



# Understanding, taming, and defending from adversarial examples

Benoît Bonnet

## ► To cite this version:

Benoît Bonnet. Understanding, taming, and defending from adversarial examples. Artificial Intelligence [cs.AI]. Université de Rennes, 2023. English. NNT : 2023URENS025 . tel-04223126

**HAL Id: tel-04223126**

**<https://theses.hal.science/tel-04223126>**

Submitted on 29 Sep 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601

*Mathématiques, Télécommunications, Informatique, Signal, Systèmes,  
Électronique*

Spécialité : *Signal, Image, Vision*

Par

**Benoît BONNET**

## Understanding, Taming, and Defending From Adversarial Examples

Thèse présentée et soutenue à Rennes, le 06/02/2023

Unité de recherche : Inria-Rennes

### Rapporteurs avant soutenance :

Christophe ROSENBERGER	Professeur, Université de Caen
David PICARD	Professeur, École des Ponts ParisTech

### Composition du Jury :

Président :	Gildas AVOINE	Professeur, Institut National des Sciences Appliquées Rennes
Examineurs :	Pascal FROSSARD	Professeur, École Polytechnique Fédérale de Lausanne
	Cecilia PASQUINI	Professeur, Fondazione Bruno Kessler
Co-dir. de thèse :	Teddy FURON	Directeur de recherche, Inria-Rennes
Co-dir. de thèse :	Patrick BAS	Directeur de recherche, CNRS-Lille



# ACKNOWLEDGEMENT

---

Je tiens à remercier de nombreuses personnes qui ont toutes eu une contribution dans l'achèvement de cette thèse.

Premièrement, je tiens à remercier mes deux directeurs de thèse Teddy Furon et Patrick Bas. Je suis extrêmement reconnaissant de l'encadrement que j'ai eu au cours de cette thèse. Au début de ma thèse, ils ont su me mettre sur les rails immédiatement pour commencer à expérimenter et publier rapidement. De plus, la crise sanitaire ayant commencé peu après le début de ma thèse, ils ont su m'assurer tout le suivi dont j'avais besoin pour continuer dans ma recherche malgré l'isolement. Tout le monde a appris à travailler de chez soi ces dernières années, je considère avoir eu beaucoup d'aide pour y parvenir. Je ne peux que souhaiter à tout doctorant et doctorante d'avoir un tel encadrement et une telle bienveillance de la part de ses directeurs de thèse.

Deuxièmement je tiens à remercier le centre IRISA et en particulier l'équipe Linkmedia de m'avoir accueilli. L'équipe a bien changé depuis mon arrivée: certains membres sont partis, d'autres sont arrivés et certains même sont partis puis revenus. Je voudrais remercier en particulier Hanwei Zhang qui m'a apporté beaucoup d'aide et m'a aidé à m'approprier les différents outils. Thibault Maho avec qui partager un bureau n'aura jamais été aussi plaisant, et qui aura été source de beaucoup de discussions sur nos recherches respectives. Guillaume Le Noé-Bienvenu avec qui j'ai pu passer de très bons moments en dehors du centre et qui a toujours su apporté beaucoup d'énergie à cette équipe. Et enfin Aurélie Patier avec qui tout est toujours facile même lorsqu'on est pas toujours des plus efficaces.

Enfin je tiens à remercier tout particulièrement Margot Pernet sans qui je n'aurais jamais pu aller au bout de cette thèse et à vrai dire sans qui je ne l'aurais même pas commencé. Elle a toujours été d'un soutien indéfectible et j'essaierai tous les jours de lui apporter le même soutien. Je remercie également toute ma famille et mes parents particulièrement qui se disent sans doute qu'après tout ça, il serait temps que leurs enfants arrêtent les études, mais qui seront toujours prêt à apporter le soutien nécessaire si cela devait encore durer. Et enfin, merci à mes deux compagnons de télétravail: Tiny et Gontrand qui auront su rendre la maison bien plus vivante pendant ces périodes d'isolement.





# TABLE OF CONTENTS

---

<b>1</b>	<b>Contributions</b>	<b>9</b>
<b>2</b>	<b>Résumé en Français</b>	<b>11</b>
2.1	Introduction . . . . .	11
2.2	Résumé des Chapitres . . . . .	13
2.3	Résumé des Contributions . . . . .	15
2.3.1	Vers une Evaluation Equitable . . . . .	16
2.3.2	Les Images Adverses . . . . .	17
2.3.3	Défenses et Perspectives . . . . .	18
2.3.4	Un Dernier Mot . . . . .	18
<b>3</b>	<b>Introduction</b>	<b>19</b>
<b>I</b>	<b>Introduction on Adversarial Samples</b>	<b>23</b>
<b>4</b>	<b>Classifier Model and Adversarial Examples</b>	<b>25</b>
4.1	Introduction . . . . .	26
4.2	Deep convolutional networks . . . . .	27
4.2.1	A Brief History . . . . .	27
4.2.2	Model and Data Notation . . . . .	32
4.2.3	Classifier Model . . . . .	33
4.2.4	Training a Classifier . . . . .	34
4.3	Adversarial Examples . . . . .	36
4.3.1	Adversarial Vulnerabilities of DNNs . . . . .	36
4.3.2	Fooling a classifier . . . . .	36
4.3.3	Usual White-box Attacks . . . . .	39
4.4	Chapter Conclusion . . . . .	41

<b>5</b>	<b>Adversariality: Analysis and Defenses</b>	<b>42</b>
5.1	Introduction . . . . .	43
5.2	Understanding Adversarial Examples . . . . .	43
5.2.1	Data Is The Enemy . . . . .	44
5.2.2	Learning the Hard Way . . . . .	50
5.2.3	Adversarial Examples Are Inevitable . . . . .	53
5.3	Defending Against Evasion . . . . .	57
5.3.1	Reactive Defenses . . . . .	57
5.3.2	Data Reforming . . . . .	57
5.3.3	Adversarial Detection . . . . .	59
5.3.4	Proactive Defenses . . . . .	60
5.4	Chapter Conclusion . . . . .	65
<b>II</b>	<b>Contributions</b>	<b>67</b>
<b>6</b>	<b>Benchmarking Models Against White-box and Black-box Attacks</b>	<b>69</b>
6.1	Introduction . . . . .	70
6.2	Difficulties . . . . .	70
6.2.1	Notation . . . . .	70
6.2.2	The best effort mode . . . . .	71
6.2.3	Worst case attacks . . . . .	71
6.2.4	Metrics . . . . .	71
6.3	The benchmark . . . . .	73
6.4	Fast Attacks . . . . .	74
6.4.1	Fast black-box attacks . . . . .	74
6.4.2	White-Box Attacks and “Best-effort Mode” . . . . .	74
6.4.3	Quantization . . . . .	76
6.5	Experiments . . . . .	76
6.5.1	Selecting the worst case attacks . . . . .	76
6.5.2	Comparison with other benchmarks . . . . .	77
6.5.3	Benchmarking models . . . . .	77
6.6	Chapter Conclusion . . . . .	79

<b>7</b>	<b>Generating Quantized Adversarial Images</b>	<b>81</b>
7.1	Introduction . . . . .	82
7.1.1	Contributions . . . . .	84
7.1.2	Outlines . . . . .	84
7.2	Smart Quantization in the Spatial Domain . . . . .	85
7.2.1	Problem Statement . . . . .	85
7.2.2	Solution . . . . .	87
7.3	Smart Quantization in the JPEG Domain . . . . .	89
7.3.1	JPEG Compression . . . . .	89
7.3.2	Quantization . . . . .	90
7.4	Experimental Work . . . . .	91
7.4.1	Implementation Details and Setup . . . . .	91
7.4.2	Investigations on the Search of $\lambda^*$ . . . . .	92
7.4.3	Impact of Quality Factor in JPEG Domain . . . . .	93
7.4.4	Impact of the Degree of Freedom . . . . .	96
7.4.5	Quantization on Different Classifiers and Attacks . . . . .	97
7.4.6	Transferability . . . . .	100
7.4.7	JPEG Compression as a Defense . . . . .	101
7.5	Chapter Conclusion . . . . .	102
<b>8</b>	<b>Quantization Using Steganographic Strategies</b>	<b>105</b>
8.1	Introduction . . . . .	106
8.2	Related Works . . . . .	107
8.2.1	Steganalysis for forensic purposes . . . . .	107
8.2.2	Adversarial examples . . . . .	108
8.2.3	Defenses . . . . .	109
8.2.4	Steganographic costs . . . . .	110
8.2.5	Looking at Adversarial Examples with Stega Glasses . . . . .	111
8.3	Steganographic Post-Processing . . . . .	112
8.3.1	Optimal post-processing . . . . .	112
8.3.2	Our proposal . . . . .	113
8.3.3	Simplification for quadratic stego-costs . . . . .	114
8.4	Experimental Investigation . . . . .	115
8.4.1	Experimental setup . . . . .	115

## TABLE OF CONTENTS

---

8.4.2	Robustness of recent classifiers: there is free lunch . . . . .	115
8.4.3	Detection with forensics detectors . . . . .	116
8.4.4	Post-processing with a Steganographic Embedder . . . . .	118
8.4.5	Training on adversarial images with GINA costs . . . . .	118
8.5	Chapter Conclusion . . . . .	120
<b>9</b>	<b>Scale and Rescaling as Means of Robustness</b>	<b>121</b>
9.1	Introduction . . . . .	122
9.2	Related Works . . . . .	123
9.3	Problem formulation . . . . .	124
9.3.1	Including the downscaler inside the network . . . . .	124
9.3.2	Attacking in the large or small image space . . . . .	125
9.4	Experimental Works . . . . .	125
9.4.1	Models, data, and attacks . . . . .	125
9.4.2	Scenario A: comparison with theoretical results . . . . .	127
9.4.3	Scenario B: attacking through downscaling . . . . .	128
9.4.4	Scenario B: transferability . . . . .	128
9.4.5	Scenario B: ensemble model . . . . .	129
9.5	Chapter Conclusion . . . . .	130
	<b>Conclusion</b>	<b>131</b>
	<b>Bibliography</b>	<b>135</b>
<b>III</b>	<b>Appendix A</b>	<b>157</b>
<b>IV</b>	<b>Appendix B</b>	<b>161</b>
9.6	Introduction and Related Works . . . . .	162
9.7	Problem Formulation . . . . .	162
9.7.1	Building Adversarial Examples . . . . .	164
9.7.2	Optimization . . . . .	164
9.8	Experimental Work . . . . .	165
9.8.1	Experimental Setup . . . . .	165
9.9	Discussion . . . . .	168

# CONTRIBUTIONS

---

BONNET, Benoît, FURON, Teddy, et BAS, Patrick. What if adversarial samples were digital images?. In : Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security. 2020. p. 55-66.

BONNET, Benoit, FURON, Teddy, et BAS, Patrick. Et si les images adverses étaient des images?. In : Actes de la conférence CAID 2020. 2020.

BONNET, Benoit, FURON, Teddy, et BAS, Patrick. Fooling an automatic image quality estimator. In : MediaEval 2020-MediaEval Benchmarking Initiative for Multimedia Evaluation. 2020. p. 1-4.

BONNET, Benoît, FURON, Teddy, et BAS, Patrick. Forensics through stega glasses: the case of adversarial images. In : International Conference on Pattern Recognition. Springer, Cham, 2021. p. 453-469.

MAHO, Thibault, BONNET, Benoît, FURON, Teddy, et al. RoBIC: A benchmark suite for assessing classifiers robustness. In : 2021 IEEE International Conference on Image Processing (ICIP). IEEE, 2021. p. 3612-3616.

BONNET, Benoit, FURON, Teddy, et BAS, Patrick. Generating Adversarial Images in Quantized Domains. IEEE Transactions on Information Forensics and Security, 2021, vol. 17, p. 373-385.

BONNET, Benoit, FURON, Teddy, et BAS, Patrick. Impact du redimensionnement sur les images adverses. In : GRETSI 2022-XXVIIème Colloque francophone de traitement du signal et des images. 2022.

BONNET, Benoît, FURON, Teddy, et al. Impact of Downscaling on Adversarial Images. In : 2022 IEEE International Conference on Image Processing (ICIP). IEEE, 2022.



# RÉSUMÉ EN FRANÇAIS

---

## 2.1 Introduction

Le terme d'Intelligence Artificielle (IA) a été prononcé pour la première fois par John McCarthy lors d'un atelier de réflexion en 1955 [102]. Cet événement eut lieu au Dartmouth College à Hanover dans le New Hampshire. Ce fut Claude Shannon lui-même, le *père de la théorie de l'information* [135] qui suggéra cet atelier. Tous considèrent cette conférence comme la pierre angulaire de l'IA. La création de cette discipline était bien en avance sur la technique, puisque la puissance de calcul nécessaire était absente à l'époque. Ce fût un événement majeur qui marqua aussi bien le début de la discipline que le premier *hiver de l'IA* [28].

De nos jours, l'Intelligence Artificielle est un terme au sens très large qui permet de désigner les algorithmes qui ne fonctionnent pas selon des instructions pré-définies. Le principe de ces programmes est de traduire une entrée (c'est-à-dire des données) en sortie (action, décision, interprétation, etc.). Différentes familles existent au sein de l'IA. En particulier, l'Apprentissage Automatique (ou Machine Learning) a su émerger parmi le reste. Dans cette famille, on trouve un sous-ensemble d'algorithmes: l'Apprentissage Profond (Deep Learning). La spécificité de ces derniers est de pouvoir manipuler des données de grandes dimensions. Les champs d'applications se répartissent majoritairement en trois catégories:

- Le Traitement Automatique de la Langue (TAL).
- Le Traitement du Signal Audio-Numérique (TSA).
- La Vision par Ordinateur (VO).

C'est à cette dernière catégorie que s'intéresse cette thèse. La VO a récemment connu un fort gain de popularité en 2022 avec les modèles génératifs tels que DALLÉ-2 [119], Imagen [128], ou StableDiffusion [122]. Mais il ne s'agit là que d'une sous-discipline récente de la VO. L'application principale de la VO reste la classification à laquelle nous allons nous intéresser. Il s'agit simplement de répondre à la question suivante:



“Que représente cette image?”

Le challenge ImageNet [32] est un parfait exemple de cette tâche. C’était une compétition annuelle permettant de classer les performances des classifieurs actuels. Le but étant de classer des images dans 1 000 catégories (ou *classes*) prédéfinies, avec plusieurs millions d’exemples au total. C’est précisément de cette compétition qu’a émergé la domination de l’Apprentissage Profond. En 2012, Krizhevsky et al. remporte la compétition avec le Réseau de Neurones Profond (DNN pour Deep Neural Network) nommé AlexNet [79]. Il s’agissait du premier DNN à remporter le concours. Concours qui ne sera par la suite remporté que par d’autres DNNs qui apporteront leur lot d’innovations. Il s’agit du début de l’ère de l’Apprentissage Profond.

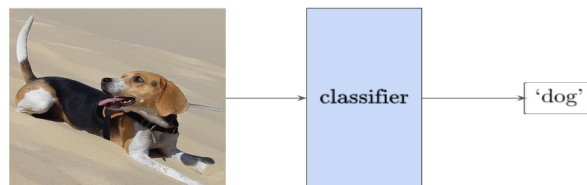


Figure 2.1 – Un classifieur permet de trier des images en catégories appelées *labels*. Le classifieur connaît d’abord une étape d’entraînement durant laquelle il apprend à effectuer cette répartition.

La notion de sécurité n’est pas toujours présente dans une tâche de classification. Certaines applications peuvent toutefois s’avérer sensibles dans leur contexte. Il peut s’agir de défense nationale par exemple (reconnaissance et cartographie), de filtre sur des réseaux sociaux (empêcher le partage de contenu inapproprié), ou de véhicules autonomes (systèmes de navigation). Ces véhicules se basent essentiellement sur de la Vision par Ordinateur.

Il apparaît essentiel d’explorer toutes les potentielles menaces qui nuiraient à l’intégrité de ces systèmes. Celles-ci sont nombreuses. Certaines interviennent dès la phase d’apprentissage:

- Empoisonnement: généralement dans le cas de données d’entraînement collectées automatiquement. Les données sont volontairement dégradées pour détériorer les performances du modèle entraîné par la suite. [106].
- Porte Dérobée: La personne qui déploie le modèle peut l’avoir entraîné à réagir différemment sur des données spécifiques. Il peut donc ensuite se soustraire à une reconnaissance ou détection, ou empêcher une bonne classification [92].

D’autres menaces concernent l’intégrité d’informations importantes, liées à la confidentialité:

- Inversion de Modèle: l'attaquant cherche à recréer les données qui ont été utilisées lors de l'entraînement [46, 168].
- Inférence d'Appartenance: similaire à la précédente, l'attaquant essaye ici de trouver si des données spécifiques qu'il détient ont été utilisées lors de l'entraînement du modèle [60].
- Extraction de Modèle: une telle attaque vise à cloner un modèle existant avec la plus grande précision [147]. Cela passe généralement par une abondance de requêtes contre laquelle il peut être nécessaire de se protéger.

Enfin il reste les menaces qui interviennent lors de l'usage final du modèle:

- Attaque par Déni de Service: ces attaques sont bien connues de tous les systèmes fonctionnant par requête. L'attaquant inonde le service de demandes pour le rendre inutilisable à d'autres utilisateurs.
- Attaque par reprogrammation: l'attaquant modifie un modèle existant pour l'utiliser à d'autres fins. [147][39].
- Attaque par Évasion ou Attaque Adversaire: le sujet de cette thèse. Une telle attaque manipule les données d'entrée de manière imperceptible pour tromper un classifieur.

Les attaques par évasion créent ce que l'on appelle des exemples adversaires ou *adverses*. Ces exemples sont des données qui ressemblent à des données normales ou *naturelles* mais qui trompent le DNN. La prédiction est incohérente avec ce qu'un humain perçoit. Cette thèse propose une étude approfondie de ces exemples: leur création, leur viabilité, pourquoi ils existent et comment s'en protéger. Voici les différents chapitres autour desquels s'articule cette thèse. Les deux premiers chapitres comportent des éléments théoriques ainsi qu'une revue de la littérature. Les chapitres suivants s'appuient sur des contributions publiées au cours de cette thèse.

## 2.2 Résumé des Chapitres

Le **chapitre 4** permet de définir la base du manuscrit. Nous expliquons dans un premier temps brièvement l'histoire de la classification d'image. Cela amène à suivre les avancées qui ont mené au début de l'ère du Deep Learning avec l'arrivée d'Alexnet. Cette première partie du chapitre définit également les bases de l'entraînement des réseaux de neurones artificiels. La deuxième partie de ce chapitre part à la découverte des vul-

néralités adverses. Les différents cadres d'attaques sont expliqués ainsi que le principe de base des attaques boîtes-blanches. Celui-ci est intrinsèquement lié à la procédure d'entraînement. Enfin les différentes notations ainsi que les attaques utilisées tout au cours de cette thèse sont expliquées.

**Le chapitre 5** revient sur l'origine des vulnérabilités adversaires. Dans une première partie, nous cherchons à comprendre la raison de leur existence. Toutes les sources potentielles de faille sont explorées. Comprendre la raison d'une faille du système permet de chercher dans la bonne direction pour corriger celle-ci. Dans une deuxième partie, nous explorons donc les défenses qui existent. Elles sont catégorisées selon leur mécanisme de fonctionnement et leur spécificité.

**Le chapitre 6** correspond à notre premier chapitre de contribution. Ces travaux ont pour but d'établir un banc d'évaluation des réseaux pour des attaques en boîte-blanche et en boîte-noire. C'est le seul chapitre qui parlera d'attaque boîte-noire. Nous y définissons également les métriques et les courbes d'évaluation d'une attaque sur un réseau. Ces courbes sont utilisées tout au long de cette thèse.

**Le chapitre 7** parle de travaux qui partent d'une observation simple: la plupart des travaux sur les exemples adverses ne produisent pas d'image. Même les illustrations les plus connues dans la littérature ne sont en réalité pas des images adverses. La raison est simple: les attaques sont considérées comme existant dans l'espace des données traitées par le DNN. Cet espace correspond au pré-traitement que subissent les images. Or cet espace est pseudo-continu, considéré par la plupart comme continu. Lorsque le pré-traitement est inversé, les valeurs de pixels ne sont pas entières. Il ne s'agit donc pas d'images numériques. Arrondir est dans bien des cas une mauvaise solution qui détruit le signal adverse. Ce chapitre propose une quantification de ces objets aussi bien dans le domaine spatial que JPEG. Les images ainsi créées restent bien adverses.

**Le chapitre 8** est un chapitre qui est lui aussi lié à des notions de quantification. Comme pour le chapitre précédent, nous cherchons à quantifier des images adverses. Cette fois-ci en revanche, nous nous inspirons des travaux dans le domaine de la stéganographie pour le faire. Cette science a pour objectif de dissimuler des informations au sein d'un medium. Une image par exemple. Les stratégies cherchent évidemment à rendre le signal inséré le plus discret possible. Nous mettons donc au point dans ce chapitre des

détecteurs pour identifier les images adverses. Certains de ces détecteurs sont employés en stéganalyse, la science antagoniste de la stéganographie. Nous montrons alors quels gains de détectabilité peuvent être obtenus selon les différentes stratégies de quantification.

**Le chapitre 9** correspond à des travaux liés à la compréhension et à la défense. Plutôt que d'étudier un mécanisme de défense, nous explorons l'impact des techniques de redimensionnement employées au cours du pré-traitement des images. Les réseaux de neurones sont capables de traiter des tailles différentes d'images, définies par leur architecture et leur entraînement. Nous cherchons donc à savoir quelle stratégie est la meilleure pour la défense à ce sujet. Dans une partie théorique, nous évaluons d'abord l'impact de la taille d'image sur le signal adverse. Nous explorons ensuite les effets des différentes méthodes de redimensionnement pour trouver la meilleure stratégie à obtenir.

## 2.3 Résumé des Contributions

La classification d'images est la principale application de la VO. Les réseaux de neurones artificiels dominant largement le domaine. Cependant, on les sait vulnérables à des petites perturbations sur les entrées: les exemples adversaires. Dans le Chap. 4 nous avons brièvement expliqué le fonctionnement de ces réseaux, leur entraînement, et la fabrication d'images adversaires. Dans le Chap. 5 nous avons étudié la raison même de l'existence de ces exemples ainsi que des travaux qui visent à s'en défendre. Mais une des conclusions de cette première partie est que les réseaux de neurones sont vulnérables aux attaques de par leur fonctionnement même.

Le but de cette thèse était d'étudier ces exemples dans un contexte réaliste. Nous avons pris plusieurs décisions dans cette direction. Nous étudions par exemple la tâche de classification que nous jugeons la plus réaliste: Imagenet. Les images qui composent le jeu de données ressemblent à des images qui seraient utilisées pour une application réelle. Et les exemples adverses devraient également être des images réalistes. Dans les chapitres 6 et 7, nous cherchons à quantifier des exemples adversaires dont les valeurs ne sont pas entières. Dans le Chap. 7, il s'agit de minimiser la distorsion dans le domaine spatial (PNG) ou fréquentiel (JPEG). Dans le Chap. 8, nous cherchons à minimiser la détectabilité.

Dans le Chap. 8, nous jouons également le rôle de la défense en mettant au point des détecteurs en partie inspiré de la stéganalyse. On y observe que la détection est efficace

lorsque l'on connaît exactement le type d'attaque (et quantification). Lorsque l'on change la stratégie de quantification, dans le but de la rendre moins détectable, les détecteurs précédents ne parviennent plus vraiment à détecter les adversaires. Et même lorsque l'on réentraîne des détecteurs sur des adversaire plus discrets, ils ne parviennent pas à retrouver leur performances initiales (données quantifiées 'naïvement').

Dans le Chap. 9, nous étudions également l'attaque et la défense. On y analyse l'impact de la taille et du redimensionnement sur les images adverses. La taille d'entrée d'un réseau varie, même pour une même tâche comme Imagenet. Et lorsque l'image n'est pas à la bonne taille, elle doit être redimensionnée. Cette étape affecte évidemment le signal adverse sur des données attaquées. On conclue dans une première partie que les réseaux utilisant des plus petites tailles semblent plus robustes aux attaques. Mais il s'agit aussi d'un équilibre entre la performance et la robustesse: le 'No Free Lunch' décrit dans le Chap. 5. Une deuxième observation est que le redimensionnement affecte logiquement plus le signal lorsque de l'anti-crénélage est employé. Celui-ci n'affecte pas les performances du réseau et devrait toujours être utilisé. Enfin, chaque attaque va être spécifique à la méthode de redimensionnement utilisée. La transférabilité est négligeable. Mais nous montrons également que l'utilisation d'un ensemble de modèle peut suffire à attaquer toutes ces méthodes simultanément. Cela rajoute évidemment de la distorsion lors de l'attaque.

### 2.3.1 Vers une Evaluation Equitable

Nous n'avons travaillé qu'avec Imagenet. Nous considérons que les données plus petites peuvent être pratique pour étudier les exemples adversaires, mais que les travaux publiés devraient s'appliquer à Imagenet pour être comparables. Mais même lorsque les articles utilisent Imagenet, les données sont rarement les mêmes. Une pratique commune est par exemple de récupérer *aléatoirement* 1 000 images du jeu de validation, ce que nous avons fait dans certains de nos travaux. Mais il n'y a aucun consensus sur ces données. Cette méthode n'est peut-être pas la meilleure dans le cadre d'évaluation et comparaison. Nous avons l'exemple en Sect. 5.3.2 des travaux de [159] qui ont extrait 5 000 images. Le taux de classification correcte est de 100% sur ces images (par le DNN étudié). Ces données ne semblent donc pas réellement aléatoires. Ce qui se rapproche le plus d'une bonne pratique est d'utiliser des jeux de données dédiés à la discipline, le *Neurips Adversarial Challenge* [80] 2017 par exemple.

Le deuxième critère d'évaluation serait comment l'attaque est utilisée. Les exemples adverses sont généralement construits selon:

1. Une contrainte de Distorsion: l'exemple est construit avec une distorsion donnée, quelle que soit sa classification à l'issue.
2. Une contrainte de Succès: l'attaque continue de pousser la distorsion jusqu'à ce que l'exemple soit adverse.
3. Une Optimisation: l'attaque minimise la distorsion tout en s'assurant que l'exemple est adverse.

Chacun de ces scenarii peut être utilisé choisi selon l'intérêt des auteurs. Par exemple pour des travaux de détection, utiliser une contrainte de Distorsion est intéressante. Le signal a toujours la même intensité et l'exemple est *probablement* loin derrière la frontière de classe. A l'opposée, chercher à détruire le signal adverse par traitement quelconque sera plus compliqué. Ces travaux vont plutôt travailler avec un scénario d'Optimisation.

Dans l'optique de travailler avec le *meilleur* des deux mondes, nous avons principalement travaillé avec le scénario d'optimisation. Mais ce choix n'était peut-être pas toujours optimal. Notamment lorsque nous avons étudié des questions de transférabilité (chapitres 7 et 9). Nous considérons toutefois que ce scénario reste le plus réaliste. Pour cela nous avons principalement utilisé l'attaque BP pour laquelle nous avons développé une version *best-effort* (Chap. 6). Cette attaque fonctionne très rapidement et produit des exemples adverses basse-distorsion de manière très fiable.

## 2.3.2 Les Images Adverses

Nous avons beaucoup étudié la création d'*images* adverses. Travailler sur un scénario boîte-blanc signifie connaître le modèle. Mais probablement pas d'avoir accès aux représentations intermédiaires des données. Ce qui est souvent le cas des exemples adverses qui sont construits dans le domaine continu (tenseurs) et non discret (pixels). Dans les chapitres 7 et 8, nous expliquons que cette tâche n'est pas nécessairement triviale, surtout dans le cadre d'optimisation.

Une attaque dans le monde réelle se basera probablement sur des images postées en ligne, ou bien même physiques (autocollants par exemple). Des images en ligne pourraient être utilisées afin de contourner des filtres de contenu, ou pour protéger ses propres données contre de la collecte automatique de données. Nous avons travaillé sur cette dernière problématique dans un court article [10] visible en Annexe A. Des applications dans le monde physique pourraient par exemple servir à tromper des véhicules autonomes, ou bien à éviter des systèmes de reconnaissance faciale.

Produire des images adverses est donc en concordance avec cette thèse: faire des attaques réalistes. Toutefois, le scénario boîte-blanche n'est pas toujours le plus réaliste. Mais cela pourrait le devenir avec des attaques plus avancées d'extraction de modèles [147].

### 2.3.3 Défenses et Perspectives

Nous avons considéré l'aspect défenses plusieurs fois au cours de cette thèse. Notamment pour la détections (chapitre 8), pour les ensembles de modèles (chapitres 7 et 9), et enfin sur les bonnes pratiques quant à la taille et au redimensionnement (chapitre 9). La sécurité est un domaine itératif et chaque contribution en attaque permet de progresser en défense. Par exemple dans le chapitre 8, nous jouons le rôle itératif des deux côtés.

Nous avons également travaillé sur de la *robustification* de modèles récemment. Mais ces travaux restent pour le moment non publiés. Un aperçu de ces travaux est donné en Annexe B. Le principe de ces travaux est de rendre les modèles plus robustes via un réentraînement non-supervisé. Cette méthode est semblable au *logit pairing* (décrit dans le Chap. 5), mais l'appariement s'effectue dans le domaine des *features*.

### 2.3.4 Un Dernier Mot

Nous avons parlé plusieurs fois de la compétition entre attaque et défense. Cela donne parfois naissance à des critiques biaisées sur les travaux de l'autre camp. Le manque d'unification des moyens d'évaluation est bien souvent un frein à la progression. En conclusion nous allons parler de travaux récents de Guo et al. [56]. Ils expliquent que les réseaux les plus robustes aujourd'hui pourraient être finalement aussi robustes que les neurones biologiques. Pour cela, ils caractérisent un équivalent de signal adverse pour l'oeil humain. A partir de ce modèle, ils évaluent que nous pourrions être tout aussi sensibles à des signaux adverses précis. Au cours des dernières années, les progrès ont été tels dans l'Apprentissage Profond que l'on peut souvent être subjugués par leurs performances. Cela nous mène probablement à juger leurs vulnérabilités de manière critique. Nous sommes toutefois nous-mêmes vulnérables dans notre traitement de l'information. Les illusions d'optique sont un parfait exemple d'exploitation de ces vulnérabilités. La différence reste sans doute que nous sommes généralement conscients que notre perception est trompée.

# INTRODUCTION

---

The term Artificial Intelligence (AI) was first coined at a workshop by John McCarthy in 1955 [102]. Taking place at Dartmouth College (Hanover, New Hampshire), this workshop was proposed by Claude Shannon himself, the *father of information theory* [135]. This event is widely considered to be the cornerstone of AI. The lack of computational power and memory however led this new field to immediately know an *AI winter* [28], longing for breakthroughs.

Artificial Intelligence is now a broad term to designate algorithms that are not based on hard-coded rules. These programs can translate an input, *i.e.* data, to an output. Different families of algorithms exist in AI. In particular, Machine Learning has emerged as the most flexible and adaptive system. Deep Learning algorithms are the largest models of Machine Learning. Their specificity is that they tackle large-dimensional data. Fields of applications can be sorted into three main categories:

- Natural Language Processing (NLP) which uses text-based data.
- Audio Signal Processing (ASP) which uses audio-based data.
- Computer Vision (CV) which uses image-based data.

The latter is at the heart of this thesis. Each of these fields can be divided into multiple sub-categories. CV has recently known an explosion in popularity with image generation. Models like DALLÉ-2 [119], Imagen [128], or StableDiffusion [122] have brought light on the impressive possibilities offered by AI.

Yet the first and most popular task in CV remains image classification. It answers one simple question:

“What does this image represent?”

The ImageNet [32] challenge well represents this task. It is a yearly competition that establishes a classification benchmark on 1,000 object categories (or *classes*) over millions of images. This is precisely on this task that Deep Learning emerged and started its dominance. In 2012, Krizhevsky et al. won the challenge with their Deep Neural Network



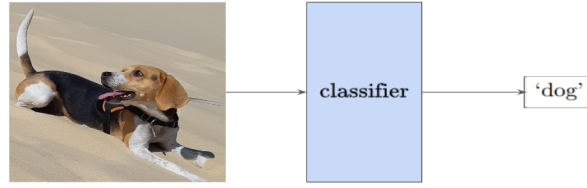


Figure 3.1 – A classifier sorts images in categories called labels. The classifier underwent a *training* procedure during which it learned to recognize a set of classes.

(DNN) AlexNet [79]. It was the first DNN to win the contest that would then only be won by other DNN models, marking the beginning of the Deep Learning era.

While most classification tasks are not linked to any notion of security, some might have crucial applications. For example in the case of national defense and security (scouting, mapping biometrics); in the case of online content filter (possibility of letting harmful or shocking content through); or in the case of autonomous vehicles, which could affect a broader population. Navigation systems rely mostly on CV tasks tied together. On such an important task, one must investigate the security threats. And there are many. Some involve malicious intervention during the training phase:

- Poisoning: usually in the case of automatically collected data. The data is intentionally degraded to worsen the performance of the later trained model [106].
- Backdooring: anyone who trains a model might teach the model to react differently to specific data. This lets the same person to have secret knowledge on how to break classification [92].

Some other security threats deal with extracting crucial information, closely related to data privacy:

- Model Inversion: the attacker tries to recreate data used during the training phase [46, 168].
- Membership Inference: similar to the previous one, the attacker here tries to find whether the specific data in their hands was used to train a model [60].
- Model Extraction: such attack aims at cloning the attacked model with the highest fidelity [147].

The last category of threats involves downstream usage of the model:

- Denial of Service (DoS): these well-known attacks apply to all request-based systems. The attack is simple: a user floods the system with requests to either bring it down or prevent other users from using it.
- Reprogramming Attack: the attacker modifies an existing model to perform another

task. This other task is done through manipulating data [39].

- Evasion or Adversarial Attack: the hearth of this thesis. Such attacks manipulate data in an imperceptible fashion so that it fools a classifier.

Evasion attacks create adversarial examples. Such samples of data are similar to other data processed by the DNN. They however exploit a specific vulnerability that breaks prediction. This thesis is an extensive study of those: their creation, their tangibility, why they are made possible, and ways to protect from them. Here are the different chapters that compose this thesis. The first two are essentially theoretical and a review of existing works. The following chapters describe our published contributions.

**Chapter 4** sets the basics for the rest of this manuscript. It first explains the basics of image classification with Deep Neural Networks. It then offers an explanation of how attacks are performed: which setup and the technique involved. Finally, this chapter introduces the most common attacks seen throughout this thesis.

**Chapter 5** is divided in two main sections narrowly tied together. The first part of this chapter revolves around explainability and tries to understand the very existence of adversarial examples. The second part of this chapter describes defenses that take advantage of this knowledge.

**Chapter 6** is our first contribution. It is the creation of a benchmark. The goal is to evaluate the robustness of existing models trained on Imagenet on both white-box and black-box attacks.

**Chapter 7** This work stems from a simple observation: most works on adversarial examples work in the floating-point domain. While these examples are indeed adversarial, they are not images and cannot be saved as such. An image indeed is constituted of only integer values. This work investigates the quantization of such examples in both spatial and JPEG domains. Images created through this method remain adversarial.

**Chapter 8** is a work that gets its inspiration from the field of steganography. This science aims at hiding information within media and images in particular. The adverse science is steganalysis which aims at detecting the presence of said information. The idea behind this study is twofold: use steganography techniques to intelligently quantize adversarial samples and hide the presence of the adversarial signal; and use techniques of steganalysis to detect said adversarial images.

**Chapter 9** is also a contribution about defense. Rather than exploring another defense mechanism, it studies the impact of image scale and the rescaling of the adversarial

signal. DNNs can process various image sizes and it is not trivial to understand which size will have the best behavior regarding its robustness. Bring theory head to head with experimentation yields best-practice to adopt for the defender to protect a model.

PART I

# Introduction on Adversarial Samples

---



# **CLASSIFIER MODEL AND ADVERSARIAL EXAMPLES**

---

## 4.1 Introduction

A Deep Neural Network (DNN) as a classifier is trained to classify images by the object represented in the picture. This is for instance the well-known ImageNet challenge encompassing a thousand classes. The state-of-the-art proposes impressive results as classifiers now do a better job than humans with fewer classification errors and much faster timings. The advent of the AlexNet DNN in 2012 is often seen as the turning point of ‘Artificial Intelligence’ in Computer Vision.

Yet, the recent literature of adversarial examples reveals that these classifiers are vulnerable to specific image modifications. The recent field of adversarial attacks explores ways of fooling DNNs in this fashion since the works of Szegedy et al. [141]. The goal of an attack is to modify an image with little distortion so that its predicted label differs from the ground truth. The perturbation is often a weak signal usually invisible to the human eye. Almost surely, no human would incorrectly classify these adversarial images. The perturbation is *a priori* both classifier and image specific. This topic is extremely interesting as it challenges the ‘Artificial Intelligence’ qualification too soon attributed to Deep Learning.

The literature considers three setups:

- **White-Box:** The attacker knows the classification model architecture and parameters. Most attacks use the very core strength of DNNs to fool them: gradient back-propagation. The very first attacks were FGSM [53], IFGSM [81] and DeepFool [104], later on improved by PGD [98], CW [21], or BP [166].
- **Black-Box:** The attacker only queries the model and observes its output. Attacks can not exploit the gradient. They thus either locally estimate it (HopSkipJump [24] or GeoDa [118]) or probe the class frontier such as SurFree [99].
- **Gray-Box:** The attacker has partial knowledge of his/her target, for instance, the classification model is public but some front-end defense mechanisms are secret.

These attacks are associated with two possible goals:

- **Targeted:** The attacker determines which class should the classifier predict over the adversarial sample.
- **Untargeted:** The attacker only needs the final predicted class to differ from the ground-truth.

In this thesis, we explore mostly the setup of **white-box** and **untargeted** attacks. This configuration is the most favorable for the attacker. It allows us to study further the

behavior of DNNs and unfold their secrets.

Adversarial examples are an emerging field in Information Forensics and Security, addressing the vulnerabilities of Machine Learning algorithms. They are also interesting in non-security applications. They constitute a means to better understand how deep learning models work (or do not work) [108]. This comes in addition to the generalization of the model measured by the accuracy over the validation set. They are key to investigating the frontiers between classes in the image space. In this non-security perspective, the quest for more robust models often resorts to adversarial training, smooth labeling, and network architecture modifications. These mechanisms aim at pushing away the frontiers from the training samples.

## 4.2 Deep convolutional networks

### 4.2.1 A Brief History

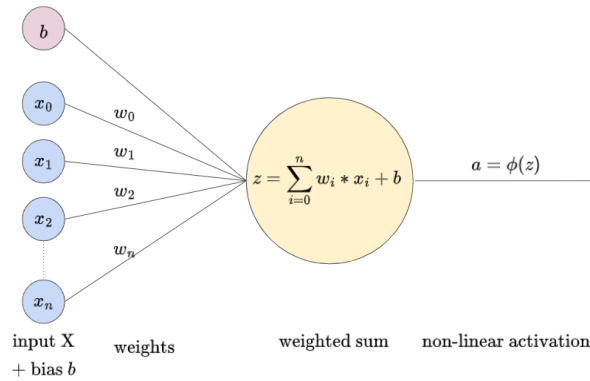


Figure 4.1 – Illustration of an artificial neuron.

Deep Learning is an incremental science. Current state-of-the-art models use many bricks that were laid down over the years. Slowly at first, for a lack of hardware and thus results. And then increasingly fast since the advent of AlexNet [79].

In 1957, F. Rosenblatt conceptualized and simulated the first artificial neuron further theorized in his published work in 1961 [124]. Psychologist at first and then a neurobiologist, he drew inspiration from biological neurons to model the artificial neuron. The concept is simple: a linear combination of multiple inputs is calculated (Fig. 4.1). A non-linear activation function acts as a threshold that ‘fires’ or simply outputs a signal if the



processed input signal is sufficient. The artificial neuron is *activated*. From multiple neurons, stacked in *layers* (or *fully-connected layer*), one can build a Multi-Layer-Perceptron (MLP). Layers (or hidden layers when they are intermediary) are rows of perceptrons stacked one after another (Fig. 4.2). The MLP is capable of extracting complex information and relationship within the input data.

Another notable step towards modern Deep Learning was studied by H.J. Kelley in 1960 [76]. In his works, Kelley set the basics for gradient back-propagation for complex systems optimization. This contribution is arguably less visible than the perceptron. It is however the very mechanism that made *learning* possible. The learning phase consists in assigning the right weights  $\theta$  to the model. This learning phase, further explained in Sect. 4.2.4, will long remain the main obstacle of Deep Learning.

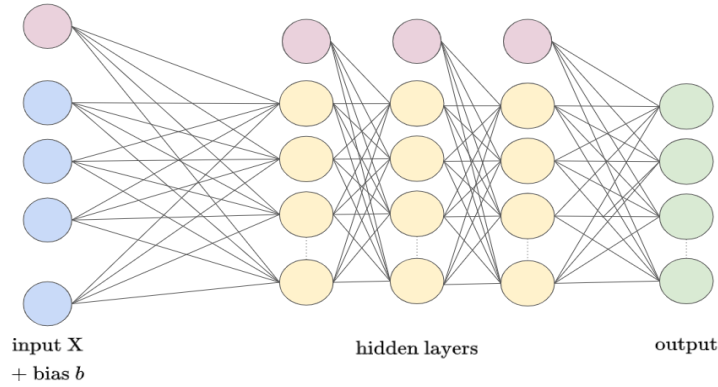


Figure 4.2 – Illustration of a Multi-Layer Perceptron. Output is  $C$ -dimensional. The size and number of hidden layers are parameters chosen when designing the model.

## Towards Convolutions

The very first convolutional neural network (CNN) was developed by K. Fukushima in 1970 [48]. All moderns Deep Neural Networks (DNNs) are CNNs. In fact, the only difference is purely arbitrary. A DNN is simply a CNN that stacked enough layers to be deemed “Deep”.

Called Neocognitron, this CNN introduced convolutional layers as well as pooling layers. These two new additions are very important features of DNNs. Convolutional layers are specific layers of artificial neurons. A filter is a group of neurons, that has few weights and only considers a small neighborhood of the input, for example,  $5 \times 5$ . And a layer consists of multiple filters. The first convolutional layer might extract basic

information such as straight lines and colors. The second layer will calculate a combination of these extracted *features* which yields slightly more complex information such as angles for example. The complexity of the patterns gets increasingly important. In the final layers of a modern DNN, features may represent a class itself.

