FC Muller-Lyer. Optische Urteilstauschungen. *Archiv fur Anatomie und Physiologie, Physiologische Abteilung*, 1889.
 - [140] Nitin Nair, Chinchu Thomas, and Dinesh Babu Jayagopi. Human Activity Recognition Using Temporal Convolutional Network. In *Proceedings of the 5th international Workshop on Sensor-based Activity Recognition and Interaction*, 2018.
 - [141] Preetum Nakkiran. A Discussion of ‘Adversarial Examples Are Not Bugs, They Are Features’: Adversarial Examples are Just Bugs, Too. *Distill*, 4(8):e00019–5, 2019.
 - [142] Shoei Nashimoto, Daisuke Suzuki, Noriyuki Miura, Tatsuya Machida, Kohei Matsuda, and Makoto Nagata. Low-cost Distance-spoofing Attack on FMCW Radar and its Feasibility Study on Countermeasure. *Journal of Cryptographic Engineering*, 2021.
 - [143] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep Neural Networks Are Easily Fooled: High Confidence Predictions For Unrecognizable Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

- [144] Jorge Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151):773–782, 1980.
- [145] Jim Oeppen and James W Vaupel. Broken Limits to Life Expectancy, 2002.
- [146] David Ouyang, Bryan He, Amirata Ghorbani, Curt Langlotz, Paul A Heidenreich, Robert A Harrington, David H Liang, Euan A Ashley, and James Y Zou. Interpretable Ai For Beat-To-Beat Cardiac Function Assessment. *medRxiv*, 2019.
- [147] Utku Ozbulak, Wesley De Neve, and Arnout Van Messem. How The Softmax Output Is Misleading For Evaluating The Strength Of Adversarial Examples. *Workshop on Security in Machine Learning, Advances in Neural Information Processing Systems*, 2018.
- [148] Utku Ozbulak, Arnout Van Messem, and Wesley De Neve. Impact Of Adversarial Examples On Deep Learning Models For Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019.
- [149] Utku Ozbulak, Arnout Van Messem, and Wesley De Neve. Not All Adversarial Examples Require A Complex Defense: Identifying Over-Optimized Adversarial Examples With Iqr-Based Logit Thresholding. In *International Joint Conference on Neural Networks (IJCNN)*, 2019.
- [150] Utku Ozbulak, Manvel Gasparyan, Wesley De Neve, and Arnout Van Messem. Perturbation Analysis of Gradient-based Adversarial Attacks. *Pattern Recognition Letters*, 2020.
- [151] Utku Ozbulak, Jonathan Peck, Wesley De Neve, Bart Goossens, Yvan Saeys, and Arnout Van Messem. Regional Image Perturbation Reduces L_p Norms of Adversarial Examples While Maintaining Model-to-model Transferability. *Workshop on Uncertainty and Robustness in Deep Learning, International Conference on Machine Learning*, 2020.
- [152] Utku Ozbulak, Esra Timothy Anzaku, Wesley De Neve, and Arnout Van Messem. Selection of Source Images Heavily Influences the Effectiveness of Adversarial Attacks. *CoRR*, abs/2106.07141, 2021.

- [153] Utku Ozbulak, Maura Pintor, Arnout Van Messem, and Wesley De Neve. Evaluating adversarial attacks on imagenet: A reality check on misclassification classes. *arXiv preprint arXiv:2111.11056*, 2021.
- [154] Utku Ozbulak, Baptist Vandersmissen, Azarakhsh Jalalvand, Ivo Couckuyt, Arnout Van Messem, and Wesley De Neve. Investigating the significance of adversarial attacks and their relation to interpretability for radar-based human activity recognition systems. *Computer Vision and Image Understanding*, 2021.
- [155] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep Learning vs. Traditional Computer Vision. In *Science and Information Conference*. Springer, 2019.
- [156] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations Of Deep Learning In Adversarial Settings. *CoRR*, abs/1511.07528, 2015.
- [157] Nicolas Papernot, Patrick D. McDaniel, and Ian Goodfellow. Transferability In Machine Learning: From Phenomena To Black-Box Attacks Using Adversarial Samples. *CoRR*, abs/1605.07277, 2016.
- [158] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation As A Defense To Adversarial Perturbations Against Deep Neural Networks. *IEEE Symposium on Security and Privacy*, 2016.
- [159] Harold Pashler, Mark McDaniel, Doug Rohrer, and Robert Bjork. Learning Styles: Concepts and Evidence. *Psychological science in the public interest*, 2008.
- [160] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch, 2017. URL <https://pytorch.org/>.
- [161] Jonathan Peck, Joris Roels, Bart Goossens, and Yvan Saeys. Lower Bounds on the Robustness to Adversarial Perturbations. In *Advances in Neural Information Processing Systems*, 2017.

