Intégrité des réseaux de neurones : défenses contre les exemples adverses et leur transférabilité

Rémi Bernhard

N°d'ordre NNT : 2021LYSEM033

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Ecole des Mines de Saint-Etienne

Ecole Doctorale N° 488
Sciences, Ingénierie, Santé

Spécialité de doctorat : Microélectronique
Discipline :
Sciences et technologies de l'information et de la communication

Soutenue publiquement/à huis clos le 30/11/2021, par :
Rémi Bernhard

Intégrité des réseaux de neurones : défenses contre les exemples adverses et leur transférabilité

Devant le jury composé de :

Mermillod, Martial Professeur des universités Université Grenoble-Alpes Président

Bouchara, Frédéric Maître de conférences Université de Toulon Rapporteur
Furon, Teddy Chargé de recherche hors classe INRIA RENNES Rapporteur
Rosenberger, Christophe Professeur des universités ENSICAEN Examinateur

Dutertre, Jean-Max Professeur des universités Ecole Nationale Supérieure des Mines de Saint-Etienne Directeur de thèse
Moëllic, Pierre-Alain Ingénieur de recherche CEA-Leti Invité

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	MATHEMATIQUES APPLIQUEES	M. Batton-Hubert
MECANIQUE ET INGENIERIE	S. Drapier, professeur	INFORMATIQUE	O. Boissier, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	SCIENCES DES IMAGES ET DES FORMES	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	N. Absi, Maitre de recherche
SCIENCES ET GENIE DE L'ENVIRONNEMENT	V.Laforest, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

ABSI	Nabil	MR	Génie industriel	CMP
AUGUSTO	Vincent	MR	Génie industriel	CIS
AVRIL	Stéphane	PR	Mécanique et ingénierie	CIS
BADEL	Pierre	PR	Mécanique et ingénierie	CIS
BALBO	Flavien	PR	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR	Mathématiques appliquées	FAYOL
BEIGBEDER	Michel	MA	Informatique	FAYOL
BILAL	Blayac	DR	Sciences et génie de l'environnement	SPIN
BLAYAC	Sylvain	PR	Microélectronique	CMP
BOISSIER	Olivier	PR	Informatique	FAYOL
BONNEFOY	Olivier	PR	Génie des Procédés	SPIN
BORBELY	Andras	DR	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR	Génie Industriel	FAYOL
BRUCHON	Julien	PR	Mécanique et ingénierie	SMS
CAMEIRAO	Ana	PR	Génie des Procédés	SPIN
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR	Génie Industriel	CMP
DEBAYLE	Johan	MR	Sciences des Images et des Formes	SPIN
DEGEORGE	Jean-Michel	MA	Génie industriel	Fayol
DELAFOSSE	David	PR	Sciences et génie des matériaux	SMS
DELORME	Xavier	PR	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
BERGER-DOUCE	Sandrine	PR	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR	Mécanique et ingénierie	SMS
DUTERTRE	Jean-Max	PR	Microélectronique	CMP
EL MRABET	Nadia	MA	Microélectronique	CMP
FAUCHEU	Jenny	MA	Sciences et génie des matériaux	SMS
FAVERGEON	Loïc	MR	Génie des Procédés	SPIN
FEILLET	Dominique	PR	Génie Industriel	CMP
FOREST	Valérie	PR	Génie des Procédés	CIS
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GAVET	Yann	MA	Sciences des Images et des Formes	SPIN
GERINGER	Jean	MA	Sciences et génie des matériaux	CIS
GONDRAN	Natacha	MA	Sciences et génie de l'environnement	FAYOL
GONZALEZ FELIU	Jesus	MA	Sciences économiques	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GRIMAUD	Frederic	EC	Génie mathématiques et industriel	FAYOL
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR	Génie des Procédés	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR	Génie des Procédés	SPIN
ISMAILOVA	Esma	MC	Microélectronique	CMP
KERMOUCHE	Guillaume	PR	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	DR	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	DR	Mécanique et ingénierie	FAYOL
LIOTIER	Pierre-Jacques	MA	Mécanique et ingénierie	SMS
MEDINI	Khaled	EC	Sciences et génie de l'environnement	FAYOL
MOLIMARD	Jérôme	PR	Mécanique et ingénierie	CIS
MOULIN	Nicolas	MA	Mécanique et ingénierie	SMS
MOUTTE	Jacques	MR	Génie des Procédés	SPIN
NAVARRO	Laurent	MR	Mécanique et ingénierie	CIS
NEUBERT	Gilles	PR	Génie industriel	FAYOL
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
O CONNOR	Rodney Philip	PR	Microélectronique	CMP
PICARD	Gauthier	PR	Informatique	FAYOL
PINOLI	Jean Charles	PR	Sciences des Images et des Formes	SPIN
POURCHEZ	Jérémy	DR	Génie des Procédés	CIS
ROUSSY	Agnès	MA	Microélectronique	CMP
SANAUR	Sébastien	MA	Microélectronique	CMP
SERRIS	Eric	IRD	Génie des Procédés	FAYOL
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
VALDIVIESO	François	PR	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR	Génie industriel	CIS
YUGMA	Gallian	MR	Génie industriel	CMP