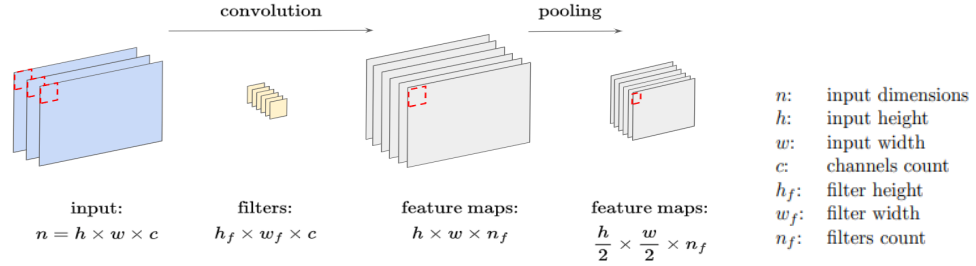


Figure 4.3 – Illustration of the first convolution operation in a DNN, followed by a pooling. Size is decreased by a factor 2.

The second contribution of this work is the pooling layers. A specificity of convolutional layers is that they preserve spatial dimension. On a color image, the 3-dimensional input has a size  $h \times w \times c$ . Where  $h$  is the number of pixels in the *height* dimension,  $w$  in the *width* dimension and  $c$  the *depth* dimension (or channels, *e.g.*: Red, Green and Blue). After a convolutional layer, this input is transformed to another object of dimension  $h \times w \times n_f$ , where  $n_f$  is the number of filters in said layer. Pooling layers are ways of reducing the size of this object. Pooling is an operation done on a small neighborhood as well, usually  $2 \times 2$ , which effectively reduces the size by a factor 2 (Fig. 4.3). The operation can be *max*-pooling: the maximum value of the neighborhood is kept, the rest is tossed; or *average*-pooling: values are averaged over the neighborhood. Combined with convolutions, this operation leads to a necessity in image classification: translation equivariance. This means that an object will be recognized as such by the system regardless of its position on the image.

Once all the different elements were available Lecun et al. [83] built Lenet5 in 1998. It is the first CNN with a practical application: reading hand-written digits on postcodes. It is also the first CNN to outperform the competition on any specific task. The so-called LeNet5 was trained using back-propagation, which was the missing link to make the training procedure work. Its name stems from the 5 *hidden* layers. There are 3 convolutional layers (with 2 pooling layers after the first two), and 2 fully-connected layers.

Convolutional layers act as feature extractors to build semantic information for the fully-connected layers. The final fully-connected layer then outputs a score for each class

(before activation) called *logits*. These logits are transformed to a probability mapping using a *Softmax* activation (Eq. (4.1)). Softmax had been investigated before on MLPs [15].

$$\text{Softmax}(x)_k = \frac{e^{x_k}}{\sum_{i=1}^C e^{x_i}} \quad (4.1)$$

LeNet5 is the result of almost a decade of research after Lecun had proposed the first conceptual LeNet in 1989. Its success was made possible by the progress in computer hardware. This is also the reason of the stagnation over the next 14 years. Computational power was simply not enough to make significant progress.

### Alexnet: A Deep Learning Revolution

Although the works of Lecun et al. [83] brought light on the possible success of CNNs, the real Deep Learning revolution came in 2012. Krizhevsky et al. released AlexNet[79] and caused a major paradigm shift in Computer Vision. Their prowess was to win ImageNet Large Scale Visual Recognition Challenge (ILSVRC), or simply ImageNet [32] challenge this year. It is a famous event within the CV community in which classifier models compete with each other in correctly predicting images on 1,000 different classes. Alexnet in terms of architecture is similar to LeNet5, but with 8 hidden layers this time. It can be called the first *Deep* Neural Network.

Saying that Krizhevsky et al. merely capitalized on increased computational power would be greatly undermining their work. Alexnet brought its load of much-needed innovations to the community. Most of which tackle issues that were unsolved until then. The first is a new activation function called the Rectified Linear Unit (ReLU):

$$\text{ReLU}(x) = \max(0, x) \quad (4.2)$$

Researchers had mostly experimented with two activation functions until then: sigmoid:  $f(x) = (1 + e^{-ax})^{-1}$ , with  $a > 0$  the slope factor) and the hyperbolic tangent:  $f(x) = \tanh(ax)$ . Figure 4.4 shows an important property of these functions: their gradient is null outside a small range centered around 0. ReLU has two possible values of gradient: 0 for negative input and 1 for positive input. This simple function allowed Krizhevsky et al. to build a deeper CNN. As is further explained in Sect. 4.2.4, the gradient is at the heart of the training procedure. Having a small value of gradient until then caused the phenomenon of *vanishing gradient*. This is caused by the gradient of all the layers being

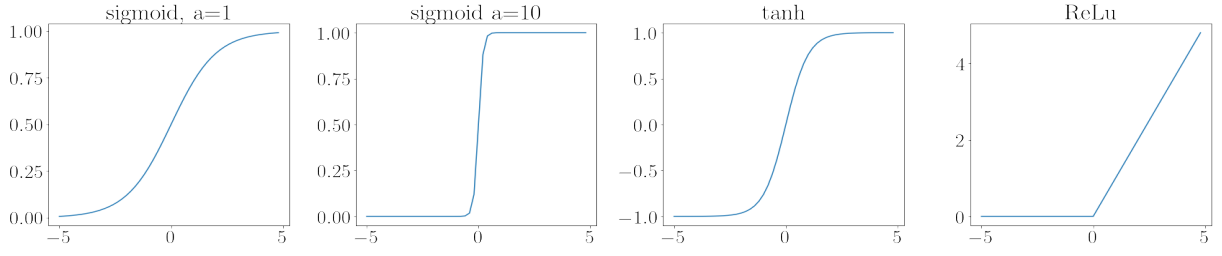


Figure 4.4 – Comparison of different activation functions around 0. First and second curves show the impact of the factor  $a$  in the sigmoid function

multiplied together:  $\lim_{n \rightarrow \infty} \prod_i^n x_i = 0, |x_i| < 1$ . When the gradient is superior to 1, the opposite phenomenon of *exploding gradient* appears:  $\lim_{n \rightarrow \infty} \prod_i^n x_i = \pm\infty, |x_i| > 1$ . Both issues are fixed with the ReLU function:  $\lim_{n \rightarrow \infty} \prod_i^n x_i = \pm 1, |x_i| = 1$ .

Another innovation is the introduction of Local Response Normalization (LRN). This step normalizes the output of convolutions layers. This prevents specific filters from having a response so large that it takes over the others. This heavily impairs training and keeps the DNN stuck in non-favorable local minima. This normalization remained in use for some years. In 2015, Ioffe et al. introduced Batch Normalization [69] (or BatchNorm) which is a data-specific normalization. It adds trainable parameters that normalize outputs of convolutions with regards to the training set. Although it is more effective and helped reach higher accuracy, the drawback of BatchNorm is that it is very dependent on the training data. If test data comes from a different distribution, its effectiveness can be compromised.

AlexNet also popularized Dropout, developed by the same team [63]. It is a regularization technique that randomly ‘turns off’ neurons during training phase. This prevents decision paths from being reinforced every step and taking over the other.

Finally, the generalization of AlexNet over such diverse test data was made possible through data augmentation. Simply put, data augmentation is a technique that modifies the input to enrich the training set. Modifications vary from random cropping, translation, rotation, and vertical or horizontal flipping of the images. It can be seen as another regularization technique that prevents overfitting.

All these additions made Deep Learning emerge from a potential technology to the standard of Computer Vision. In the years that followed, the competition was won by only deeper and larger DNNs. Although the challenge was discontinued in 2017, Imagenet is still widely used as a dataset today. Accuracy on test set is still a good benchmark to rank vision models, and it is the largest labeled image dataset that can often be useful for

pretraining. Models have nowadays become much larger, and keep increasing in terms of parameter count. But let us not downplay the progress that was made following Alexnet.

GoogleNet or Inception V1 [140] in 2014 introduced many tricks and novel techniques. They use two outputs to their DNN: one intermediary and one final. The first output serves as a way to propagate further information through the gradient that ended up being too weak otherwise. They also created *Inception layers* that combine and concatenate simultaneous convolutions of different sizes. Among these sizes, the surprising  $1 \times 1$  convolution kernels were introduced, serving as depth reduction.

In 2015 Simonyan et al. introduced VGG family of DNNs [137]. These DNNs ranging from VGG-11 to VGG-19 were parameters-heavy at the time. Their smallest implementation used 133M parameters and their largest 144M. As a comparison, Alexnet ‘only’ had 62.4M and GoogleNet 6.4M. The architecture did not use many innovations. Their creators argue that their main contribution is the use of small  $3 \times 3$  convolutions kernels. The real prowess remains that they successfully trained this many layers. It was a feature itself at the time.

Lastly, Resnets were released in 2018 [13] and are another notable contribution. Their main contribution is using ‘skip’ connections within the convolutional layers to transmit the information from previous layers to the rest of the network. These connections are simply an identity function that ensures a complete transmission of the information.

Many other DNNs were developed over the years and continued performing better and better as is shown on Fig. 4.5. Among the most recent models, Attention networks made a noticeable impact [150]. Attention networks apply priority over certain regions of an image, deeming what part of it is more relevant to classification. Another important breakthrough that tends to become the standard is the Transformer network or Self-Attention network. Inspired by NLP model Bert [33], ViT [37] popularized the use of transformers for vision. Transformers tend to generalize better than other DNNs and avoid biases.

We will now see notations used throughout this manuscript to define models, images, and attacks.

### 4.2.2 Model and Data Notation

Let  $I \sim \mathcal{D}$  be a digital image in the domain  $\llbracket 0, 255 \rrbracket^n$  where  $n$  is the number of pixels and  $\llbracket 0, 255 \rrbracket := \{0, 1, \dots, 255\}$ . This image is preprocessed before feeding a DNN. This stage is defined during the training phase of the DNN to improve its learning capability



Figure 4.5 – Top-1 accuracy (*i.e.*: correctly predicted images) on DNNs since Alexnet. The interactive chart is available at <https://paperswithcode.com/sota/image-classification-on-imagenet>

over the data. After the DNN was successfully trained, it expects data preprocessed in the same fashion to make a prediction. This preprocessing is usually done in two steps: range reduction from  $[0, 255]$  to  $[0, 1]$ , and normalization:

$$x := \frac{I/255 - \mu_{data}}{\sigma_{data}}, \quad (4.3)$$

where  $\mu_{data}$  and  $\sigma_{data}$  are respectively the mean and standard deviation computed over the training data. These constants are usually channel-specific (*i.e.* each channel has its own normalization), sometimes set to arbitrary values such as 0.5. We call  $x \sim \mathcal{X}$  the tensor image which lies in a pseudo-continuous ensemble. Its values are in the floating-point domain.

### 4.2.3 Classifier Model

Let  $f : [0, 255]^n \rightarrow [0, 1]^C$  be a classifier mapping a tensor image  $x \sim \mathcal{X}$  to class logits  $\hat{l}$  for  $C$  classes.  $\hat{l}$  is a  $C$ -dimensional vector. Probability vector  $\hat{y}$  is obtained with a Softmax function:  $\hat{y} = \text{Softmax}(\hat{l})$  (Eq. (4.1)). The predicted class is defined as:

$$\hat{c}(x) := \underset{k}{\operatorname{argmax}} \hat{y}_k(x). \quad (4.4)$$

The classifier makes a correct prediction if  $\hat{c}(x) = c(x)$ , where  $c(x)$  denotes the ground-truth class of  $x$ .  $c(x)$  class is embedded in a one-hot ground-truth vector  $y$ , another  $C$ -dimensional vector. The  $c$ -th value is 1, rest is 0. A perfect classifier yields  $\hat{y} = y$ . But to achieve this, the classifier must first undergo a training phase. This is how it learns to make a good mapping of input image to correct classification.

#### 4.2.4 Training a Classifier

Training a classifier is the very first step to having a functioning model. A model does not initially ensure  $\hat{c}(x) = c(x)$ . It is first initialized with random weights under some form of uniform Gaussian distribution. The probability of making a correct prediction is entirely random:  $1/C$ . As discussed previously in Sect. 4.2.1, the training phase is far from easy. We will see in Sect. 4.3 that this procedure is also at the heart of White-Box attacks. We will now have a quick overview of this procedure.

DNNs are complex systems with millions to billions of parameters nowadays. The problems that DNNs try to solve are non-convex, high-dimensional, and optimized over a large volume of data. No optimal parametrization can be computed in a single step. The solution comes from iterative approximation.

To achieve this, a particular algorithm is helpful: gradient descent (shown in Eq. (4.5)). This method, when it converges successfully, finds a good approximation of the solution by minimizing an *cost* function or *loss* function. Convergence is however far from trivial and almost certainly the algorithm ends in a local minimum. It has been observed that these local minima are however close to the theoretical global minimum. Some even argue that local minima are global minima [75].

Formally, parameters  $\theta$  of the model are updated at each iteration with the following rule:

$$\theta_{t+1} \leftarrow \theta_t - \eta \times \frac{\partial \mathcal{L}}{\partial \theta_t} \quad (4.5)$$

Where  $\theta_t$  represent the set of parameters  $\theta$  at iteration  $t$ .  $\mathcal{L}$  is a loss function, usually *categorical cross-entropy* in the case of classification (Eq. (4.9)). And  $\eta$  is the *learning rate*. The gradient is calculated using the chain rule:

$$\frac{\partial x}{\partial z} = \frac{\partial x}{\partial y} \times \frac{\partial y}{\partial z} \quad (4.6)$$

The gradient is calculated from the output, we call this procedure *back-propagation*.

Let us consider a DNN that has  $l$  layers. Parameters  $\omega^h$  of a hidden layer  $h$  are thus updated as follows:

$$\omega_{t+1}^h \leftarrow \omega_t^h - \eta \times \frac{\partial \omega_t^{h+1}}{\partial \omega_t^h} \times \cdots \times \frac{\partial \omega_t^l}{\partial \omega_t^{l-1}} \times \frac{\partial \mathcal{L}}{\partial \omega_t^l} \quad (4.7)$$

Gradient descent updates the weights  $\theta_t$  layer per layer in order to minimize the loss. This is the basic gradient descent algorithm. The optimal set of parameters  $\theta^*$  is thus:

$$\theta^* = \operatorname{argmin}_{\theta} \mathbb{E}_{x \sim \mathcal{X}} \mathcal{L}(\theta, x, y) \quad (4.8)$$

Most training procedures nowadays use slight variations of gradient descent. Stochastic Gradient Descent (SGD) with Momentum [113] adds an aggregation of past gradients to control the descent. This reduces bias on a single step and quickens convergence. Root Mean Square Propagation (RMSProp) [64] is another optimization algorithm that uses a mobile exponential average of gradients to regularize the update term. The most popular optimizer remains Adam [77] which simply is a combination of both SGD and RMSProp.

A loss function translates the error that a classifier makes on its predictions. In multi-label classification, the *categorical cross-entropy* is the most common. It is defined as follows:

$$\mathcal{L} = - \sum_{k=1}^C y_k \cdot \log \hat{y}_k \quad (4.9)$$

This differentiable loss thus computes the error between prediction and ground-truth. Minimizing this loss effectively improves the accuracy of the DNN. The procedure is the following: the DNN is ‘fed’ a batch of images, then the loss (Eq. (4.9)) is computed over this batch, and finally, parameters of the DNN are updated using Eq. (4.5). A batch of images is used rather than a single image as another regularization method. This ensures better convergence. Single images generate too much bias over one step that takes time to be corrected. At the end of the training phase, we have ideally  $\theta \approx \theta^*$ . This means that the classifier reached the highest accuracy possible.

These were the basics of classifiers and training. We will see how adversarial examples in a white-box setup are crafted using similar methods. In fact, attacks can be seen as a form of gradient descent performed over the image rather than the DNN.



## 4.3 Adversarial Examples

### 4.3.1 Adversarial Vulnerabilities of DNNs

Adversarial attacks were brought to light by the works of Szegedy et al.[53, 141]. The famous image (Fig. 4.6) of a panda turned into a gibbon through an invisible perturbation has been seen by many deep learning enthusiasts.

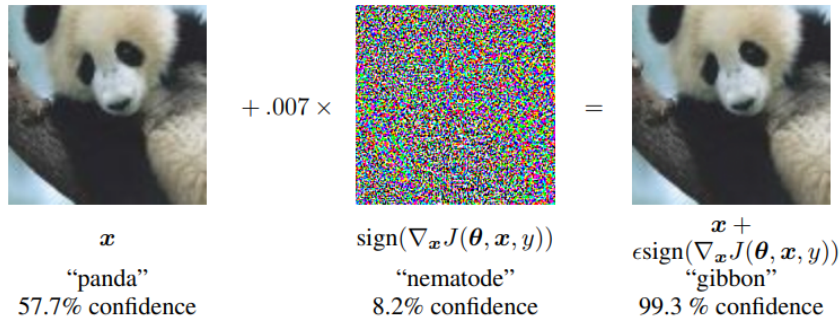


Figure 4.6 – Famous illustration of an adversarial example from the work of Goodfellow et al. [53]. The perturbation signal is voluntarily amplified to understand the addition (see the factor 0.007).

This image is an *adversarial example*, and the perturbation can be called the adversarial signal. It is the result of an evasion *attack*. Although this signal may look like noise, it is carefully crafted. The mechanism behind it is the very same one used to train a model: gradient descent using back-propagation.

Adversarial examples can be a surprising result. Especially now that state-of-the-art DNNs perform better than a single human on a classification task. This has been observed in medical diagnosis [17, 121] for instance. These works [53, 141] gave birth to a whole new field dedicated to adversarial examples. Works vary from theoretical studies of their very existence, to how to craft them in different setups, with better performance, or how to defend against them. We will now study the crafting procedure of these adversarial examples. We will see in Chap. 7 that this panda is however not an adversarial *image*.

### 4.3.2 Fooling a classifier

Evasion attacks are performed at evaluation stage. The DNN is trained, and ready to operate on its task and its weights are frozen. The attack leaves DNN in its exact same state, only the input is modified to affect the prediction. The optimization problem

seen in Eq. (4.5) takes place on the image instead. But this time the goal is to degrade prediction, so it can rather be seen as a gradient *ascent*:

$$x_a \leftarrow x_0 + \eta \times \frac{\partial \mathcal{L}}{\partial x_0} \quad (4.10)$$

Where  $x_a$  is the created adversarial image from an image  $x_0$  we call *natural*. The gradient is calculated using back-propagation as seen in Eq. (4.7). Chain rule is propagated one further (Fig. 4.7). Note that the attacker has access to gradients only in a white-box setup (or a gray-box setup to some extent). Unless stated otherwise, we consider white-box attacks throughout this thesis.

A completed attack forges an adversarial sample  $x_a$  such that:

$$\hat{c}(x_a) \neq c(x_0). \quad (4.11)$$

The resulting class  $\hat{c}(x_a)$  is either chosen by the attacker in a *targeted* scenario or any class that verifies Eq. (4.11) in an *untargeted* scenario. An attack optimizes the perturbation  $x_a - x_0$  according to a given metric, usually the  $\ell_0, \ell_1, \ell_2$  or  $\ell_\infty$ -norm. This gives the following optimization problem on a generic  $\ell_m$ -norm:

$$x_a^* := \min_{\hat{c}(x_a) \neq c(x_0)} \|x_a - x_0\|_m. \quad (4.12)$$

Throughout this thesis, we consider attacks in their  $\ell_2$ -norm form. This is common practice in image processing: The PSNR gives the logarithmic scale of the  $\ell_2$ -norm. Indeed, for natural images (*i.e.* ImageNet [32], but not MNIST),  $\ell_2$  reflects distortion perceived by humans *when comparing similar images with very low distortion*. Researchers also often consider the case of  $\ell_\infty$ -norm (maximum distortion over a pixel) or more rarely  $\ell_0$ -norm (number of modified pixels).

Equation (4.11) displays the paradigm of an *untargeted* scenario. In a *targeted* setup, the attacker picks the predicted label or *adversarial* class. The optimization performed on the image is this time a normal gradient descent:

$$x_a \leftarrow x_0 - \eta \times \frac{\partial \mathcal{L}}{\partial x_0} \quad (4.13)$$

$$\mathcal{L} = - \sum_{i=1}^C y_i^t \cdot \log \hat{y}_i \quad (4.14)$$

Where  $y_i^t$  is a one-hot vector. The 1-value is on the targeted class  $t$ . The difference between targeted and untargeted attacks lies mostly in this  $+$  or  $-$  sign. The optimization is a gradient descent just like during the training phase. The objective is to minimize the loss characterizing the prediction of the targeted class. The following works will now be presented in their untargeted version.

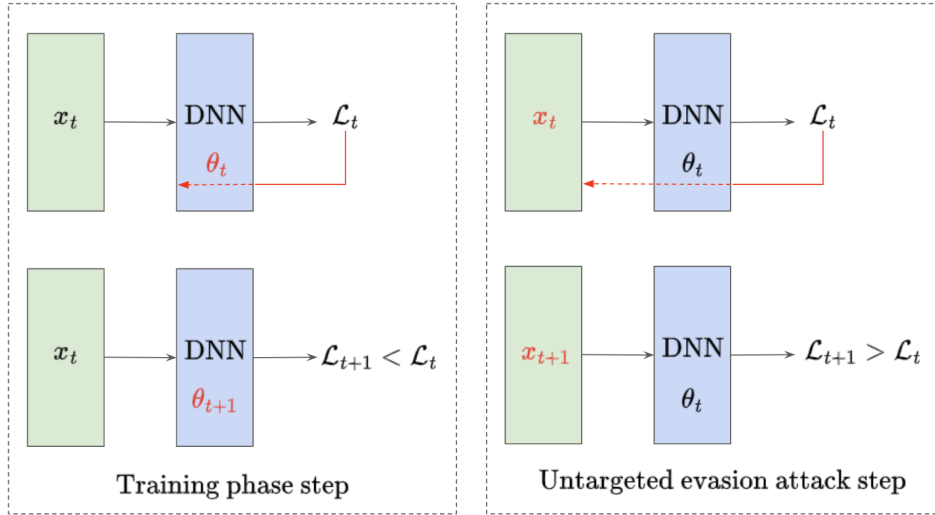


Figure 4.7 – Similarity and difference between an evasion attack and training phase.

These definitions and optimization problems apply to classification tasks. They can however easily be transferred to other DNN-based tasks. In Computer Vision, adversariality has been studied in various scenarii such as:

1. Object segmentation [2, 45, 62]
2. Object detection and tracking [89, 156]
3. Video recognition models [72, 87]

But adversariality is not limited to Computer Vision. This phenomenon is seen in numerous other application fields and tasks:

1. Audio (speech-to-text) [20, 26]
2. Text (sentiment analysis) [73, 153]
3. Reinforcement Learning (prevent the agent from learning) [67, 90]
4. Anomaly detection [78]

This is just a mere overview of what is possible. Adversariality has been studied in most Deep Learning application, and literature is very rich in this domain. This thesis is however about classification in Computer Vision. The next sections will dive into the construction of adversarial examples in a white-box setup.

### 4.3.3 Usual White-box Attacks

White-box attacks have access to the entire model and parameters. This is convenient to use gradients, as seen in Eq. (4.10). We now define a slightly different loss that we call adversarial loss  $L_{\text{adv}}$ . This will prove useful to craft perturbations:

$$L_{\text{adv}}(x) := \hat{y}_c(x) - \hat{y}_a(x), \quad (4.15)$$

Where  $\hat{y}_c(x)$  is the predicted probability of  $x$  belonging to the ground truth class  $c(x_0)$ . The second term  $\hat{y}_a(x)$  is the probability of the adversarial class. Note that both terms can be replaced by logits  $\hat{l}_c(x)$  and  $\hat{l}_a(x)$  for similar results. In a *targeted* scenario, class  $a = t$  is a parameter of the attack. In an *untargeted* scenario, it is the best prediction excluding the ground-truth of the original:

$$a = \arg \max_{k \neq c} \hat{y}_k(x). \quad (4.16)$$

Using this trick, we still consider the attack to be untargeted, but the adversarial class is guided. Attacks are however iterative processes and the adversarial class can change over time. This is not the case when the probability  $\hat{y}_a(x)$  is already high. It does change more frequently when  $\hat{y}_c(x) \approx 1$ .

The optimization problem is thus similar to a targeted scenario as shown in Eq. (4.13). Loss needs to be decreased to ensure adversariality instead of increased. Moreover, it has an interesting property that justifies its use. When  $L_{\text{adv}}(x) < 0$ , the predicted class is not the ground-truth label  $c$  and  $x$  is adversarial. It thus provides both a direction and a stopping criterion. We will now introduce notable attacks that we use throughout this thesis. Attacks are defined with this loss  $L_{\text{adv}}$ .

### Fast Gradient Sign Method FGSM

FGSM [53] is the first and most basic attack. It uses the sign of the gradient on the original image:

$$x_a = \text{clip}_{[0,255]}(x_0 - \epsilon \times \text{sign}(\nabla L_{\text{adv}}(x_0))), \quad (4.17)$$

where  $\text{clip}_I$  is the clipping of the component within the interval  $I$ . This attack is not iterative, it is a single-step process relying on only one parameter  $\epsilon$ . Note that the adversarial sample is a quantized image only for an integer value of  $\epsilon$ .

## Iterative Fast Gradient Sign Method iFGSM

iFGSM [81] is an iterative version of FGSM.

$$x_a^{(i+1)} = \text{clip}_{[0,255]} \left( x_a^{(i)} - \epsilon \times \text{sign} \left( \nabla L_{\text{adv}}(x_a^{(i)}) \right) \right). \quad (4.18)$$

This attack uses two parameters: the descent rate  $\epsilon$  previously seen in FGSM and the number of iterations  $N_{\text{iter}}$ .

## Projected Gradient Descent PGD

This attack is an iterative attack whose updates are defined as follows in its  $\ell_2$ -norm version PGD<sub>2</sub> [98]:

$$x_a^{(i+1)} = \text{clip}_{[0,255]} \left( \text{proj}_\alpha \left( x_a^{(i)} - \epsilon \frac{\nabla L_{\text{adv}}(x_a^{(i)})}{\|\nabla L_{\text{adv}}(x_a^{(i)})\|} \right) \right). \quad (4.19)$$

PGD<sub>2</sub> revolves around a projection  $\text{proj}_\alpha$  on the ball centered on  $x_0$  of radius  $\alpha$ . This projection is effective only if the  $\ell_2$ -norm of the perturbation exceeds  $\alpha$ . PGD<sub>2</sub> also uses an  $\ell_2$  normalized gradient to have better control of the perturbation update. This attack uses 3 parameters: the radius  $\alpha$ , the descent rate  $\epsilon$ , and the number of iterations  $N_{\text{iter}}$ . Other versions of PGD can use different norms, which affects the projection step.

## Boundary Projection BP

BP [166] is a fast two-step iterative attack. The first step quickly finds an adversarial sample while the second step refines it by reducing its distortion. Stage 1 is defined as follows:

$$x_a^{(i+1)} = \text{clip}_{[0,255]} \left( x_a^{(i)} - \gamma_i \epsilon \frac{\nabla L_{\text{adv}}(x_a^{(i)})}{\|\nabla L_{\text{adv}}(x_a^{(i)})\|} \right), \quad (4.20)$$

where  $\gamma_i$  is an acceleration term ranging from a predetermined  $\gamma_{\min}$  at the first iteration to 1 at the final one. Stage 2 refines the adversarial sample found through projection using the same parameters and normalized gradient of  $L_{\text{adv}}$ .

### Carlini & Wagner CW

This attack minimizes the following Lagrangian formulation in its  $\ell_2$ -norm version CW<sub>2</sub> [21]:

$$J(x_a, \mu) = \|x_a - x_0\|^2 + \mu |L_{\text{adv}}(x_a) - m|_+ \quad (4.21)$$

where  $|z|_+ = \max(0, z)$  and  $m \leq 0$  is a margin. The minimum of this equation is found with ADAM optimizer within an inner loop. The outer loop does a line search over  $\mu$ . When both outer and inner loops are done, the adversarial sample with the least distortion is returned. This attack uses five parameters: the amount of iterations over both loops, the margin  $m$ , and the learning rate and momentum for ADAM.

## 4.4 Chapter Conclusion

This chapter allowed us to set the basics for the rest of this thesis. We first saw an overview of the history of DNNs and how each of the essential bricks was brought to the community over a long period of time. This was until the advent of Alexnet. Then, successive DNNs started to use the same bricks and improve on them. We briefly reviewed notable innovations and architecture principles. This allowed us to define a classifier model and study the important first phase of every classification project: the training phase.

We however exposed that this training phase, the strength of DNNs, can have an inconvenient side-effect. This is the existence of adversarial examples in a white-box setup. It is also worth mentioning that many Black-Box attacks exist as well. It is a whole different paradigm that is more constrained for the attacker. Attacks can be based on prediction scores [96, 163]; or only decision ([24, 86, 99, 118]). Some aim at estimating the gradient to reproduce similar mechanisms, while others do not.

In this thesis, we however mostly consider DNNs to be a white-box. The only exception is Chap. 6 which partly uses black-box attacks. We then studied in little more detail some popular white-box attacks. All these attacks are studied throughout this thesis. Many others exist and listing all of them would be tedious. This was not the purpose of this chapter. We will now dive into a less practical chapter and analyze the very existence of adversarial examples. Understanding how they exist also leads to defending from them. This will be the second part of the chapter.

# **ADVERSARIALITY: ANALYSIS AND DEFENSES**

---

## 5.1 Introduction

Within every security topic, two sides of the coin coexist. Attack and Defense. The field of adversarial examples is no exception to this rule. Security is played as an iterative game where each party makes a move one after another. The attacking side is however usually one step ahead. A flaw in the system is found. The Attacker sees the opportunity to exploit it. The defense then fixes the issue and hopefully focuses on extending the solution to other similar potential flaws. Being one step behind as the defender is obviously not desirable. This led companies over the world to hire *ethical* hackers (also known as *white-hat* hackers). Their purpose is simple: break a system to identify threats that will need fixing. The pattern we see here is simple:

Identify a flaw and attack → Gain knowledge over the system → Defend and fix the issue → Identify further risks and start over.

Defense and understanding of a system are thus closely tied together. DNNs are still vastly considered to be black-boxes. We understand the operations done within but we do not know what is the purpose of each intermediate step. Even DNNs with the fewest parameters have millions of them.

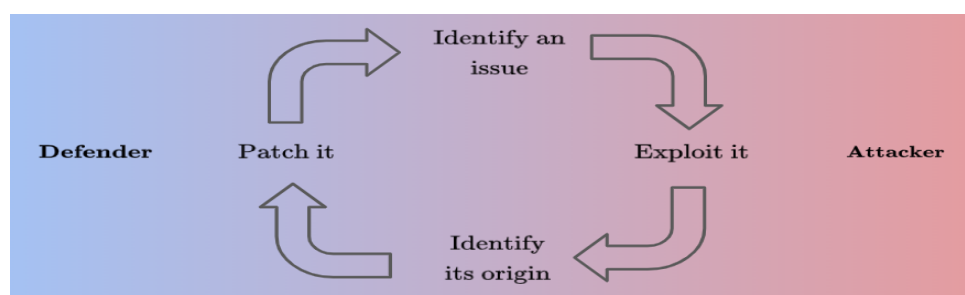


Figure 5.1 – Illustration of the iterative game between attack and defense. The frontier between both sides is always blurry. There is an everlasting race to find issues.

## 5.2 Understanding Adversarial Examples

A classifier is usually built from two bricks: convolutional layers stacked together and a fully-connected layer. The purpose of the convolutional layers is to extract intermediary representations or *features*. The fully-connected layer creates a probability map given the existing features.



The first convolutional layers extract simple information such as contours, texture, and color. The following layers piece together these small bits to extract larger and more complex patterns. The final convolutional layer builds a feature map: a high-dimensional vector representing features on the whole picture.

In terms of explainability we can for example easily analyze the first layer of convolution and understand the patterns considered by the DNN. The first layer usually looks for straight or curved lines, and color presence. But as we get to deeper layers, it is harder and harder to understand the macro-patterns, made of small patterns pieced together.

The construction of feature maps is part of the black-box. A human for instance might identify a car through the presence of two or more wheels, a parallelepiped-like structure, a windshield, etc. The features extracted and used to make a prediction by the DNN are however unknown. And they vary greatly from one model to another. ResNet-18 has a 512-dimensional feature map while ResNet-50 has a 2048-dimensional feature map. This is true for any architecture, a bigger model (more parameters) usually uses a higher-dimensional feature map. Features extracted between two networks simply cannot be the same. All the intermediary phases (convolutional operations) are thus unintelligible. We can understand how the whole system works, but we do not know the steps leading to the result.

The very existence of adversarial examples can be useful to understand and unfold the black-box. Understanding how the DNN is fooled gives insight into how it was correct in the first place. First, we need to identify the possible culprits. We denote three of them:

- Data: DNNs are vulnerable to adversarial examples because the data it was fed during the training phase is incomplete or misleading.
- Training Phase: The DNN is not taught correctly how to process images or given too much freedom.
- DNNs themselves: adversarial examples are simply a feature of Deep Learning.

Obviously, neither of these three is the sole responsible. We need to dive into each category and investigate their share of responsibility.

### 5.2.1 Data Is The Enemy

Data is commonly pointed at when it comes to adversarial robustness. How could a DNN be robust if its data is misleading? The strength and weakness of Deep Learning is the amount of data. This can be mitigated with data augmentation, but only to some extent. To train a good classifier, one needs an enormous amount of data. Table 5.1

gives an overview of the most popular datasets in image classification. Even for small classification problems such as MNIST or CIFAR-X, the amount of data is considerable. But as we tackle harder tasks, the amount becomes overwhelming, especially in the case of ImageNet. Most of our work throughout this thesis was done using ImageNet classifiers. We will now consider this dataset for the rest of this section.

Name	Image Size	Image Count	Classes Count
MNIST	$28 \times 28$	60,000	10
CIFAR-10	$32 \times 32$	60,000	10
CIFAR-100	$32 \times 32$	60,000	100
Tiny-ImageNet	$64 \times 64$	100,000	200
MS-COCO	$> 224 \times 224$	330,000	80 + 91
ImageNet	$> 224 \times 224$	$> 14\text{M}$	1,000
ImageNet-21k	$> 224 \times 224$	$> 14\text{M}$	21,841

Table 5.1 – Most popular image classification datasets. MS-COCO has 80 object categories and 91 ‘other’ categories such as background, sky, ground...

## Data labeling

It would be tedious work to check the label of every single image within Imagenet. It would take over 38 years for someone to cover the whole dataset if they were to inspect 1,000 images a day. So how exactly were these images annotated?

It was done through crowdsourcing with Amazon’s Mechanical Turk (AMT)<sup>1</sup>. A bot first queries the internet to find images of a specific class. Humans then validate whether the queried class is indeed present in the image. The protocol seems pretty sound at first but there are many flaws with it.

## Dogs ? Dogs. And some more

The choice of classes within the 1,000 existing is questionable. It seems obvious that reaching a *thousand* categories was a goal in itself. It is probable that some categories were added just to reach that number. We can for instance hope that people who participated

---

1. The name is inspired by a famous automaton named the Mechanical Turk, built in 1770. It was supposed to be a self-operating machine that played chess. It was however quickly discovered that it was a hoax and a human was operating the ‘automaton’, hiding within the table that displayed the chess set...

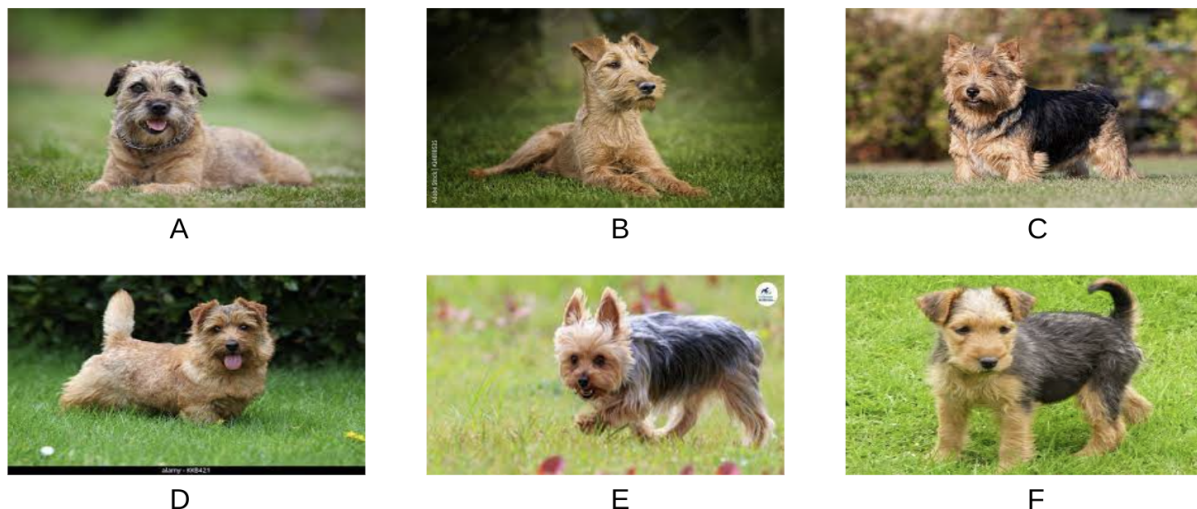


Figure 5.2 – Image A: 182: ‘Border Terrier’, Image B: 184: ‘Irish Terrier’, Image C: 185: ‘Norwich Terrier’, Image D: 186: ‘Norfolk Terrier’, Image E: 187: ‘Yorkshire Terrier’, Image F: 189: ‘Lakeland Terrier’.

in AMT were dog enthusiasts. We count no less than 118 dog breeds within classes. Many breeds are easily distinguishable from one another. Additionally, people surely looked up what each of them looked like prior to confirming their presence. But even so, this task is likely complicated. We extracted an example of 6 different classes, displayed on Fig. 5.2. This will allow you to test your own differentiation abilities on these breeds. This task is definitely not trivial.

Labeling error is indeed found throughout the even most popular datasets as per Northcutt et al. [107]. They estimate that the validation set on Imagenet has at least 6% error. It is also suggested that more recent models overfit on these errors. Overfitting on validation or test sets is an indirect consequence of cross-validation. While the DNN is not trained directly on these sub-sets, it is evaluated on them. Thus the model that will be picked in the end is the one that has the best accuracy on validation sets. This might come at the cost of indirect overfitting.

An improved (*e.g.* more curated) dataset is released along with this work [107], available at <https://labelerrors.com/>. This website also offers good visualization of the existing errors in the data. It is worth seeing for yourself if you were to work with this set.

Trained with this cleaner Imagenet, a Resnet-18 outperforms off-the-shelf Resnet-50 while on cleaner CIFAR-10, VGG-11 outperforms VGG-19. Resnet-50 should perform better (resp. VGG-19). They however generalized on wrong data which hinders prediction

confidence or even classification on clean data. Low prediction confidence is a strong enemy. It means the class *frontier* could easily be crossed. The image lies close to it, and thus a small perturbation will suffice to fool said classifier.

Finally, some labels in ImageNet are plainly duplicates:

- 638: ‘maillot’ and 639: ‘maillot, tank suit’
- 134: ‘crane’ and 517: ‘crane’
- 657: ‘missile’ and 744: ‘projectile, missile’
- 620: ‘laptop, laptop computer’ and 681: ‘notebook, notebook computer’
- 836: ‘sunglass’ and 837: ‘sunglasses, dark glasses, shades’

Images from both 657: ‘missile’ and 744: ‘projectile, missile’ could be from either class (as is illustrated on Fig. 5.3). The same goes for 620 and 681, 836 and 837, and 638 and 639 (not shown for decency reasons, but we invite you to see for yourself that both class contain the exact same type of images). In the case of 134: ‘crane’ and 517: ‘crane’ there is however a difference since the former designate birds and the latter building machines. But surely this ambiguity led to errors.



Figure 5.3 – We extracted images from the validation set for two classes. Top row: images of class 657: ‘missile’. Bottom row: images of class 744: ‘projectile, missile’.

Labeling error is not the only threat here. Feature maps will more likely be extremely correlated. This results in high class proximity within the feature space. Section 5.2.3 provides a further analysis of this phenomenon.

Many people participated in labeling these images. While the effort is admirable, we can still question the quality of such work. It is hard to find statistics about this whole

work, data is unavailable. We can however hope that there was a lot of overlap on reviewed images. Having each image checked by multiple people would effectively reduce the error made. Still, labeling is a long and tedious task and the community is grateful to have access to such a large dataset. Labeling errors is the human factor. But it is not the only issue with ImageNet and image datasets in general. Images themselves can be misleading and poor lessons for the DNN to learn.

## Data representation

An important work by Birodkar et al. [9] analyzes redundancy in datasets. They for instance estimate that ImageNet or CIFAR-10 contain at least 10% of semantic redundancy. Images within the same class overlap a lot in visual information which hinders learning and proper generalization. A basic example of redundancy is the DNN that learns to recognize the street to predict a car. Streets are not a ‘car’ per se, but appear to be closely related. If the classifier is then shown a car on a beach, it might struggle. This is an example of ‘background’ redundancy as is shown on Fig. 5.4.

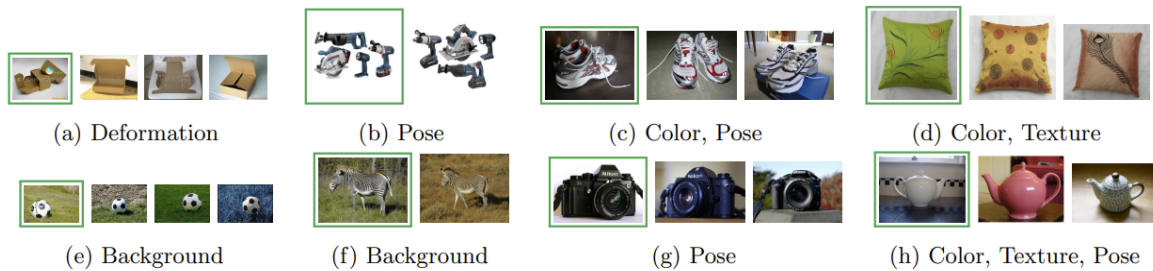


Figure 5.4 – Figure 1 from [9] that helps visualize different types of redundancies within ImageNet pictures. Images highlighted with a green box are kept for an improved dataset while the others are deemed redundant. They are discarded.

Other issues are pointed at by Berer et al. [8]. The main complaint is the single annotation per image. Many images have multiple possible labels represented. Having a single label is misleading because the DNN might find the exact features of a secondary class but it is taught to ignore it. This in turn is detrimental to the learning procedure. Features of all present classes are mixed and the fully-connected layer adjusts its decision based on them. The harm caused is twofold:

- Secondary classes count for the prediction of the primary one. An image containing only an object from a secondary class will have less confidence on its correct label. The primary class might have a non-null and even non-negligible probability.



- Features from secondary classes are not discriminant for prediction. The construction of their features is tuned down. This results in poor learning of these off-classes.