- [162] Jonathan Peck, Bart Goossens, and Yvan Saeys. Detecting Adversarial Examples with Inductive Venn-ABERS Predictors. In *27th European symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2019)*, 2019.
- [163] Cristina Pena-Betancor, Marta Gonzalez-Hernandez, Francisco Fumero-Batista, Jose Sigut, Erica Medina-Mesa, Silvia Alayon, and Manuel Gonzalez de la Rosa. Estimation Of The Relative Amount Of Hemoglobin In The Cup And Neuroretinal Rim Using Stereoscopic Color Fundus Images. *Investigative Ophthalmology & Visual Science*, 2015.
- [164] Dzung L Pham, Chenyang Xu, and Jerry L Prince. Current Methods In Medical Image Segmentation. *Annual review of biomedical engineering*, 2000.
- [165] George Philipp, Dawn Song, and Jaime G Carbonell. The Exploding Gradient Problem Demystified-definition, Prevalence, Impact, Origin, Tradeoffs, and Solutions. *CoRR*, abs/1712.05577, 2017.
- [166] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 1999.
- [167] Qasim Mahmood Rajpoot and Christian Damsgaard Jensen. Video Surveillance: Privacy Issues And Legal Compliance. In *Promoting Social Change and Democracy through Information Technology*. IGI global, 2015.
- [168] Valerie L Reese and Rita Dunn. Learning-style Preferences of a Diverse Freshmen Population in a Large, Private, Metropolitan University by Gender and GPA. *Journal of College Student Retention: Research, Theory & Practice*, 2007.
- [169] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 1951.
- [170] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks For Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention*, 2015.

- [171] Frank Rosenblatt. The perceptron: A Probabilistic Model for Information Storage and Organization In the Brain. *Psychological Review*, 1958.
- [172] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The Odds Are Odd: A Statistical Test For Detecting Adversarial Examples. In *International Conference on Machine Learning*, 2019.
- [173] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear Total Variation Based Noise Removal Algorithms. *Physica D: nonlinear phenomena*, 1992.
- [174] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning Internal Representations by Error Propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [175] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.
- [176] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial Manipulation Of Deep Representations. In *International Conference on Learning Representations*, 2016.
- [177] E. Santos, L. Santos, R. Veras, M. Frazao, and D. Leite. A Semiautomatic Superpixel Based Approach to Cup-to-Disc Ratio Measurement. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018.
- [178] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *International Conference on Artificial Neural Networks*, 2010.
- [179] Cordelia Schmid and Roger Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997.
- [180] Jürgen Schmidhuber. Who Invented Backpropagation? <https://people.idsia.ch/~juergen/who-invented-backpropagation.html>, 2020.

- [181] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-Cam: Why Did You Say That? Visual Explanations From Deep Networks Via Gradient-Based Localization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [182] Mehmet Saygın Seyfioğlu, Ahmet Murat Özbayoğlu, and Sevgi Zubeyde Gürbüz. Deep Convolutional Autoencoder for Radar-Based Classification of Similar Aided and Unaided Human Activities. *IEEE Transactions on Aerospace and Electronic Systems*, 2018.
- [183] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are Adversarial Examples Inevitable? In *International Conference on Learning Representations*, 2019.
- [184] Adi Shamir, Odelia Melamed, and Oriel BenShmuel. The Dimpled Manifold Model of Adversarial Examples in Machine Learning. *CoRR*, abs/2106.10151, 2021.
- [185] Ali Shahin Shamsabadi, Ricardo Sanchez-Matilla, and Andrea Cavallaro. Colorfool: Semantic Adversarial Colorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [186] Mahmood Sharif, Lujo Bauer, and Michael K Reiter. On the Suitability of lp-norms for Creating and Preventing Adversarial Examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.
- [187] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. *CoRR*, abs/1704.02685, 2017.
- [188] Carl-Johann Simon-Gabriel, Yann Ollivier, Bernhard Schölkopf, Léon Bottou, and David Lopez-Paz. Adversarial Vulnerability Of Neural Networks Increases With Input Dimension. *CoRR*, abs/1802.01421, 2018.
- [189] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks For Large-Scale Image Recognition. *International Conference on Learning Representations*, 2015.