Mise à jour : 04/02/2021

Remerciements

Dans un premier temps, je souhaite remercier Frédéric Bouchara ainsi que Teddy Furon d'avoir accepté d'être les rapporteurs de cette thèse. Leurs conseils et commentaires à la fois avisés et pertinents, qui ont fait suite à une lecture minutieuse de ce manuscrit, ont grandement contribué à son amélioration.
Je tiens à remercier Martial Mermillod d'avoir présidé le jury de cette thèse. Son expérience en psychologie et neurocognition, ainsi que sa force de proposition, ont permis une collaboration aussi intéressante que fructueuse.
Enfin, je remercie Christophe Rosenberger pour son rôle d'examinateur dans le jury de thèse, ainsi que pour ses commentaires judicieux lors de la soutenance.

Dans un second temps, je tiens à exprimer ma profonde gratitude envers Pierre-Alain Moëllic et Jean-Max Dutertre, pour m'avoir encadré et accompagné durant ces trois années de thèse. Sans leur présence à mes côtés tout au long de cette aventure, cette thèse n'aurait sans aucun doute pas été aussi agréable à mener, tant en termes professionnels qu'en termes humains. Enfin, je dirais qu'ils m'ont permis de grandir, dans le domaine de la recherche, mais aussi humainement.
Je remercie Pierre-Alain de m'avoir apporté sa vision globale de la sécurité algorithmique et matérielle de l'apprentissage statistique. Ses conseils de recherche ont toujours été les bienvenus, tant par leur pertinence que leur côté novateur. Son expérience m'a permis de prendre la hauteur sur les différents enjeux, actuels et concrets, de notre domaine de recherche. Ses conseils en matière de présentation orale, qui se sont révélés particulièrement pertinents, m'accompagneront tout au long de mon expérience professionnelle. Surtout, je remercie Pierre-Alain pour sa bienveillance sincère, sa bonne humeur, son optimisme, et toutes les autres qualités humaines qui font de lui un encadrant d'exception. Chaque jour de cette thèse a été marquée par le plaisir de collaborer avec lui. Pierre-Alain a su faire preuve de sincérité, d'honnêteté et de dévouement tout au long de la thèse, et m'a transmis un état d'esprit positif qui m'a vraiment élevé humainement. Enfin, je dirais qu'il a été une composante essentielle de la réussite de cette thèse.
Je remercie Jean-Max d'avoir été le directeur de cette thèse. Son expérience de la sécurité matérielle m'a permis d'apprendre à connaître de nouveaux horizons de recherche. Son suivi constant et méticuleux de mon travail a été une composante essentielle de cette thèse. Ses remarques pertinentes permettaient d'explorer des pistes toujours intéressantes. Ses relectures de mes différents travaux ont été les plus complètes qu'il m'a été donné de voir, et je lui en suis très reconnaissant pour ça. Jean-Max a aussi été une personne avec des qualités humaines incroyables, marqué par la bienveillance et la bonne humeur. Les discussions avec lui ont toujours été enrichissantes et positives. Je garderai de Jean-Max l'image de quelqu'un de sincèrement gentil, qui a à cœur de s'occuper des étudiants.

Dans un troisième temps, je souhaite remercier toutes les personnes du laboratoire dans lequel j'ai effectué cette thèse. Les enseignants chercheurs, tous passionnés par leur domaine, et toujours heureux de partager leurs idées et connaissances. L'équipe administrative, toujours disponible et pleine de bons conseils. Enfin, toute l'équipe de doctorants avec qui j'ai eu le plaisir de travailler pendant ces trois ans. La solidarité, l'entraide et les conversations au sein du laboratoire, autour d'un verre ou d'un repas vont me manquer. Je tiens aussi à remercier les chercheurs et doctorants du Cea-Leti à Grenoble et de l'université de Grenoble-Alpes pour la collaboration fructueuse que nous avons pu mener !

Enfin, je remercie ma famille et mes amis, qui me soutiennent depuis toujours, et sans qui je n'en serai pas arrivé là.

Abstract

Adversarial examples refer to a type of attack where an adversary maliciously modifies some input, in order to fool a model at inference time. In this thesis, we aim at better understanding the vulnerability of neural networks to adversarial examples, and at developing efficient schemes to thwart their transferability, one of their worrisome aspect.