These issues will especially be problematic in the case of strongly correlated classes. Figure 5.5 from their work illustrates some of these situations. In the case of 477: "carpenter's kit, tool kit", one can assume that 784: 'screwdriver' will be present in many instances (Fig. 5.6 displays a few examples).



Figure 5.5 – Illustration from [8] showing images with multiple classes within. Red: original labels found in the dataset. Green: suggested improved labels if multiple annotations are considered.



Figure 5.6 – Looking through images of 477: "carpenter's kit, tool kit" reveals an logical abundance of 784: 'screwdriver'.

Playfully titled 'Are we done with ImageNet?' [8], their work questions the relevance of ImageNet nowadays. We just showed that there is indeed room for improvement. A lot of improvement. But it is still the best data there is for image classification. A DNN is likely to be trained on it to at least learn a diversity of features, before being fine-tuned on another specific task. This is especially convenient if this new task contains little data. Finally and mostly, it is still the ultimate benchmark in the community. Hopefully, it will be supplanted in a near future, but it is our best option at the time.

### 5.2.2 Learning the Hard Way

The way information is treated stems from the training phase that we originally defined. The DNN is taught to minimize an error, without clear instructions on how to achieve this. Training a DNN is entirely empirical and usually requires multiple attempts before having a satisfying result. Convergence of gradient descent is never ensured and can suffer from many phenomenons detrimental to the training procedure:

- Overshoot: learning rate too high, the gradient descent is unable to remain in a minimum and diverges.
- Undershoot: learning rate too small, the loss function gets stuck in a local minimum.
- Gradient explosion: the back-propagation of the gradient increases its value multiplicatively which makes the algorithm diverge.
- Gradient vanishing: the back-propagation of the gradient decreases its value multiplicatively which stops the convergence.

Finally, a DNN in its converged state is inherently dependent on initialization. This first step is done in a well-thought-out fashion but remains random. So how do we know that two different DNNs (same architecture) that underwent the exact same training phase will end up in the same state? The answer is simple: we know that they do not and that they are not the same classifier. The landscape of a loss function is highly non-smooth. This is due to the high dimensionality of the problem tackled. Two DNNs might end up with the same loss scores on training and validation sets. But they will not be equal. This is one of the prowess of Deep Learning: the loss of a converged DNN finishes in a local minimum that is close in value to the global minimum. Yet we cannot identify this global minimum and can only assume that gradient descent made the loss close to it.

### Generalization

The strength of DNNs lies in their ability to generalize. Images are complex data and extracting the right information is not an easy task. This led Computer Vision community to investigate invariance in their classification systems, well before DNNs. Classifiers are usually built from two bricks: feature extraction and category sorting. It so happens that DNNs have both within a single network even if both tasks are not done with the same layers (for instance convolutional layers for feature extraction and fully-connected for classification).

Before DNNs, independent feature extractors were used for this first half of classification. Notably Scale-Invariant Feature Transform (SIFT) [94] that introduced, as its name suggests, scale invariance. Another Oriented FAST and Rotated BRIEF<sup>2</sup>(ORB) [126] introduced rotation *robustness*. Complete invariance is equivalent to a perfect conceptualization of an object. Formally, let us note  $x \in \mathbb{R}^n$  a class representation,  $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_f}$  a feature extractor and  $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$  an input transformation:

$$f(x) = f(g(x)) \quad (5.1)$$

In a perfect extractor, this is true for any transformation or composition of transformations. A transformation is a modification of the representation of a given object. This includes translation, rotation, color change, scale, viewpoint, contrast, saturation, etc. No extractor is perfect to the point of reaching perfect equality between both terms of Eq. (5.1).

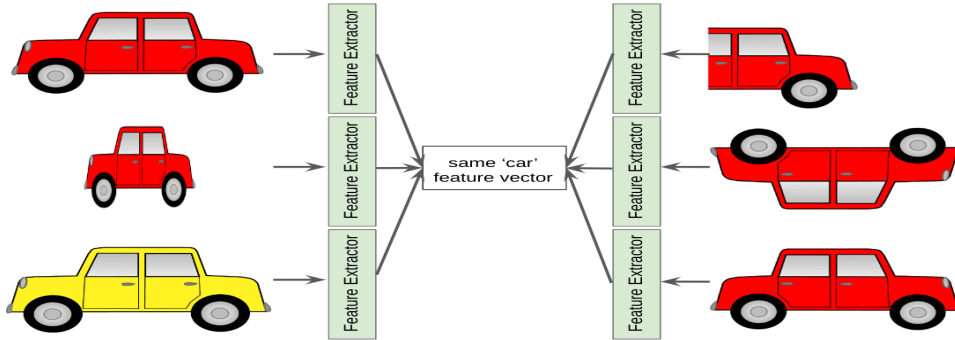


Figure 5.7 – Illustration of a perfect feature extractor. Every feature vector is equal, regardless of the transformation applied to the image. The object is perfectly conceptualized.

The role of convolutional and pooling layers is to bring scale and translation invariance to the DNN. Other invariances are however acquired during the learning stage. These are taught through the sheer amount of data as well as data augmentation. The amount of data required by DNN is mostly to improve generalization and push the DNN away from training set biases. But how much generalization is *too much* generalization?

2. FAST being itself an improved version of SIFT[125] and BRIEF another feature extractor [18] robust to pixel intensity. We can certainly notice a pattern of puns well-appreciated by the community.



## Excessive Invariance

The work of Jacobsen [71] points this excess in state-of-the-art models. DNNs develop invariances to a lot of contextual information that should not be ignored. They propose an invariance-based attack akin to adversarial attacks. Their attack however adds perturbation to an image such that its logits matches the logits of another image. Images shown on Fig. 5.8 display the surprising inability of a DNN to differentiate entirely different images (more examples are available in the full article [71]). They study this phenomenon through two different tools.

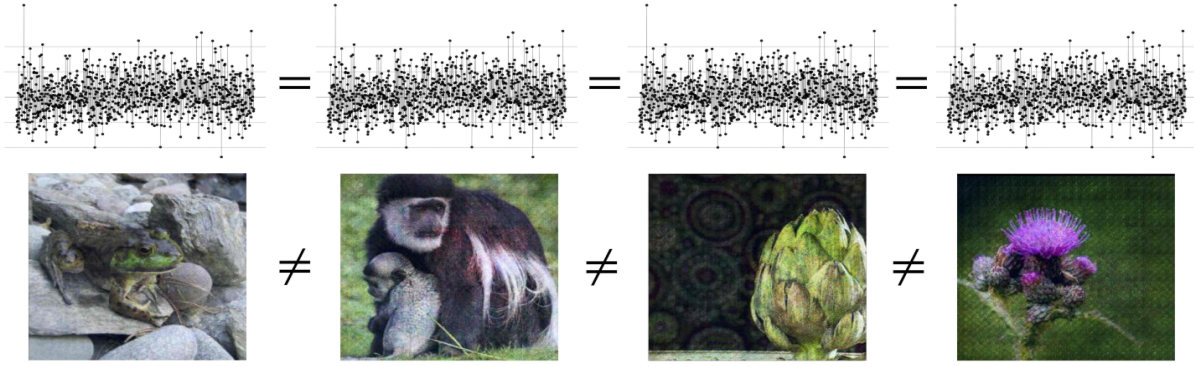


Figure 5.8 – Figure 1 of [71] showing visually different images that share the same logits (displayed above) through a ResNet152. The first image is unmodified.

The first one is a toy dataset called *adversarial spheres* [50]. It is a simple classification problem in which both categories are two concentric high-dimensional spheres of different radii. This dataset was introduced to study the detrimental effect of high dimensions inherent to image data. This translates into high data sparsity and many degrees of freedom for the attack to be performed. This is a prime example of *curse of dimensionality* [6].

The second one is an iRevNet [70] model. It is a DNN invertible up until the last projection onto class vectors. By removing this last projection, convert it to a *fully*-invertible network called fi-RevNet. This network has two outputs  $z_s$  for classification (semantic variables) and  $z_n$  (nuisance variables). Its classification performances on ImageNet are only slightly lower than the competition. It is thus a decent classifier.

It is demonstrated that a perfect classifier should maximize the conditional mutual information  $I(y; z_s | z_n)$  while decreasing the mutual information  $I(y; z_n)$  [71]. The identified culprit is cross-entropy loss. A training phase using this cost function will only increase  $I(y; z_s)$ . This hinders the capacity of DNNs to understand the presence of possible *nui-*

sance that could be linked to a given class. This can be read in echo with the presence of redundancies studied above in Sect. 5.2.1.

Adversarial Spheres [50] were also used in other works to blame BatchNorm [49]. We briefly mentioned BatchNorm [69] in Sect. 4.2.1 as an improvement to LRN. It is a normalization, applied batch-wise onto data after going through a layer. BatchNorm operate a 0-mean and unit standard-deviation as well as the following operation:  $x_{BN} \leftarrow \gamma * x_{Norm} + \beta$ . Where  $x_{BN}$  is the input after BatchNorm and  $x_{Norm}$  after normalization. Both *gamma* and *beta* are parameters that were however learned during the training phase. Adversarial images are not part of the same distribution as natural images. It is easy to see how this could backfire, and [49] provides experimental observation of it.

We have so far identified many possible culprits for adversarial vulnerabilities. Among them: data labels, data itself, data dimension, cross-entropy loss, BatchNorm, etc. It seems that there is room for improvement. However, any classifier, even a human, will misclassify an image at a certain level of distortion. It is time to look into this capacity in DNNs.

### 5.2.3 Adversarial Examples Are Inevitable

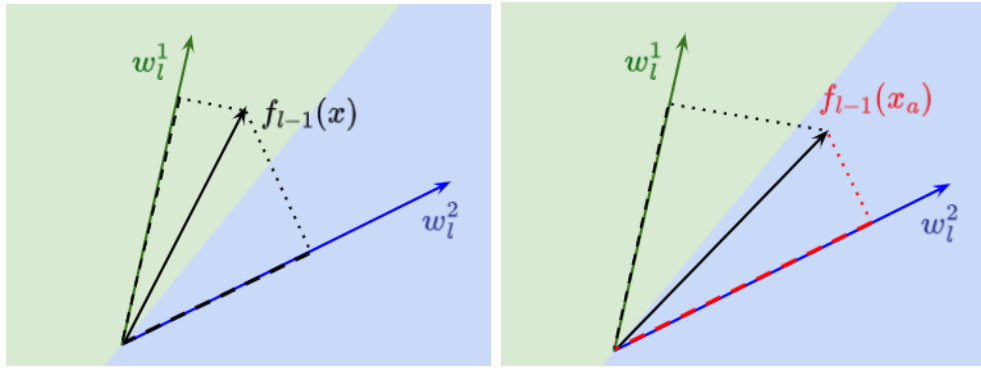


Figure 5.9 – Illustration of class projection on a 2-label problem. Dashed lines illustrate the scalar product of  $f_{l-1}(x)$  on each class vector. The natural image  $x$  has a greater logit score on class 1. The modified adversarial image  $x_a$  has been modified so it crosses class frontier. It is now classified as label 2.

The works of Cubuk et al. in 2017 titled “Intriguing Properties of Adversarial Examples” [30] as a nod to Szegedi et al. [141] state in their abstract:

“Here we argue that the origin of adversarial examples is primarily due to an inherent uncertainty that neural networks have about their predictions. We show that the functional form of this uncertainty is independent of architecture, dataset, and training

protocol; and depends only on the statistics of the logit differences of the network, which do not change significantly during training.”

It is a strong statement about adversarial vulnerability. In the previous section, we identified bricks of DNNs that *increased* an existing flaw in the system. But this work [30] offers a pessimistic view of the problem. They argue that the success of an attack is affected by two elements. The first one is the input-logit Jacobian of the model and the logits themselves. In the case of a semi-random attack, this had previously been quantized by [43]. There is a clear dependency on dimensions and distances in the feature space. This subspace is very informative to understand adversariality. It is convenient to have a look at the construction of logits to gain further knowledge on the problem.

### Class Projection and Logits

Logits are the scores calculated for each class considered by a model. This is before going through a SoftMax activation that turns logits into class probabilities. So what are these scores exactly?

Let us consider a DNN  $f(x)$  built from two bricks: convolutional layers and a fully-connected layer. This DNN has  $l$  layers. We call  $f_i(x)$  the output of the  $i$ -th layer whose weights are  $\Omega_i$ . An input image  $x$  is transformed into a feature vector  $f_{l-1}(x)$  after convolutions. The weights of the fully-connected layer are such that:

$$\Omega_l = \begin{bmatrix} \text{---} & \omega_l^1 & \text{---} \\ \text{---} & \omega_l^2 & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \omega_l^C & \text{---} \end{bmatrix} \quad (5.2)$$

Logits are obtained through matrix multiplication:  $\Omega_l \times f_{l-1}(x)$ . The logit score of class  $k$  is thus a scalar projection of  $f_{l-1}(x)$  onto  $\omega_l^k$ . For clarity we note these vectors:  $f(x)$ , resp.  $\omega^k$  for the rest of this section. Seeing logits as projection eases geometric interpretation (Fig. 5.9). A scalar product is dependent on two values: norms of both vector and the cosine of the angle between them. Class logit scores are thus determined by two values  $\|\omega^k\|$  and  $\cos(\widehat{\omega^k, f(x)})$ . Table 5.2 tells us that class vectors do not differ by a lot in norm. The most discriminant value is thus the angle  $\cos(\widehat{\omega^k, f(x)})$ .

It is interesting to note that  $f(x) \in \mathbb{R}_+^{n_f}$  but  $\omega^k \in \mathbb{R}^{n_f}, \forall k \in \{0, 1, \dots, C\}$ . Feature vector is indeed obtained after a ReLu activation, which makes all its values  $\geq 0$ . There

Model	$n_f$	$\ \omega^k\ $	$\text{std}(\ \omega^k\ )$	$\min(\ \omega^k\ )$	$\max(\ \omega^k\ )$
VGG16:	4,096	1.18	0.08	0.98	1.43
Resnet18:	512	1.57	0.10	1.30	1.80
Resnet50:	2,048	0.87	0.05	0.72	1.06
EfficientNet-b0:	1,280	2.38	0.16	1.83	3.01

Table 5.2 – Class vectors in the fully-connected layer of popular models.  $n_f$  is the input or *feature* dimension.

is no such restriction on  $\omega^k$  <sup>3</sup>.

### Class Proximity

Label #	182	184	185	186	187	189
182	<i>1.00</i>	<b>0.42</b>	<b>0.43</b>	0.37	0.23	0.39
184	<b>0.42</b>	<i>1.00</i>	0.41	0.31	0.21	0.40
185	<b>0.43</b>	0.41	<i>1.00</i>	<b>0.62</b>	<b>0.44</b>	0.36
186	0.37	0.31	<b>0.62</b>	<i>1.00</i>	<b>0.43</b>	0.36
187	0.24	0.21	<b>0.44</b>	<b>0.43</b>	<i>1.00</i>	0.28
189	0.39	0.40	0.36	0.36	0.27	<i>1.00</i>

Table 5.3 – Correlations of classes vector in the fully-connected layer of Resnet18. Studied classes are expected to have high correlations: 182:Border Terrier, 184:Irish Terrier, 185:Norwich Terrier, 186:Norfolk Terrier, 187:Yorkshire Terrier, and 189:Lakeland Terrier

We discussed earlier in Sect. 5.2.1 the problem of class proximity. Frontiers between two classes  $k$  and  $l$  is highly dependent on  $\cos(\omega^k, \omega^l)$ . If this value is small, confidence can be affected by this phenomenon. A high logit score on either class yields a somewhat high value on the other. And works of Cubuk et al. [30] tell us that close logit scores are linked to high vulnerability.

We can calculate  $\cos(\omega^k, \omega^l)$  easily through the definition of scalar product:

$$\cos(\omega^k, \omega^l) = \frac{\sum_{i=0}^{n_f} \omega_i^k \omega_i^l}{\|\omega^k\| \|\omega^l\|} \quad (5.3)$$

Let us note  $C\omega$  the symmetric matrix in  $\mathbb{R}^{C \times C}$  such that  $C\omega_{k,l} = \cos(\omega^k, \omega^l)$ . Table 5.3 gives us the values of  $C\omega$  for dog labels mentioned in Sect. 5.2.1. For comparison, Tab. 5.4

3. Features therefore lies in a tiny fraction of  $\omega^k$  space  $\mathbb{R}_+^{n_f}$ :  $\frac{1}{2^{n_f}}$ . In the case of even the smallest  $n_f$  of Resnet18, it represents  $7.46 \times 10^{-153}$  % of it.

Label#	182	184	185	186	187	189
579	0.02	-0.04	0.00	0.06	0.02	-0.03
580	-0.01	0.02	0.01	0.04	-0.02	-0.03
581	-0.04	-0.04	-0.02	-0.10	-0.05	-0.05
582	-0.04	0.00	-0.03	0.00	-0.06	-0.11
583	-0.07	-0.04	0.02	-0.01	0.01	0.00
584	-0.08	-0.06	-0.03	-0.01	0.08	-0.13

Table 5.4 – Correlations of the same classes vector as Tab. 5.3. New classes are expected to have low correlations: 579: grand piano, grand, 580: greenhouse, nursery, glasshouse, 581: grille, radiator grille, 582: grocery store, 583: guillotine, and 584: hair slide

gives coefficients for a random set of consecutive labels. These are not *a priori* semantically related to dog breeds. The difference is flagrant. The correlation between dog breeds is sometimes really important. In the untargeted paradigm, these classes are especially vulnerable.

Then a solution appears: train class vectors to be uncorrelated to each other. But this would affect accuracy. Insisting once more on dog breeds: classes vectors from these classes *need* to be correlated. It appears that a trade-off exists between robustness and accuracy. It is the (in)famous No Free Lunch Theorem.

### No Free Lunch Theorem

‘No Free Lunch’ is a phrase often seen in adversarial-related works. It states that additional robustness on a given trained system comes necessarily at the cost of accuracy. The works of Tsipras et al. [148] discussed this theorem. Through experimentation, they evidence this equilibrium. It persists even in simple settings, or a theoretical case of infinite data [148]. This even holds true on adversarially robust models (see Sect. 5.3.4). In short, DNNs are *doomed* to be vulnerable to adversariality. It is an indirect consequence of their efficiency.

Using toy examples, Zhang et al. [167] push the analysis further. They propose a theoretical model of the trade-off. Robustness is basically defined by two terms: natural error and boundary error. Natural error is a classification error on untouched images. Boundary error is a measure of the margin of confidence around natural images. But then again, further works such as [117] exploit this definition of the trade-off to increase it. Once an issue is understood, we can improve on it. The future of robust DNNs is

thus not hopeless. We investigated in this section the reasons for adversarial vulnerability. There is room for improvement on some aspects of it. Data, in the case of Imagenet for instance, seems to have a lot of room for improvement. But literature is rich in novel ideas to defend DNNs. We will now study existing defenses.

## 5.3 Defending Against Evasion

Adversarial vulnerabilities necessarily exist. At least to some extent. This does not mean we cannot defend from them. In this final section of the chapter, we will dive into existing defenses. We can sort defenses into two distinct categories:

1. Reactive defenses: the defender acknowledges the vulnerability of a DNN. It does not try to interfere with the attack. Focus is set instead on the corruption of data. The goal is to either detect image manipulation or to remove (*reform*) the adversarial signal.
2. Proactive defenses: the defender tries to patch the vulnerability and/or make it more difficult to attack their system. More distortion is required to build an adversarial sample.

Both strategies are the source of numerous works. We do not try to benchmark the following defenses. The purpose of this section is to give insight into each strategy and sub-strategies.

### 5.3.1 Reactive Defenses

Two strategies exist to counter an adversarial example: detection or reformation. Attack-wise, the arms race is about reducing the distortion of the perturbation. Papers on either strategy thus assume low-distortion samples. When it comes to data reforming, this eases the job. But this might make the detection job harder to accomplish. We will now see the different approaches that have been considered by researchers.

### 5.3.2 Data Reforming

We explained in Chap. 4 that white-box attacks rely mostly on back-propagating the gradient. In a classifier, convolutional filters all overlap on the image. This generates interference in the gradient computed on the image. Moreover, attacks are usually run in

a multiple-step process. This implies even local stability (image-wise) of the adversarial signal. Therefore perturbation usually lies in the high-frequency domain. This is a crucial observation from which reforming stems. It is easier to remove a *noisy* signal than a low-frequency one. Within the iterative attack-defense security game, this inspired research on low-frequency attacks [57].

To remove high-frequencies, the most common approach is a form of compression. This technique especially aims at reducing local variance to decrease overall information. The works of Guo et al. [58] in particular use four different methods (Fig. 5.10). The first one is bit-depth reduction. Natural 256-level 8-bit images ( $\{0, 1, \dots, 255\}$ ) are converted to images with as low as 8-level 3-bit images. The loss of top-1 clean accuracy on Imagenet amounts to  $\approx 8\%$ . Considering the amount of compression, this is pretty low. DNNs are resilient to compression. The second method is JPEG compression. They use a quality factor of 75 for their experiments. This does not amount to a lot of compression and clean accuracy is almost unaffected. The third method is total-variation (TV) minimization. TV measures the norm ( $\ell_2$ ) of the difference between pixels and their neighborhood. Minimizing TV builds an image locally smooth. The resulting compression is strong, which results in a loss of  $\approx 15\%$  of clean accuracy on Imagenet. Finally, they use image *quilting* as a means of compression. The image is split into patches along a predefined grid. Each patch is replaced by its closest counterpart within a database. This results in a tremendous impact on clean accuracy: a loss of  $\approx 40\%$ . A greater gain in robustness is at the cost of worse clean accuracy. Except for bit-depth reduction, which gains in robustness do not compensate for the loss of accuracy.

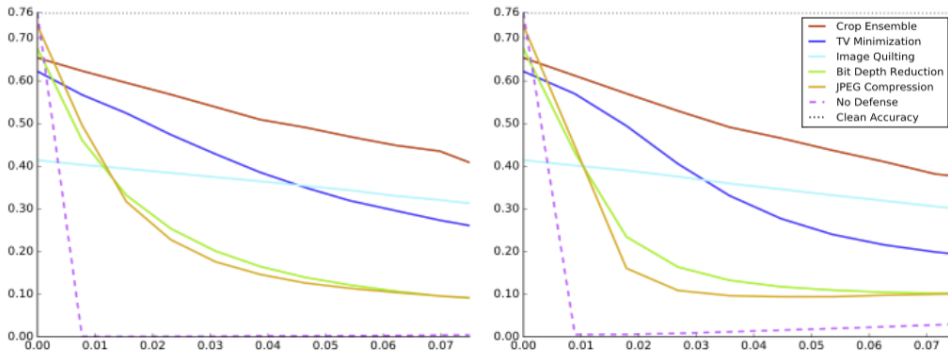


Figure 5.10 – Experiments of [58] over Imagenet validation dataset (50,000 images). y-axis is the accuracy, x-axis the  $L_2$ -dissimilarity:  $\frac{\|x_0 - x_a\|}{\|x_0\|}$ . Resnet-50 is attacked with Deepfool (left) and C&W (right).

Most works in adversarial reforming use derivatives of these ideas. It is however difficult to make a fair comparison between different methods. Different norms are used to craft adversarial samples ( $\ell_2$ ,  $\ell_\infty$  or less common  $\ell_0$ ). Different data is used as well. Interesting work from Xie et al. [159] uses a subset of 5,000 images from the validation set. But clean accuracy on this data is 100%. This means that data was cherry-picked. This is problematic for a fair evaluation. Their idea is to perform random cropping and padding within the classifier. But without a unified evaluation, it is impossible to draw conclusions.

Some works however differ from the others. A defense called *PixelDefend* [138] aims at projecting data back onto a known manifold (training data). They justifiably argue that adversarial examples lie in low-probability regions of feature maps. This idea is seen in other works such as Samangouei et al. [130] that uses a Generative Adversarial Network (GAN [54]) to do so.

In general, reforming data comes at a cost: natural accuracy. Unsurprisingly, these methods drag us back to No Free Lunch (Sect. 5.2.3). Hopefully, the impact of such methods is greater on manipulated data. But it will still alter clean images, which in turn decreases accuracy. Reforming data can be favorable, especially with very low distortion, but is to be used cautiously.

### 5.3.3 Adversarial Detection

Detection is a very special case of defense. It does not respond to a form of trade-off between accuracy and robustness. Neither is *directly* affected by detection. Accuracy will however indirectly be affected by false positives (image is not classified). An important aspect of this method is that it requires a *detectable* signal. If a network has very low robustness, adversarial images are crafted with very little perturbation. They become *indistinguishable* from natural images. Thankfully a detector is far more effective than the human eye. But a detector might require a somewhat robust network to work for. The problem of No Free Lunch is just displaced to another step of the whole process.

One of the cornerstones of detection is Feature Squeezing [162]. It is heavily inspired by works on reformation. It was first developed using bit-depth reduction and spatial smoothing. But any method seen previously can be used. The idea is to use a reformation method or *squeezer* on an image. Logits outputs of clean and reformed data are compared. If the difference is above a learned threshold, the input image is deemed adversarial.

Multiple squeezers can be used in parallel to increase confidence. Method is illustrated on Fig. 5.11. Feature Squeezing works under an important assumption: the adversarial



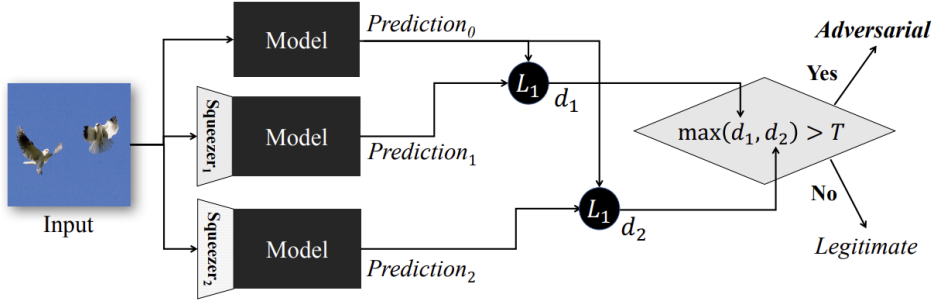


Figure 5.11 – Illustration of Feature Squeezing from [162]. Two *squeezers* are used here, but more could be added.

image is more affected by the squeezer than a natural image. Reformation works under the same assumption. Feature Squeezing has a big advantage over its competitors as it is indirectly improved as a new squeezer is available.

This detector, as well as others, comes at the cost of training time. Most other works build and train DNNs. In the paper of Metzen et al [103], sub-DNNs are integrated into the original classifier. Each convolutional layer has a secondary output that goes through a sub-classifier. The spatial size of feature maps decreases along the DNN. Each sub-classifier is thus built differently. The idea behind is that the distributions of adversarial images and natural images are vastly unaligned. Resulting feature maps thus lie in improbable spots of feature spaces.

This idea is seen throughout many works. Ma et al. [97] for instance train classifiers with natural, noisy, and adversarial images. Their argument is that noise does not bring a natural image far out of distribution. But noise counters the adversarial signal to some extent. It therefore drags adversarial images back to the distribution of natural images. Learning how either will react to added noise is key to detection.

We mentioned earlier that detection does not respond to the No Free Lunch theorem. However such methods will fail if a DNN allows very low-distortion adversarial examples to exist. Proactive defenses can be helpful either on their own or combined with detection. We will now study these defenses

### 5.3.4 Proactive Defenses

What is striking at first is how adversarial samples look unaltered even to an expert eye. This is due to the signal being diluted over a whole image, and also to the great

vulnerability of DNNs. But this vulnerability can be decreased to some extent. This is the goal of proactive defenses. We identified earlier some reasons in Sect. 5.2. Proactive defenses have indeed worked at fixing most of these possible issues. This starts with data.

### Enhancing Data

It has been observed times and times that DNNs do achieve human-level performances on natural images. DNNs could even surpass panels of experts. This can be seen in medical diagnosis [17, 121] for instance. We can assume that there is nothing inherently wrong with the training procedure. DNNs do learn what they are taught. However humans may not be the best teachers. We saw previously (Sect. 5.2.1) that data could sometimes be improved. But we may also not be able to teach the classifier to learn the right elements. As per [68], DNNs learn features that are not robust. For instance, texture or colors can be somewhat useful, but mostly non-informative. Contours and shapes seem better strategies.

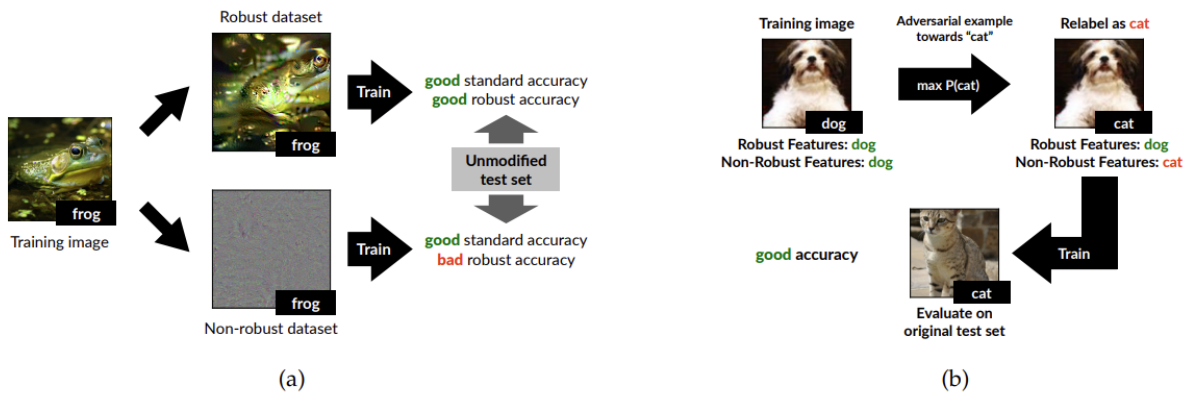


Figure 5.12 – Illustration of training experimented by Ilyas et al. [68]. A robust dataset is used on the diagram (a) and a non-robust dataset is used on diagram (b).

The authors build two different datasets. The first one is obtained by removing non-robust features. The resulting images may come as surprising as the pictured class seems less visible (illustration (a) of Fig. 5.12). Models that learn from this data both end up having a good clean accuracy and robustness. More surprisingly, they build a dataset of visually mislabeled data (illustration (b) of Fig. 5.12). These images are adversarial examples on a teacher DNN. The student DNN ends up having poor robustness but good clean accuracy. This is really telling about the weak features that DNNs learn. Forcing a DNN to learn strong features (first set of data) gives it good robustness.

The other side of data are the labels. It is argued by some works that a one-hot encoding of the pictured class is not a good lesson for the classifier. We saw for instance the case of 477: "carpenter's kit, tool kit" and 784: 'screwdriver' (Fig. 5.6). A solution some have come up with is label distillation. The idea is to use class probabilities that another classifier (called *teacher*) outputs. The idea dates back to 2015 [65] and was initially introduced for dimensionality reduction. The improvement brought to this method by Papernot et al. [110] is to use the same architecture for both student and teacher. In an iterative fashion, the student can become the teacher of a new student, and so on.

These works increased robustness by improving the quality of the training data. But we saw earlier that data was not the only one responsible for adversarial vulnerability and we can have a look at how to modify the building bricks of DNNs to improve their robustness.

## Model Tweaking

We discussed at length the problematic of class proximity in the feature subspace in Sect. 5.2.3. Some works aim at changing the way classification is performed. Instead of relying on scalar projection (Sect. 5.2.3), the works of Mustafa et al. [105] instead rely on building class polytopes. Each polytope is trained to be maximally separated from other polytopes. This decreases the correlation between classes, which in turn increases robustness. Their training objective includes two terms: one to minimize the distance from a feature vector to its class centroid, and the other to push centroids apart.

Figure 5.13 displays a projection of different images in the feature space. We notice the direction vectors for classical correlation Softmax training and the disposition of centroids in their method. Experiments are run on MNIST and CIFAR-10. These are *easy* classification tasks. When it comes to ImageNet, more correlation between classes exists semantically. We mentioned in Sect. 5.2.3 that correlation between vectors may be necessary for correct classification. To insist once more on dog breeds: they do share many characteristics and could not be entirely separated in well-trained feature space. Trying to reproduce such results on Imagenet may not be possible.

The idea is interesting and the visualization is really telling of the classical class proximity paradigm. Especially when feature vectors have small norms, an adversarial example might be at a close distance (see the left side of Fig. 5.13). It is however hard to tell whether this could work on a more complex Imagenet.

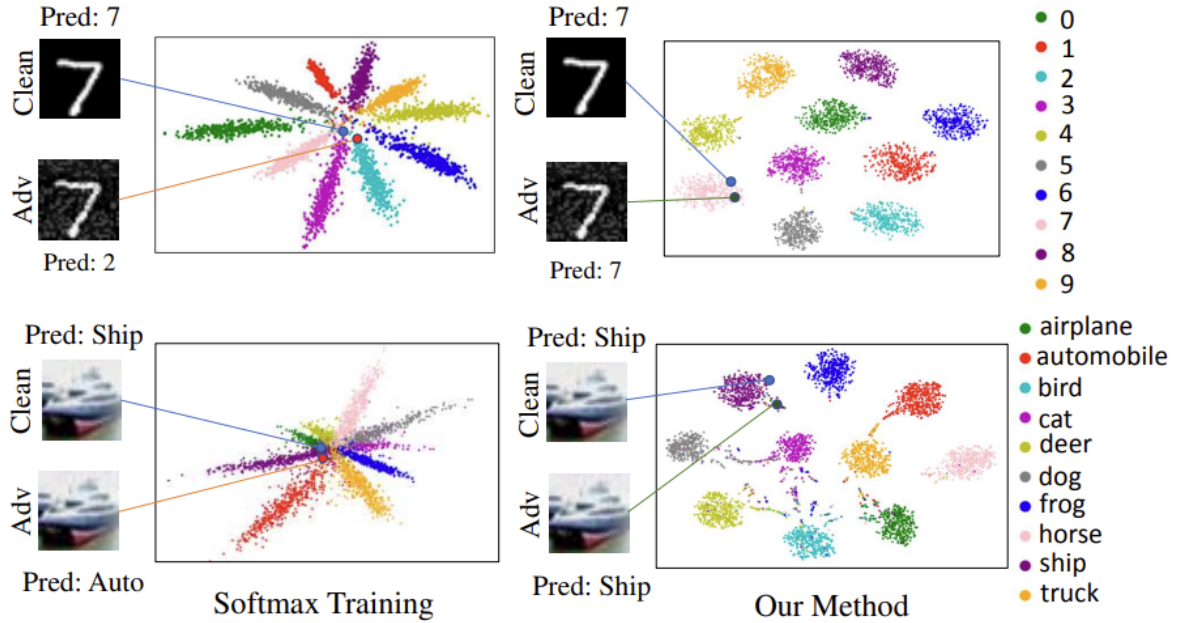


Figure 5.13 – 2D projection of feature vectors of natural images, extracted from [105]. Comparison is given between their method and classical correlation and Softmax training. The projection of natural images and their adversarial counterpart is also displayed.

Other attempts at improving classification and feature space exist. For instance, the works of Pang et al. [109] also changed the Softmax and cross-entropy to a ‘MaxMahalanobis center (MMC)’ loss. Execution is different, but the idea remains the same. Their training aims at creating class centroids. They however acknowledge the additional difficulty of complex Imagenet. They, therefore, propose an improved loss called ‘elastic Max-Mahalanobis center loss’. They however do not provide substantial experimental work on Imagenet to confirm gains in robustness.

Other works have studied modifying models for robustness purposes. For example, Dhillon et al. [34] who suggest a random pruning of activations (preferably lower magnitude ones) at inference. A good aspect of their method is that it does not require additional training and can be performed easily. But once again they do not experiment on Imagenet. We did try to perform pruning on Imagenet models to obtain absolutely no gain until accuracy degradation. Our experimental work was brief, and we do not dismiss this work as potential good practice.

Works that attempt at modifying models to earn robustness are not necessarily the most promising. In terms of proactive defenses, the next section is by far the most active.

## Adversarial Training

The very idea of adversarial training (or retraining) came early with the works of Goodfellow et al. in 2015 [53]. The idea is simple although it exists in different forms. The main idea is to train a model using adversarial examples in the training set. Then the optimization problem lies in minimizing a normal classification loss:

$$\theta^* = \operatorname{argmin}_{\theta} \mathbb{E}_{x \sim \mathcal{X}} \max_{\|x_a - x\| \leq \epsilon} \mathcal{L}(\theta, x_a, y) \quad (5.4)$$

This optimization is done over a constrained attack. Under a given norm,  $\|x_a - x\|$  can not exceed  $\epsilon$ . This is akin to a classical training procedure as previously seen. The other optimization is a little different:

$$\theta^* = \operatorname{argmax}_{\theta} \mathbb{E}_{x \sim \mathcal{X}} \min_{c \neq \hat{c}(x_a)} \|x_a - x\| \quad (5.5)$$

Here the goal is to maximize the margin between classes frontier. The lowest distortion (unconstrained) adversarial example  $x_a$  is pushed further away from  $x$ . Of course, both of these optimizations are done in parallel with *regular* training. This ensures both correct classification and good robustness. In practice, only the first formulation is used for adversarial training.

Ideally, adversarial training is achieved with knowledge of the threat model. Knowing how  $x_a$  is generated ensures better robustness against the attack. The first adversarial training suggested in [53] used for instance FGSM (Sect. 4.3.3). In general, adversarial training with gradient-based attacks transfers well to other attacks. It can be however preferable to use iterative attacks. This is what was done by Madry et al. [98] in which they both introduced and trained against PGD (Sect. 4.3.3). While their robustness is unmatched, it comes at the cost of a great loss of accuracy. Obviously, adversarial training falls right into the No Free Lunch.

Another important work by Kannal et al. [74] brought many novel ideas to the field. First, they minimize the distance between logits rather than reducing classification error. The optimization problem is close to the first one (Eq. (5.4)):

$$\theta^* = \operatorname{argmin}_{\theta} \mathbb{E}_{x \sim \mathcal{X}} \max_{\|x_a - x\| \leq \epsilon} \|\hat{l}(x_a) - \hat{l}(x)\| \quad (5.6)$$

This is Adversarial Logits Pairing (ALP). The adversarial example is constrained and built using a PGD attack. The norm reduced is  $\ell_2$  but they argue that optimization would

work for other norms. Other ideas in this work are interesting. This first one is Clean Logits Pairing (CLP) in which they match logits of *different* images of the same class together. They empirically show that this increased model robustness. The best results were obtained with added Gaussian noise for regularization. And the final interesting result they show is Logits Squeezing in which they simply reduce the norm of logits. They obtain increased robustness as well from this experiment. This last result can be a little surprising when considering the previous illustration of feature space (Fig. 5.13). It seems preferable to have large norms to increase the distance between classes. Their experiments however show otherwise.

Plenty of other works exist in this model *robustification*, but most derive from these. Adversarial training (or sometimes *retraining*) is probably one of the hottest topics defense-wise. Protecting a model from this vulnerability is preferably done upfront. This works also display really good results. Some adversarialy trained models can be attacked only through visible perturbations. To a human that is.

## 5.4 Chapter Conclusion

The efficiency of defenses is sometimes questioned. Some defenses rely on preventing correct gradient calculation, often through randomization. This kind of defense is largely proven ineffective by the works of Athalye et al. [3]. They developed the Backward-Pass Differentiable Approximation (BPDA) that can be used in many scenarii to approximate gradient. The attack is then performed as if the defense was absent and the results are convincing. Other critics [120] claim that more recent works in robustification are simply mitigating overfitting in adversarial training. Finally, some argue that robustification is performed only on small distortion increments [111]. This regime would not be effective against larger increments and would bear no added robustness against black-box attacks.

Still, each form of defense is interesting differently. A combination of both proactive and reactive defenses might be desirable. The synergy between robustness and detection is strong for instance. High-distortion examples are however not desirable for reformation. These techniques are more effective against small signals. But their advantage over others is that they can usually be implemented without further training. If time is a constraint to deploy a system, this situation can be favorable. Otherwise, the first solution might yield the best results.

Finally, No Free Lunch Theorem may not be so rigid. Works have shown that the

trade-off can be shifted towards one or another quantity. Notably, Xie et al. improved the accuracy of EfficientNet [158] as well as its robustness. Either quantity is however only mildly improved. Their work is inspired by adversarial training, they introduce *AdvProp* as a form of adversarial crafting. Overfitting is mitigated and No Free Lunch is defeated.

PART II

# Contributions

---





# **BENCHMARKING MODELS AGAINST WHITE-BOX AND BLACK-BOX ATTACKS**

---

## 6.1 Introduction

A deluge of research papers now propose defenses to block an attacker, and adaptive attacks against these defenses. This is an endless arms race, and systematic benchmarks to evaluate the state of the threat are greatly required.

It is currently extremely difficult to have a clear view on what is truly working in this domain. The cliché is that no two papers report the same statistics for the same attack against the same model over the same image set. This is mostly due to that an attack is an algorithm with many parameters. Its power is indeed highly dependent of these parameters. These values are rarely specified in research papers.

There exist benchmarks in the litterature, such as **ARES** [36], **RobustBench** [29], **RobustVision** [82], **ADBD** [23]. They aim at providing a better understanding of the robustness of image classifiers. Yet, they fall short because their slowness prevents them from tackling large image dataset like ImageNet. They only operate on CIFAR-10 or MNIST. Also, they resort to attacks which are not all state-of-the-art.

This chapter proposes **RoBIC**, to consider these concerns and develop a benchmark tool to measure the robustness of image classifiers in a modern setup.

## 6.2 Difficulties

This section explains the difficulties for setting up a benchmark measuring the robustness of image classifiers.

### 6.2.1 Notation

An attack is a process forging an image  $I_a = \mathcal{A}(I_o, M, \Pi)$ , where  $I_o$  is the original image,  $M$  is the target model, and  $\Pi$  is a set of attack parameters. The ground truth label of  $I_o$  is denoted by  $y_o$ . The boolean function  $\mathbb{1}(I_a, y_o) = [M(I_a) \neq y_o]$  tells whether the attack deludes classifier  $M$  in the untargeted attack scenario: the prediction  $M(I_a)$  is not the ground truth. The distortion between  $I_o$  and  $I_a$  is denoted by  $d(I_a, I_o)$ .

Some statistics like the probability of success and the average distortion are extracted from the adversarial images forged from the test set. They depend on the attack  $\mathcal{A}$  and its set of parameters  $\Pi$ . Therefore, it can not play the role of a measure of robustness of a given model. The first difficulty is to get rid off the impact of parameters  $\Pi$ .