- [190] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models And Saliency Maps. In *Workshop, Proceedings of 2th International Conference on Learning Representations (ICLR)*, 2014.
- [191] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian Surface Editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2004.
- [192] Angelo Sotgiu, Ambra Demontis, Marco Melis, Battista Biggio, Giorgio Fumera, Xiaoyi Feng, and Fabio Roli. Deep Neural Rejection Against Adversarial Examples. *CoRR*, abs/1910.00470, 2019.
- [193] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving For Simplicity: The All Convolutional Net. *CoRR*, abs/1412.6806, 2014.
- [194] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 2014.
- [195] Pete Staples. Thinking About Buying A Smart Home Device? Heres What You Need To Know About Security. <https://www.forbes.com>, 2019. Accessed: 2019-07-26.
- [196] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is Robustness the Cost of Accuracy?—A Comprehensive Study on the Robustness of 18 Deep Image Classification Models. In *Proceedings of the European Conference on Computer Vision*, 2018.
- [197] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One Pixel Attack For Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation*, 2019.
- [198] 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. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

- [199] Lu Sun, Mingtian Tan, and Zhe Zhou. A Survey of Practical Adversarial Example Attacks. *Cybersecurity*, 2018.
- [200] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Representation from Predicting 10,000 Classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
- [201] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties Of Neural Networks. *International Conference on Learning Representations*, 2014.
- [202] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [203] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking The Inception Architecture For Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [204] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-Rmsprop: Divide The Gradient By A Running Average Of Its Recent Magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.
- [205] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *CoRR*, abs/2105.01601, 2021.
- [206] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On Adaptive Attacks to Adversarial Example Defenses. *Advances in Neural Information Processing Systems*, 2020.
- [207] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based Zeroth Order Optimization Method for Attacking Black-Box Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

- [208] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically Realizable Adversarial Examples for Lidar Object Detection. In *Conference on Computer Vision and Pattern Recognition*, 2020.
- [209] John W Tukey. *Exploratory Data Analysis*, volume 2. Reading, Mass., 1977.
- [210] Baptist Vandersmissen, Nicolas Knudde, Azarakhsh Jalalvand, Ivo Couckuyt, André Bourdoux, Wesley De Neve, and Tom Dhaene. Indoor Person Identification Using A Low-Power Fmcw Radar. *IEEE Transactions on Geoscience and Remote Sensing*, 2018.
- [211] Baptist Vandersmissen, Nicolas Knudde, Azarakhsh Jalalvand, Ivo Couckuyt, Tom Dhaene, and Wesley De Neve. Indoor Human Activity Recognition Using High-Dimensional Sensors And Deep Neural Networks. *Neural Computing and Applications*, 2019.
- [212] Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, Łukasz and Polosukhin, Illia. Attention is All you Need. In *Advances in Neural Information Processing Systems*, 2017.
- [213] Hongjun Wang, Guangrun Wang, Ya Li, Dongyu Zhang, and Liang Lin. Transferable, Controllable, and Inconspicuous Adversarial Attacks on Person Re-identification With Deep Mis-Ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [214] Ke Wang, Guangyu Wang, Ning Chen, and Ting Chen. How Robust Is Your Automatic Diagnosis Model? In *IEEE International Conference on Bioinformatics and Biomedicine*, 2019.
- [215] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. Interacting With Soli: Exploring Fine-Grained Dynamic Gesture Recognition In The Radio-Frequency Spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 2016.
- [216] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. *CoRR*, abs/1705.08292, 2017.