### 6.2.2 The best effort mode

The parameters  $\Pi$  have a huge impact on the power of an attack. For instance, some attacks like FGSM [53], I-FGSM [81], PGD [81] are distortion constrained in the sense that  $\Pi$  is strongly connected to a distortion budget. If this budget is small, the probability of the success of the attack is small. If it is large, this probability is close to 1 but the distortion is too big. Hence, it is hard to find the best setting to make these attacks competitive. Our strategy, so-called ‘best effort mode’, reveals the intrinsic power of an attack by finding the best setting for any image:  $I_a = \mathcal{A}(I_0, M, \Pi^*)$  with

$$\Pi^* = \arg \min_{\Pi: \mathbb{1}(\mathcal{A}(I_0, M, \Pi), y_0)=1} d(\mathcal{A}(I_0, M, \Pi), I_0). \quad (6.1)$$

The best effort mode makes the measurement of the robustness independent from an arbitrary global setting  $\Pi$ . Yet, it is costly in terms of computations. Attacks with few parameters are preferred since the search space is smaller.

### 6.2.3 Worst case attacks

A second difficulty is to make the robustness score independent of the attack. Ideally, we would like to know the worst case attack to certify the robustness of a model. An option proposed by benchmarks RobustVision [82] and ARES [36] is to consider a set of  $J = 11$  attacks as outlined in Tab. 6.1. This is again costly as each image of the test set has to be attacked  $J$  times. Yet, a benchmark happens to be useful if it is fast enough so to assess the robustness of many models. The best effort mode over an ensemble of attacks is out of reach. This is the reason why we need to focus on fast worst case attacks in the sense that they achieve their best effort mode within limited complexity. Section 6.4 focuses on these attacks.

### 6.2.4 Metrics

In this thesis, we present results in two forms: graphs or table. We call *operating curve* the graph showing the success-rate of an attack (y-axis) at a given  $\ell_2$ -distortion (x-axis) such as in Fig. 6.1. We believe this curves to represent fair evaluation of optimized attacks. Distortion is measured in the pixel domain  $\llbracket 0, 255 \rrbracket$  as

$$d(x_a, x_o) := \|x_a - x_o\|_2 / \sqrt{n}, \quad (6.2)$$

Benchmark	Domain	Nb. attacks	Measures	Runtime
RoBIC	$\llbracket 0, 255 \rrbracket^n$	1 WB + 1 BB	Half-distortion $\ell_2$	43s
RobustBench [29]	$[0, 1]^n$	3 WB + 1 BB	Success-Rate for fixed budget ( $\ell_2$ or $\ell_\infty$ )	48s
ADBD [23]	$[0, 1]^n$	1 BB	Distance $\ell_\infty$	360s
RobustVision [82]	$[0, 1]^n$	6 WB + 5 BB	Median Distance $\ell_2$	200s
ARES [36]	$[0, 1]^n$	5 WB + 10 BB	Success-Rate vs Budget ( $\ell_2$ , $\ell_\infty$ or queries)	Too long

Table 6.1 – Benchmarks Comparison. Average Runtimes per ImageNet Image with ResNet50 [98].

where  $N$  is the number of pixels. This is easily interpretable: If for any pixel  $i$ ,  $x_{a,i} = x_{o,i} \pm \epsilon$  (as in FGSM) then  $d(x_a, x_o) = \epsilon$ . The operating curve sums up the impact of an attack against a classifier over a set of test images  $\mathcal{S}_{\text{test}}$  by the following function:

$$d \rightarrow P(d) = \frac{|\{x_o \in \mathcal{S}_{\text{test}} : d(x_a, x_o) \leq d\}|}{|\mathcal{S}_{\text{test}}|}. \quad (6.3)$$

Note that  $P(0) = 1 - \eta > 0$ , where  $\eta$  is the accuracy of the classifier. This is the fraction of original images which are misclassified, hence considered as already adversarial.

Figure 6.1 shows operating curves of several attacks against two well-known classifiers. We choose a complexity budget allowing an attack to perform at its best capacity under the best effort mode: FGSM runs on  $N_{\text{iter}} = 30$  iterations, BP on  $N_{\text{iter}} = 50$ , IFGSM, PGD<sub>2</sub> on  $N_{\text{iter}} = 10 \times 10$ , and CW<sub>2</sub> on  $10 \times 100$  iterations (*i.e.* outer loop  $\times$  inner loop). We formulate three remarks:

- CW<sub>2</sub> requires even more iterations to successfully attack all images against ResNet-50.
- BP achieves by far the best trade-off between complexity, distortion, and success rate.
- These attacks yield adversarial images with unquantized pixel values, and the average distortion is lower than 0.5 for most images. Rounding these values to  $\llbracket 0, 255 \rrbracket$  erases the adversarial perturbation in most pixel positions so that the result is no longer an adversarial image [12]. Chapter 7 studies this thoroughly.

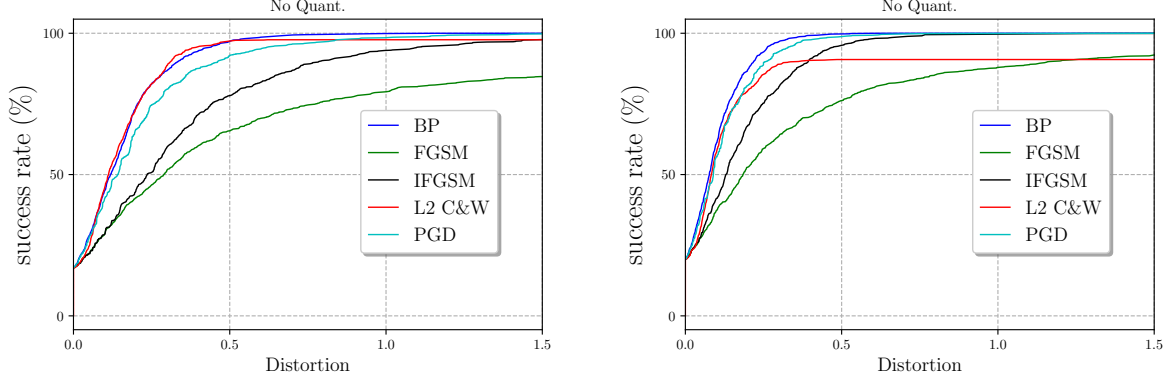


Figure 6.1 – Operating curves of EfficientNet-b0 (left) and ResNet-50 (right) against four attacks in *best-effort mode* and without quantization.

## 6.3 The benchmark

This section justifies the recommendations made in our benchmark and defines the measure of robustness.

**Pixel domain.** Our benchmark is dedicated to image classification. As a consequence, the distortion is defined on the pixel domain: An image  $I$  is defined in the space  $\llbracket 0, 255 \rrbracket^n$  with  $n = 3RC$  pixels for 3 color channels,  $R$  rows and  $L$  columns. Most papers in the field measure distortion after the transformation of the image in a tensor  $x \in \mathcal{X}^n$ . This is a mistake preventing a fair comparison: for most models  $\mathcal{X} = [0, 1]$ , but for some others  $\mathcal{X} = [-1, 1]$  or  $\mathcal{X} = [-3, 3]$ .

We outline that an adversarial image is above all an image, *i.e.* a discrete object  $I_a \in \llbracket 0, 255 \rrbracket^n$ . Again, most attacks output a continuous tensor  $x_a \in \mathcal{X}^n$ , neglecting the quantization. This is a mistake: in real-life, the attacker has no access to  $x_a$ , which is an auxiliary data internal of the model.

**Measure of robustness.** Let us define the accuracy function  $\eta(d) := 1 - P(d)$ . The value  $\eta(0)$  is the classical accuracy of the model over original images. Function  $\eta(d)$  is by construction non increasing and should converge to 0 as the distortion  $d$  increases. After observing many accuracy functions  $\eta$  for different models and attacks, we notice that they share the same prototype:

$$\eta(d) = \eta(0) e^{-\lambda d} \quad \text{with } \lambda \in \mathbb{R}^+. \quad (6.4)$$

Like in nuclear physics, we define the half-distortion  $d_{1/2}$  as the distortion needed to reduce

to half the initial accuracy:

$$\eta(d_{1/2}) = \eta(0)/2, \quad d_{1/2} = \lambda^{-1} \log(2). \quad (6.5)$$

This approximation is verified experimentally with an average coefficient of determination  $R^2$  of 99%. The half-distortion  $d_{1/2}$  will be the keystone of the proposed metric of robustness. A model is then characterized by three separated concepts: its generalization ability  $\eta(0)$  and its robustnesses  $d_{1/2}$  against black-box and white-box attacks.

## 6.4 Fast Attacks

The recent trend in adversarial examples is to design fast attacks with state-of-the-art performances.

### 6.4.1 Fast black-box attacks

In the black-box decision based setup, the attacker can query a model and observes the predicted class. The complexity of the attack is gauged by the number of queries  $K$  needed to find an adversarial image of low distortion.

There has been a huge improvement on the amount of queries recently. Brendel *et al.* report in the order of one million of queries for one image in one of the first decision based black-box BA [14, Fig. 6]. Then, the order of magnitude went down to tens of thousands [24, Fig. 4] [86, Fig. 5] and even some thousands in [118, Fig. 2]. Current benchmarks use others black-box attacks, which are either decision-based (**Square Attack** [1] in **RobustBench** [29] is score-based), or not state-of-the-art (like Gaussian noise in **RobustVision** [82], or BA [14] in **ARES** [36]).

**SurFree** [99] and **RayS** [23] are the only *decision-based* papers with less than one thousand of calls on ImageNet. Yet, **RayS** [23] is designed to minimize the  $\ell_\infty$  distortion, whereas **SurFree** [99] targets  $\ell_2$ . Sect. 6.5 investigates which attack is the best candidate for a fast benchmark.

### 6.4.2 White-Box Attacks and “Best-effort Mode”

White-box attacks are a family of processes parametrized by one or more parameters. One parameter setting may not be adequate from one classifier to another, and for

any image. This explains why experimental results in this literature lack reproducibility. The complexity is usually gauged by the number of gradient computations. Current benchmarks use different white-box attacks: **RobustBench** [29] relies on PGD [81] (with 2 parameters  $\Pi$ ), **RobustVision** [82] use DeepFool [104], and **ARES** [36] CW [21].

In this chapter, we propose the concept of *best-effort mode* enabling a fair comparison of attacks and classifiers. It consists of automatically setting the attack to perform as well as possible on a given iteration budget. It finds the parameters such that the attack is successful and the  $\ell_2$ -norm of the perturbation is minimized. (The attack parameters are usually defined within ranges and the optimization may fail providing adversarial).

The implementation of CW is already optimized and therefore needs no tweaking. Two parameters are still defined by the user: boundaries of research and the number of iterations for iterative attacks.

**FGSM** This attack depends on one parameter  $\epsilon$ . *Best-effort* simply means running a binary search to find the lowest value that successfully crafts an adversarial perturbation.

**iFGSM** This attack also depends on one parameter  $\epsilon$ , for a given number of iterations. We perform the iterative search in the same fashion as previously.

**PGD** Our *best-effort* mode runs a binary search on the radius  $\alpha$ . The iterations budget is equally distributed between the number of iterations and the binary search (e.g. if  $N_{\text{iter}} = 100$ , 10 radii are tested with  $N_{\text{run}} = 10$  iterations each), while  $\epsilon$  is set to  $2\alpha/N_{\text{run}}$ . With this value of  $\epsilon$ , adversarial samples are not projected back onto the  $\ell_2$ -ball of radius  $\alpha$  at least within the first half of the  $N_{\text{run}}$  iterations. Our experiments confirm that this empirical choice is good.

**BP** This attack leads to very good results when set up correctly. It finds an adversarial sample (stage 1) that is then refined (stage 2). For stability, we aim at finishing stage 1 within roughly the same number of iterations for each image. Inspired by Deepfool [104], this is done through a first-order approximation of the loss:

$$L_{\text{adv}}(x_o + u) \approx L_{\text{adv}}(x_o) + u^\top \nabla L_{\text{adv}}(x_o). \quad (6.6)$$



Branching this linearization with (4.20) gives the value of  $\alpha$  canceling the loss within  $\kappa$  iterations:

$$\alpha = \frac{L_{\text{adv}}(x_o)}{\|\nabla L_{\text{adv}}(x_o)\|_2 \sum_{j=1}^{\kappa} \gamma_j}. \quad (6.7)$$

We set  $\kappa = \lceil 0.2 \times N_{\text{iter}} \rceil$  and experimentally observe that stage 1 is more or less completed when desired, leaving  $\approx 0.8 \times N_{\text{iter}}$  iterations to stage 2.

Section 6.5 compares PGD, C&W and PGD to show differences in complexity.

### 6.4.3 Quantization

The adversarial samples are quantified in the pixel domain to create images. The first option considers the quantization as a post-processing not interfering with the attack. The second option performs quantization at the end of any iteration. These options are tested on several black and white box attacks. The quantization will be a post-processing for white-box attacks as detailed in Chap. 7, whereas the second option give better results on black-box attacks.

## 6.5 Experiments

All the attacks are run on 1000 ImageNet images from ILSVRC2012 validation set with size  $n = 3 \times 224 \times 224$ .

### 6.5.1 Selecting the worst case attacks

**Black box attacks:** Figure 6.2 compares the evolution of the half-distortion (6.5) in function of the query amount for four decision-based black-box attacks: **SurFree** [99], **RayS** [23], **GeoDA** [118], and **QEBA** [86]. **SurFree** and **RayS** reach their best effort within 3000 queries, while **QEBA** and **GeoDA** do not since their  $d_{1/2}$  still decrease after 5000 queries. Yet, **SurFree** obtains quantified adversarials with much lower distortion. Therefore, our benchmark only needs this attack. The number of queries is kept at 5000 to be sure to reach the optimal value of  $d_{1/2}$ .

**White box attacks:** Figure 6.3 compares three white-box-attacks in the best effort mode: PGD [81], CW [21], and BP [166] with our trick 6.7. They all reach the same  $d_{1/2}$  when given a large complexity budget. Yet, BP converges faster than the others. Our benchmark uses this version of BP to evaluate the white-box-robustness.

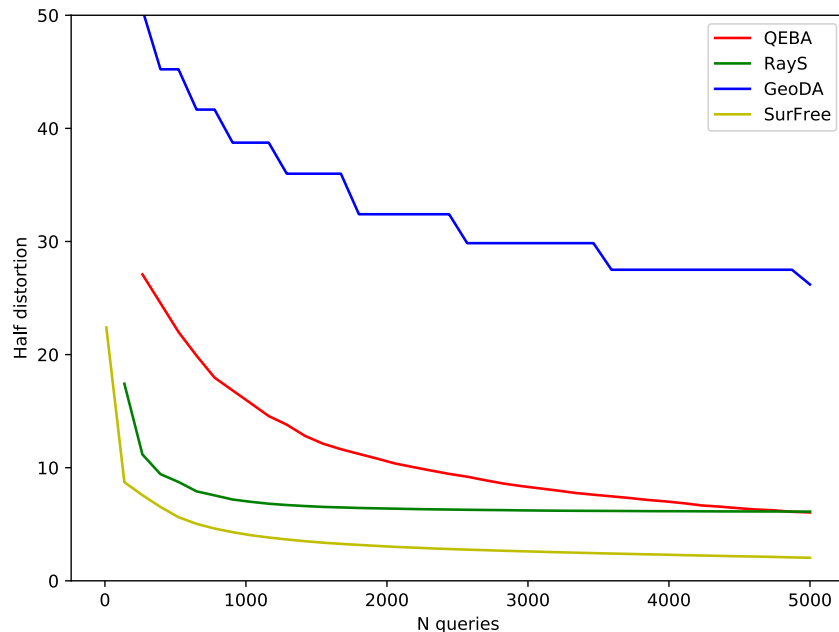


Figure 6.2 – Evolution of  $d_{1/2}$  with the complexity budget for black box setup. Attacks on EfficientNet [142]

### 6.5.2 Comparison with other benchmarks

Table 6.1 lists several benchmarks. Most of them evaluate the robustness as the success-rate under a prescribed  $\ell_2$  or  $\ell_\infty$  distortion budget. But, these budgets are set arbitrarily or even not constant within the same benchmark for **RobustML**. Our half-distortion (6.5) is parameter-free. It returns an accurate, reliable and fair measurement of robustness.

Some benchmarks need many attacks to get a full vision of the robustness: **ARES** [36] and **RobustVision** [82] use 11 attacks. This is too time-consuming. On the contrary, **ADBD** [23] focuses on a single black-box attack, which is indeed outdated. **RobustBench** [29] condenses four attacks in one measure elegantly: for a given image, if the first simple attack does not succeed within the distortion budget, then the second more complex one is launched *etc.* The total runtime heavily depends on the distortion budget. Yet, black-box and white-box attacks use different mechanisms. Our benchmark reports a measurement for each separately.

### 6.5.3 Benchmarking models

Table 6.2 compares standard models from *timm* [157] and *torchvision* [101] libraries. Here are some intriguing results.

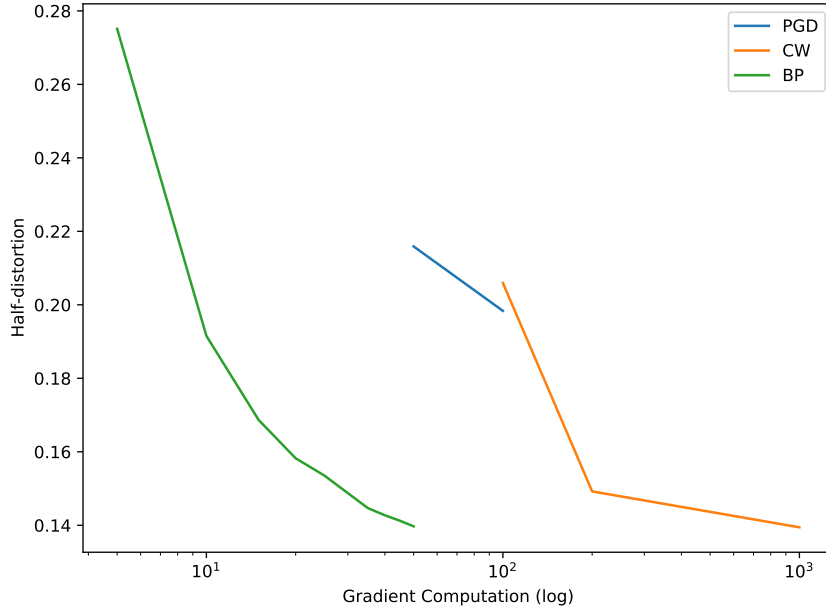


Figure 6.3 – Evolution of  $d_{1/2}$  with the complexity budget for white box setup. Attacks on EfficientNet [142]

**Robustness in white box *vs.* black box.** One does not imply the other. Fig. 6.4 even shows a negative correlation. However, some models escape this rule. For instance, VGG16 is neither robust in black box nor in white box. EfficientNet AdvProp [158] follows the opposite trend. We believe that black-box robustness reveals the complexity of the borders between classes, and white-box robustness indicates how close natural images are from the borders. This highlights the importance of having two different measurements.

**The importance of the training procedure.** There is on average a factor 20 between the half-distortions in white and black box. This factor drops to 4 and 10 for the models adversarially trained: ResNet50 [98], EfficientNet AdvProp [158].

Table 6.2 lists four EfficientNet models sharing the same architecture but different training procedures. Their accuracies are similar but there is up to a factor of 2 between the robustnesses. The same holds on the three variants of Resnet50. The gaps in accuracy and robustness are noticeable with standard models from *timm* [157] and *torchvision* [101]. It is even more visible with adversarial training from [98]: the gain in robustness is impressive but at the cost of a big drop in accuracy.

Model	Parameters (millions)	Accuracy $\eta(0)$	$d_{1/2}$	
			white box	black box
AlexNet [137]	62.38	56.8	0.19	2.17
CSPResNeXt50 [152]	20.57	84.6	0.13	4.48
DualPathNetworks 68b [88]	12.61	83.8	0.08	3.82
MixNet Large [144]	7.33	84.2	0.12	2.96
MobileNetV2 [131]	5.83	80.1	0.09	2.90
ReXNet 200 [31]	16.37	85.4	0.14	3.89
RegNetY 032 [116]	19.44	85.8	0.11	4.94
SEResNeXt50 32x4d [66]	27.56	<b>85.9</b>	0.12	<b>5.01</b>
VGG16 [137]	138.00	74.9	0.09	2.44
EfficientNet AdvProp [158]	5.29	84.3	<b>0.31</b>	4.35
EfficientNet EdgeTPU Small [142]	5.44	82.8	0.15	3.16
EfficientNet NoisyStudent [160]	5.29	82.7	0.19	2.37
EfficientNet [142]	5.29	82.8	0.17	3.56
ResNet50 (torchvision) [61]	25.56	77.9	0.10	2.77
ResNet50 (timm) [61]	25.56	80.5	0.15	4.35
ResNet50 AdvTrain [98]	25.56	60.8	<b>2.56</b>	<b>9.88</b>

Table 6.2 – Benchmark of models with 1.000 ImageNet Images

## 6.6 Chapter Conclusion

This chapter introduced a rigorous benchmark based on a new and independent measurement of robustness: the half distortion. RoBIC is faster than the other benchmarks. This allows to tackle larger images which is more realistic.

In addition to the accuracy, RoBIC gives the black box robustness, and white box robustness. We believe that the first indicates how far away the class boundaries lie from the images whereas the last reflects how curved are the boundaries. As the other benchmarks, two limitations hold: The network must be differentiable to run a white box attack, and deterministic to run a black box attack.

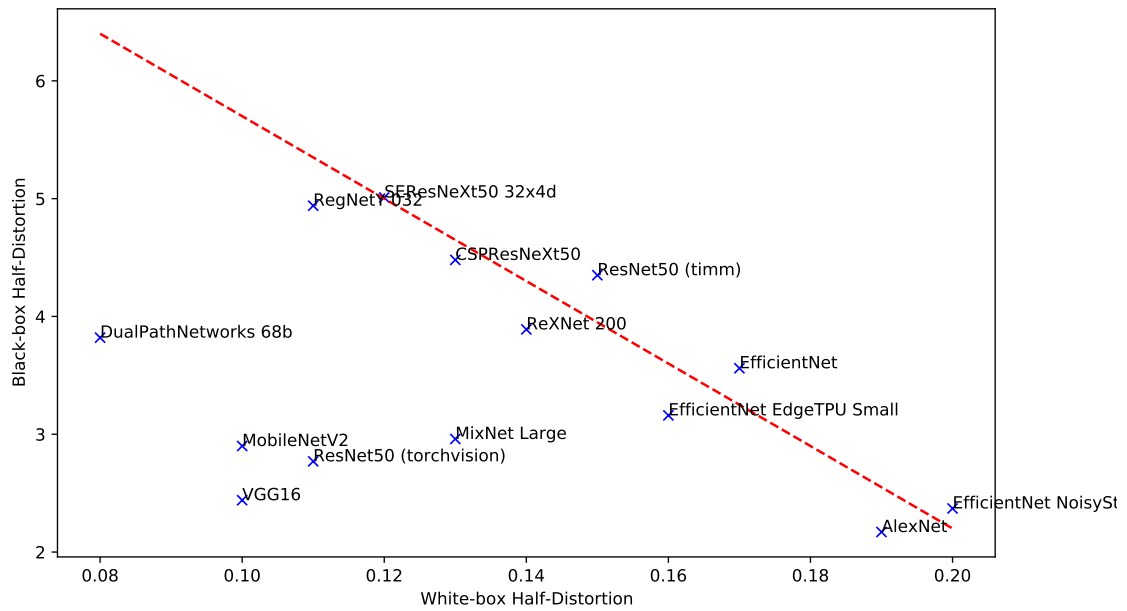


Figure 6.4 – Black-box  $d_{1/2}$  as a function of white-box  $d_{1/2}$ .

# GENERATING QUANTIZED ADVERSARIAL IMAGES

---

## 7.1 Introduction

Recent attacks aim at reducing the distortion (usually measured as  $\ell_2$  or  $\ell_\infty$  norm), increasing the probability of success, and speeding up the process. Even if some learning procedures result in more robust classifiers [98] and some images are harder to attack, recent white-box attacks craft perturbation invisible to the human eye in most cases provided their complexity budget is large enough.

Regardless their complexity, very few attacks consider the specificity of the medium. A raster image is in its digital form a 3-dimensional matrix of integers, such as the PNG image format. JPEG images [151] are coded as integer matrices representing DCT coefficients in different color spaces. To forge an adversarial *image* rather than just a sample encoded in a floating-point tensor, one needs to craft an integral perturbation. Added to the original image, the result must remain within the defined boundaries (*i.e.*  $[0, 255]^n$  with  $n$  the number of pixels in the spatial domain).

Attacks rarely address this constraint. It is sometimes argued that the attack is performed inside the classifier in the white-box setup and thus is not required to be integral. While debatable, we consider this assumption to be very niche. Ironically every attack still clips their samples within the boundaries of a pre-processed image (*i.e.*  $[0, 255]$ ). The white-box setup means that the attacker can replicate the model in his/her garage to prepare an attack that will later on be deployed against a remote classifier service analyzing integral images.

The first idea that comes to mind is to round pixel-wise the crafted perturbation to the nearest integer. This is not working. Perturbations are so small that they are partially erased (set to zero) by rounding. Table 7.1 gives a first insight of this problem. This preliminary experiment is run on 1,000 randomly selected images from ImageNet. The same images are used throughout this chapter. The studied classifier is EfficientNet-b0. Rounding an optimized attack significantly increases the accuracy (decreases the success rate of the attack). An alternative is to round after every step of an iterative attack like PGD<sub>2</sub>. This requires significantly more distortion at every iteration so that the perturbation is not erased by rounding. This is displayed as PGD<sub>2</sub> *round* in Table 7.1: it succeeds in beating classification but generates 64% more distortion than our quantization. Figure 7.1 illustrates further these results.

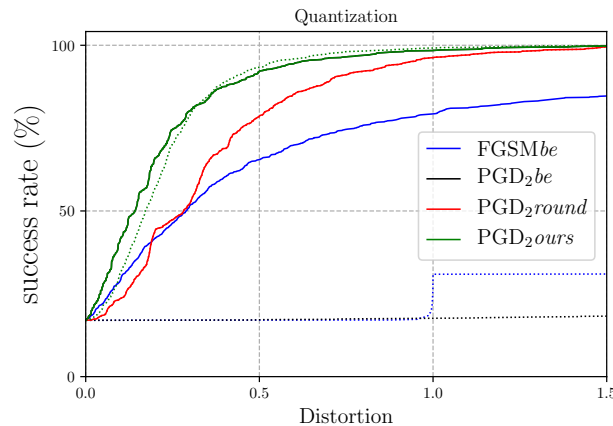


Figure 7.1 – Operating curves of EfficientNet-b0 against FGSM and PGD in *best-effort mode* with floating point (plain) or quantized (dotted) pixel values.

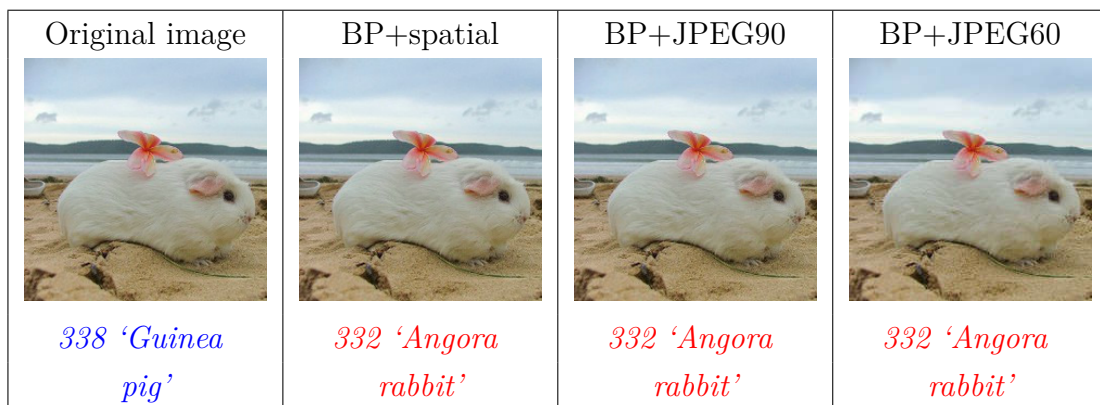


Figure 7.2 – Example of adversarial images quantized with our method. Attacked network is EfficientNet-b0. The predicted label is displayed below.

JPEG compression smoothes the image by cancelling high frequencies, especially at low quality factor. This has little effect on the accuracy of the classifier over natural images while adversarial perturbations are very sensitive to it. Even an attack with an increased distortion budget does not easily fool a classifier after a JPEG compression especially at low-quality factor. Table 7.1 shows that JPEG compressing images forged by FGSM or PGD does not create adversarial examples.

This is the reason why some works propose JPEG as a defense against adversarial attacks [93, 134]. Backward Pass Differentiable Approximation (BPDA [3]) was developed to beat this defense and can be used to create JPEG adversarial images as well. BPDA approximates the JPEG compression by a differentiable transformation. This approxima-



tion is less accurate for low JPEG quality factors. BPDA then fails to attack some images although it generates more distortion than our quantization. Table 7.1 shows that our method JPEG quantizes images which remain adversarial almost surely with a distortion comparable to the compression itself (see section 7.4).

### 7.1.1 Contributions

This chapter proposes a quantization dedicated to adversarial perturbation so that samples can be saved as adversarial images as shown in Fig.7.2. It is a post-process to be used on top of any attack. This method however relies on gradients available in the white-box setup. It is quick in the sense that it typically needs fewer iterations than the attack per se. Our intensive experimental study shows that forging real images, be it in the raster or JPEG format, is not a hard constraint for the attacker when quantization is properly managed. In other words, our quantization adds little to no extra distortion.

Quantization is restricted to a desired range of values providing control over  $\ell_\infty$ -distortion (see Sect. 7.2). We also extend this method to the JPEG domain (see Sect. 7.3). This proves to be challenging since this compression erases high-frequencies typical of an adversarial perturbation. Finally we also propose *best-effort* mode for multiple attacks (see Sect. 6.4.2). This mode finds the best parameter setting for each original image in order to reveal the intrinsic power of an attack. This allows a fair comparison of the attacks.

Note that generating adversarial contents in the quantized domain can also be used in another context. The Euclidean distortion can for example be replaced by a steganographic cost to increase the undetectability of adversarial perturbation (see Chap. 8)). The attacker may also target a network which is not a classifier, like a regression function evaluating the visual quality (see Appendix A).

### 7.1.2 Outlines

Sections 7.2 and 7.3 detail our improved post-processing in both spatial and JPEG domains. Section 7.4 presents experimental results in various scenarios as well as a thorough study of the impact of each parameter. This further motivates our choice of default parameters. Example images are also displayed throughout this section.

Our code is available at [gitlab.inria.fr/bbonnet/adversarial-quantization](https://gitlab.inria.fr/bbonnet/adversarial-quantization)

Table 7.1 – Accuracy (in %) and mean average distortion of FGSM and PGD<sub>2</sub> attacks against EfficientNet-b0 in *best-effort* mode (see section 6.4.2) over 1,000 randomly selected images from ImageNet.

Attack	floats		PNG		JPEG90		JPEG75		JPEG60	
	Acc.	Dist.	Acc.	Dist.	Acc.	Dist.	Acc.	Dist.	Acc.	Dist.
None	83.0	0.00	83.0	0.00	83.0	3.0	81.0	5.2	81.0	6.4
FGSM	11.6	0.33	66.4	0.41	83.0	3.0	81.2	5.2	81.4	6.4
PGD <sub>2</sub>	<b>0.2</b>	0.16	81.7	0.04	83.0	3.0	81.2	5.2	81.4	6.4
PGD <sub>2</sub> <i>round</i>	<b>0.2</b>	0.28	<b>0.2</b>	0.28	-	-	-	-	-	-
PGD <sub>2</sub> <i>BPDA</i>	-	-	-	-	<b>0.6</b>	3.9	6.6	5.8	12.5	6.8
PGD <sub>2</sub> <i>ours</i>	<b>0.2</b>	0.16	<b>0.2</b>	0.17	<b>0.6</b>	3.0	<b>0.6</b>	5.2	<b>1.0</b>	6.5

## 7.2 Smart Quantization in the Spatial Domain

We saw in 4.2.2 the principle of pre-processing  $I_0$  to  $x_0$ . Values in tensor  $x_0$  are encoded as floating-point variables so that their domain is *pseudo-continuous*. Yet, Eq. (4.3) shows that there are only 256 different possible values for a given entry of  $x_0$ . White-box attacks modify  $x_0$  into  $x_a \in [0, 1]^n$  which entries may not equal one of the 256 admissible values. This means that by reversing the preprocessing (4.3), the attacker gets  $x_a \in [0, 255]^n$  whose pixel values may not be integers. For readability, we integrate the preprocessing to our models as the first layer. The sequel focuses on images  $x \in [0, 255]^n$  whereas original images are in  $\llbracket 0, 255 \rrbracket^n$ . In particular,  $x_0 \in \llbracket 0, 255 \rrbracket^n$ .

Assume that an attack has forged the adversarial sample  $x_a \in [0, 255]^n$ , which is not quantized, *i.e.* pixel values are a priori not in  $\llbracket 0, 255 \rrbracket^n$ . This section presents our mechanism carefully quantizing the pixel values to keep adversarial images adversarial.

### 7.2.1 Problem Statement

Our mechanism casts  $x_a \in [0, 255]^n$  to  $x_q \in \llbracket 0, 255 \rrbracket^n$ . Its goal is to solve the optimization problem defined in (4.12) with the additional integral constraint:  $x_a$  is eventually replaced by  $x_q \in \llbracket 0, 255 \rrbracket^n$ . Since we are working in a white-box environment, our method can rely on the following quantities:

- the original image  $x_0 \in \llbracket 0, 255 \rrbracket^n$ ,
- the unquantized adversarial image  $x_a \in [0, 255]^n$ ,
- the adversarial loss  $L_{\text{adv}}(x)$  and its gradient  $\nabla L_{\text{adv}}(x)$  (see Eq. (4.15)).

We introduce the following weak signals:

$$u := x_a - x_0, \quad (7.1)$$

$$q := x_q - x_a. \quad (7.2)$$

The quantization noise  $q$  plays the central role in our approach. We redefine the distortion and the loss functions w.r.t. this variable:

$$D(q) := \|x_q - x_0\|^2 = \|u + q\|^2 \quad (7.3)$$

$$L(q) := L_{\text{adv}}(x_q) = L_{\text{adv}}(x_a + q). \quad (7.4)$$

There is obviously a trade-off between these two quantities. For instance, the choice

$$q^\dagger = \arg \min D(q) = -u \quad (7.5)$$

cancels the perturbation and makes  $x_q = x_o$  not adversarial.

Finding the adversarial image  $x_q$  minimizing distortion  $D(q)$  can be expressed as:

$$\min_{q \in \mathcal{Q}, L(q) < 0} D(q), \quad (7.6)$$

where  $\mathcal{Q}$  is the set of admissible solutions. Remark that  $x_q - x_o = q + u \in \mathbb{Z}^N$  since it is the difference of two integer vectors. This implies that  $q \in \mathcal{Q} \subset \mathbb{Z}^n - u$ , *i.e.* the grid  $\mathbb{Z}^n$  shifted by translation  $-u$ . For instance, quantizing by rounding the perturbation gives

$$q = [x_a] - x_a = [x_o + u] - (x_o + u) = [u] - u, \quad (7.7)$$

where  $[u]$  is the closest integer value of  $u$  component-wise.

We add the other constraint of  $q$  being of limited amplitude. We introduce a new parameter  $d_f \in \mathbb{N}_*$ , so-called *degree of freedom* which reflects the number of choices per component:

$$q_i + u_i = \begin{cases} [u_i] & \text{if } d_f = 0 \\ [u_i] \text{ or } \lceil u_i \rceil & \text{if } d_f = 1 \\ [u_i] - 1, [u_i], \text{ or } [u_i] + 1 & \text{if } d_f = 2 \\ \dots & \end{cases}$$

The case  $d_f = 0$  amounts to rounding the perturbation and there is no freedom to choose another integer. This implies that the  $\ell_\infty$ -norm of the quantization noise is bounded by  $\|q\|_\infty \leq 1/2$ . In the general case, this norm is bounded by  $\|q\|_\infty \leq (d_f + 1)/2$ . The case  $d_f = \infty$  means that the perturbation is quantized to integers but there is no control on the norm  $\|q\|_\infty$ .

In the end, the set of admissible quantization noises is defined by the product space  $\mathcal{Q} = \otimes_{i=1}^N \mathbb{Q}_i$  with

$$\mathbb{Q}_i = \begin{cases} \{[u_i] - \frac{d_f}{2}, \dots, [u_i] + \frac{d_f}{2}\} - u_i & \text{if } d_f \text{ is even,} \\ \{[u_i] - \frac{d_f-1}{2}, \dots, [u_i] + \frac{d_f-1}{2}\} - u_i & \text{if } d_f \text{ is odd.} \end{cases} \quad (7.8)$$

Note that the number of admissible solutions is exponential with  $N$ :  $|\mathcal{Q}| = (d + 1)^N$ .

We now assume that  $\forall q \in \mathcal{Q}$ ,  $q$  is small enough to make a first order approximation of the loss:

$$L(q) := L_{\text{adv}}(x_q) \approx L_{\text{adv}}(x_a) + q^\top g, \quad (7.9)$$

where  $g := \nabla L_{\text{adv}}(x_a)$ . For instance, the choice

$$q_i^\dagger = \begin{cases} \min(\mathbb{Q}_i) & \text{if } \text{sign}(g_i) > 0 \\ \max(\mathbb{Q}_i) & \text{if } \text{sign}(g_i) < 0 \end{cases} \quad (7.10)$$

minimizes the first order approximation of  $L(q)$ . For  $d_f$  large enough, this certainly ensures that  $L(q) < 0$  and  $x_q$  is adversarial but the distortion is large. The solution to problem (7.6) can consequently be seen as a compromise between (7.5) minimizing the distortion and (7.10) minimizing the loss.

## 7.2.2 Solution

The solution of (7.6) is given by a Lagrangian formulation. Define the following functional:

$$J_\lambda(q) := D(q) + \lambda L(q), \quad (7.11)$$

where  $\lambda \in \mathbb{R}_+$  is the Lagrangian multiplier balancing adversariality and distortion quantities. Suppose we know how to efficiently minimize that functional by  $q_\lambda^* := \min_{\mathcal{Q}} J_\lambda(q)$ ,  $\forall \lambda \in \mathbb{R}_+$ . The expected behavior along  $\lambda$  is for  $L(q_\lambda^*)$  to *decrease* while  $D(q_\lambda^*)$  *increases* (see Fig. 7.3). For instance,

1. When  $\lambda = 0$ , all importance is given to  $D(q)$ . This results in a distortion-based quantization  $q^\dagger$  erasing the perturbation  $u$  as seen in (7.5).
2. When  $\lambda \rightarrow +\infty$ , all importance is given to  $L(q)$ . This results in a gradient-based quantization  $q^\ddagger$  of (7.10).

Since the distortion strictly increases with  $\lambda$ , the optimal solution of problem (7.6) is then  $q_{\lambda^*}^\star$  where  $\lambda^* = \min\{\lambda : L(q_\lambda^\star) < 0\}$ . We practically compute this optimal solution in a two step approach.

### Minimizing the functional

Finding the minimum of  $J_\lambda$  is difficult, except if we rely on approximation (7.9), then we can write that

$$J_\lambda(q) \approx \|u + q\|^2 + \lambda L_{\text{adv}}(x_a) + \lambda q^\top g \quad (7.12)$$

is convex and thus is minimized when  $\nabla J_\lambda(q) = 0$ . This happens for  $q = \tilde{q}_\lambda$ , where

$$\tilde{q}_\lambda := -\frac{\lambda}{2}g - u. \quad (7.13)$$

Yet this solution is not admissible because it does not belong to  $\mathcal{Q}$  a priori. We rewrite the approximation (7.12) as

$$J_\lambda(q) \approx \|q - \tilde{q}_\lambda\|^2 + \frac{\lambda^2}{4}\|g\|^2 + \lambda(g^\top \tilde{q}_\lambda + L_{\text{adv}}(x_a)) \quad (7.14)$$

to outline that the minimizer on  $\mathcal{Q}$  is just its closest element to  $\tilde{q}_\lambda$ . This amounts to first quantize  $\tilde{q}_\lambda$  onto  $\mathbb{Z}^N - u$  and then clip:

$$q_{\lambda,i}^\star = \text{clip}_{[\min(\mathbb{Q}_i), \max(\mathbb{Q}_i)]}([- \lambda g_i / 2] - u_i). \quad (7.15)$$

### Finding the optimal $\lambda^*$

The relaxation of the integral constraint and the linearisation of the loss provides a first approximation of  $\lambda^*$ . Inserting (7.13) in (7.9) yields:

$$\lambda_c := \frac{2(L_{\text{adv}}(x_a) - u^\top g)}{\|g\|^2}. \quad (7.16)$$

We find the value of  $\lambda^*$  by looking around  $\lambda_c$ . Similarly to our previous work [12], we run a line search in the interval  $[0.01\lambda_c, 100\lambda_c]$ . For every value tested, we compute the optimal perturbation (7.15), add it to  $x_a$ , and submit this to the classifier. If it is adversarial, then the value of  $\lambda$  is decreased. It is increased otherwise. In other words, the computation of the best quantization noise  $q_\lambda^*$  given  $\lambda$  relies on the linear approximation (7.9), but the finding of  $\lambda^*$  implies to evaluate the classifier.

## 7.3 Smart Quantization in the JPEG Domain

The JPEG file format represents an image as a 3-dimensional tensor of scaled DCT coefficients quantized to integers. Extending our quantization to the DCT domain enables us to craft JPEG adversarial images.

### 7.3.1 JPEG Compression

The JPEG compression is schematically done in four stages (excluding entropic source coding). A  $RGB$  image (Red, Green, Blue) is linearly converted to  $YC_bCr$  (Luminance, blue-Chroma, red-Chroma). This linear transform ensures that all values lie in the range  $[0, 255]$ . Then each channel undergoes the  $8 \times 8$  block DCT-transform. The resulting DCT coefficients are finally divided by quantization steps which depend on their frequency bin and the quality factor. A lower quality factor increases the quantization steps s.t. the following quantization loses more information.

This pipeline is linear and thus can be summarized by  $X = Jx + C$ , where  $X \in \mathbb{R}^n$  stores the scaled DCT coefficients and  $C$  is a constant vector encoding the shift in the conversion  $RGB$  to  $YC_bCr$ . We have supposed here that  $n = 3LC$  where the numbers of columns  $C$  and lines  $L$  are multiple of 8. The matrix  $J \in \mathbb{R}^{n \times n}$  encodes the color conversion, the block DCT, and the division by the quantization steps. It is cumbersome to express it due to the flattening of images in  $n$  dimensional vectors. The main properties are that  $J$  is invertible and that it is not an isometry, *i.e.*  $\|Jx\| \neq \|x\|$  in general. This is due to the scaling with the quantization steps but also to the color domain conversion [169].

Vectors in this JPEG domain are denoted with capital letters:  $X_o$ ,  $X_a$ ,  $X_q$ ,  $U$ , and  $Q$

with:

$$U = X_a - X_o = J(x_a - x_o) = Ju, \quad (7.17)$$

$$Q = X_q - X_a = J(x_q - x_a) = Jq. \quad (7.18)$$

The scenario is the following: As in the previous section, an attack forges  $x_a$  and we have to craft its JPEG version. This amounts to convert it in the JPEG domain,  $X_a = Jx_a + C$ , and to quantize these coefficients with care so that the image remains adversarial. Note that  $X_o = Jx_o + C$  is a priori not an element of  $\mathbb{Z}^N$ , unless the original image was already quantized in the JPEG domain.

### 7.3.2 Quantization

We write the problem by focusing on the quantization noise  $Q$  in the JPEG domain. Since  $x_q = J^{-1}(X_q - C)$ , Eq. (7.9) is written as:

$$\begin{aligned} L_{\text{adv}}(x_q) &= \mathcal{L}(J^{-1}(X_a + Q - C)) = \mathcal{L}(x_a + J^{-1}Q) \\ &\approx L_{\text{adv}}(x_a) + (J^{-1}Q)^\top g, \end{aligned} \quad (7.19)$$

where  $g$  is the gradient of the loss function in the pixel domain. As for the Euclidean distortion, we have

$$\|x_q - x_o\|^2 = \|u + q\|^2 = \|u + J^{-1}Q\|^2. \quad (7.20)$$

Finally, the Lagrangian functional defined in (7.11) becomes:

$$D(J^{-1}Q) + \lambda L(J^{-1}Q). \quad (7.21)$$

Following the same reasoning as in the spatial domain, the minimum of this functional when relaxing the quantization constraint amounts to set  $J^{-1}Q$  to  $\tilde{q}_\lambda$  given in (7.13). Equivalently:

$$\tilde{Q}_\lambda = -\frac{\lambda}{2}Jg - U. \quad (7.22)$$

Yet, this time the rounding is done with respect to  $X_a$  since we need  $X_a + Q$  to be an integral vector:

$$Q_\lambda^* = \left\lceil -\frac{\lambda}{2}Jg - U + X_a \right\rceil - X_a. \quad (7.23)$$

Like in the spatial domain, this value is clipped to belong to the set  $\mathcal{Q}$ , defined in (7.8) replacing  $u$  by  $U$ . Note that if the original image is in JPEG format with the same quality factor so that  $X_o$  is an integer vector, then  $Q_\lambda^* = \left\lceil \frac{-\lambda}{2} Jg \right\rceil - U$ , and we recover a quantization similar to (7.15).

## 7.4 Experimental Work

### 7.4.1 Implementation Details and Setup

We make the following implementation choices.

One can either implement the transformation  $J^{-1}$  as a preprocessing layer like previously discussed in Sect. 4.2.2. This allows to directly feed the classifier with JPEG images. Attacks also naturally adapt to this new object as the gradient *back-propagates* through the transformation layer. Yet, this approach makes the attack domain-specific. Our choice of design is to implement our quantization separately on top of any attack. Our method forges JPEG images from a *spatial* adversarial sample  $x_a$  resulting from an attack on  $x_o$ .

Our implementation builds on the Python library `Pillow` to be as close as possible to the official JPEG standard. Two differences remain: JPEG may apply a sub-sampling on the  $C_b$  and  $C_r$  channels. Taking into account sub-sampling is straightforward with back-propagation and auto-differentiation. For the sake of simplicity, we work on JPEG images without sub-sampling and the color channels have all the same size. JPEG may apply clipping when converting from one color domain to another. These border effects produce small information losses. We do not apply this lossy step to keep the transformation linear.

The experiments use 1,000 PNG versions of images from the validation set of Imagenet ILSVR 2012 [127]. Unless stated otherwise, the attacked classifier is EfficientNet-b0 [142]. EfficientNet in its b0 configuration is a recent and lightweight classifier that achieves high accuracy on ImageNet. Our previous work [12] shows that distortion created by quantization is proportional to the distortion created by the attack. We thus choose BP to be the default attack. It performs well with few iterations in its *best-effort* setup as seen in Fig. 6.1. The research on the optimal value of  $\lambda$  is done by default over 10 iterations. Finally, both spatial and JPEG quantization are run by default with degree of freedom  $d_f = 1$  unless specified otherwise.

The protocol is the following. For the spatial domain,  $x_q = x_o$  if the original image is already misclassified, otherwise BP produces  $x_a$  that our method quantizes to  $x_q$ . For the



JPEG domain,  $x_o$  is first converted in the JPEG format. If this triggers a misclassification, then this JPEG version of  $x_o$  is the adversarial image  $X_q$ . Otherwise, BP produces  $x_a$  from the JPEG original that our method quantizes to  $X_q$ .

Operating curves in both domains are displayed as follow:

- Distortion is calculated w.r.t. the original spatial image.
- Misclassified original images in each domain are considered as already adversarial at null distortion.

Note that compressed JPEG images do not have a null distortion since they differ from the original spatial image. For the sake of clarity, we however consider they do. This choice is further motivated in Section 7.4.3.

### 7.4.2 Investigations on the Search of $\lambda^*$

Figure 7.3 shows the behavior of the adversarial loss and the distortion as functions of  $\lambda$ , for one image in the spatial and the JPEG domains. To plot these curves we use the quantized (7.15) (resp. (7.22) for JPEG) and unquantized solution (7.23) (resp. (7.13)). The same behavior is observed on other images through other classifiers up to a change of ranges of values.

Without quantization, the distortion is the same in both domains and it strictly increases w.r.t.  $\lambda$  as predicted by (7.13) or (7.22). With quantization and clipping (with  $d = 1$ ), the distortion increases much more in the JPEG domain because of the coarser quantization steps in the high-frequency bins. It also does not start at 0 but at the distortion induced by the regular JPEG compression of the original image.

In the spatial domain, the adversarial losses (with or without quantization and/or clipping) are well-approximated by (7.9) for small perturbation, *i.e.* when  $\lambda$  is small. In particular, they converge to  $L_{\text{adv}}(x_o)$  when  $\lambda \rightarrow 0$ . The linear approximation is useful for predicting when the loss cancels. Adding quantization and clipping constrains the problem and we observe that  $\lambda^* > \lambda_c$ . This implies a stronger distortion. Of course, the linear approximation is very wrong when predicting losses below  $-1$ .

The picture is less clear in the JPEG domain. The approximation holds true in the beginning and until  $L_{\text{adv}}(X_q)$  reaches 0. The approximation  $\lambda_c$  remains extremely relevant. However  $L_{\text{adv}}$  sometimes becomes non-monotonic as  $\lambda \rightarrow \infty$  because of the strong distortion.

For this reason, our search of  $\lambda^*$  slightly differs. It starts from  $\lambda_c$  given in (7.16). This is the same value for both spatial and JPEG domains. In the spatial domain, a line search

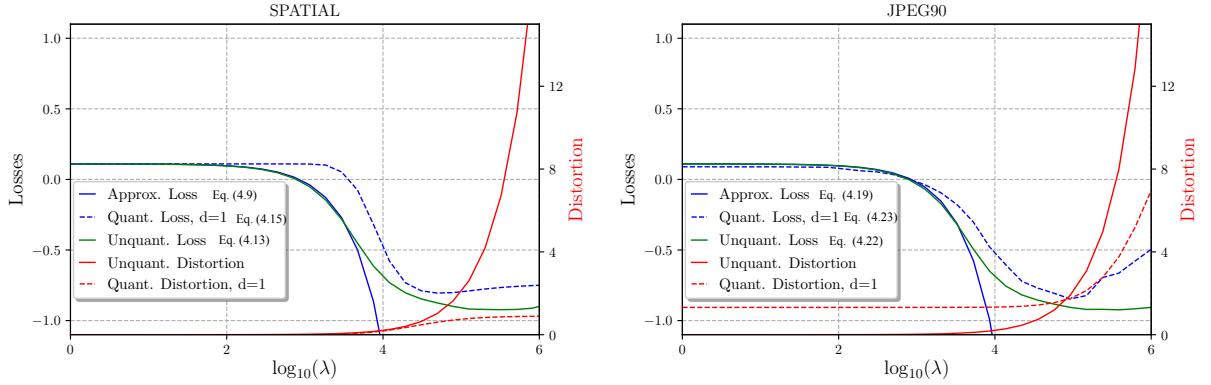


Figure 7.3 – Comparing the approximated loss (7.9) (resp. (7.19)) with the loss without rounding:  $L(\tilde{q}_\lambda)$  in (7.13) (resp.  $L(J^{-1}\tilde{Q}_\lambda)$  in (7.22)) and the loss with quantization  $L(q_\lambda^*)$  in (7.15) (resp.  $L(J^{-1}Q_\lambda^*)$  in (7.23)) as a function of  $\lambda$  in the (left) spatial domain, resp. (right) JPEG90. Distortion is also displayed with scale on the right.

within  $[0.01 \cdot \lambda_c, 100 \cdot \lambda_c]$  works well because the loss is almost monotonically decreasing. In the JPEG domain, the loss is less predictable and we instead sample  $n$  values in this interval:

$$\lambda_i : = \lambda_c \cdot 10^{\alpha_i}, \quad (7.24)$$

$$\alpha_i := 2 \cdot \frac{n - 2i}{n}. \quad (7.25)$$

The lowest value tested that verifies  $L_{adv} < 0$  is necessarily the best since distortion strictly increases with  $\lambda$ .

It is expected that the line search in the spatial domain is more efficient than the uniform sampling in the JPEG domain. This is indeed illustrated by Fig. 7.4. Quantization in the spatial domain converges faster thanks to the line search. However, for both approaches, searching for  $\lambda^*$  with  $n = 10$  steps is sufficient. Note that each step only makes a forward pass through the classifier network. In comparison running BP makes  $N_{\text{iter}} = 50$  forward and backward passes. Our quantization is thus faster than the attack. It gets slowed down by the DCT transform in JPEG domain however.

### 7.4.3 Impact of Quality Factor in JPEG Domain

The quality factor of JPEG determines the values of the quantization steps applied on the DCT coefficients. A lower quality factor leads to a bigger loss in information and

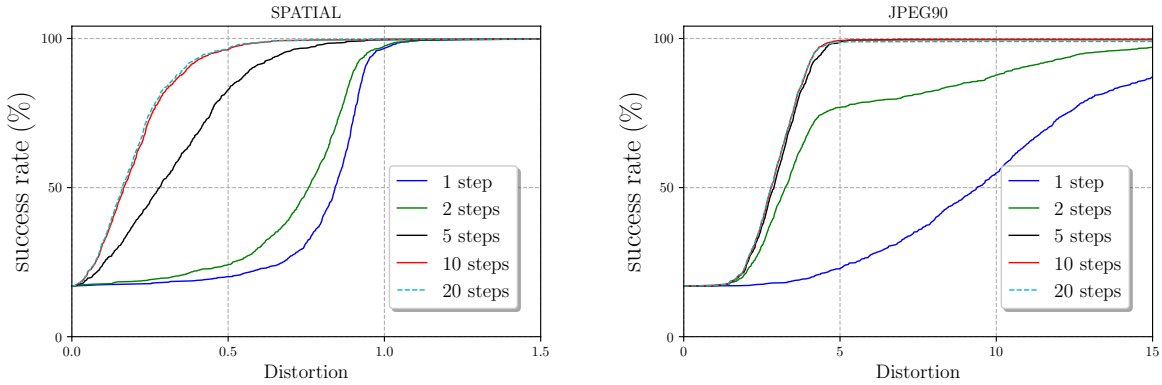


Figure 7.4 – The impact of the number of tested values of  $\lambda$  on the operating curve. EfficientNet, BP,  $d_f = 1$ , spatial (left) and JPEG90 (right).

degradation of the image. This is especially true for high frequencies of the image which are coarsely quantized. The gradients computed during attacks look like noisy patterns and the resulting adversarial perturbation is thus in the high-frequency range. Its distortion needs to be amplified to preserve the adversarial property of the perturbation at a lower quality factor.

Figure 7.8 shows operating curves for adversarial images crafted for different JPEG quality factors (plain curves) which confirm this last statement. For the sake of comparison, the distortion of the JPEG compression on the original images is also displayed as a cumulative sum over all the 1,000 images (dashed curves). We observe that our adversarial quantization returns a distortion very close to regular JPEG compression. This leads to the interesting result that some adversarial images quantized in JPEG often have the same distortion (or even *less* in few cases) than just the original image compressed. Figure 7.13 illustrates this result with JPEG75. On three of the four examples displayed, distortion of the adversarial image is equal or very close to the distortion of the compressed image. In other words, a perturbation being compliant to JPEG is not a strong constraint for the attacker as it is almost distortion-free.

The price to pay is a small extra complexity thanks to our method. This is necessary as the JPEG compression alone is not working as an attack. Table 7.1 indicates that only 17.0% (JPEG90 and JPEG100) or 19.0% (JPEG75 and JPEG60) of compressed original images are misclassified.

We plot the same operating curve as previously considering JPEG compression as an attack on Fig. 7.8 (dash plots). Its discloses which images were naturally adversarial

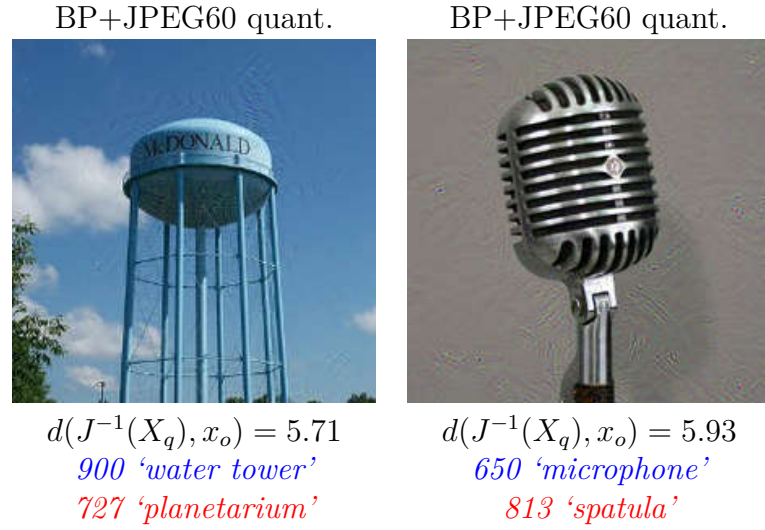


Figure 7.5 – Visible artifacts on images quantized as JPEG60. Predicted class is displayed in red, ground-truth in blue.

after JPEG compression and which ones needed to be attacked and quantized by our method. There is no correlation between the distortion due to JPEG and the chance that it succeeds as an attack. Indeed, most of these images are already misclassified in PNG and still adversarial once in JPEG.

Figure 7.6 shows the operating curve w.r.t. to the Euclidean distance from the original image *compressed to JPEG format*. These curves start at a success rate of 17.0% (resp. 19.0%) since a null distortion corresponds to misclassified original JPEG images. Except for these specific images, our method forges an attacked JPEG image different than its original JPEG version although both of them are equivalently far away from the original PNG image.

Distortion is mostly imperceptible when the quality factor is high. It does start to be noticeable with JPEG75. Fig. 7.13 displays for example at the last row some little artifacts at the bottom of the lighter which are not typical from a JPEG compression. On JPEG60, quantization artifacts are more frequent and important. Figure 7.5 displays two examples. These patterns are especially visible on smooth image regions usually not affected by JPEG compression. Quantized attacks remain imperceptible when the image is highly textured.

We conclude this section by the following remarks.

- The classifier is very robust to JPEG compression alone.
- For any tested quality factor, almost all images are successfully attacked and quan-

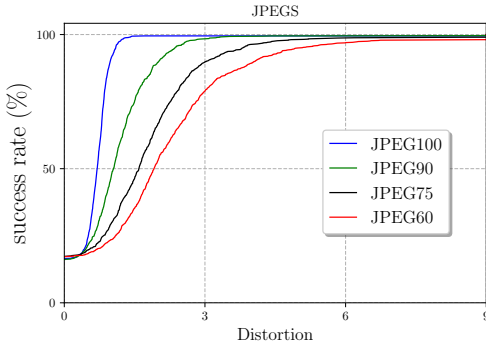


Figure 7.6 – Operating curve of Efficientnet-b0 against BP + JPEG quantization ( $d_f = 1$ ). Distortion is calculated w.r.t. to the original image compressed in JPEG domain  $X_o$ .

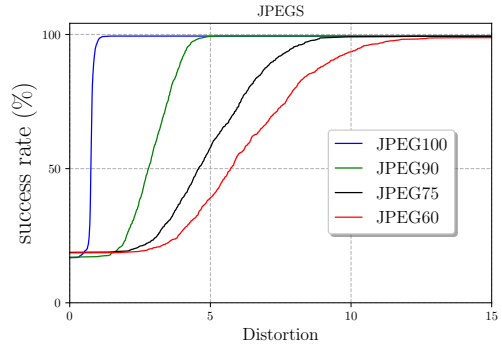


Figure 7.7 – Operating curve of Efficientnet-b0 against BP + JPEG quantization ( $d_f = 1$ ). Distortion is calculated w.r.t. to original spatial image  $x_o$ . Images that are already adversarial after compression are considered to have null distortion for readability.

tized. A few images were unsuccessful at low quality factors (1% for JPEG60).

- Little distortion is added to the inherent distortion of the JPEG compression.
- Yet, at low quality factor, the adversarial perturbation is not typical from JPEG compression artifact.

Figure 7.7 summarizes the results but this time assuming that misclassified images have null distortion. We consider this procedure to carry out the main information about the efficiency of the attack and we use this setup to display the following results.

#### 7.4.4 Impact of the Degree of Freedom

A previous iteration of this work [12] considered only quantization in the spatial domain with  $d_f = 1$ . The method here is more general with higher degrees of freedom. Is that useful?

When  $d_f = 0$ , quantization is equivalent to rounding the perturbation samples to the nearest integer. Figures 7.10 shows that this mostly leads to unsuccessful attack. Indeed, the perturbation is a weak signal partially destroyed by rounding. The success rate converges to approximately 20% for the different JPEG quality factors, close to the proportion of misclassified original images (see Fig. 7.8). This demonstrates the robustness of the classifier against JPEG.

When attacking EfficientNet-b0,  $d_f = 1$  seems enough to quantize almost every image

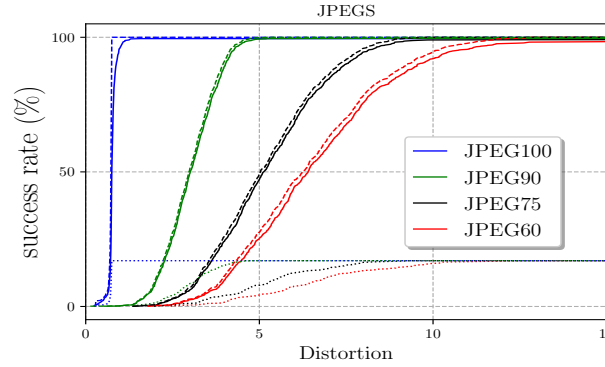


Figure 7.8 – The operating curve of Efficientnet-b0 against BP + JPEG quantization with  $d_f = 1$  (plain) and against JPEG compression alone (dot). For the sake of comparison, the cumulative distribution function of the distortion due to JPEG compression is also displayed (dashed). Distortion is measured from the original spatial image  $x_o$ .

in both domains and the benefit of increasing  $d_f$  seems rather low. A higher value of  $d_f$  is not necessarily a better choice. This is particularly visible in Fig. 7.10 with the quality factor 100.

The reason lies in the metrics used. The distortion (6.2) is proportional to the  $\ell_2$ -norm, *i.e.* the square root of the squared-difference, summed over all pixels. Adding +2 on one coefficient thus costs 2 whereas adding +1 on two coefficients costs  $\sqrt{2}$ . The degree of freedom constrains the  $\ell_\infty$  norm of the total perturbation  $u + q$  (or  $U + Q$  in the JPEG domain). This clipping increases the spreading of the perturbation over all the coefficients: since the coefficients with large gradient amplitude can not host a large perturbation,  $\lambda$  increases to compensate this clipping on the other components. This more uniform distribution of the perturbation energy over the coefficients yields a lower Euclidean distortion but also a lower perceptual impact. Figure 7.9 shows two images quantized with two different quality factors each quantized with  $d_f = 1$  and  $d_f = \infty$ . Their Euclidean distortion w.r.t. the original image is similar but the artefact are more visible for  $d_f = \infty$  (quantization but no clipping). We notice that this holds for any image.

#### 7.4.5 Quantization on Different Classifiers and Attacks

This study considers four recent classifiers: EfficientNet-b0 [142] and its adversarially trained counterpart EfficientNet-b0-advprop [158]; RegNetX-032 [116], and the older ResNet50 [61].

Figure 7.12 shows two images misclassified by two different classifiers. It is interesting

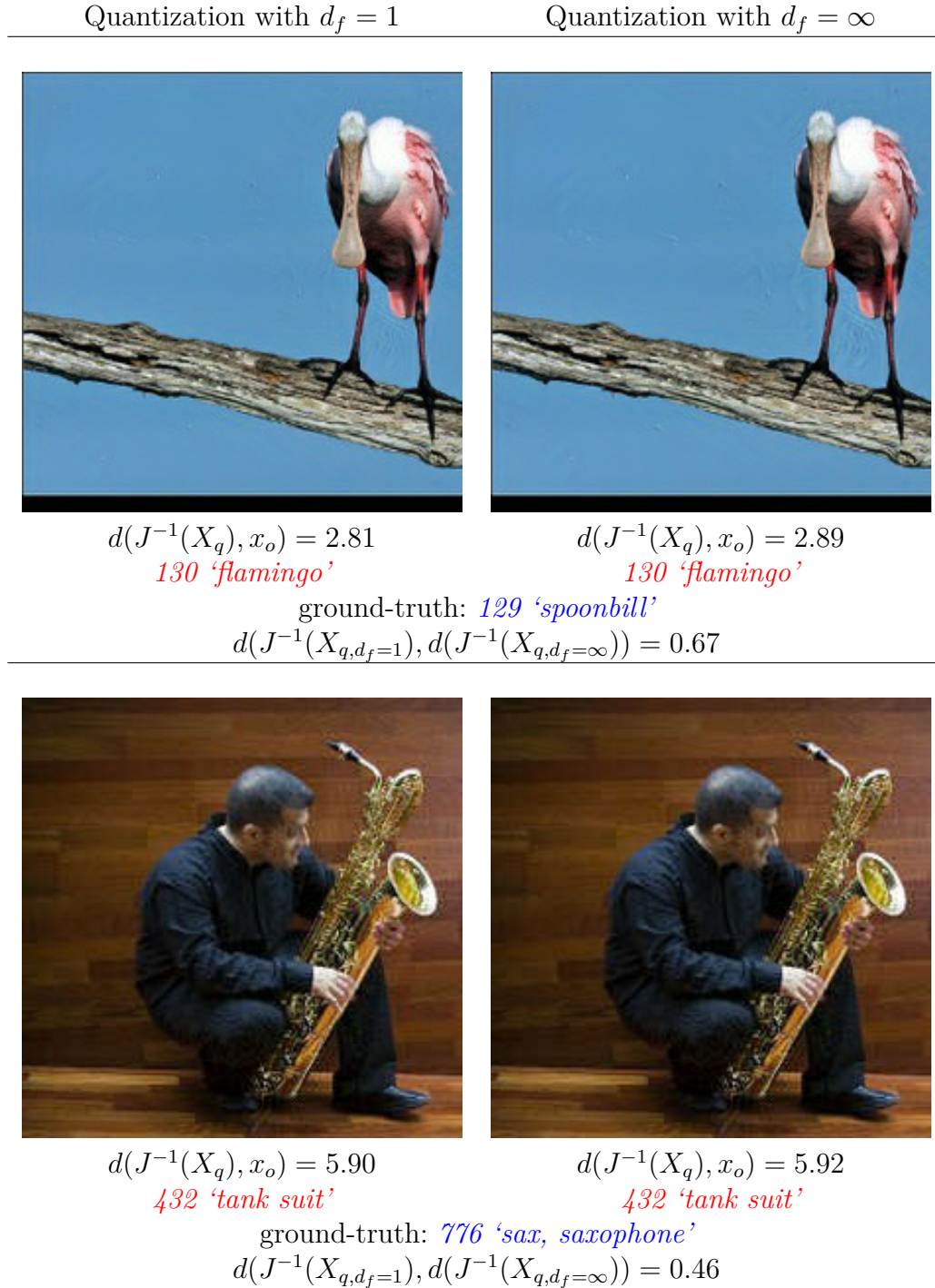


Figure 7.9 – Visual artifacts for two adversarial images on JPEG90 (top) and JPEG60 (bottom) with and without clipping. Visual artifacts on JPEG60 are due to the compression factor. Distortion is naturally high. JPEG 90 however represents low compression. Notice the non-smooth sky around the flamingo. Effect is especially visible for  $d_f = \infty$ .

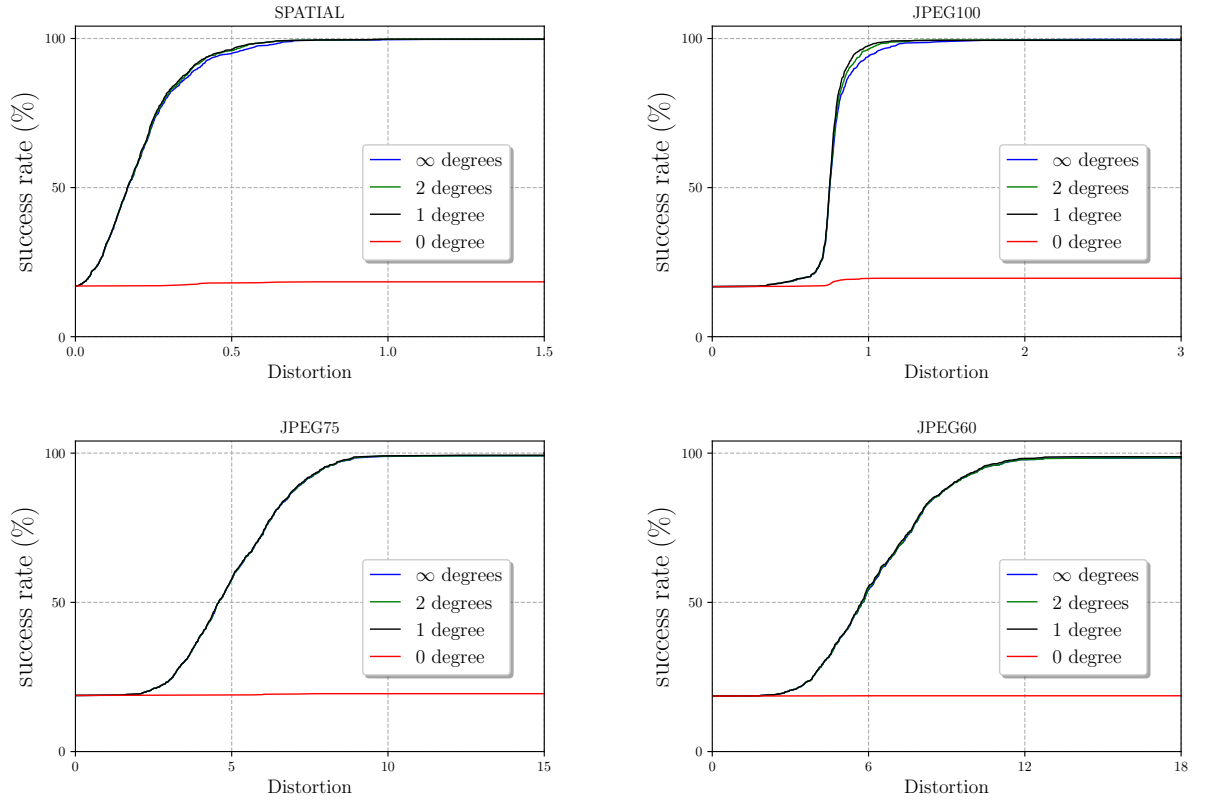


Figure 7.10 – The operating curves of EfficientNet-b0 against BP in the spatial and JPEG domain with different degree of freedom. Distortion is measured from the original spatial image.

to note that both neural networks misclassified the right image as the same class (828: tray) whereas the left image is misclassified with different labels yet semantically very close. These classification errors are also understandable from a human point of view. As a final comment on classifiers: all images misclassified by EfficientNet-b0 (170 total) are misclassified by RegNetX-032 as well, and RegNetX-032 misclassified 6 more images (176 total).

Figure 7.11 shows the operating curves of all four classifiers. They are attacked with BP or PGD<sub>2</sub> and quantized in both spatial and JPEG90 domains. Gradients from one classifier to another vary with the number and nature of hidden layers. This affects how an attack behaves. The hierarchy between unquantized attacks however remains the same from one classifier to another as seen in Fig. 6.1. This order remains after quantization in the spatial domain: BP outperforms PGD<sub>2</sub>.

Nevertheless, the differences between classifiers and attacks are barely noticeable in the



JPEG domain. Distortion added by JPEG compression takes over as Sec. 7.4.3 explains and imposes the common shape of the operating curve. This shows the adaptability of our quantization w.r.t. which neural network is attacked.

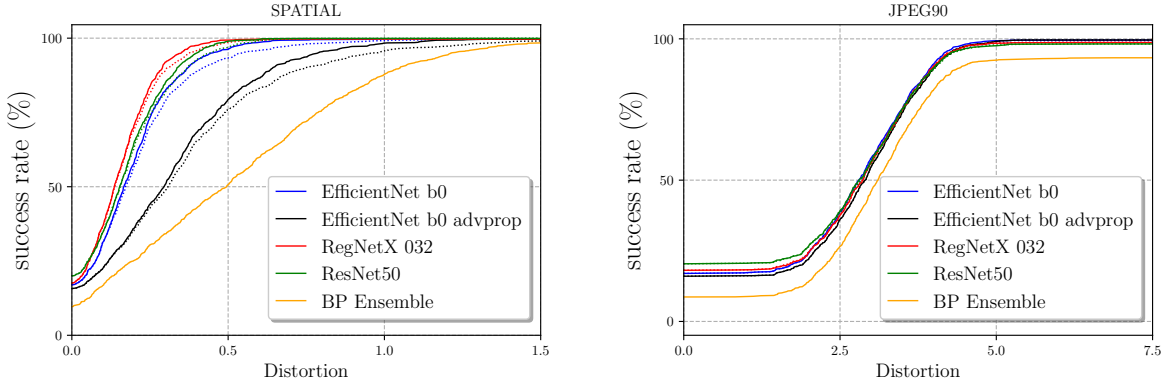


Figure 7.11 – The operating curves of several classifiers against BP (plain) and  $\text{PGD}_2$  (dotted) with quantization in spatial (left) and JPEG90 (right) domains.

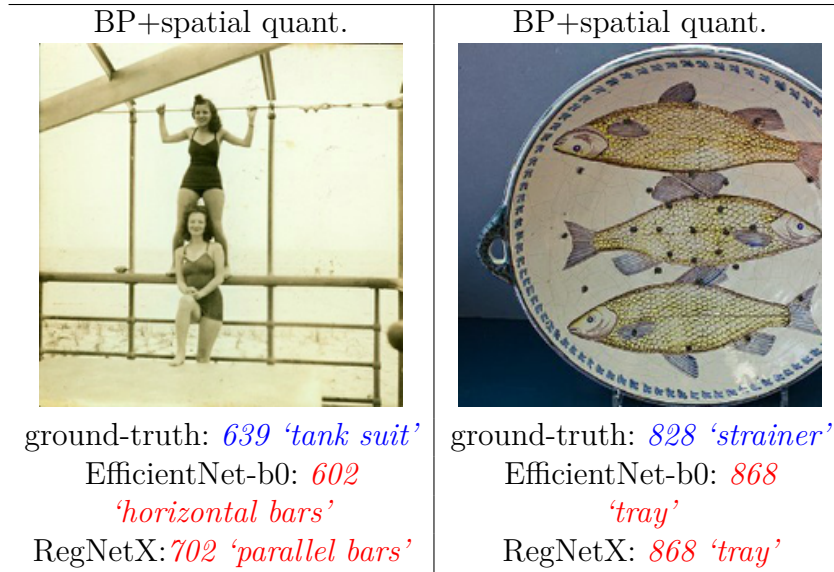


Figure 7.12 – Misclassification of two images from our dataset through two classifiers: EfficientNet-b0 and RegNetX-032.

### 7.4.6 Transferability

Our quantization method aims at creating an adversarial image with minimum distortion. The image therefore lies just behind the frontier of correct classification for the

targeted classifier. Therefore, no transferability to other deep neural networks is guaranteed. We consider the scenario where the attacker knows that the deployed classifier belongs to an ensemble but she/he does not know which one exactly. The goal is to forge images adversarial for all the classifiers in the ensemble. Our strategy is to aggregate the losses of the classifiers with the maximum operator so that we focus on the most robust element of the ensemble and to aggregate their gradients with the average operator like in Expectation over Transformation (EoT [3, 4]). Figure 7.11 shows that beating all the classifiers in the ensemble does not amount to beat the most robust one (*i.e.* EfficientNet-b0 advprop). More distortion is required instead in spatial domain. In JPEG domain we observe that the slope of the ensemble curve is similar to the curve of any single model. The distortion created by the quantization is still on par with compression alone. We do note however that quantizing for several classifiers is a more difficult task in JPEG. The final accuracy is 6.7% in JPEG, 0% in spatial domain.

#### 7.4.7 JPEG Compression as a Defense

As mentioned several times throughout this chapter, JPEG compression usually erases adversarial perturbations while not spoiling the accuracy of the classifier over natural images. For this reason, JPEG compression has been studied as a means of defense against adversarial samples [93, 134]. It acts as a low-pass filter reforming the input image. Table 7.2 shows this is indeed true for our spatially quantized images. While a quality factor of 90 reforms 42% of our adversarial PNG images, a quality factor of 60 reforms  $\approx 70\%$ . This proves to be a very effective defense against adversarial samples.

However, our adversarial images quantized in the JPEG domain are naturally more robust to JPEG compression. The results show interesting properties:

- The performance of the attack is maximized when the quality factor matches the one used at the defense.
- Compressing at the same quality factor does however reform few images ( $< 1\%$  in all three cases) because JPEG is not idempotent.
- Quantized adversarials at a given quality factor are robust to defenses with a higher quality factor.

Table 7.2 – Accuracy (in %) of EfficientNet-b0 equipped with a JPEG compression as a defense front-end reformer against our quantized *best-effort* BP.

Attack	Defense				
	None	JPEG100	JPEG90	JPEG75	JPEG60
Spatial	<b>0.1</b>	4.0	42.1	63.0	70.4
JPEG100	0.6	1.0	39.8	71.4	76.6
JPEG90	0.6	<b>0.7</b>	<b>0.6</b>	60.0	69.8
JPEG75	0.6	<b>0.7</b>	2.4	<b>0.6</b>	14.7
JPEG60	1.0	1.0	1.3	6.4	<b>1.1</b>

## 7.5 Chapter Conclusion

We have proposed a method (improved from an initial work [12]) to effectively quantize adversarial samples in order to craft adversarial images in spatial or JPEG domains. This quantization guarantees adversariality while minimizing the distortion. It runs within few *forward* calls to the network making it faster than simple attacks and it conveniently operates on top of any white box attacks for broader usability.

When dealing with JPEG compression, the distortion induced by the attack is a very small fraction of the distortion induced by sole compression, and crafting adversarial images in JPEG at low-quality factors also provides robustness to countermeasures based on JPEG compression.

The presented methodology is moreover ubiquitous and it could be transferred to other domains such as JPEG2000 [165] or HEIF [59], and the optimization setup can also be used for other metrics than classification (for example regression as proposed in [10]) and on other distances such as steganographic costs [11] in order to generate adversarial images that are less prone to be statistically detected.

















Natural	BP+spatial quant.	JPEG75 compression	BP+JPEG75 quant.
 <p>2 'white shark' <math>d(x_q, x_o) = 0.0</math></p>	 <p>3 'tiger shark' <math>d(x_q, x_o) = 0.02</math></p>	 <p>2 'white shark' <math>d(J^{-1}(X_0), x_o) = 3.15</math></p>	 <p>3 'tiger shark' <math>d(J^{-1}(X_q), x_o) = 3.15</math></p>
 <p>791 'shopping cart' <math>d(x_q, x_o) = 0.0</math></p>	 <p>161 'basset hound' <math>d(x_q, x_o) = 0.53</math></p>	 <p>791 'shopping cart' <math>d(J^{-1}(X_0), x_o) = 6.88</math></p>	 <p>161 'basset hound' <math>d(J^{-1}(X_q), x_o) = 7.52</math></p>
 <p>754 'radio, wireless' <math>d(x_q, x_o) = 0.0</math></p>	 <p>766 'rotisserie' <math>d(x_q, x_o) = 0.17</math></p>	 <p>754 'radio, wireless' <math>d(J^{-1}(X_0), x_o) = 3.63</math></p>	 <p>766 'rotisserie' <math>d(J^{-1}(X_q), x_o) = 3.65</math></p>
 <p>626 'lighter, light' <math>d(x_q, x_o) = 0.0</math></p>	 <p>470 'candle, taper' <math>d(x_q, x_o) = 0.21</math></p>	 <p>626 'lighter, light' <math>d(J^{-1}(X_0), x_o) = 3.72</math></p>	 <p>470 'candle, taper' <math>d(J^{-1}(X_q), x_o) = 3.79</math></p>

Figure 7.13 – Examples of attacked images with spatial and JPEG75 quantizations. JPEG75 compression of the original image is also displayed in the third column.



# QUANTIZATION USING STEGANOGRAPHIC STRATEGIES

---

## 8.1 Introduction

The connection between adversarial examples and forensics/anti-forensics is obvious. First, adding an adversarial perturbation to delude a processing is an image manipulation per se and therefore detecting adversarial examples is a forensic task by itself. Second, techniques forging adversarial examples are also used to fool forensics detectors as proposed in [55][5]. In this case, the adversarial attack is a counter-forensics strategy to conceal an image manipulation.

Paper [115] makes the connection between adversarial examples and information hiding (be it watermarking or steganography). Both fields modify images (or any other type of media) in the pixel domain so that the content is moved to a targeted region of the feature space. That region is the region associated to a secret message in information hiding or to a wrong class in adversarial examples. Indeed, paper [115] shows that adversarial examples benefits from ideas proven efficient in watermarking, and vice-versa.

This chapter contributes to the same spirit by investigating what both steganography and steganalysis bring to the the “cat-and-mouse” game of adversarial examples. There are two natural ideas:

**Steganalysis** aims at detecting weak perturbations in images. This field is certainly useful for the defender.

**Steganography** is the art of modifying an image while being non-detectable. This field is certainly useful for the attacker.

These two sides of the same coin allow to mount a defense and to challenge it in return, as done in other studies [3, 19, 146]. This chapter aims at revealing the status of the game between the attacker and the defender at the time of writing, *i.e.* when both players use up-to-date tools: state-of-the-art image classifiers with premium steganalyzers, and best-in-class steganography embedders. This chapter proposes three contributions:

- Assess the robustness of very recent image classifiers, EfficientNet [142] and its robust version [158],
- Apply one state-of-the-art steganalyzer (SRNet [13]) for forensics purposes, *i.e.* to detect adversarial images,
- Use the best steganographic schemes to craft counter-forensics perturbations reducing the detectability: HILL [84] uses empirical costs, MiPod [133] models undetectability from a statistical point of view, while GINA [85, 155] synchronizes embeddings on color channels.

Section 8.2 reviews the connections between forensics, steganography, and adversarial examples. Our main contribution on counter-forensics and experimental results are detailed in Sect. 8.3 and 8.4.

## 8.2 Related Works

### 8.2.1 Steganalysis for forensic purposes

Steganalysis has always been bounded to steganography, obviously. Yet, a recent trend is to resort to this tool for other purposes than detecting whether an image conceals a secret message. For instance, paper [114] claims the universality of SRM and LBP steganalyzers for forensic purposes detecting image processing (like Gaussian blurring, gamma correction) or splicing. The authors of [41] used this approach during the IEEE IFS-TC image forensics challenge. The same trend holds as well on audio forensics [95]. As for camera model identification, the inspiration from steganalysis (co-occurrences, color dependencies, conditional probabilities) is clearly apparent in [149].

This reveals a certain versatility of steganalysis. It is not surprising since the main goal is to model and detect weak signals. Modern steganalyzers are no longer based on hand-crafted features like SRM [47]. They are no more no less than Deep Neural Networks like Xu-Net [161] or SRNet [13]. The frontier between steganalysis and any two-class image classification problem (such as image manipulation detection) is blurred. Yet, these networks have a specific structure able to focus on weak signal detection. They for example avoid subsampling or pooling operations in order to preserve high frequency signals, they also need large databases combined with augmentation techniques and curriculum learning to converge [164].

However, this general-purpose strategy based on steganalysis method has some drawbacks. It lacks fine-grained tampering localization, which is often an issue in forensics [40]. Paper [25] goes a step further in the cat-and-mouse game with an counter-forensic method: knowing that the defender uses a steganalyzer, the attacker modifies the perturbation (accounting for a median filtering or a contrast enhancement) to become less detectable.

As for adversarial images detection, this method is not new as well. The authors of [132] wisely see steganalysis detection as a perfect companion to adversarial re-training. This last mechanism fights well against small perturbations. It however struggles in correctly classifying coarser and more detectable attacks. Unfortunately, this idea is supported with



a proof of concept (as acknowledged by the authors): the steganalyzer is rudimentary, the dataset is composed of tiny images (MNIST). On the contrary, the authors of [91] outline that steganalysis works better on larger images like ImageNet (ILSVRC-2016). They however use a deprecated classifier (VGG-16 [137]) with outdated steganalyzers based on hand-crafted features (SPAM and SRM).

### 8.2.2 Adversarial examples

As mentioned multiple times throughout this manuscript, an *untargeted* adversarial attack aims at finding the optimal point:

$$x_a^\star = \arg \min_{x: \hat{c}(x) \neq c(x_o)} \|x - x_o\|, \quad (8.1)$$

where  $\|\cdot\|$  is usually the Euclidean distance.

Discovering this optimal point is difficult because the space dimension  $n$  is large. Even in a white-box scenario, all attacks are sub-optimal iterative processes.

As outlined in Chap. 7, definition (8.1) is very common in literature, yet it is incomplete. The final goal of the attacker is to create an adversarial image  $I_a \in \mathcal{D}$  in the pixel domain, not  $x_a \in \mathcal{X}$ . Applying the inverse pre-processing is not solving the issue because this a priori makes non integer pixel values. Rounding to the nearest integer,  $I_a = \text{round}(x_a)$ , is simple but ineffective. Most networks are so vulnerable that  $u = x_a - x_0$  is a weak signal partially destroyed by rounding.  $x_a$  is no longer adversarial. Note that DDN is a rare example of a powerful attack natively offering quantized pixel values.