- [217] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip Connections Matter: On the Transferability of Adversarial Examples Generated with ResNets. In *International Conference on Learning Representations*, 2020.
- [218] Nan Wu, Jason Phang, Jungkyu Park, Yiqiu Shen, Zhe Huang, Masha Zorin, Stanislaw Jastrzebski, Thibault Fevry, Joe Kat-snelson, Eric Kim, Stacey Wolfson, Ujas Parikh, Sushma Gad-dam, Leng Lin, Kara Ho, Joshua Weinstein, Beatriu Reig, Yim-ing Gao, Hildegard Pysarenko, and Krzysztof Geras. Deep Neu-ral Networks Improve Radiologists' Performance in Breast Can-cer Screening. *IEEE Transactions on Medical Imaging*, 2019.
- [219] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial Examples For Semantic Seg-mentation And Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [220] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating Adversarial Effects Through Randomization. *International Conference on Learning Representations*, 2018.
- [221] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial Examples Improve Image Recognition. In *Proceedings of the IEEE Conference on Com-puter Vision and Pattern Recognition*, 2020.
- [222] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [223] Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Quanfu Fan, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. Structured Adversarial Attack: Towards General Implementa-tion and Better Interpretability. *International Conference on Learning Representations*, 2019.
- [224] Ziang Yan, Yiwen Guo, and Changshui Zhang. Deep Defense: Training DNNs with Improved Adversarial Robustness. In *Ad-vances in Neural Information Processing Systems*, 2018.
- [225] Li Yao and Ying Qian. DT-3DResNet-LSTM: An Architecture for Temporal Activity Recognition in Videos. In *Pacific Rim Conference on Multimedia*. Springer, 2018.

- [226] Michał Zajac, Konrad Zolna, Negar Rostamzadeh, and Pedro O Pinheiro. Adversarial Framing for Image and Video Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.
- [227] Matthew D Zeiler and Rob Fergus. Visualizing And Understanding Convolutional Networks. In *Proceedings of the IEEE European Conference on Computer Vision*. Springer, 2014.
- [228] Semir Zeki. *A Vision of the Brain*. Blackwell scientific publications, 1993.
- [229] Chong Zhang, Huan Zhang, and Cho-Jui Hsieh. An Efficient Adversarial Attack for Tree Ensembles. *CoRR*, abs/2010.11598, 2020.
- [230] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. A Comprehensive Survey Of Vision-Based Human Action Recognition Methods. *Sensors*, 2019.
- [231] Yuchen Zhang and Percy Liang. Defending against Whitebox Adversarial Attacks via Randomized Discretization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019.
- [232] Mingmin Zhao, Tianhong Li, Mohammad Abu Alsheikh, Yonglong Tian, Hang Zhao, Antonio Torralba, and Dina Katabi. Through-Wall Human Pose Estimation Using Radio Signals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [233] Yu Zhao, Rennong Yang, Guillaume Chevalier, Ximeng Xu, and Zhenxing Zhang. Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors. *Mathematical Problems in Engineering*, 2018, 2018.
- [234] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. Towards Large yet Imperceptible Adversarial Image Perturbations with Perceptual Color Distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [235] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features For Discriminative

- Localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [236] Aniketh Reddy Zihao Ding and Aparna Joshi. Reproducibility in Machine Learning. <https://blog.ml.cmu.edu/2020/08/31/5-reproducibility/>, 2020.
- [237] Friedrich Zöllner. Ueber eine neue Art von Pseudoskopie und ihre Beziehungen zu den von Plateau und Oppel beschriebenen Bewegungsphänomenen. *Annalen der Physik*, 1860.
- [238] Jasper Zuallaert, Frédéric Godin, Mijung Kim, Arne Soete, Yvan Saeys, and Wesley De Neve. SpliceRover: Interpretable Convolutional Neural Networks for Improved Splice Site Prediction. *Bioinformatics*, 2018.



A minor perturbation can easily trick machine learning models the same way a simple set of lines can easily trick the human visual system.