Chapter 7 proposed a post-processing  $\mathbf{Q}$  on top of any attack that makes sure  $x_q = \mathbf{Q}(x_a)$  is (i) an image (integral constraint), (ii) remains adversarial, and (iii) has a low Euclidean distortion  $\|x_q - x_0\|$ . This chapter follows the same approach but adds another constraint: (iv) be non-detectable.

Figure 8.1 shows the characteristic function measuring the probability of success of an attack [12] as a function of the distortion budget ( $L_2$ -norm) against landmark classifiers in the history of ImageNet challenge. Unlike other chapters, distortion is not normalized w.r.t.  $n$ . The characteristic function starts at  $1 - \eta$ , where  $\eta$  is the accuracy of the classifier: a proportion  $1 - \eta$  of original images are naturally adversarial since there are misclassified. As we know, the accuracy of the networks increases as time goes by: AlexNet (2012) [79] < VGG-16 (2015) [137] < GoogLeNet (2015) [139] < ResNet-50 [61] (2016) < EfficientNet-b0 [142] (2019). On the other hand, the robustness to this attack can be measured by the

Table 8.1 – Robustness of recent classifiers against PGD<sub>2</sub> followed by quantization

	Acc (%)	$P_{suc}$ (%)	$\overline{L}_2$
Alexnet	57.0	100	104
VGG-16	75.0	100	56.5
GoogLeNet	77.2	99.8	72.9
ResNet-50	80.0	97.2	81
Vanilla EfficientNet-b0 [142]	82.8	99.1	115
Robust EfficientNet [158]	84.3	98.5	192

average distortion necessary for hacking the images (cf. Tab. 8.1). This reveals a different hierarchy: ResNet-50 and VGG-16 are quite fragile contrary to the old AlexNet. Overall, the recent EfficientNet is both more accurate and more robust.

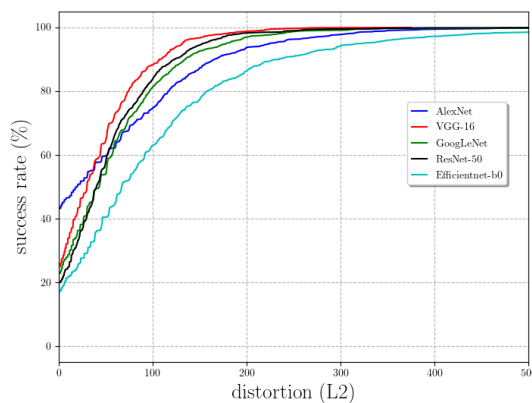


Figure 8.1 – Characteristic function of quantized attack (PGD in best-effort) against well known (vanilla) classifiers for ImageNet.

### 8.2.3 Defenses

The literature proposes four types of defenses or counter-attacks against adversarial examples white-box attacks as seen in Chap. 5. In this work, we consider detection as means of defense against our quantized adversarial images. We evaluate steganalysis as a candidate for this task. We evaluate its efficiency against standard quantization from the previous chapter as well as a steganographic-oriented quantization explained below

### 8.2.4 Steganographic costs

Undetectability is usually tackled by the concept of costs in the steganographic literature: each pixel location  $i$  of a given cover image is assigned a set of costs  $(w_i(\ell))_\ell$  that reflects the detectability of modifying the  $i$ -th pixel by  $\ell$  quantum. Usually,  $w_i(0) = 0$ ,  $w_i(-\ell) = w_i(\ell)$ , and  $w_i(|\ell|)$  is increasing. The goal of the steganographer is to embed a message  $\mathbf{m}$  while minimizing the empirical steganographic distortion:

$$D(\ell) := \sum_{i=1}^n w_i(\ell_i). \quad (8.2)$$

This is practically achieved using Syndrome Trellis Codes [44]. This chapter proposes to use the steganographic distortion (instead of  $L_1$ ,  $L_2$  or  $L_\infty$  norms in adversarial literature) in order to decrease detectability.

Note that this distortion is additive, which is equivalent to consider that each pixel modification yields a detectability independent from the others. Yet, one strategy takes into account potential interactions between neighboring modifications. The image is first decomposed into disjoint lattices to be sequentially embedded where costs are then sequentially updated after the embedding over one lattice [85].

This work uses three families of steganographic costs. The first one, HILL [84], is empirical and naive, but has nevertheless been widely used in steganography thanks to its simplicity. The cost map  $\mathbf{w}$  associated to  $\pm 1$  is computed using two low-pass averaging filters  $\mathbf{L}_1$  and  $\mathbf{L}_2$  of respective size  $3 \times 3$  and  $15 \times 15$  and one high pass filter  $\mathbf{H}$ : ( $*$  means convolution)

$$\mathbf{w} = \frac{1}{|\mathbf{I} * \mathbf{H}| * \mathbf{L}_1} * \mathbf{L}_2, \text{ with } \mathbf{H} = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}. \quad (8.3)$$

The second one, derived from MiPod [133], assumes that the residual signal is distributed as  $\mathcal{N}(0, \sigma_i^2)$  for the original image, and  $\mathcal{N}(\ell_i, \sigma_i^2)$  for the stego image. The variance  $\sigma_i^2$  is estimated on each pixel using Wiener filtering and a least square approximation on a basis of cosine functions. The cost is the log likelihood ratio between the two distributions evaluated at 0, *i.e.*:

$$w_i(\ell_i) = \ell_i^2 / \sigma_i^2. \quad (8.4)$$

Unlike HILL, this model handles modifications other than  $\pm 1$ .

The last one is a cost updating strategy favoring coherent modifications between pixels

within a spatial or color neighborhood. It is called GINA [155] and it is derived from CMD [85]. It splits the color images into 4 disjoint lattices per channel, i.e. 12 lattices. The embedding performs sequentially starting by the green channel lattices. The costs on one lattice is updated according to the modifications done on the previous ones as:

$$w'_i(\ell_i) = \frac{1}{9}w_i(\ell_i), \text{ if } \text{sign}(\ell_i) = \text{sign}(\mu_i), \quad (8.5)$$

with  $\mu_i$  the average of the modifications already performed in the spatial or colour neighborhood of location  $i$ .

### 8.2.5 Looking at Adversarial Examples with Stega Glasses

First, note that adversarial images recently became a source of inspiration for steganography: paper [145] proposes the concept of steganography with an adversarial embedding fooling a DNN-based steganalyzer. References [7] and [136] propose both to cast the problem of adversarial embedding as a game-theoretical problem. A protocol to train efficiently new adversaries and to generate less detectable stego contents using a min max strategy is presented in [7]. The reference [136] solves the game between one embedder and one steganalyst using both different levels of adversarial perturbations.

Paper [132] stresses however one fundamental difference between steganography and adversarial examples: Steganalysis has two classes, where the class ‘cover’ distribution is given by Nature, whereas the class ‘stego’ distribution is a consequence of designed embedding schemes. On the other hand, a *perfect* adversarial example and an original image are distributed as by the class  $\hat{c}(x_a)$  or  $c(x_o)$ , which are both given by Nature.

We stress another major difference: Steganographic embedding is essentially a stochastic process. Two stego-contents derived from the same cover are different almost surely with STC [44]. This is a mean to encompass the randomness of the messages to be embedded. This is also the reason why steganographic embedders turns the costs  $(w_i(\ell))_\ell$  into probabilities  $(\pi_i(\ell))_\ell$  of modifying the  $i$ -th pixel by  $\ell$  quantum. These probabilities are derived to minimize the detectability under the constraint of an embedding rate given by the source coding theorem:

$$R = -\frac{1}{n} \sum_i \sum_{\ell_i} \pi_i(\ell_i) \log_2 (\pi_i(\ell_i)) \text{ bits.} \quad (8.6)$$

In contrast, an attack is a deterministic process always giving the same adversarial version

of one original image. Adversarial imaging does not need these probabilities.

## 8.3 Steganographic Post-Processing

This section presents the use of steganography in our post-processing  $Q$  mounted on top of any adversarial attack.

### 8.3.1 Optimal post-processing

Starting from an original image, we assume that an attack has produced  $x_a$ . The problem is that  $x_a \in [0, 255]^n$ , *i.e.* its pixel values are a priori not quantized. Our post-processing specifically deals with that matter, outputting  $x_q = Q(x_a) \in \{0, \dots, 255\}^n$ . In this chapter, we call  $\ell$  the **quantized**<sup>1</sup> perturbation:

$$u := x_a - x_0 \in \mathbb{R}^n, \quad (8.7)$$

$$\ell := x_q - x_0 \in \mathbb{Z}^n. \quad (8.8)$$

The design of  $Q$  amounts to finding a good  $\ell$ . This is more complex than just rounding perturbation  $u$ .

We first restrict the range of  $\ell$ . We define the degree of freedom  $d_f$  as the number of possible values for each  $\ell_i$ ,  $1 \leq i \leq n$ . This is an even integer greater than or equal to 2. The range of  $\ell_i$  is centered around  $u_i$ . For instance, when  $d_f = 2$ ,  $\ell_i \in \{\lfloor u_i \rfloor, \lceil u_i \rceil\}$ . In general, the range is given by

$$\mathcal{L}_i := \{\lfloor u_i \rfloor - d_f/2, \dots, \lfloor u_i \rfloor - 1, \lfloor u_i \rfloor, \dots, \lfloor u_i \rfloor + d_f/2 - 1\}. \quad (8.9)$$

Over the whole image, there are  $d_f^n$  possible sequences for  $\ell$ .

We now define two quantities depending on  $\ell$ . The *classifier loss* at  $x_q = x_a - u + \ell$ :

$$L(\ell) := \hat{y}_{c_0}(x_a - u + \ell) - \hat{y}_{c_a}(x_a - u + \ell), \quad (8.10)$$

where  $c_0$  is the ground truth class of  $x_o$  and  $c_a$  is the predicted class after the attack. When the attack succeeds, it means that  $x_a$  is classified as  $c_a \neq c_0$  because  $\hat{y}_{c_a}(x_a) > \hat{y}_{c_0}(x_a)$  so that  $L(u) < 0$ . Our post-processing cares about maintaining this adversariality. This

---

1. Notation differs from previous chapter. Here  $\ell$  is equal to  $u + q$  seen previously

constrains  $\ell$  s.t.  $L(\ell) < 0$ .

The second quantity is the *detectability*. We assume that a black-box algorithm gives the stego-costs  $(w_i(\ell))_\ell$  for a given original image. The overall detectability of  $x_q$  is gauged by  $D(\ell)$  as given by (8.2). In the end, the optimal post-processing  $\mathbf{Q}$  minimizes detectability while maintaining adversariality:

$$\ell^\star = \arg \min_{\ell: L(\ell) < 0} D(\ell). \quad (8.11)$$

### 8.3.2 Our proposal

The complexity for finding the solution of (8.11) a priori scales as  $O(d_f^n)$ . Two ideas from the adversarial examples literature help reducing this cost. First, the problem is stated as an Lagrangian formulation as in [21]:

$$\ell_\lambda = \arg \min D(\ell) + \lambda L(\ell). \quad (8.12)$$

where  $\lambda \geq 0$  is the Lagrangian multiplier. This means that we must solve this problem for any  $\lambda$  and then find the smallest value of  $\lambda$  s.t.  $L(\ell_\lambda) < 0$ .

Second, the classifier loss is linearized around  $x_a$ , *i.e.* for  $\ell$  around  $u$ :  $L(\ell) \approx L(u) + (\ell - u)^\top g$ , where  $g = \nabla L(u)$ . This transforms problem (8.12) into

$$\ell_\lambda = \arg \min \sum_{i=1}^n w_i(\ell_i) + \lambda(u_i - \ell_i).g_i. \quad (8.13)$$

The solution is now tractable because the functional is separable: we can solve the problem pixel-wise. The algorithm stores in  $d_f \times n$  matrix  $W$  the costs, and in  $d_f \times n$  matrix  $G$  the values  $((u_i - \ell_i).g_i)_i$  for  $\ell_i \in \mathcal{L}_i$  (8.9). For a given  $\lambda$ , it computes  $W + \lambda G$  and looks for the minimum of each column  $1 \leq i \leq n$ . In other words, it is as complex as  $n$  minimum findings, each over  $d_f$  values, which scales as  $O(n \log d_f)$ .

Note that for  $\lambda = 0$ ,  $\mathbf{Q}$  quantizes  $x_{a,i}$  ‘towards’  $x_{0,i}$  to minimize detectability. Indeed, if  $\ell_i = 0$  is admissible ( $0 \in \mathcal{L}_i$  holds if  $|u_i| \leq d_f/2$ ), then  $\mathbf{Q}(x_{a,i}) = x_{0,i}$  at  $\lambda = 0$ .

On top of solving (8.13), a line search over  $\lambda$  is required. The linearization of the loss being a crude approximation, we make calls to the network to check that  $\mathbf{Q}(x_a)$  is adversarial: When testing a given value of  $\lambda$ ,  $\ell_\lambda$  is computed to produce  $x_q$  that feeds the classifier. If  $x_q$  is adversarial then  $L(\ell_\lambda) < 0$  and we test a lower value of  $\lambda$  (giving more importance to the detectability), otherwise we increase it. The search is performed

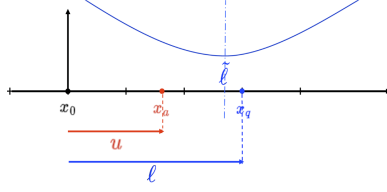


Figure 8.2 – Rounding the minimizer when the stego-cost is quadratic.

over  $\log_2(n)$  steps. The images we used are of dimension  $224 \times 224 \times 3$  which gives 18 steps. Optimal  $\lambda$  varies widely in value between different images.

### 8.3.3 Simplification for quadratic stego-costs

We now assume that the stego-costs obey to the following expression:  $w_i(\ell) = \ell^2/\sigma_i^2$  as in (8.4). This makes the functional of (8.13) (restricted to the  $i$ -th pixel) equals to  $\ell_i^2/\sigma_i^2 - \lambda g_i \ell_i + \lambda p_i$  which minimizer is  $\tilde{\ell}_i = \lambda g_i \sigma_i^2/2$ .

Yet, this value in general is not an integer belonging to  $\mathcal{L}_i$  (8.9). This issue is easily solved because a quadratic function is symmetric around its minimum, therefore the minimum over  $\mathcal{L}_i$  is its value closest to  $\tilde{\ell}_i$  as shown in Fig. 8.2. The range  $\mathcal{L}_i$  being nothing more than a set of consecutive integers, we obtain a closed form expression:

$$\ell_{\lambda,i} = \min(\max(\lceil \lambda g_i \sigma_i^2/2 \rceil, \lceil u_i \rceil - d_f/2), \lceil u_i \rceil + d_f/2 - 1), \quad (8.14)$$

where  $\lceil \cdot \rceil$  is the rounding to the nearest integer. The post-processing has now a linear complexity.

In this equation, the min and max operate a clipping so that  $\ell_{\lambda,i}$  belongs to  $\mathcal{L}_i$ . This clipping is active if  $\tilde{\ell}_i \notin \mathcal{L}_i$ , which happens if  $\lambda \geq \bar{\lambda}_i$  with

$$\bar{\lambda}_i := \begin{cases} \left\lceil \left| \frac{2[p_i] - d_f}{g_i \sigma_i^2} \right| \right\rceil_+ & \text{if } g_i < 0 \\ \left\lceil \left| \frac{2[p_i] + d_f - 2}{g_i \sigma_i^2} \right| \right\rceil_+ & \text{if } g_i > 0, \end{cases} \quad (8.15)$$

where  $|a|_+ = a$  if  $a > 0$ , 0 otherwise. This remark is important because it shows that for any  $\lambda > \max_i \bar{\lambda}_i$ , the solution  $\ell_\lambda$  of (8.14) remains the same due to clipping. Therefore, we can narrow down the line search of  $\lambda$  to  $[0, \max_i \bar{\lambda}_i]$ .

## 8.4 Experimental Investigation

### 8.4.1 Experimental setup

Our experimental work uses 18,000 images from ImageNet of dimension  $224 \times 224 \times 3$ . This subset is split in 1,000 for testing and comparing, 17,000 for training. An image is attacked only if the classifier predicts its correct label beforehand. This happens with probability equaling the accuracy of the network  $\text{Acc}$ . We measure  $\overline{L}_2$  the average Euclidean distance of the perturbation  $\ell$  and  $P_{suc}$  the probability of a successful attack *only over correctly labeled images*.

We attack the networks with 4 different attacks: FGSM [53], PGD<sub>2</sub> [98], CW [21] and DDN [123]. All these attacks are run in a *best-effort* fashion with a complexity limited to 100 iterations. This means that for FGSM and PGD<sub>2</sub> the distortion is gradually increased until the image is adversarial. For more complex CW and DDN attacks, different parameters are used over a total maximum of 100 iterations. The final attacked version is the adversarial image with the smaller distortion. Since DDN is the only attack that creates integer images, the other 3 are post-processed either by the enhanced quantization [12], which is our baseline, or by our method explained in Sect. 8.3.2.

The adversarial image detectors are evaluated by the true positive rate  $\text{TPR}_5$  when the false positive rate  $\text{FPR}$  is fixed to 5%.

### 8.4.2 Robustness of recent classifiers: there is free lunch

Our first experiment compares the robustness of the famous ResNet-50 network to the recent classifiers: the vanilla version of EfficientNet-b0 [142] and its robust version trained with AdvProp [158]. Note that the authors of [158] apply adversarial re-training for improving accuracy. As far as we known, the robustness of this version was not yet established.

Figure 8.3 shows the same characteristic function as in Fig. 8.1 with this time the vanilla EfficientNet-b0 against its robust version. Table 8.1 gives measurements  $P_{suc}$  and  $\overline{L}_2$  as a summary of the characteristic function shown in Fig 8.1. This confirms that modern classifiers are more accurate and more robust (lower  $P_{suc}$  and/or bigger  $\overline{L}_2$ ). This is indeed a surprise: It pulls down the myth of ‘No Free Lunch’ in adversarial machine learning literature [35, 148] (the price to pay for robustifying a network is allegedly a lower accuracy).



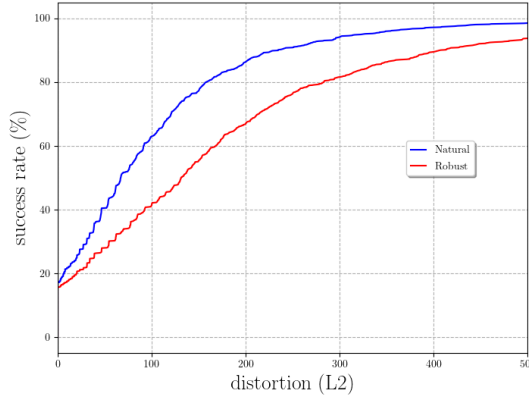


Figure 8.3 – Characteristic function of attack [12] (PGD in best effort with quantization) against Vanilla EfficientNet-b0 and its Robust counterpart. Distortion is not normalized for  $n$ .

### 8.4.3 Detection with forensics detectors

We use three steganalyzers to detect adversarial images. Their training set is composed of 15,651 pairs of original and adversarial images. The latter are crafted with *best-effort* FGSM against vanilla EfficientNet-b0.

The first detector is trained on SRM feature vectors [47], with dimensions 34,671. SRM is a model that applies to only one channel. It is computed on the luminance of the image in our experimental work. The classifier separating these high-dimensional vectors into two classes is the linear regularized classifier [27]. The second detector is based on the color version of SRM: SCRMQ1 [52] with dimension 18,157. The classifier is the same. The third detector is SRNet [13], one of the best detectors in steganalysis. Training is

Table 8.2 – Detection probabilities ( $\text{TPR}_5$ ) with forensics detectors of adversarial images targeting classifier vanilla EfficientNet-b0 [142]

	$P_{suc}$	$\bar{L}_2$	SRM(%)	SCRMQ1(%)	SRNet(%)
FGSM+[12]	89.7	286	72.00	83.3	<b>93.5</b>
PGD <sub>2</sub> +[12]	98.6	113	65.02	83.1	<b>93.8</b>
CW+[12]	89.7	97	68.78	83.6	<b>94.5</b>
DDN	83.2	186	79.53	91.9	<b>94.8</b>

Table 8.3 – Undetectability of steganographic embedding on PGD<sub>2</sub> against the vanilla model (Van) and its robust version (Rob).

	$d_f$	$P_{suc}$ (%)		$\overline{L_2}$		SCRMQ1(%)		SRNet(%)	
		Van	Rob	Van	Rob	Van	Rob	Van	Rob
[12]	2	98.6	98.3	<b>101</b>	<b>167</b>	83.1	84.6	93.8	90.1
HILL	2	98.6	98.3	113	177	78.0	76.6	87.6	88.5
HILL	4	<b>98.9</b>	<b>98.5</b>	125	181	76.0	73.3	87.4	88.2
MiPod	2	98.3	98.3	176	242	77.4	76.2	86.6	87.7
MiPod	4	98.7	98.0	164	247	74.4	70.2	84.5	87.7
GINA	2	98.5	98.1	283	337	24.4	32.4	68.3	<b>82.9</b>
GINA	4	98.8	98.2	300	330	<b>18.6</b>	<b>24.3</b>	<b>50.9</b>	85.2

performed on 180 epochs: The first 100 with a learning rate of  $10^{-3}$ , the remaining 80 with  $10^{-4}$ . Data augmentation is also performed during training. First, there is a probability  $p_1 = 0.5$  of mirroring the pair of images. Then, there is another probability  $p_2 = 0.5$  of rotating them by 90 degrees.

**The attacks:** Table 8.2 shows the results of detection on all 4 attacks. PGD<sub>2</sub> achieves a high  $P_{suc}$  at almost a third of the distortion FGSM would obtain. DDN and CW being harder to optimize attain both lower  $P_{suc}$  and higher distortion under the given constraints. For the rest of the study we therefore focus on PGD<sub>2</sub> to give the best attacking setup with reasonable complexity.

**The detectors:** Table 8.2 gives also the TPR<sub>5</sub> associated to the detectors. Although [91] achieves good performances with SRM, we do not obtain the high detection rates reported in the reference. This can be due to both finer attacks (best effort mode) and quantization. Our results show also that the detectors generalize well: although trained to detect images highly distorted by FGSM, they can detect as well and sometimes even better more subtle attacks like CW. Moreover, SRNet always outperforms SCRMQ1 and is the most accurate of the three detectors. From Tab. 8.2, we can also deduce that PGD<sub>2</sub>+ [12] is the worst-case scenario for defense. The probability of fooling both the classifier EfficientNet-b0 and the detector SRNet in this setup combines to only  $0.88 \times (1 - 0.933) = 5.9\%$ .

#### 8.4.4 Post-processing with a Steganographic Embedder

We now play the role of the attacker. We use  $\text{PGD}_2$  with best effort as the base attack to compare the detectability of four post-processings: The non-steganographic insertion [12] as a baseline, HILL (8.3), MiPod (8.4), and GINA (8.5). GINA uses the quadratic method explained in Sect. 8.3.3 sequentially over the 12 lattices. Quadratic stego-costs are updated with CMD strategy (8.5). Each lattice contributes to a  $1/12$  of the initial classification loss.

Table 8.3 illustrates how each strategy is detected by either SCRMQ1 or SRNet. Both detectors are trained on FGSM with [12] quantization as ‘stego’ images crafted on their respective network. Distortion increases with each method and along the degree of freedom  $d_f$ . The use of Steganographic costs therefore enables to reduce the detectability while increasing the  $L_2$  distortion.

From the attacker perspective, the best strategy to fool the detector  $\text{PGD}_2$  is GINA costs with  $d_f = 4$ . This scenario now has 48.0% chance of fooling both Vanilla EfficientNet-b0 and SRNet and 80.4% with SCRMQ1 as the detector. Fig. 8.4 shows the two examples with highest distortion on EfficientNet-b0 that still fool SRNet. The added distortion remains imperceptible to the human eye even in these cases.

The conclusion on Robust EfficientNet-b0 is however different. Since the distortion needed to attack the network is higher, it is consequently expected that the detectors will be more accurate. If SCRMQ1 detects GINA distortion slightly better than on Vanilla EfficientNet-b0, SRNet is however very efficient to detect each strategy even if it was trained on FGSM.

#### 8.4.5 Training on adversarial images with GINA costs

We finally play the role of the defender again. We want to detect GINA perturbation with the highest possible TPR. To achieve this we retrain our detectors in the same setups as before, but with images using GINA perturbation as adversarial images. Since Tab. 8.3 shows that in most cases  $d_f = 4$  is indeed the worst-case for the defense side, we attacked the training set of “cover” images with  $\text{PGD}_2$  and GINA costs with  $d_f = 4$ .

The first result we report is that under the same setup, SRNet was never able to distinct both distributions of images. The average confidence on the whole test set is roughly 50%. Trying to train SRNet with a finer learning rate did not lead to any better result. There is probably a set of *hyperparameters* that would lead to a more effective



'Angora rabbit'



'woolen'

'hare',  $L_2 = 488$ 'knot',  $L_2 = 449$ 

Figure 8.4 – Top row: Cover images with their label below. Bottom row: adversarial images with steganographic embedding GINA ( $d_f=4$ ). Below them are their new label and the distortion

training. However this result illustrates that GINA distortion is harder to detect.

Table 8.4 shows  $\text{TPR}_5$  for SCRMQ1 under such training setup. The detector is able to detect GINA mechanism at a higher rate than in Tab. 8.3 but generalizes poorly on other attacks. A conclusion to this final experiment is that GINA can be stealthy to general detectors, but it is still better detected after another iteration of the defender. The detection accuracy is however lower when using GINA costs, and drops from 83.1% to 68.5%. The price of detecting GINA is also to become more specific and to lose performance on the other attacks.

Table 8.4 – Detection on SCRMQ1 after training on adversarial images embedded with GINA ( $d_f=4$ )

	$d$	SCRMQ1(%)	
		Van	Rob
[12]	2	55.9	56.7
HILL	2	53.4	53.6
HILL	4	50.4	53.9
MiPod	2	56.1	55.9
MiPod	4	53.9	54.9
GINA	2	<b>77.7</b>	78.4
GINA	4	68.5	<b>79.7</b>

## 8.5 Chapter Conclusion

This chapter explores both sides of adversarial image detection with steganographic glasses.

On the Attack side, our work using distortions designed for steganographic purposes is able to reduce the detection rates. Steganographic distortion target specific regions and pixels of an image to quantize the attack. The  $L_2$  distortion increases w.r.t. the original attack, but remains imperceptible by the human eye (Fig. 8.4) and less detectable by a targeted detector. This paper consequently shows the possibility of tweaking an attack to make it harder to detect while remaining invisible.

On the Defense side, we use SRNet [13], state-of-the-art in steganalysis to detect adversarial images. Training it on images attacked with the basic FGSM shows excellent performance. Detection also generalizes well even on the finest attacks such as PGD<sub>2</sub> [98] and CW [21].

Finally both Attack and Defense are affected by the considered neural network. The effect of adversarial training on EfficientNet-b0 [Xie:2019aa] is twofold: it increases the classification accuracy as well as robustifying the network. An increased robustness translates into a higher attacking distortion, which itself translates into a higher detectability.

# SCALE AND RESCALING AS MEANS OF ROBUSTNESS

---

## 9.1 Introduction

The topic of adversarial examples has recently attracted a huge interest with more than 3,000 papers published in the last four years. As far as image classification is concerned, most articles consider toy datasets like MNIST composed of  $28 \times 28$  images (10 classes) and CIFAR  $32 \times 32$  images (10 or 100 classes). These are far from nowadays typical image sizes. Few papers deal with the more realistic ImageNet dataset composed of large images (1000 classes). Yet, classifiers usually process these images once downscaled to  $224 \times 224$  or  $256 \times 256$ . This is still about the size of thumbnails on Internet websites, so not yet a modern image. Indeed, no work uses the *original* ImageNet with larger images.

This chapter considers forging realistic, *i.e.* large, adversarial images under a white-box setup for two scenarios:

- A. The image classifier can natively process large images. This means that the model underneath is a wide and deep neural network.
- B. The image classifier first downscales the images to a smaller size, say  $224 \times 224$ , and then uses a neural network to make a prediction.

Scenario A questions how the energy of the adversarial perturbation evolves with the size of the input data. On one hand, the classifier gets more pixels to make a decision which stems into a higher accuracy on natural images. On the other hand, the attacker has more degrees of freedom to delude the classifier. This chapter shows experimental evidence that a bigger size indeed benefits more to the attacker (under some conditions).

In scenario B, the downscaling is part of the classifier box. The small image is thus an internal data that the attacker cannot have access to: The goal is to forge an adversarial version of the large image and not its downscaled version. White-box attacks rely on computing the gradient of a loss function w.r.t. a neural network using back-propagation. The difficulty is therefore how to propagate further this information through the downscaling back to the space of large images. Another point of interest is the choice of downscaling method for the defender and whether its knowledge is key to forge adversarial examples for the attacker.

In the end, we also compare the distortion of perturbations that delude a classifier according to scenario A or B. This yields the best practice for the defender.

## 9.2 Related Works

Several theoretical works study how the dimension  $n$  of the inputs makes the forgery of adversarial examples easier. However, they use different arguments and terminology. It is difficult to verify that they evidence the same phenomenon.

We denote by  $n$  the dimension of the input and call by *input deviation* its natural standard deviation  $\sigma_X(n)$ . If the input  $x \in \mathbb{R}^n$  is a centered random vector, it is typically measured by  $\sqrt{E[\|x\|^2]}/n$  where  $\|\cdot\|$  is the Euclidean distance. In the same way, we measure the *adversarial distortion*  $\sigma_A(n)$  by the typical value of  $\sqrt{E[\|u\|^2]}/n$  of the adversarial perturbation  $u$  for signals of dimension  $n$ .

The early attempt [50] considers a synthetic example where data points are uniformly distributed over spheres of constant radius  $R$ : the input deviation  $\sigma_X(n)$  obviously scales as  $R/\sqrt{n}$ . Paper [50] then proves that the  $\ell_2$  norm of the adversarial perturbation in expectation scales as  $O(1/\sqrt{n})$  for a fixed classification accuracy. This translates into an adversarial distortion  $\sigma_A(n)$  scaling as  $O(1/n)$ .

Papers [35, 42] generalizes this result to many data distributions verifying the Talagrand  $W_2$  transportation-cost inequality. They state that the  $\ell_2$  norm of the perturbation scales as the natural “*intra-class noise level*”  $\sigma_X$  (for a fixed accuracy). For instance, if  $x \sim \mathcal{N}(\mu_k, \sigma_X^2 I_n)$  for class  $k$ , then the  $\ell_2$  norm of the attack is proportional to the power of  $x$ , which remains constant as  $n$  increases (see [35, Sec. 2.5.2]).

We propose to summarize this literature by the following rule of thumb:

$$\sigma_A(n) \propto \sigma_X(n)/\sqrt{n}. \quad (9.1)$$

More precisely, [35] shows that the proportional constant is upper bounded by  $\kappa(n) = \sqrt{-2 \log(1 - \eta(n))} + \sqrt{\pi/2}$ , where  $\eta(n)$  is the accuracy of the classifier.

How do these theoretical results transfer to images? For square color images of size  $\ell$ , there are  $n = 3\ell^2$  pixel values. One argument is that the diameter of the hypercube  $[0, 255]^n$  grows as  $255\sqrt{n}$ , which reflects the typical distance between inputs, hence a constant input deviation. In the same way, up or downscaling does not change the histogram of the pixel values, yielding a constant standard deviation  $\sigma_X(n)$ . Plugging this hand-waving justification into (9.1) yields an adversarial distortion scaling as  $\kappa(n)/\sqrt{n}$ .

On the contrary, paper [38] claims that the expectation of the  $\ell_2$  norm of the perturbation vanishes as  $O(1/\sqrt{n})$  assuming images are  $1/f^2$  power spectrum processes. This would make  $\sigma_A(d) = O(1/n)$  contradicting the previous paragraph.



No work provide clear experimental evidence. Paper [22] investigates this topic on simple datasets MNIST and CIFAR. By upscaling to higher resolutions, it finds that network accuracy and attack efficiency remain the same whatever the dimension. This is not convincing because upscaling does not create any new information.

In the end, this section shows that there is no clear report about adversarial example as the dimension of the inputs grows as far as realistic image classification is concerned.

## 9.3 Problem formulation

This section proposes to include the downscaling inside the neural network, *i.e.* this is scenario B in Sect. 9.1. It outlines that running an adversarial attack in the large image space or the small image space does not produce the same effect.

### 9.3.1 Including the downscaler inside the network

The classifier is a neural network composed of several layers. Each layer applies a linear transformation (be it a convolution or a fully connected layer) followed the non-linear activation function  $\phi(\cdot)$ .

$$a_k = \phi(z_k), \quad (9.2)$$

$$z_k = W_k a_{k-1} + b_k, \quad (9.3)$$

with  $a_k$  the output of the  $k$ -th layer  $\forall k$  s.t.  $1 \leq k \leq K$ . The final vector  $a_K$  is the predicted logit per class (usually the last layer has no non-linearity). Since the downscaling is also a linear function, it can be incorporated inside the network as the layer 0 without activation:  $a_0 := x = DX$ , where  $X$  is the large  $L \times L$  input image, and  $D$  the  $3\ell^2 \times 3L^2$  matrix downscaling to a smaller  $\ell \times \ell$  image  $x$ .

For instance, if  $\ell = L/2$ , one pixel of  $x_o$  is interpolated from 4 pixels of  $X_o$ , and under a proper flattening we have:

$$D = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & \delta_4 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad (9.4)$$

where  $\{\delta_i\}$  are the positive weights of the downscaling kernel (summing up to 1). Another

case is the nearest neighbor interpolation where there is single 1 in each row of  $D$ , *e.g.*  $\delta_1 = 1$ ,  $\delta_i = 0$  for  $i \neq 1$ .

### 9.3.2 Attacking in the large or small image space

In an untargeted white-box setup, the attacker usually defines the loss  $\mathcal{L}(X) = a_K(c_o) - \max_{c \neq c_o} a_K(k)$ , where  $c_o$  is the ground-truth label of image  $X_o$ . The attack aims at finding a perturbation  $u$  s.t.  $\mathcal{L}(X_o + u)$  is negative and  $\|u\|$  is small. White box attacks use the gradient of this loss or its first term:

$$\begin{aligned} \nabla_X \mathcal{L}(X) &:= (d\mathcal{L}(X)/dX)^\top = D^\top g \quad \text{with} \\ g &:= W_1^\top \phi'_1 \dots W_{K-1}^\top \phi'_{K-1} W_{K-1}^\top \nabla \mathcal{L}(a_K) \end{aligned} \tag{9.5}$$

With  $\phi'_k$  a shortcut for  $\phi'(a_k)$ . In other words, by back-propagating through the downscaler, the gradient in the large image space is nothing more than the gradient  $g$  in the small image space through the matrix  $D^\top$ .

The attack usually computes  $X_o - \epsilon \nabla_X \mathcal{L}(X_o) = X_o - \epsilon D^\top g$ . The downscaling maps this to  $x_o - \epsilon D D^\top g$ . Yet, the same attack in the small input space would give  $x_o - \epsilon g$ .

With the nearest neighbor interpolation,  $DD^\top = I_\ell$  and there is no difference. If  $L$  is a multiple of  $\ell$  as in (9.4),  $DD^\top = (\sum_i \delta_i^2) I_\ell$ , with  $\sum_i \delta_i^2 \leq 1$ . There is a loss of energy but the downscaled perturbation stays colinear with  $g$ .

If  $L$  is not a multiple of  $\ell$  or if the downscaler is not the nearest neighbor interpolation, there is also a misalignment because  $DD^\top$  is not proportional to identity matrix  $I_\ell$ . This is especially true when several pixels of the small image depend on the same pixel of the large image, *e.g.* due to anti-aliasing. This shows that the downscaling has an impact on the forgery of adversarial images.

## 9.4 Experimental Works

### 9.4.1 Models, data, and attacks

We experiment with four families of classifiers: EfficientNet [142] and its Lite implementation, EfficientNet-V2 [143], and NFNet family [16]. One family is composed of several models tackling  $\ell \times \ell$  color images with, for instance,  $\ell$  ranging from 224 to 600 for EfficientNet, or from 256 to 576 for NFNet (see Fig. 9.1). Family members share the same

architecture but with wider activation maps and are trained with the same procedure. This is important for a fair comparison.

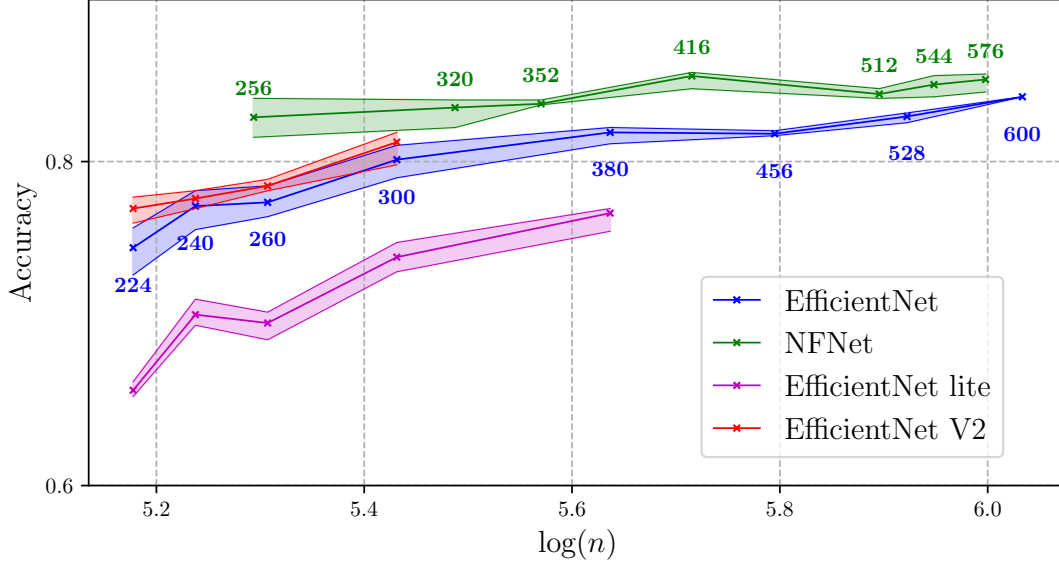


Figure 9.1 – Accuracies with downscaling. Stripes define max and min accuracies over the 6 downscalers for a given model. Numbers are the sizes  $\ell$  of the downsampled images,  $n = 3\ell^2$ .

We created a subset of 1,000 images from the validation set of Imagenet 2012. Each image is the first occurrence of a class within the original dataset. Each image is first *center-cropped* and resized to  $L \times L$  with  $L = 600$  and bilinear interpolation (default on OPENCV library). This is the image size natively processed by EfficientNet-b7, the largest size used by the CNN of our experimental work.

For scenario B, we use four different downscaling methods: *Nearest*, *Bilinear*, *Bicubic* and *Area*. The first three methods interpolate one pixel from the 1, 4, or 16 (resp.) closest pixels. *Area* performs an average pooling on the image. Paper [112] highlights the lack of antialiasing in PyTorch implementation except for method *Area*. We implement two antialiasing methods: average and Gaussian. Both use a kernel size  $k_{size} = \lceil \ell/L \rceil$ . Gaussian values are generated with a standard deviation  $\sigma = 1.6 \times k_{size}$  as inspired by SIFT scale space construction [94]. Figure 9.1 shows the accuracies of all models and over all downscalers (except when  $\ell = L$ ). Our observations are twofold: the accuracy increases with the model size  $\ell$ ; and the downscaling method has little impact.

We choose the white-box attack BP [166] in its *best-effort* [100] implementation because of its high probability of success, low distortion, and speed. Distortion is measured

as the Root Mean Squared Error:  $d(x_a, x_0) = \|x_a - x_0\|/\sqrt{n}$  where  $x_0$  (resp.  $x_a$ ) is the original (resp. adversarial) image. For scenario B, we implement downscaling as a first layer in order to *back-propagate* the gradient as detailed in Sect. 9.3.

### 9.4.2 Scenario A: comparison with theoretical results

We measure the adversarial distortion for different image sizes by downscaling the baseline images to the input size  $\ell$  of the model with bilinear interpolation. The attacker modifies these  $\ell \times \ell$  images in this scenario. According to section 9.2 Eq. (9.1), we measure the normalized adversarial distortion  $\sigma_A(n)/\kappa(n)$  as a function of  $n = 3\ell^2$ .

Figure 9.2 plots this function in logarithmic scale. These experimental results *partially* confirm the theory found in literature. For three families, in the range  $\ell \in [224, 416]$ , this gives almost a trend line whose slope is  $\approx -1/2$ . This gives more credit to papers [35, 42, 50] rather than [38] predicting  $-1$ . Yet, in the range  $\ell \in [456, 600]$  the adversarial deviation is stable or even increasing, and the family EfficientNet-Lite completely violates the theory.

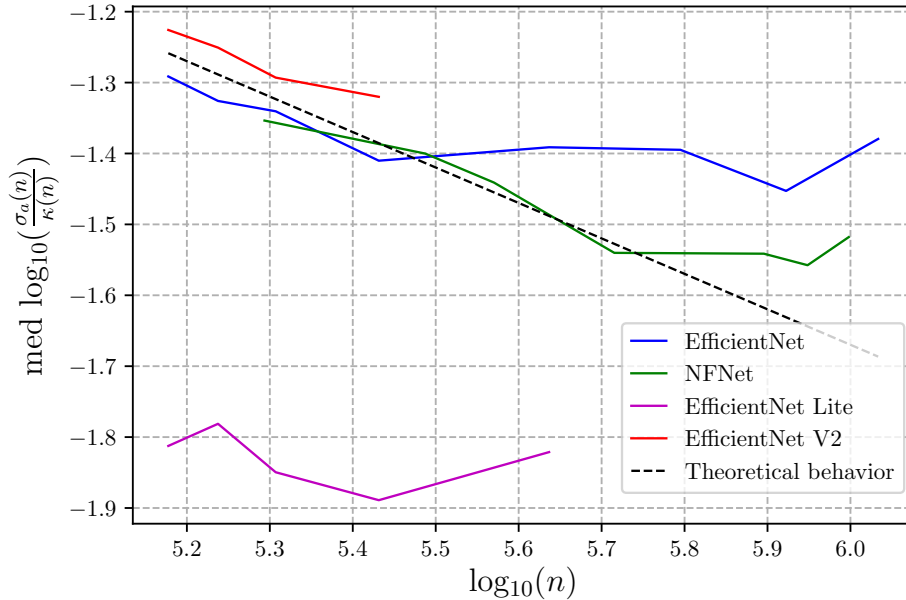


Figure 9.2 – Experimental measurement of the normalized adversarial distortion  $\sigma_A(n)/\kappa(n)$  as a function of  $\log_{10}(n)$ .

### 9.4.3 Scenario B: attacking through downscaling

Figure 9.3 shows the success rate of the attack logically increases with distortion. As Sect. 9.3.2 suspected, there is a noticeable hierarchy in the downscaling methods, especially on smaller sizes like  $\ell = 256$  for NFNet-F0. With *Nearest*, any pixel on the smaller image matches exactly a pixel on the bigger one. The perturbation signal is crafted on these pixels and entirely goes through the downscaling. In every other method, the adversarial signal is diluted through the *back-propagation* on neighboring pixels. This creates a counter effect detrimental to the attack. *Nearest* is thus easier to attack because it requires less distortion. Conversely, methods with anti-aliasing such as *Area* are the best options as a defense.

The impact of the downscaling decreases as the network takes a bigger input size. This results in narrower differences in the plotted curves for  $\ell = 512$  with NFNet-F4 (Fig. 9.3).

We observe the same behavior with the other network families. For instance, with Efficient-Net, Tab. 9.1 second column shows that the attack distortion goes up as  $\ell$  increases for *Nearest*, whereas *Area* is more robust when downscaling a lot, *e.g.* down to b0 ( $\ell = 224$ ).

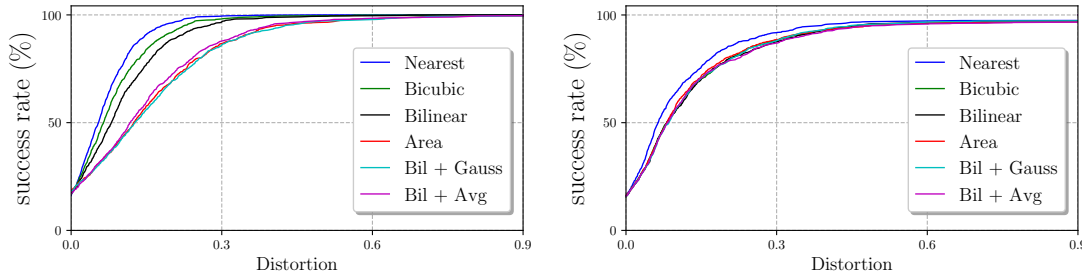


Figure 9.3 – Attacking NFNet-F0 (left,  $\ell = 256$ ) and NFNet-F4 (right,  $\ell = 512$ ) through every downscaling method with the attack BP.

### 9.4.4 Scenario B: transferability

This section assumes that the attacker does not know which downscaling method is used. The attack targets a specific interpolation and is tested through another one. Table 9.2 shows these results when downscaling from  $\ell = 600$  to  $\ell = 224$ .

Transferability of the attack is poor except for *Bilinear* (attack) displaying good transferability over *Bicubic* (defense). Our explanation is that BP adds very small perturbations. Adversarial examples thus lie right behind the frontier of the ground-truth class for

Table 9.1 – Distortion for attacking 90% of the images for the EfficientNet family

Model	Downscaling		Ensemble	
Size $\ell$	Nearest	Area	Average	Worst
b0: 224	0.15	0.39	<b>0.54</b>	0.53
b1: 240	0.16	0.37	<b>0.56</b>	0.47
b2: 260	0.17	0.37	0.47	<b>0.49</b>
b3: 300	0.17	0.34	<b>0.46</b>	0.44
b4: 380	0.21	0.33	0.40	<b>0.54</b>
b5: 456	0.26	0.33	0.38	<b>0.42</b>
b6: 528	0.25	0.31	0.36	<b>0.37</b>
b7: 600	<b>0.37</b>	<b>0.37</b>	<b>0.37</b>	<b>0.37</b>

a targeted classifier. A change in the downscaling method modifies these frontiers which in return classify the sample accurately again.

Table 9.2 – Accuracy (%) when downscaling from size  $L = 600$  to  $\ell = 224$  for EfficientNet-b0

Defense	Attack					
	Bil.	Bic.	Area	Near.	BilG	BilAg
Bil	0.7	70.7	74.6	75.1	73.5	8.2
Bic.	5.6	0.9	75.6	75.9	74.1	9.6
Area	72.8	72.8	0.3	72.8	69.8	71.9
Near.	75.6	75.6	35.5	0.8	69.4	75.6
BilG	73.4	73.5	72.6	73.4	0.50	72.2
BilAg	74.0	73.9	73.2	75.0	72.2	0.7

### 9.4.5 Scenario B: ensemble model

In order to achieve better transferability, the attacker fights against an ensemble of classifiers. For a given model (*e.g.* EfficientNet-b0), we gather all downscaling methods in an ensemble of classifiers. This gives birth to as many gradients, which need to be aggregated into a single one to be exploited by BP attack. We explore two options:

1. A basic averaging of the gradients over all the ensemble, also referred to as *EOT* [154].
2. A worst-case gradient selection, a trick inspired by Deepfool [104].

For a given classifier / downscaler, the second method estimates the distance  $d_{adv}$  an image is from the frontier as if the classifier were linear [104]:  $d_{adv} = \frac{L_{adv}(x)}{\|\nabla L_{adv}(x)\|}$ . The classifier

with the biggest distance is deemed as the worst for the attacker, who then targets it in the next iteration of the attack. In other words, the aggregated gradient is simply the gradient of the worst classifier of the ensemble.

For both methods, an image is deemed adversarial if it deludes all the classifiers of the ensemble. For this experiment we use another subset of 100 randomly picked images from the validation set of Imagenet 2012 as a matter of computational time. Table 9.1 shows the distortion at which 90% of the images are successfully misclassified. Here are the lessons: Attacking an ensemble consumes more distortion than targeting a single classifier ; so guaranteed transferability is possible at the cost of a bigger distortion. There is no clear better option for attacking an ensemble. Distortion increases as the scale goes down. A better defense strategy is to run a model on small scale images and to randomly pick a downscaler over an ensemble at each inference.

## 9.5 Chapter Conclusion

In this chapter, we have studied the impact of downscaling when attacking large images. The best practice appears to be scenario B: downscaling to a small size  $\ell$  with a wide downscaling kernel like ‘Area’ or ‘Bil+Gauss’.

The transferability of an adversarial attack to other interpolation kernels is nearly insignificant when running optimized (low-distortion) attacks. This opens the door to a random resizing method as a means of defense. The attacker can however circumvent this issue by attacking an ensemble, if the pool of downscaling methods is disclosed. Nevertheless, this defense increases the distortion by  $\approx 35\%$  compared to the best pure strategy without sacrificing accuracy.

# CONCLUSION AND PERSPECTIVES

---

Image classification is the main application of Computer Vision. Deep Learning-based classifiers quickly became the best option to tackle a such task. They were however found to be vulnerable to specific perturbations. This created the field of adversarial examples. In Chap. 4, we briefly introduce Deep Neural Networks, how they are trained, and how they are fooled. We then describe a few White-Box attacks that are used throughout the manuscript. We then study the existence of a such vulnerability in Chap. 5. Many culprits are identified. Adversarial vulnerability comes from the shared responsibility of data, architecture, and training phase. But eventually, we conclude that DNNs are vulnerable by *design*.

The purpose of this thesis is to study adversarial examples in a realistic context. We make many choices in this direction. First, we work on a classification task that we estimate to be realistic: the Imagenet classification task. This task is what resembles the most a real-world application. Adversarial examples should also be realistic images. We discuss in Chap. 7 how most works on adversarial examples do not craft *images*. Instead, they work in the floating-point domain and generate data that can be processed by an off-the-shelf DNN but cannot be saved as an image. Chapter 7 shows that quantization is not a trivial task, especially when considering a lossy image format such as JPEG.

Chapter 8 also aims at crafting spatial images. We consider both the perspective of a defender and the attacker. First, we build detection systems that could prevent a DNN from working on a corrupted image. Then, we modify our quantization method to follow steganographic strategies. We finally build detection systems to detect these new examples. This gives insight into the iterative attack-defense game. Each time a player moves, the other answers. We conclude that detection can be prevented in many cases, even after retraining on these stealthy images. One of our detectors is still able to perform a little. The other one could not even be trained successfully on stealthy adversarial examples.

Chapter 9 is also a contribution that studies both perspectives. We analyze the impact that image size can have on adversarial examples. Images from Imagenet are processed in different sizes that are DNN-specific. We show that models using larger images usually



---

are more accurate. They are also larger in terms of parameters, which is a reason for this performance. However, the scaling of adversarial examples (distortion-wise) is almost entirely a function of the input dimension. And this does not act in favor of larger models. The “No Free Lunch theorem” is at stake and we can not draw a hard conclusion on what model to adopt. This is obviously very task-specific. Yet, smaller models have comparable performances and offer more robustness. This should be known when deploying a classifier. The second point in our conclusion is that resizing obviously affects adversarial examples. If the image will undergo a downscaling, the adversarial signal should remain effective. An evident aspect of resizing is that it should rely on antialiasing to dilute the signal. Attacks are downscaler-specific. Defenses could thus use a random resizing method to hinder the attack. However, we show that even a such strategy can be defeated using an ensemble model if the resizing options are known. But this comes at the cost of higher distortion.

## Towards Fair Evaluation

We worked solely using ImageNet. Smaller datasets can be useful to understand adversarial examples or as a mere playground. We believe however that most works should be applied to Imagenet for fair comparisons. This is not the case. But even then, data is rarely the same. A common practice, that we ourselves did, is to pick random images from the validation set. Say 1,000. But there is no consensus on which images to pick. We had the example in 5.3.2 of the works of [159] that randomly picked 5,000 images. Accuracy on these images is 100%. Should we still consider data random? It appears to be cherry-picked. A best practice would be the use of unified data such as Neurips Vision Challenge [80].

The other main aspect is the attack used and *how* it is used. Adversarial examples can be crafted along:

1. Distortion constraint: create a sample with a given distortion and evaluate whether it is indeed adversarial or not.
2. Success constraint: push the attack until the sample is adversarial.
3. Optimize: ensure adversariality at minimum distortion.

Each scenario can be pushed to the authors’ best interest. For instance, the distortion constraint can be convenient for detection. If the signal always is of the same intensity *and*

---

probably far beyond the adversarial frontier. We observed this when reviewing literature for Chap. 8 .

Once again, in a matter of studying the best of both worlds, we mostly studied the Optimize setup. But this may not always be adequate. Transferability is a such situation. Pushing attacks further is probably a better option and we did not. Still, we consider this scenario to be the most likely. An attack like BP, in its best-effort implementation (see Chap. 6) is impressively effective with few iterations as shown in Chap. 6.

## Extend the Image Paradigm

We studied at length the creation of adversarial images and not *samples*. We believed this last point to be an important factor. Having access to the model (white-box) does not mean you have access to inner representations of your data.

A likely attack will use images online or even in the real world. Online adversarial images could be used to bypass content filters online or to protect your data from data collection. This was the purpose of our short work [10] (see Appendix A). Real-world images could be applied to endanger passengers of autonomous vehicles or to evade facial recognition or any detection system.

Making images was thus in line with our direction in this thesis: making attack realistic. Arguably, a white-box setup is not the most likely setup. But its likelihood increase with the advent of model extraction (euphemism for stealing) attacks [147].

## About Defense and Perspectives

We considered the Defense side a few times throughout this thesis. Notably when considering detectors in Chap. 8, the effect of ensemble models (Chap. 7 and Chap. 9), or best-practice for image size and resizing in Chap. 9. Contribution on the attack side can also be considered an iteration towards better defense. Especially in Chap. 8 when we both built detection and studied the effect of steganographic strategies to fool detectors.

We also have been working on adversarial training recently but our work remains unpublished to this day. Insight into our experiments and results is displayed in Appendix B. Our method yields robustification with unsupervised retraining. It is akin to logit pairing, only it is performed on feature maps.

---

## A Final Word

We mentioned many times that both attack and defense compete against each other. This can sometimes give birth to biased critics of opposing works. The lack of unity in the evaluation may sometimes hinder progression. As a conclusion, we would like to bring attention to a recent work by Guo et al. [56]. They argue that currently robust DNN could be as robust as biological neurons. This paper runs experiments to characterize similar adversarial signals in the human eye. Leveraging this model, they go on to show that the human eye is as easily fooled as robust models. We can easily be baffled by the performance of DNNs and forget that we ourselves can be vulnerable to adversity. We are all familiar with optical illusions. Yet, we think that DNNs should not be fooled under any circumstances. The only difference may lie in the ability we have to understand that we are being fooled.

# NOTATIONS

---

$f$	Classifier model
$\theta$	Parameters of a model
$I_0$	Natural (unmodified) image
$I_a$	Adversarial image
$x_0$	Tensor natural image
$x_a = x_0 + u$	Tensor adversarial image
$u$	Perturbation (adversarial signal)
$\mathcal{X}$	Distribution of natural images
$\mathcal{D}$	Distribution of tensor natural images
$\hat{l} = f(x, \theta)$	Logits output
$\hat{y} = f(x, \theta)$	Class probability output
$y \in \{0, 1\}^C$	One-hot ground-truth vector
$n$	Number of pixels
$C$	Number of classes
$c = \operatorname{argmax}_k y$	Ground-truth class
$\hat{c} = \operatorname{argmax}_k \hat{y}$	Predicted class

# LIST OF FIGURES

---

2.1	Un classifieur permet de trier des images en catégories appelées <i>labels</i> . Le classifieur connaît d'abord une étape d'entraînement durant laquelle il apprend à effectuer cette répartition. . . . .	12
3.1	A classifier sorts images in categories called labels. The classifier underwent a <i>training</i> procedure during which it learned to recognize a set of classes. .	20
4.1	Illustration of an artificial neuron. . . . .	27
4.2	Illustration of a Multi-Layer Perceptron. Output is $C$ -dimensional. The size and number of hidden layers are parameters chosen when designing the model. . . . .	28
4.3	Illustration of the first convolution operation in a DNN, followed by a pooling. Size is decreased by a factor 2. . . . .	29
4.4	Comparison of different activation functions around 0. First and second curves show the impact of the factor $a$ in the sigmoid function . . . . .	31
4.5	Top-1 accuracy ( <i>i.e.</i> : correctly predicted images) on DNNs since Alexnet. The interactive chart is available at <a href="https://paperswithcode.com/sota/image-classification-on-imagenet">https://paperswithcode.com/sota/image-classification-on-imagenet</a> . . . . .	33
4.6	Famous illustration of an adversarial example from the work of Goodfellow et al. [53]. The perturbation signal is voluntarily amplified to understand the addition (see the factor 0.007). . . . .	36
4.7	Similarity and difference between an evasion attack and training phase. . .	38
5.1	Illustration of the iterative game between attack and defense. The frontier between both sides is always blurry. There is an everlasting race to find issues. . . . .	43
5.2	Image A: 182: 'Border Terrier', Image B: 184: 'Irish Terrier', Image C: 185: 'Norwich Terrier', Image D: 186: 'Norfolk Terrier', Image E: 187: 'Yorkshire Terrier', Image F: 189: 'Lakeland Terrier'. . . . .	46

---

5.3	We extracted images from the validation set for two classes. Top row: images of class 657: ‘missile’. Bottom row: images of class 744: ‘projectile, missile’.	47
5.4	Figure 1 from [9] that helps visualize different types of redundancies within ImageNet pictures. Images highlighted with a green box are kept for an improved dataset while the others are deemed redundant. They are discarded.	48
5.5	Illustration from [8] showing images with multiple classes within. Red: original labels found in the dataset. Green: suggested improved labels if multiple annotations are considered.	49
5.6	Looking through images of 477: “carpenter’s kit, tool kit” reveals an logical abundance of 784: ‘screwdriver’.	49
5.7	Illustration of a perfect feature extractor. Every feature vector is equal, regardless of the transformation applied to the image. The object is perfectly conceptualized.	51
5.8	Figure 1 of [71] showing visually different images that share the same logits (displayed above) through a ResNet152. The first image is unmodified.	52
5.9	Illustration of class projection on a 2-label problem. Dashed lines illustrate the scalar product of $f_{l-1}(x)$ on each class vector. The natural image $x$ has a greater logit score on class 1. The modified adversarial image $x_a$ has been modified so it crosses class frontier. It is now classified as label 2.	53
5.10	Experiments of [58] over Imagenet validation dataset (50,000 images). y-axis is the accuracy, x-axis the $L_2$ -dissimilarity: $\frac{\ x_0 - x_a\ }{\ x_0\ }$ . Resnet-50 is attacked with Deepfool (left) and C&W (right).	58
5.11	Illustration of Feature Squeezing from [162]. Two <i>squeezers</i> are used here, but more could be added.	60
5.12	Illustration of training experimented by Ilyas et al. [68]. A robust dataset is used on the diagram (a) and a non-robust dataset is used on diagram (b).	61
5.13	2D projection of feature vectors of natural images, extracted from [105]. Comparison is given between their method and classical correlation and Softmax training. The projection of natural images and their adversarial counterpart is also displayed.	63
6.1	Operating curves of EfficientNet-b0 (left) and ResNet-50 (right) against four attacks in <i>best-effort mode</i> and without quantization.	73

---

6.2	Evolution of $d_{1/2}$ with the complexity budget for black box setup. Attacks on EfficientNet [142] . . . . .	77
6.3	Evolution of $d_{1/2}$ with the complexity budget for white box setup. Attacks on EfficientNet [142] . . . . .	78
6.4	Black-box $d_{1/2}$ as a function of white-box $d_{1/2}$ . . . . .	80
7.1	Operating curves of EfficientNet-b0 against FGSM and PGD in <i>best-effort mode</i> with floating point (plain) or quantized (dotted) pixel values. . . . .	83
7.2	Example of adversarial images quantized with our method. Attacked network is EfficientNet-b0. The predicted label is displayed below. . . . .	83
7.3	Comparing the approximated loss (7.9) (resp. (7.19)) with the loss without rounding: $L(\tilde{q}_\lambda)$ in (7.13) (resp. $L(J^{-1}\tilde{Q}_\lambda)$ in (7.22)) and the loss with quantization $L(q_\lambda^*)$ in (7.15) (resp. $L(J^{-1}Q_\lambda^*)$ in (7.23)) as a function of $\lambda$ in the (left) spatial domain, resp. (right) JPEG90. Distortion is also displayed with scale on the right. . . . .	93
7.4	The impact of the number of tested values of $\lambda$ on the operating curve. EfficientNet, BP, $d_f = 1$ , spatial (left) and JPEG90 (right). . . . .	94
7.5	Visible artifacts on images quantized as JPEG60. Predicted class is displayed in red, ground-truth in blue. . . . .	95
7.6	Operating curve of Efficientnet-b0 against BP + JPEG quantization ( $d_f = 1$ ). Distortion is calculated w.r.t. to the original image compressed in JPEG domain $X_o$ . . . . .	96
7.7	Operating curve of Efficientnet-b0 against BP + JPEG quantization ( $d_f = 1$ ). Distortion is calculated w.r.t. to original spatial image $x_o$ . Images that are already adversarial after compression are considered to have null distortion for readability. . . . .	96
7.8	The operating curve of Efficientnet-b0 against BP + JPEG quantization with $d_f = 1$ (plain) and against JPEG compression alone (dot). For the sake of comparison, the cumulative distribution function of the distortion due to JPEG compression is also displayed (dashed). Distortion is measured from the original spatial image $x_o$ . . . . .	97

---

7.9	Visual artifacts for two adversarial images on JPEG90 (top) and JPEG60 (bottom) with and without clipping. Visual artifacts on JPEG60 are due to the compression factor. Distortion is naturally high. JPEG 90 however represents low compression. Notice the non-smooth sky around the flamingo. Effect is especially visible for $d_f = \infty$ . . . . .	98
7.10	The operating curves of EfficientNet-b0 against BP in the spatial and JPEG domain with different degree of freedom. Distortion is measured from the original spatial image. . . . .	99
7.11	The operating curves of several classifiers against BP (plain) and PGD <sub>2</sub> (dotted) with quantization in spatial (left) and JPEG90 (right) domains. .	100
7.12	Misclassification of two images from our dataset through two classifiers: EfficientNet-b0 and RegNetX-032. . . . .	100
7.13	Examples of attacked images with spatial and JPEG75 quantizations. JPEG75 compression of the original image is also displayed in the third column. . .	103
8.1	Characteristic function of quantized attack (PGD in best-effort) against well known (vanilla) classifiers for ImageNet. . . . .	109
8.2	Rounding the minimizer when the stego-cost is quadratic. . . . .	114
8.3	Characteristic function of attack [12] (PGD in best effort with quantization) against Vanilla EfficientNet-b0 and its Robust counterpart. Distortion is not normalized for $n$ . . . . .	116
8.4	Top row: Cover images with their label below. Bottom row: adversarial images with steganographic embedding GINA ( $d_f=4$ ). Below them are their new label and the distortion . . . . .	119
9.1	Accuracies with downscaling. Stripes define max and min accuracies over the 6 downscales for a given model. Numbers are the sizes $\ell$ of the down-scaled images, $n = 3\ell^2$ . . . . .	126
9.2	Experimental measurement of the normalized adversarial distortion $\sigma_A(n)/\kappa(n)$ as a function of $\log_{10}(n)$ . . . . .	127
9.3	Attacking NFNet-F0 (left, $\ell = 256$ ) and NFNet-F4 (right, $\ell = 512$ ) through every downscaling method with the attack BP. . . . .	128
9.4	Illustration of our training procedure. Features are extracted from a natural image $x_0$ . An adversarial example of this image $x_a$ is built in parallel. Features of $x_a$ are then extracted. . . . .	163



---

9.5	Comparison between different optimization methods seen in Sect. 9.7.2. Attack is performed with $N_{iter} = 1$ and $\epsilon = 1$ . . . . .	167
9.6	Effect of the value of $\epsilon$ on our method. Optimization is done through MSE and attack is performed $N_{iter} = 1$ . State-of-the-Art [129](SotA) is displayed.	167
9.7	Effect of the number of steps $N_{iter}$ used during the attack. The attack is performed with $\epsilon = 1$ and the training procedure is optimized with MSE .	168

# LIST OF TABLES

---

5.1	Most popular image classification datasets. MS-COCO has 80 object categories and 91 ‘other’ categories such as background, sky, ground...	45
5.2	Class vectors in the fully-connected layer of popular models. $n_f$ is the input or <i>feature</i> dimension.	55
5.3	Correlations of classes vector in the fully-connected layer of Resnet18. Studied classes are expected to have high correlations:182:Border Terrier, 184:Irish Terrier, 185:Norwich Terrier, 186:Norfolk Terrier, 187:Yorkshire Terrier, and 189:Lakeland Terrier	55
5.4	Correlations of the same classes vector as Tab. 5.3. New classes are expected to have low correlations:579: grand piano, grand, 580: greenhouse, nursery, glasshouse, 581: grille, radiator grille, 582: grocery store, 583: guillotine, and 584: hair slide	56
6.1	Benchmarks Comparison. Average Runtimes per ImageNet Image with ResNet50 [98].	72
6.2	Benchmark of models with 1.000 ImageNet Images	79
7.1	Accuracy (in %) and mean average distortion of FGSM and PGD <sub>2</sub> attacks against EfficientNet-b0 in <i>best-effort</i> mode (see section 6.4.2) over 1,000 randomly selected images from ImageNet.	85
7.2	Accuracy (in %) of EfficientNet-b0 equipped with a JPEG compression as a defense front-end reformer against our quantized <i>best-effort</i> BP.	102
8.1	Robustness of recent classifiers against PGD <sub>2</sub> followed by quantization	109
8.2	Detection probabilities (TPR <sub>5</sub> ) with forensics detectors of adversarial images targeting classifier vanilla EfficientNet-b0 [142]	116
8.3	Undetectability of steganographic embedding on PGD <sub>2</sub> against the vanilla model (Van) and its robust version (Rob).	117
8.4	Detection on SCRMPQ1 after training on adversarial images embedded with GINA ( $d_f=4$ )	120

---

9.1	Distortion for attacking 90% of the images for the EfficientNet family . . .	129
9.2	Accuracy (%) when downscaling from size $L = 600$ to $\ell = 224$ for EfficientNet-b0 . . . . .	129
9.3	Correlations of $C\omega$ matrices from different classifiers. Three architectures are studied: EfficientNet-b0, Resnet18, and Resnet50. Three robustification methods are studied: 1 [129]( $\epsilon = 0.05$ ), 2 [129]( $\epsilon = 0.5$ ), and 3 [98]( $\epsilon = 3.0$ ). Results are symmetrical, only half is displayed for clarity. . . . .	166

# BIBLIOGRAPHY

---

- [1] M. Andriushchenko et al., « Square attack: a query-efficient black-box adversarial attack via random search », *in: ECCV*, 2020.
- [2] 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, pp. 888–897.
- [3] Anish Athalye, Nicholas Carlini, and David A. Wagner, « Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples », *in: ICML*, 2018, pp. 274–283.
- [4] Anish Athalye et al., « Synthesizing robust adversarial examples », *in: International conference on machine learning*, PMLR, 2018, pp. 284–293.
- [5] Mauro Barni, Matthew C Stamm, and Benedetta Tondi, « Adversarial multimedia forensics: Overview and challenges ahead », *in: 2018 26th European Signal Processing Conference (EUSIPCO)*, IEEE, 2018, pp. 962–966.
- [6] Richard Bellman, « Dynamic programming », *in: Science* 153.3731 (1966), pp. 34–37.
- [7] S. Bernard et al., « Explicit Optimization of min max Steganographic Game », *in: IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 812–823.
- [8] Lucas Beyer et al., « Are we done with imagenet? », *in: arXiv preprint arXiv:2006.07159* (2020).
- [9] Vighnesh Birodkar, Hossein Mobahi, and Samy Bengio, « Semantic Redundancies in Image-Classification Datasets: The 10% You Don’t Need », *in: arXiv preprint arXiv:1901.11409* (2019).
- [10] Benoit Bonnet, Teddy Furon, and Patrick Bas, « Fooling an Automatic Image Quality Estimator », *in: MediaEval 2020 - MediaEval Benchmarking Initiative for Multimedia Evaluation*, Online, United States, Dec. 2020, pp. 1–4, URL: <https://hal.archives-ouvertes.fr/hal-03132891>.

- 
- [11] Benoit Bonnet, Teddy Furon, and Patrick Bas, « Forensics through stega glasses: the case of adversarial images », *in: International Conference on Pattern Recognition*, Springer, 2021, pp. 453–469.
- [12] Benoit Bonnet, Teddy Furon, and Patrick Bas, « What If Adversarial Samples Were Digital Images? », *in: Proc. of ACM IH&MMSec '20*, Denver, CO, USA, 2020, pp. 55–66, ISBN: 9781450370509, DOI: 10.1145/3369412.3395062.
- [13] Mehdi Boroumand, Mo Chen, and Jessica Fridrich, « Deep residual network for steganalysis of digital images », *in: IEEE Transactions on Information Forensics and Security* 14.5 (2018), pp. 1181–1193.
- [14] W. Brendel, J. Rauber, and M. Bethge, « Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models », *in: ICLR*, 2018.
- [15] John Bridle, « Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters », *in: Advances in neural information processing systems* 2 (1989).
- [16] Andy Brock et al., « High-performance large-scale image recognition without normalization », *in: International Conference on Machine Learning*, PMLR, 2021, pp. 1059–1071.
- [17] Antoine Buetti-Dinh et al., « Deep neural networks outperform human expert’s capacity in characterizing bioleaching bacterial biofilm composition », *in: Biotechnology Reports* 22 (2019), e00321.
- [18] Michael Calonder et al., « Brief: Binary robust independent elementary features », *in: European conference on computer vision*, Springer, 2010, pp. 778–792.
- [19] Nicholas Carlini and David Wagner, « Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods », *in: arXiv:1705.07263* (2017).
- [20] Nicholas Carlini and David Wagner, « Audio adversarial examples: Targeted attacks on speech-to-text », *in: 2018 IEEE security and privacy workshops (SPW)*, IEEE, 2018, pp. 1–7.
- [21] Nicholas Carlini and David Wagner, « Towards evaluating the robustness of neural networks », *in: IEEE Symp. on Security and Privacy*, 2017.
- [22] Nandish Chattopadhyay et al., « Curse of Dimensionality in Adversarial Examples », *in: 2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8, DOI: 10.1109/IJCNN.2019.8851795.

- 
- [23] J. Chen and Q. Gu, « RayS: A Ray Searching Method for Hard-Label Adversarial Attack », *in: SIGKDD*, 2020, ISBN: 9781450379984, DOI: 10.1145/3394486.3403225, URL: <https://doi.org/10.1145/3394486.3403225>.
- [24] Jianbo Chen, Michael I Jordan, and Martin J Wainwright, « Hopskipjumpattack: A query-efficient decision-based attack », *in: 2020 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2020, pp. 1277–1294.
- [25] Z. Chen et al., « A gradient-based pixel-domain attack against SVM detection of global image manipulations », *in: IEEE WIFS*, 2017, pp. 1–6.
- [26] Moustapha Cisse et al., « Houdini: Fooling deep structured prediction models », *in: arXiv preprint arXiv:1707.05373* (2017).
- [27] Rémi Cogranne et al., « Is ensemble classifier needed for steganalysis in high-dimensional feature spaces? », *in: Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, IEEE, 2015, pp. 1–6.
- [28] Daniel Crevier, *AI: the tumultuous history of the search for artificial intelligence*, Basic Books, Inc., 1993.
- [29] F. Croce et al., « RobustBench: a standardized adversarial robustness benchmark », *in: arXiv preprint arXiv:2010.09670* (2020).
- [30] Ekin D Cubuk et al., « Intriguing properties of adversarial examples », *in: arXiv preprint arXiv:1711.02846* (2017).
- [31] D. Han et al., « ReXNet: Diminishing Representational Bottleneck on CNN », *in: arXiv e-prints arXiv:2007.00992* (2020).
- [32] Jia Deng et al., « Imagenet: A large-scale hierarchical image database », *in: 2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
- [33] Jacob Devlin et al., « Bert: Pre-training of deep bidirectional transformers for language understanding », *in: arXiv preprint arXiv:1810.04805* (2018).
- [34] Guneet S Dhillon et al., « Stochastic activation pruning for robust adversarial defense », *in: arXiv preprint arXiv:1803.01442* (2018).

- 
- [35] Elvis Dohmatob, « Generalized No Free Lunch Theorem for Adversarial Robustness », in: *Proceedings of the 36th International Conference on Machine Learning*, ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov, vol. 97, Proceedings of Machine Learning Research, PMLR, June 2019, pp. 1646–1654, URL: <https://proceedings.mlr.press/v97/dohmatob19a.html>.
- [36] Y. Dong et al., « Benchmarking Adversarial Robustness on Image Classification », in: *CVPR*, 2020.
- [37] Alexey Dosovitskiy et al., « An image is worth 16x16 words: Transformers for image recognition at scale », in: *arXiv preprint arXiv:2010.11929* (2020).
- [38] Simant Dube, *High Dimensional Spaces, Deep Learning and Adversarial Examples*, 2018, arXiv: 1801.00634 [cs.CV].
- [39] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein, « Adversarial reprogramming of neural networks », in: *arXiv preprint arXiv:1806.11146* (2018).
- [40] W. Fan, K. Wang, and F. Cayre, « General-purpose image forensics using patch likelihood under image statistical models », in: *IEEE Int. Workshop on Information Forensics and Security (WIFS)*, 2015, pp. 1–6.
- [41] Sundus Farooq, Muhammad Haroon Yousaf, and Fawad Hussain, « A generic passive image forgery detection scheme using local binary pattern with rich models », in: *Computers & Electrical Engineering* 62 (2017), pp. 459–472, ISSN: 0045-7906, DOI: <https://doi.org/10.1016/j.compeleceng.2017.05.008>.
- [42] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi, « Adversarial vulnerability for any classifier », in: *NeuroIPS 2018*, Montreal, Canada, Dec. 2018, URL: <https://hal.archives-ouvertes.fr/hal-01990465>.
- [43] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard, « Robustness of classifiers: from adversarial to random noise », in: *Advances in neural information processing systems* 29 (2016).
- [44] Tomáš Filler, Jan Judas, and Jessica Fridrich, « Minimizing additive distortion in steganography using syndrome-trellis codes », in: *IEEE Transactions on Information Forensics and Security* 6.3 (2011), pp. 920–935.
- [45] Volker Fischer et al., « Adversarial examples for semantic image segmentation », in: *arXiv preprint arXiv:1703.01101* (2017).

- 
- [46] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart, « Model inversion attacks that exploit confidence information and basic countermeasures », *in: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.
- [47] Jessica Fridrich and Jan Kodovsky, « Rich models for steganalysis of digital images », *in: Information Forensics and Security, IEEE Transactions on* 7.3 (2012), pp. 868–882.
- [48] Kunihiko Fukushima and Sei Miyake, « Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition », *in: Competition and cooperation in neural nets*, Springer, 1982, pp. 267–285.
- [49] Angus Galloway et al., « Batch normalization is a cause of adversarial vulnerability », *in: arXiv preprint arXiv:1905.02161* (2019).
- [50] Justin Gilmer et al., *Adversarial Spheres*, 2018, arXiv: 1801.02774 [cs.CV].
- [51] Micah Goldblum et al., « Adversarially robust distillation », *in: Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 04, 2020, pp. 3996–4003.
- [52] Miroslav Goljan, Jessica Fridrich, and Rémi Coganne, « Rich model for Steganalysis of color images », *in: 2014 IEEE International Workshop on Information Forensics and Security, WIFS 2014* (Apr. 2015), pp. 185–190, DOI: 10.1109/WIFS.2014.7084325.
- [53] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, « Explaining and Harnessing Adversarial Examples », *in: ICLR 2015, San Diego, CA, USA*, 2015.
- [54] Ian Goodfellow et al., « Generative adversarial networks », *in: Communications of the ACM* 63.11 (2020), pp. 139–144.
- [55] David Güera et al., « A counter-forensic method for CNN-based camera model identification », *in: 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, IEEE, 2017, pp. 1840–1847.
- [56] Chong Guo et al., « Adversarially trained neural representations may already be as robust as corresponding biological neural representations », *in: arXiv preprint arXiv:2206.11228* (2022).
- [57] Chuan Guo, Jared S Frank, and Kilian Q Weinberger, « Low frequency adversarial perturbation », *in: arXiv preprint arXiv:1809.08758* (2018).



- 
- [58] Chuan Guo et al., « Countering adversarial images using input transformations », *in: arXiv preprint arXiv:1711.00117* (2017).
- [59] Miska M. Hannuksela, Jani Lainema, and Vinod K. Malamal Vadakital, « The High Efficiency Image File Format Standard [Standards in a Nutshell] », *in: IEEE Signal Processing Magazine* 32.4 (2015), pp. 150–156, DOI: 10.1109/MSP.2015.2419292.
- [60] Jamie Hayes et al., « Logan: Membership inference attacks against generative models », *in: arXiv preprint arXiv:1705.07663* (2017).
- [61] K. He et al., « Deep Residual Learning for Image Recognition », *in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [62] Yang He et al., « Segmentations-leak: Membership inference attacks and defenses in semantic image segmentation », *in: European Conference on Computer Vision*, Springer, 2020, pp. 519–535.
- [63] Geoffrey E Hinton et al., « Improving neural networks by preventing co-adaptation of feature detectors », *in: arXiv preprint arXiv:1207.0580* (2012).
- [64] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky, « Neural networks for machine learning lecture 6a overview of mini-batch gradient descent », *in: Cited on* 14.8 (2012), p. 2.
- [65] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al., « Distilling the knowledge in a neural network », *in: arXiv preprint arXiv:1503.02531* 2.7 (2015).
- [66] Jie Hu, Li Shen, and Gang Sun, « Squeeze-and-Excitation Networks », *in: CVPR*, 2018.
- [67] Sandy Huang et al., « Adversarial attacks on neural network policies », *in: arXiv preprint arXiv:1702.02284* (2017).
- [68] Andrew Ilyas et al., « Adversarial examples are not bugs, they are features », *in: Advances in neural information processing systems* 32 (2019).
- [69] Sergey Ioffe and Christian Szegedy, « Batch normalization: Accelerating deep network training by reducing internal covariate shift », *in: International conference on machine learning*, PMLR, 2015, pp. 448–456.
- [70] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon, « i-revnet: Deep invertible networks », *in: arXiv preprint arXiv:1802.07088* (2018).

- 
- [71] Jörn-Henrik Jacobsen et al., « Excessive invariance causes adversarial vulnerability », *in: arXiv preprint arXiv:1811.00401* (2018).
  - [72] Linxi Jiang et al., « Black-box adversarial attacks on video recognition models », *in: Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 864–872.
  - [73] Di Jin et al., « Is bert really robust? a strong baseline for natural language attack on text classification and entailment », *in: Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 05, 2020, pp. 8018–8025.
  - [74] Harini Kannan, Alexey Kurakin, and Ian Goodfellow, « Adversarial logit pairing », *in: arXiv preprint arXiv:1803.06373* (2018).
  - [75] Kenji Kawaguchi, Jiaoyang Huang, and Leslie Pack Kaelbling, « Every local minimum value is the global minimum value of induced model in nonconvex machine learning », *in: Neural Computation* 31.12 (2019), pp. 2293–2323.
  - [76] Henry J Kelley, « Gradient theory of optimal flight paths », *in: Ars Journal* 30.10 (1960), pp. 947–954.
  - [77] Diederik P Kingma and Jimmy Ba, « Adam: A method for stochastic optimization », *in: arXiv preprint arXiv:1412.6980* (2014).
  - [78] Marius Kloft and Pavel Laskov, « Online anomaly detection under adversarial impact », *in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings*, 2010, pp. 405–412.
  - [79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, « ImageNet Classification with Deep Convolutional Neural Networks », *in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12*, Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.
  - [80] Alexey Kurakin et al., « Adversarial attacks and defences competition », *in: The NIPS'17 Competition: Building Intelligent Systems*, Springer, 2018, pp. 195–231.
  - [81] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio, « Adversarial examples in the physical world », *in: Artificial intelligence safety and security*, Chapman and Hall/CRC, 2018, pp. 99–112.
  - [82] Bethge Lab, « Robust Vision Benchmark », <https://robust.vision>, URL: %5Curl%7Bhttps://robust.vision%7D (visited on ).

- 
- [83] Yann LeCun et al., « Gradient-based learning applied to document recognition », *in: Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
  - [84] Bin Li et al., « A new cost function for spatial image steganography », *in: Image Processing (ICIP), 2014 IEEE International Conference on*, IEEE, 2014, pp. 4206–4210.
  - [85] Bin Li et al., « A strategy of clustering modification directions in spatial image steganography », *in: Information Forensics and Security, IEEE Trans. on* 10.9 (2015).
  - [86] Huichen Li et al., « Qeba: Query-efficient boundary-based blackbox attack », *in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1221–1230.
  - [87] Shasha Li et al., « Adversarial perturbations against real-time video classification systems », *in: arXiv preprint arXiv:1807.00458* (2018).
  - [88] Y. Chen and J. Li et al., « Dual Path Networks », *in: NIPS*, 2017.
  - [89] Siyuan Liang et al., « Efficient adversarial attacks for visual object tracking », *in: European Conference on Computer Vision*, Springer, 2020, pp. 34–50.
  - [90] Yen-Chen Lin et al., « Tactics of adversarial attack on deep reinforcement learning agents », *in: arXiv preprint arXiv:1703.06748* (2017).
  - [91] J. Liu et al., « Detection Based Defense Against Adversarial Examples From the Steganalysis Point of View », *in: IEEE/CVF CVPR*, 2019, pp. 4820–4829.
  - [92] Yingqi Liu et al., « Trojaning attack on neural networks », *in:* (2017).
  - [93] Zihao Liu et al., *Feature Distillation: DNN-Oriented JPEG Compression Against Adversarial Examples*, 2019, arXiv: 1803.05787 [cs.CV].
  - [94] David G Lowe, « Distinctive image features from scale-invariant keypoints », *in: International journal of computer vision* 60.2 (2004), pp. 91–110.
  - [95] Weiqi Luo et al., « Improved Audio Steganalytic Feature and Its Applications in Audio Forensics », *in: ACM Trans. Multimedia Comput. Commun. Appl.* 14.2 (Apr. 2018), ISSN: 1551-6857, DOI: 10.1145/3190575.
  - [96] Chen Ma, Li Chen, and Jun-Hai Yong, « Simulating unknown target models for query-efficient black-box attacks », *in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11835–11844.

- 
- [97] Xingjun Ma et al., « Characterizing adversarial subspaces using local intrinsic dimensionality », *in: arXiv preprint arXiv:1801.02613* (2018).
- [98] Aleksander Madry et al., « Towards Deep Learning Models Resistant to Adversarial Attacks », *in: ICLR 2018, Vancouver, BC, Canada*. 2018.
- [99] Thibault Maho, Teddy Furon, and Erwan Le Merrer, « Surfing: a fast surrogate-free black-box attack », *in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10430–10439.
- [100] Thibault Maho et al., *RoBIC: A benchmark suite for assessing classifiers robustness*, 2021, arXiv: 2102.05368 [cs.CV].
- [101] S. Marcel and Y. Rodriguez, « Torchvision the Machine-Vision Package of Torch », *in: ACM Multimedia*, 2010, DOI: 10.1145/1873951.1874254, URL: <https://doi.org/10.1145/1873951.1874254>.
- [102] John McCarthy et al., « A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955 », *in: AI magazine* 27.4 (2006), pp. 12–12.
- [103] Jan Hendrik Metzen et al., « On detecting adversarial perturbations », *in: arXiv preprint arXiv:1702.04267* (2017).
- [104] 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, pp. 2574–2582.
- [105] Aamir Mustafa et al., « Adversarial defense by restricting the hidden space of deep neural networks », *in: Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3385–3394.
- [106] Blaine Nelson et al., « Exploiting machine learning to subvert your spam filter. », *in: LEET* 8.1 (2008), p. 9.
- [107] Curtis G Northcutt, Anish Athalye, and Jonas Mueller, « Pervasive label errors in test sets destabilize machine learning benchmarks », *in: arXiv preprint arXiv:2103.14749* (2021).
- [108] Guillermo Ortiz-Jiménez et al., « Optimism in the face of adversity: Understanding and improving deep learning through adversarial robustness », *in: Proceedings of the IEEE* 109.5 (2021), pp. 635–659.

- 
- [109] Tianyu Pang et al., « Rethinking softmax cross-entropy loss for adversarial robustness », *in: arXiv preprint arXiv:1905.10626* (2019).
  - [110] Nicolas Papernot et al., « Distillation as a defense to adversarial perturbations against deep neural networks », *in: 2016 IEEE symposium on security and privacy (SP)*, IEEE, 2016, pp. 582–597.
  - [111] Nicolas Papernot et al., « Practical black-box attacks against machine learning », *in: Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.
  - [112] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu, *On Buggy Resizing Libraries and Surprising Subtleties in FID Calculation*, 2021, arXiv: 2104.11222 [cs.CV].
  - [113] Ning Qian, « On the momentum term in gradient descent learning algorithms », *in: Neural networks 12.1* (1999), pp. 145–151.
  - [114] Xiaoqing Qiu et al., « A Universal Image Forensic Strategy Based on Steganalytic Model », *in: Proc. of ACM IH&MMSec '14*, Salzburg, Austria, 2014, pp. 165–170, ISBN: 9781450326476, DOI: 10.1145/2600918.2600941.
  - [115] E. Quiring, D. Arp, and K. Rieck, « Forgotten Siblings: Unifying Attacks on Machine Learning and Digital Watermarking », *in: IEEE European Symp. on Security and Privacy*, 2018, DOI: 10.1109/EuroSP.2018.00041.
  - [116] Ilija Radosavovic et al., « Designing Network Design Spaces », *in: CVPR*, 2020.
  - [117] Aditi Raghunathan et al., « Understanding and mitigating the tradeoff between robustness and accuracy », *in: arXiv preprint arXiv:2002.10716* (2020).
  - [118] Ali Rahmati et al., « GeoDA: a geometric framework for black-box adversarial attacks », *in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8446–8455.
  - [119] Aditya Ramesh et al., « Hierarchical text-conditional image generation with clip latents », *in: arXiv preprint arXiv:2204.06125* (2022).
  - [120] Leslie Rice, Eric Wong, and J. Zico Kolter, « Overfitting in adversarially robust deep learning », *in: CoRR abs/2002.11569* (2020), arXiv: 2002.11569, URL: <https://arxiv.org/abs/2002.11569>.

- 
- [121] Jonathan G Richens, Ciaran M Lee, and Saurabh Johri, « Improving the accuracy of medical diagnosis with causal machine learning », *in: Nature communications* 11.1 (2020), pp. 1–9.
- [122] Robin Rombach et al., *High-Resolution Image Synthesis with Latent Diffusion Models*, 2021, arXiv: 2112.10752 [cs.CV].
- [123] Jérôme Rony et al., « Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses », *in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4322–4330.
- [124] Frank Rosenblatt, *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*, tech. rep., Cornell Aeronautical Lab Inc Buffalo NY, 1961.
- [125] Edward Rosten and Tom Drummond, « Fusing points and lines for high performance tracking », *in: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, Ieee, 2005, pp. 1508–1515.
- [126] Ethan Rublee et al., « ORB: An efficient alternative to SIFT or SURF », *in: 2011 International conference on computer vision*, Ieee, 2011, pp. 2564–2571.
- [127] Olga Russakovsky et al., « Imagenet large scale visual recognition challenge », *in: International journal of computer vision* 115.3 (2015), pp. 211–252.
- [128] Chitwan Saharia et al., « Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding », *in: arXiv preprint arXiv:2205.11487* (2022).
- [129] Hadi Salman et al., « Do adversarially robust imagenet models transfer better? », *in: Advances in Neural Information Processing Systems* 33 (2020), pp. 3533–3545.
- [130] Pouya Samangouei, Maya Kabkab, and Rama Chellappa, « Defense-gan: Protecting classifiers against adversarial attacks using generative models », *in: arXiv preprint arXiv:1805.06605* (2018).
- [131] Mark Sandler et al., « MobileNetV2: Inverted Residuals and Linear Bottlenecks », *in: CVPR*, 2018.
- [132] P. Schöttle et al., « Detecting Adversarial Examples - a Lesson from Multimedia Security », *in: European Signal Processing Conference (EUSIPCO)*, 2018, pp. 947–951.

- 
- [133] Vahid Sedighi, Rémi Cogranne, and Jessica Fridrich, « Content-Adaptive Steganography by Minimizing Statistical Detectability », *in: Information Forensics and Security, IEEE Transactions on* 11.2 (2016), pp. 221–234.
- [134] Uri Shaham et al., *Defending against Adversarial Images using Basis Functions Transformations*, 2018, arXiv: 1803.10840 [stat.ML].
- [135] Claude Elwood Shannon, « A mathematical theory of communication », *in: The Bell system technical journal* 27.3 (1948), pp. 379–423.
- [136] Xiaoyu Shi et al., « CNN-based steganalysis and parametric adversarial embedding: a game-theoretic framework », *in: Signal Processing: Image Communication* (2020), p. 115992.
- [137] Karen Simonyan and Andrew Zisserman, « Very Deep Convolutional Networks for Large-Scale Image Recognition », *in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015, URL: <http://arxiv.org/abs/1409.1556>.
- [138] Yang Song et al., « Pixeldefend: Leveraging generative models to understand and defend against adversarial examples », *in: arXiv preprint arXiv:1710.10766* (2017).
- [139] C. Szegedy et al., « Going deeper with convolutions », *in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
- [140] Christian Szegedy et al., *Going Deeper with Convolutions*, 2014, arXiv: 1409.4842 [cs.CV].
- [141] Christian Szegedy et al., « Intriguing properties of neural networks », *in: International Conference on Learning Representations*, 2014, URL: <http://arxiv.org/abs/1312.6199>.
- [142] Mingxing Tan and Quoc Le, « EfficientNet: Rethinking Model Scaling for CNN », *in: ICML*, 2019, URL: <http://proceedings.mlr.press/v97/tan19a.html>.
- [143] Mingxing Tan and Quoc Le, « EfficientNetV2: Smaller Models and Faster Training », *in: Proceedings of the 38th International Conference on Machine Learning*, ed. by Marina Meila and Tong Zhang, vol. 139, Proceedings of Machine Learning Research, PMLR, July 2021, pp. 10096–10106, URL: <https://proceedings.mlr.press/v139/tan21a.html>.
- [144] Mingxing Tan and Quoc V. Le, « MixConv: Mixed Depthwise Convolutional Kernels », *in: arXiv e-prints arxiv:1907.09595* (), eprint: 1907.09595 (cs.CV).

- 
- [145] W. Tang et al., « CNN-Based Adversarial Embedding for Image Steganography », *in: IEEE Transactions on Information Forensics and Security* 14.8 (2019), pp. 2074–2087.
- [146] Florian Tramèr et al., *On Adaptive Attacks to Adversarial Example Defenses*, 2020, arXiv: 2002.08347 [cs.LG].
- [147] Florian Tramèr et al., « Stealing machine learning models via prediction {APIs} », *in: 25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 601–618.
- [148] Dimitris Tsipras et al., « Robustness May Be at Odds with Accuracy », *in:* (2018), arXiv: 1805.12152 [stat.ML].
- [149] A. Tuama, F. Comby, and M. Chaumont, « Camera model identification based machine learning approach with high order statistics features », *in: EUSIPCO*, 2016, pp. 1183–1187.
- [150] Ashish Vaswani et al., « Attention is all you need », *in: Advances in neural information processing systems* 30 (2017).
- [151] Gregory K Wallace, « The JPEG still picture compression standard », *in: IEEE transactions on consumer electronics* 38.1 (1992), pp. xviii–xxxiv.
- [152] C.-Y. Wang et al., « CSPNet: A new backbone that can enhance learning capability of CNN », *in: CVPR Workshops*, 2020.
- [153] Tianlu Wang et al., « Cat-gen: Improving robustness in nlp models via controlled adversarial text generation », *in: arXiv preprint arXiv:2010.02338* (2020).
- [154] Xiao Wang et al., « Protecting neural networks with hierarchical random switching: Towards better robustness-accuracy trade-off for stochastic defenses », *in: arXiv preprint arXiv:1908.07116* (2019).
- [155] Yaofei Wang et al., « Non-Additive Cost Functions for Color Image Steganography Based on Inter-Channel Correlations and Differences », *in: IEEE Trans. on Information Forensics and Security* (2019).
- [156] Xingxing Wei et al., « Transferable adversarial attacks for image and video object detection », *in: arXiv preprint arXiv:1811.12641* (2018).
- [157] Ross Wightman, *PyTorch Image Models*, <https://github.com/rwightman/pytorch-image-models>, 2019, DOI: 10.5281/zenodo.4414861.



- 
- [158] Cihang Xie et al., « Adversarial Examples Improve Image Recognition », *in: CVPR*, 2020.
- [159] Cihang Xie et al., « Mitigating adversarial effects through randomization », *in: arXiv preprint arXiv:1711.01991* (2017).
- [160] Qizhe Xie et al., « Self-Training With Noisy Student Improves ImageNet Classification », *in: CVPR*, 2020.
- [161] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi, « Structural design of convolutional neural networks for steganalysis », *in: IEEE Signal Processing Letters* 23.5 (2016), pp. 708–712.
- [162] Weilin Xu, David Evans, and Yanjun Qi, « Feature squeezing: Detecting adversarial examples in deep neural networks », *in: arXiv preprint arXiv:1704.01155* (2017).
- [163] Jiancheng Yang et al., « Learning black-box attackers with transferable priors and query feedback », *in: Advances in Neural Information Processing Systems* 33 (2020), pp. 12288–12299.
- [164] Yassine Yousfi et al., « Breaking ALASKA: Color separation for steganalysis in JPEG domain », *in: Proc. of ACM IH&MMSec '19*, 2019, pp. 138–149.
- [165] A. Zandi et al., « CREW: Compression with Reversible Embedded Wavelets », *in: Proceedings DCC '95 Data Compression Conference*, 1995, pp. 212–221, DOI: 10.1109/DCC.1995.515511.
- [166] Hanwei Zhang et al., « Walking on the Edge: Fast, Low-Distortion Adversarial Examples », *in: IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 701–713, ISSN: 1556-6021, DOI: 10.1109/tifs.2020.3021899, URL: <http://dx.doi.org/10.1109/TIFS.2020.3021899>.
- [167] Hongyang Zhang et al., « Theoretically principled trade-off between robustness and accuracy », *in: International conference on machine learning*, PMLR, 2019, pp. 7472–7482.
- [168] Yuheng Zhang et al., « The secret revealer: Generative model-inversion attacks against deep neural networks », *in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 253–261.
- [169] Shuyuan Y. Zhu et al., « High-Quality Color Image Compression by Quantization Crossing Color Spaces », *in: IEEE Transactions on Circuits and Systems for Video Technology* 29.5 (2019), pp. 1474–1487, DOI: 10.1109/TCSVT.2018.2841642.

PART III

# Appendix A

---

# Fooling an Automatic Image Quality Estimator

Benoit Bonnet<sup>1</sup>, Teddy Furon<sup>1</sup>, Patrick Bas<sup>2</sup>

<sup>1</sup>Univ. Rennes, Inria, CNRS, IRISA Rennes, France

<sup>2</sup>Univ. Lille, CNRS, Centrale Lille, UMR 9189, CRISTAL, Lille, France

contact:benoit.bonnet@inria.fr

## ABSTRACT

This paper presents our work on the 2020 MediaEval task: "Pixel Privacy: Quality Camouflage for Social Images". Blind Image Quality Assessment (BIQA) is an algorithm predicting a quality score for any given image. Our task is to modify an image to decrease its BIQA score while maintaining a good perceived quality. Since BIQA is a deep neural network, we worked on an adversarial attack approach of the problem.

## 1 INTRODUCTION

The internet is flooded with images. This is especially true with the growth of social networks over the last decade. All this data is used to perform analysis to bring out new trends or to train predictive models. When it comes to images, deep neural networks vastly lead the landscape of machine learning. These deep neural networks are especially known to thrive on big datasets. This leads to the idea that more data leads to better models. While there certainly is truth to that affirmation, better learning mostly comes out of *better* data. Good data is data that both fits the task (e.g. people, places, objects detection) and whose quality is good. Due to the amount of available data, a human could not perform this cherry-picking of good data. Automated classifiers like BIQA [4] have been trained to assess the quality of an image. This classifier was trained on images whose quality was labeled based on the perceived quality of the media (e.g. resolution, compression artifacts). To protect one's data, images can be manipulated and slightly modified to defeat the automatic quality assessment [6]. We chose an adversarial attack approach to achieve this goal.

## 2 APPROACH

### 2.1 Adversarial Examples

Adversarial examples were first introduced by Szegedy *et al.* [8] in early 2014. They are usually studied in the case of image classification: An attack effectively crafts a perturbation of an image to a small extent but enough to fool even the best classifiers.

In this setup, an original image  $x_0$  is given as an input to the trained neural network to estimate the probabilities  $(\hat{p}_k(x_0))_k$  of being from class  $k \in \{1, \dots, K\}$ . The predicted class is given by:

$$\hat{c}(x_0) = \arg \max_k \hat{p}_k(x_0). \quad (1)$$

The classification is correct if  $\hat{c}(x_0) = c(x_0)$  the ground truth class for  $x_0$ . The goal of an attack is to craft an imperceptible perturbation  $p$  such that the adversarial sample  $x_a = x_0 + p$  verifies ideally:

$$x_a^* = \arg \min_{x: \hat{c}(x) \neq c(x_0)} \|x - x_0\|, \quad (2)$$

Where  $\|\cdot\|$  is a measure of distortion, in most cases the Euclidean distance. A small distortion makes it less likely for human to perceive that the image was manipulated.

BIQA is a deep neural network and as such is vulnerable to adversarial attacks. However BIQA is not a classifier returning a class prediction but a regressor giving a quality score  $BIQA(x) \in [0, 100]$ . The notion of adversarial sample thus needs to be redefined. In our case, we set a target score  $s_a \in [0, 100]$ . Regardless of the original score  $BIQA(x_0)$ , our adversarial sample now ideally verifies:

$$x_a^* = \arg \min_{x: BIQA(x) < s_a} \|x - x_0\|, \quad (3)$$

### 2.2 Quantization

An original image  $x_0$  in the spatial domain (e.g. PNG format) is a 3-dimensional discrete tensor:  $x_0 \in \{0, 1, \dots, 255\}^n$  (with  $n = 3 \times R \times C$ , 3 color channels,  $R$  rows and  $C$  columns of pixels). The main objective of this task is to craft images:  $x_a \in \{0, 1, \dots, 255\}^n$ . This additional constraint to the attack is yet not easy to enforce.

In a deep neural network, this input image is first *preprocessed* onto a range domain that usually reduces variance of the data. Its purpose is to ease the learning phase and thus to increase the performance of a deep neural network. This *preprocessing* is defined by design before the training stage and cannot be modified at testing. In the case of BIQA, the range domain is  $[-0.5, 0.5]^n$ .

Most attacks of the literature are performed in this domain without consideration of the transformation it represents. This leads to an adversarial sample  $x_a \in [0, 255]^n$  after reverting the preprocessing. To save this adversarial sample  $x_a$  as an image, the first step is then to round it which will erase most of the perturbation in the case of a low-distortion attack. Rounding is therefore likely to remove the adversarial property of the sample.

Paper [1] addresses this problem presenting a *post-processing* added on top of any attack to efficiently quantize a perturbation: It keeps the adversarial property while lowering the added distortion. The method is based on a classification loss to ensure adversariality defined as follows:

$$L(x) = \log(\hat{p}_{c(x_0)}(x)) - \log(\hat{p}_{\hat{c}(x)}(x)). \quad (4)$$

To adapt this method to the context of BIQA, we only need to redefine it to:

$$L(x) = BIQA(x) - s_a. \quad (5)$$

For a given  $x$ ,  $L(x) < 0$  ensures  $x$  scores under the target  $s_a$ .

### 3 EXPERIMENTAL WORK

In this task, we know the classifier (BIQA) and its parameters. We are therefore in a *white-box* setup. Most modern attacks are developed in this scenario, from the most basic FGSM [3] and IFGSM [5] to the most advanced PGD [7], C&W [2], BP [10]. FGSM is a non-iterative attack bringing a fast solution of the problem. Our work used this attack in the early stages as a proof of concept bringing a quick further understanding of the problem. Artifacts were visible. Instead all the results reported here are crafted using more the advanced PGD attack [7] in its  $L_2$  optimization version. One input parameter is the distortion budget. We run the attack over 7 iterations with different distortion budgets (whose maximum value is set to 2000). A binary search quickly finds an adversarial sample with the lowest distortion.

#### 3.1 JPEG compression

The final images will be evaluated on their JPEG [9] counterpart. This compression is done with a quality factor of 90. However there are many image compression softwares providing different results. We used the command line `$ convert` to simulate this compression.

Tables 1 and 2 show for different methods both  $P_{PNG}$  and  $P_{JPEG}$  respectively the percentage of images successfully beating the target score in the PNG domain and the JPEG domain. Additionally Table 2 shows results of the jury as well.

#### 3.2 Quantization

**3.2.1 Spatial domain.** The work [1] serves as a baseline for quantization. We only slightly adapt it as stated in Sect. 2.2. Table 1 reports our results for two target scores:  $s_a = 30$  and  $s_a = 50$ . It appears that the perturbation crafted in the pixel domain is fragile when facing a JPEG compression.

**3.2.2 DCT domain.** The final image being evaluated after a JPEG compression, we explore a method adapting the quantization [1] to the DCT domain. Using the same notations [1]: Let  $X_o$  denote the image in the DCT domain,  $X_a = X_o + P$  is the result of an initial attack like PGD, and  $X_q = X_o + P + Q$  the final quantized transformed coefficients. We solve a Lagrangian formulation:

$$X_q = X_o + P + \arg \min_Q D(Q) + \lambda L(Q), \quad (6)$$

where  $\lambda$  is the Lagrangian multiplier controlling the tradeoff between the distortion  $D(Q)$  and  $L(Q)$  defined in (5). The distortion  $D(Q)$  is defined as the squared  $L_2$  norm of added perturbation:  $D(Q) = \|\Delta \times (P + Q)\|^2$ .

The quantization noise  $Q$  is s.t.  $X_o + P + Q \in \Delta \mathbb{Z}^n$ , where  $\Delta \in \mathbb{N}^n$  is the quantization step matrix for JPEG QF=90. If we use a first order approximation of  $L(Q)$ , we can develop (6) in a second-degree polynomial function. For any coefficient  $j$ , this function is locally minimized by:

$$Q^*(j) = -P(j) - \lambda \frac{G(j)}{2\Delta(j)}, \quad (7)$$

where  $G = \nabla L(Q)|_{Q=0}$  the gradient computed at  $Q = 0$ . This minimum however does not enforce  $(P + Q^*) \in \mathbb{Z}^n$ . A simple rounding of  $(P + Q)$  will then finalize the quantization. Finally we need to control a maximum allowed distortion. If  $\lambda$  gets big,  $Q(j)$  become a very high value which is not desirable. The final value

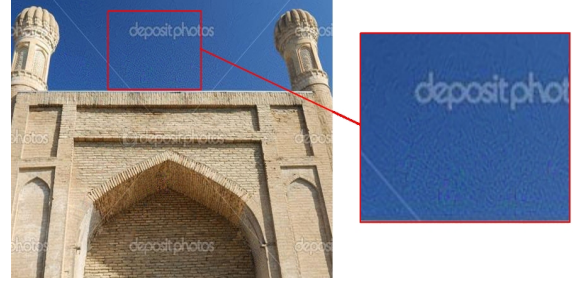


Figure 1: Image Places365\_val\_00019601c.png when quantized in the DCT domain at  $s_a = 30$ .

Table 1: Probabilities of success with a spatial Quantization

	$P_{PNG}$	$P_{JPEG}$
$s_a = 30$	<b>99.0%</b>	0.7%
$s_a = 50$	<b>100.0%</b>	11.1%

Table 2: Probabilities of success with a DCT Quantization

	$P_{PNG}$	$P_{JPEG}$	Accuracy after(JPEG90)	Number of times selected "best"
$s_a = 30$	77.5%	<b>63.8%</b>	23.82	40
$s_a = 50$	96.9 %	<b>91.6%</b>	0.91	57

for the quantized perturbation in the DCT domain is thus bounded by  $[-\frac{1}{\Delta}, \frac{1}{\Delta}]$ . These images were submitted to the jury.

## 4 RESULTS AND ANALYSIS

Tables 1 and 2 show the importance of considering the JPEG compression. When the image is quantized by the  $L_2$  optimization in the spatial domain, most images will successfully be adversarial images. However, very few of them remain adversarial after the JPEG compression. The BIQA score on most images increases up to 10 points. If the quantization is done in the DCT domain, most of them remain adversarial and the task is successful. It is however obviously more difficult to beat a lower target score  $s_a$ . An interesting property of the DCT quantization is that it creates typical JPEG artifacts as seen on Figure 1. This is especially true in low frequency images since it is harder to remain undetectable in a such situation.

## 5 DISCUSSION AND OUTLOOK

The MediaEval task was a good opportunity to extend our previous work [1] to 1) a regressor BIQA, and 2) in the DCT domain. Saving the DCT coefficients directly into a JPEG image is more consistent as it offers a better control on adversariality. Another difficulty of this task was the lack of knowledge about the compression algorithm. We therefore worked in a 'gray' box setup. The results showed that JPEG compression have a big effect on the BIQA score of, at least, adversarial images (and probably any other quality estimator). Hopefully our JPEG compression is close to the one used in the contest which allowed transferability of our adversarial images.

## REFERENCES

- [1] Benoît Bonnet, Teddy Furon, and Patrick Bas. 2020. What If Adversarial Samples Were Digital Images?. In *Proc. of ACM IH&MMSec '20*. 55–66. <https://doi.org/10.1145/3369412.3395062>
- [2] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symp. on Security and Privacy*.
- [3] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR 2015, San Diego, CA, USA*.
- [4] V. Hosu, H. Lin, T. Sziranyi, and D. Saupe. 2020. KonIQ-10k: An Ecologically Valid Database for Deep Learning of Blind Image Quality Assessment. *IEEE Transactions on Image Processing* 29 (2020), 4041–4056.
- [5] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. (2017). [arXiv:cs.CV/1611.01236](https://arxiv.org/abs/1611.01236)
- [6] Zhuoran Liu, Zhengyu Zhao, Martha Larson, and Laurent Amsaleg. 2020. Exploring Quality Camouflage for Social Images. In *Working Notes Proceedings of the MediaEval Workshop*.
- [7] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR 2018, Vancouver, BC, Canada*.
- [8] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. (2014). [arXiv:cs.CV/1312.6199](https://arxiv.org/abs/1312.6199)
- [9] G. K. Wallace. 1992. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics* 38, 1 (1992), xviii–xxxiv. <https://doi.org/10.1109/30.125072>
- [10] Hanwei Zhang, Yannis Avrithis, Teddy Furon, and Laurent Amsaleg. 2020. Walking on the Edge: Fast, Low-Distortion Adversarial Examples. *IEEE Transactions on Information Forensics and Security* (Sept. 2020). <https://doi.org/10.1109/TIFS.2020.3021899>

PART IV

# Appendix B

---

---

## 9.6 Introduction and Related Works

We discussed in Chap. 5 the existence of different techniques to make a DNN more resilient to evasion attacks. We call this resilience *robustness*, and the process *robustification*. The most commonly used method is undeniably adversarial training or retraining. The main idea is pretty simple: *feed* the DNN adversarial examples during the training phase. Optimization may differ from one work to another. Notably, we mentioned the existence of two paradigms:

- Minimize cross-entropy loss  $\mathcal{L}$  on adversarial examples. Examples are crafted with a given distortion constraint. (Eq. (5.4))
- Maximize distortion between a natural image and its closest adversarial example. (Eq. (5.5))

A slightly different optimization is suggested by Kannan et al. [74]. Called Logits Pairing, their method mixes both paradigms (Eq.(5.6)). The type of training they propose is akin to label distillation [65]. We saw in Chap. 5 the works of Papernot et al. [110] that extended this idea to improve robustness. A *teacher* DNN performs a prediction on data, which in turn is fed to a *student* DNN as labels along with the same data. Training is performed on the student as if labels were the ground-truth. Teacher DNN remains unmodified throughout the procedure. Through this distillation of knowledge, they demonstrate gains in robustness.

The works of Kannan et al. [74] add to this procedure the presence of distortion-constrained adversarial examples. Distillation is this time performed on logits rather than class probabilities. Goldblum et al. [51] also use adversarial samples but perform more classical label distillation to display gains in robustness.

Our work is in the same line of distillation. We however propose a retraining phase on features instead of logits. We use the final features (*i.e.* one step before logits) to do so. We estimate that most of the vulnerability lies in feature extraction rather than classification. We display results that back this idea in Sect. 9.8. Feature vectors are richer in information than logits and thus a better candidate for such unsupervised training.

## 9.7 Problem Formulation

Let  $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^C$  be a trained DNN, built from a feature extractor  $g(x) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_f}$  and fully-connected layer  $h(x) : \mathbb{R}^{n_f} \rightarrow \mathbb{R}^C$ . We initially clone this model to obtain

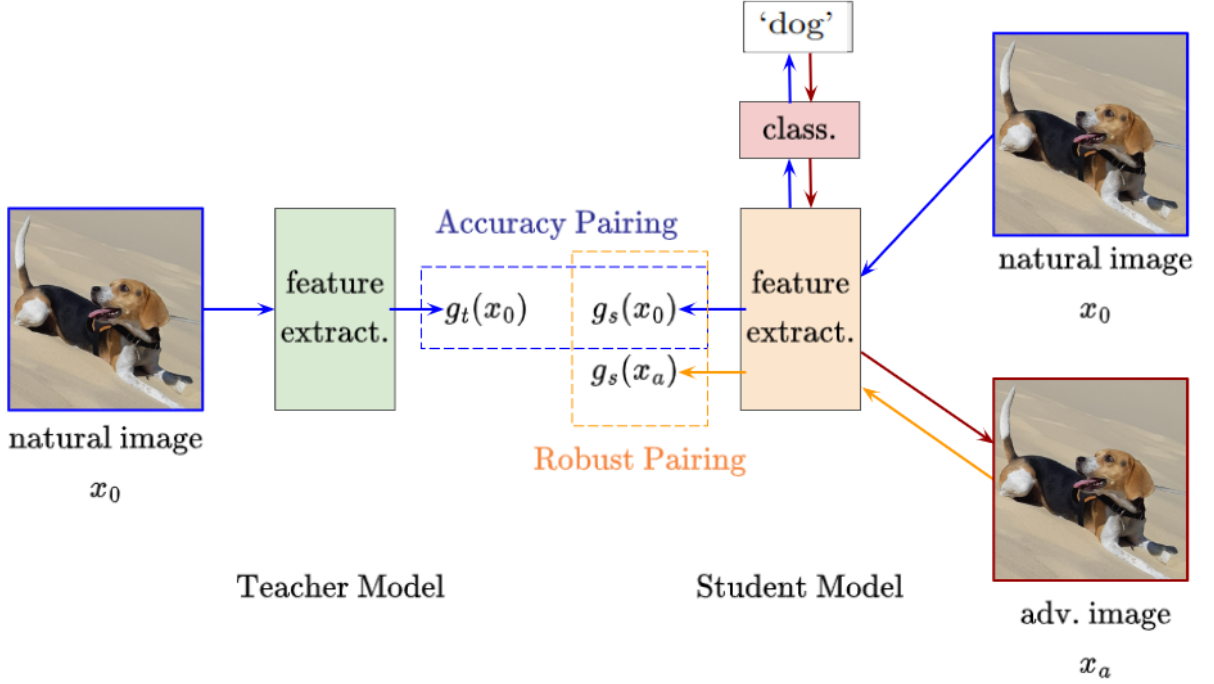


Figure 9.4 – Illustration of our training procedure. Features are extracted from a natural image  $x_0$ . An adversarial example of this image  $x_a$  is built in parallel. Features of  $x_a$  are then extracted.

$f_t(x) = h_t(g_t(x))$  a **teacher** model, and  $f_s(x) = h_s(g_s(x))$  a **student** model. The teacher remains unmodified throughout the training phase while the student is retrained.

We call  $g_t(x_0)$  natural teacher features (resp.  $g_s(x_0)$  natural student features) and  $g_s(x_a)$  adversarial student features. Note that we do not need  $g_t(x_a)$  the adversarial teacher features. Adversarial examples are built from  $f_s(x_0)$  as detailed below. In these experiments, we only the feature extraction half of the DNN. Both convolutional filters and BatchNorm are retrained. We identified BatchNorm to be partially responsible for adversarial vulnerabilities in Sect. 5.2.2.

An accurate student classifier yields  $g_s(x_0) \approx g_t(x_0)$ . Since the fully-connected layer is not trained, the best accuracy is *a priori* obtained when feature extraction is identical. A robust classifier yields  $g_s(x_a) \approx g_s(x_0)$ . A robust classifier should extract similar features out of an adversarial example. We introduce two losses:  $L_{\text{nat}} := \text{dist}(g_s(x_0), g_t(x_0))$  and  $L_{\text{rob}} := \text{dist}(g_s(x_a), g_s(x_0))$ . Where *dist* is a metrics to be defined in Sect 9.7.2. Both are optimized throughout our retraining with  $L_{\text{tot}} := L_{\text{nat}} + L_{\text{rob}}$ .



---

### 9.7.1 Building Adversarial Examples

Our training procedure is unsupervised. We can however attack images regardless of their ground-truth class  $c$  but w.r.t. their predicted class  $\hat{c}$ . Adversarial samples are also model-specific. Our student model evolves during its retraining. Adversarial samples should become more and more different between the student and the teacher model. They are strictly equal only at the first iteration.

During this first phase, we therefore build adversarial samples through  $f_s(x)$ . We use a PGD-like attack:

$$x_a^{(i+1)} = x_a^{(i)} - \sqrt{n} \frac{\epsilon}{N_{iter}} \frac{\nabla L_{adv}(x_a^{(i)})}{\|\nabla L_{adv}(x_a^{(i)})\|}. \quad (9.6)$$

With:

$$L_{adv}(x) := \hat{y}_c(x) - \hat{y}_a(x), \quad (9.7)$$

Where  $c(x)$  is the argmax of  $\hat{y}$  and  $a(x)$  its second argmax. This attack ensures a final distortion of a maximum of  $d(x_a, x_0) = \epsilon$ . Note that in the case of  $N_{iter} > 1$ , we set  $a(x)$  to remain the same as during the first iteration. This prevents the adversarial example from going back and forth between the same two classes.

### 9.7.2 Optimization

We explore several distances to achieve our robustification: Mean Squared Error (MSE), Mean Squared Logarithm Error (MSLE), and Cosine Correlation (CC).

**Mean Squared Error** This metrics is commonly seen in problems involving a pairing of vectors:

$$MSE(g_1, g_2) := \mathbb{E}[(g_1 - g_2)^2] \quad (9.8)$$

**Mean Squared Logarithmic Error** This metrics is more rarely used in such problems. In this specific task, it can be used since feature maps are strictly positive. It offers more stability over MSE whose values may explode.

$$MSLE(g_1, g_2) := \mathbb{E}[(\log(g_1 + 1) - \log(g_2 + 1))^2] \quad (9.9)$$

---

**Cosine Correlation** We worked multiple times cosine distances since Sect.5.2.3:

$$\cos(g_1, g_2) := \frac{\langle g_1, g_2 \rangle}{\|g_1\| \cdot \|g_2\|} \quad (9.10)$$

Where  $\|\cdot\|$  is the Euclidian distance. This distance needs to be maximized instead of minimized. We call Cosine Correlation our last metrics that minimizes the sine value of this angle using the trigonometric identity:

$$CC(g_1, g_2) := 1 - \cos(g_1, g_2)^2 \quad (9.11)$$

**Theoretical Scaling** We observed in Chap. 9 the scaling of adversarial examples. We established that  $\sigma_A(n)$  scaled as  $O(1/\sqrt{n})$ . This yields  $\sigma_A(n_1) \times \sqrt{n_1} = \sigma_A(n_2) \times \sqrt{n_2}$ . This is true only if  $\sigma_X(n_1) = \sigma_X(n_2)$ . Let us consider  $g(x)$  a mere downscaling operation. We can now establish the distortion that the adversarial signal *should* have in the feature space:

$$\frac{\sigma_A(g(x_a))}{\sigma_X(g(x_0))} \sqrt{n_f} = \frac{\sigma_A(x_a)}{\sigma_X(x_0)} \sqrt{n} \quad (9.12)$$

We call  $\sigma_{obj}$  the scaled distortion such that:

$$\sigma_{obj}(g(x_a)) := \sigma_X(g(x_0)) \frac{\|x_a - x_0\|_2}{\|x_0\|_2} \frac{\sqrt{n}}{\sqrt{n_f}} \quad (9.13)$$

This could be useful to reformulate  $L_{rob}$ . In the case of a Root Mean Square Error (RMSE) optimization normalized for dimension, we could write:

$$L_{rob} := \max[\sigma_A(g(x_a)), \sigma_{obj}(g(x_a))] \quad (9.14)$$

We however did not obtain substantial gains using this scaling yet and we do not provide results using this scaling.

## 9.8 Experimental Work

### 9.8.1 Experimental Setup

We use MS-COCO unlabeled 2017 dataset. There 123,000 images that we split in 120,000 for training and 3,000 for validation. Our justification to using this data is that

it represents complex scenes and we hope to extract rich feature vectors.

Our test set is *Neurips Adversarial Challenge* 2017 [80] dataset. 1,000 images akin to Imagenet compose this set. We only studied robustification on Resnet-18. In the testing phase, we attack these images with BP in best-effort mode.

The only available comparison on Resnet-18 we found comes from Salman et al. [129] (Fig. 9.6). Out of the different available model weights, we pick two:  $\epsilon = 0.05$  because clean accuracy is barely affected, and  $\epsilon = 0.5$  because it is at the frontier of a notable accuracy drop. Both stem from vanilla adversarial training *from scratch*.

During the training phase, we reduce  $L_{tot} = L_{nat} + L_{rob}$  with Adam optimizer on every convolution and BatchNorm of Resnet-18. The learning rate is set to  $10^{-4}$ . We use a batch size of 256 images. Training is performed over 10 epochs at which point we observe convergence.

## Class Vectors

Before diving into experimental work, we would like to display an interesting result. In Sect. 5.2.3 we introduced  $C\omega$  the symmetric matrix in  $\mathbb{R}^{C \times C}$  such that  $C\omega_{k,l} = \cos(\omega^k, \omega^l)$ . We now study the Cosine Correlation between  $C\omega$  of different classifiers.

	Eff.Net-b0	Res18	Res18 <sup>1</sup>	Res18 <sup>2</sup>	Res50	Res50 <sup>1</sup>	Res50 <sup>2</sup>	Res50 <sup>3</sup>
Eff.Net-b0	1.00	0.70	0.71	0.70	0.85	0.85	0.85	0.67
Res18		1.00	0.91	0.92	0.75	0.75	0.75	0.74
Res18 <sup>1</sup>			1.00	0.91	0.75	0.76	0.76	0.74
Res18 <sup>2</sup>				1.00	0.75	0.75	0.75	0.74
Res50					1.00	1.00	1.00	0.89
Res50 <sup>1</sup>						1.00	1.00	0.92
Res50 <sup>2</sup>							1.00	0.85
Res50 <sup>3</sup>								1.00

Table 9.3 – Correlations of  $C\omega$  matrices from different classifiers. Three architectures are studied: EfficientNet-b0, Resnet18, and Resnet50. Three robustification methods are studied: 1 [129]( $\epsilon = 0.05$ ), 2 [129]( $\epsilon = 0.5$ ), and 3 [98]( $\epsilon = 3.0$ ). Results are symmetrical, only half is displayed for clarity.

Table 9.3 display correlation for these matrices for different classifiers. We draw two observations. First, distances between class vectors seem to be mostly similar from one architecture to another. Second, Distances are even closer from one training to another on the same architecture. This observation led us to believe that most robustness is to be earned on feature extraction.

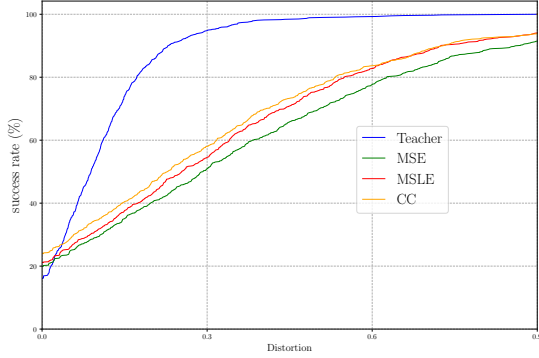


Figure 9.5 – Comparison between different optimization methods seen in Sect. 9.7.2. Attack is performed with  $N_{iter} = 1$  and  $\epsilon = 1$ .

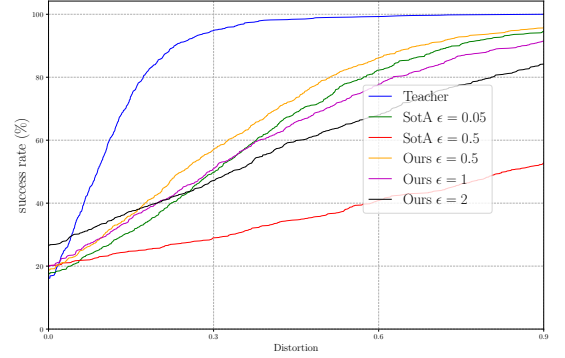


Figure 9.6 – Effect of the value of  $\epsilon$  on our method. Optimization is done through MSE and attack is performed  $N_{iter} = 1$ . State-of-the-Art [129](SotA) is displayed.

## Optimization Method

We experiment with the different distances discussed in Sect. 9.7.2. Figure 9.5 display results for each of them. Behavior is fairly similar for each of them, gains in robustness are obtained at the cost of a small loss in clean accuracy. We note that MSE however has the best trade-off. We observe good robustness as well as better accuracy than other methods.

## Adversarial Distortion

We also experiment with the effect of  $\epsilon$  during the generation of the adversarial example. Figure 9.6 display results for  $\epsilon = 0.5$ ,  $\epsilon = 1$ , and  $\epsilon = 2$ . Performances of the robust model from [129] are also visible.

We observe that our method gets better robustness with higher values of  $\epsilon$ . But this comes in exchange to clean accuracy. Overall, it seems that  $\epsilon = 1$  yields the best of both quantities. Its clean accuracy is roughly equal to the training with  $\epsilon = 0.5$  but with better robustness. Since there is a trade-off, the idea of a “best” method is debatable.

We can see that our method is not on par with [129]. Their lower-distortion training with  $\epsilon = 0.05$  seems equally as robust as our method with  $\epsilon = 0.5$ . But their clean accuracy is better. Their higher-distortion training with  $\epsilon = 0.5$  display great robustness with a loss in a clean accuracy equivalent to our method with  $\epsilon = 1$ . And our retrained model is far more vulnerable.

---

## Iterative Attacks

In this last experiment, we observe the effect of an iterative adversarial generation on robustification. Figure 9.7 display results of this experiment. We surprisingly do not observe any gain in either accuracy or robustness with multi-step generation.

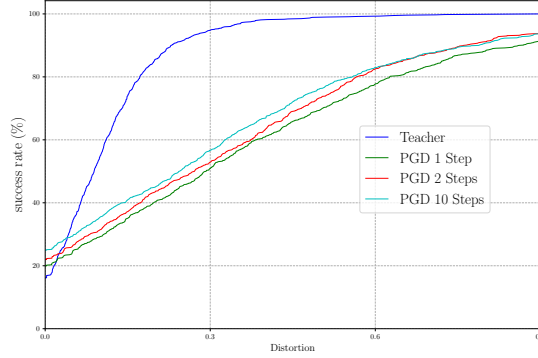


Figure 9.7 – Effect of the number of steps  $N_{iter}$  used during the attack. The attack is performed with  $\epsilon = 1$  and the training procedure is optimized with MSE

## 9.9 Discussion

Our work is currently still in progress. Experiments are iterative and we are hoping to improve our results further. We have however displayed interesting gains in robustness on DNNs. Moreover, it can be difficult to compare ourselves with the existing method. There are two reasons for that. The first is that our method proposes unsupervised retraining and the best works in model robustification train models from scratch. The use case is different since our method could be used to *robustify* DNNs trained on any task.

We identify that our data may not be the most adequate. We chose it to bring a variety of visual representations. Pseudo-classification used to create adversarial images is however heavily biased towards certain classes. Over 5% of pseudo-classification predict 562: ‘fountain’ and over 5% predict 807: ‘solar dish, solar collector, solar furnace’. Label 611: ‘jigsaw puzzle’ even appears in over 10% of all predictions. This is far above an average of 0.1% that would represent fairly all classes. These images likely do *not* represent solar dishes but instead fall in improbable regions of classification. Data is too different from Imagenet distribution that the DNN was trained on. We wanted to have rich data, agnostic to Imagenet classification but we did not use the best data. Representation within MS-COCO is very different and not adequate.





---

**Titre :** Comprendre, Apprivoiser, et se Protéger des Exemples Adversaires

**Mot clés :** Réseaux de Neurones Artificiels, Exemples Adversaires

**Résumé :** L'Intelligence Artificielle est une discipline qui a connu un fort essor au cours de ces dernières années, notamment en Vision par Ordinateur où l'application la plus commune est la classification d'image. Aujourd'hui, les réseaux de neurones artificiels profonds sont d'excellents classificateurs inférant ce que représente une image. Des travaux ont cependant rapidement montré qu'ils sont vulnérables aux attaques par évasion, aussi appelées les exemples adverses. Ces exemples sont des images qui pour un humain semblent être une représentation normale d'un objet. Mais le classifieur attaqué ne parviendra pas à prédire correctement ce qu'elles représentent.

Cette thèse étudie les mécanismes de création de ces exemples, la raison de leur existence et la vulnérabilité des classificateurs. En particulier, ce travail replace ces exemples adverses dans un contexte réaliste. Premièrement, il propose des attaques rapides même sur des grandes images avec un fort taux de succès et une distortion imperceptible ou indétectable. Deuxièmement, il ajoute la contrainte que les exemples adversaires sont avant tout des images, c'est à dire des signaux quantifiés dans le domaine spatial (format PNG) ou dans le domaine DCT (format JPEG).

---

**Title:** Understanding, Taming, and Defending from Adversarial Examples

**Keywords:** Deep Neural Networks, Adversarial Examples

**Abstract:** Artificial Intelligence is nowadays one of the most essential disciplines of computer science. These algorithms perform particularly well on Computer Vision tasks, especially classification. A classifier infers what an image represents. Nowadays Deep Neural Networks are largely used for these problems. These neural networks first undergo a training phase during which they are given many examples. These images are accompanied by labels: information on what the image represents. However, it was quickly found that the same logic used during the training phase could be used maliciously. This is the creation of Adversarial Examples through an Evasion

Attack.

Such examples are seemingly normal images. A human understands what it represents as if it was not manipulated. But the attacked classifier will make an incorrect prediction. In this manuscript, we study the creation of such examples, the reason for their existence, and the underlying vulnerability of classifiers. In particular, we study these examples in a realistic context. First, attacks are optimized (high success rate and low distortion). Second, we add the constraint that adversarial examples should be images. We thus work on spatially-quantized (PNG) or DCT-quantized images (JPEG).