Regarding the success of deep learning in various tasks, ranging from image classification to speech recognition, there is a growing will to deploy neural networks models in the everyday life. Whether these models are used in distant APIs or embedded on devices, adversarial examples therefore constitute a threat to this deployment, and more generally to the integrity of machine learning (ML) models. More particularly, in some use-cases such as autonomous driving, adversarial examples constitute a major threat to the public safety.

Among the different aspects of adversarial attacks, the phenomenon of the transferability of adversarial perturbations is particularly concerning, as it is a powerful tool for an adversary who does not have a direct access to the target model (black-box setting). Indeed, an adversarial example crafted on a substitute model can also fool another model. As the black-box setting is the most common scenario for a threat model, studying and hindering transferability would therefore allow to greatly enhance the security of machine learning models.

Relatively to our goals in this thesis, we begin by investigating at a frequency level the cause for the vulnerability of models to adversarial attacks. More particularly, we find links between robustness issues of different models and intrinsic frequencies properties of the data set considered. Then, we conduct a study based on cognitive psychology results showing the prevailing importance of low-frequency information in the human classification process. More precisely, we discover conditions ensuring robustness for models enforced to rely on low-frequency information during training.

Considering the will to deploy machine learning models in a large variety of hardware platforms, quantization schemes are used to address memory footprint and inference time issues. In the scientific literature, some works found quantization methods to provide robustness to models trained with such schemes. To better investigate this phenomenon, we conduct a study of the robustness of quantized models, considering various attack schemes and threat models. Contrary to some existing works on the subject, we showed that quantization offers no robustness. More precisely, we found that it induces a false sense of security via the phenomenon of gradient masking, which render gradients useless to find adversarial examples. Interestingly, we noticed a poor transferability of adversarial perturbations between quantized models with different bitwidths, which we explain with gradient and quantization related concerns. Moreover, we use these results as a basis for the development of an ensemble-based defense scheme.

We then tackle the transferability phenomenon, at the core of this thesis. We develop an innovative way to thwart the transferability of adversarial perturbations: the *luring effect*, implemented on a substitute model, relatively to a target model. When crafting adversarial examples on his substitute model, the adversary is *lured* into choosing some directions, which are not the directions which should have been chosen to fool the target model. The luring effect is conceptually new as it does not make the target model more robust, nor it implements a reactive scheme such as purification. We characterize the luring effect with experiments highlighting how it leads to the substitute and target models relying on different concepts (and thus displaying different vulnerabilities) to perform their predictions. Then, we show how the luring effect can be used in different threat models.

Eventually, we highlight a link between integrity and availability attacks. In many use-cases, the accuracy is not sufficient and a ML-based system requires the underlying model to output confident predictions. We design a new attack method against the availability of a system. It aims at making the underlying model output *uncertain* predictions, characterized as an example by a doubt between many labels. Importantly, we show how such attack can be implemented by leveraging the knowledge of adversarial examples crafting schemes.

Résumé

Les exemples adverses sont un type d'attaque où un adversaire modifie de manière malicieuse une entrée pour tromper un modèle à lors de l'inférence. Le but de cette thèse est de mieux comprendre la vulnérabilité des réseaux de neurones aux exemples adverses, et de développer des schémas de défense efficaces pour contrer leur transférabilité.

Au vu du succès du *deep learning* lors de nombreuses tâches, de la classification d'images à la reconnaissance de la voix, on observe une volonté croissante de déployer des modèles de réseaux de neurones. Que ces modèles soient utilisés via des APIs distantes ou embarqués sur divers appareils, les exemples adverses constituent donc une menace à ce déploiement, et plus générélement à l'intégrité des modèles de machine learning. Plus particulièrement, ils peuvent constituer une menace sérieuse à la sécurité des systèmes critiques (santé, transport, énergie, ...).

Parmi les différents aspects des attaques adverses, la transférabilité des perturbations adverses est particulièrement préoccupante, car elle représente un outil puissant pour un adversaire sans accès direct au modèle cible (scénario boîte noire). En effet, un exemple adverse élaboré sur un modèle substitut peut arriver à tromper un autre modèle. Le scénario boîte noire étant le plus commun, étudier la transférabilité est une considération de premier plan.

Au vu de ces objectifs, nous commençons par étudier au niveau fréquentiel les causes de la vulnérabilité des modèles aux attaques adverses. Plus particulièrement, nous montrons des liens entre la robustesse de modèles et des propriétés fréquentielles propres aux données avec lesquelles ils ont été entrainés. Nous conduisons ensuite une étude qui se base sur des résultats en psychologie cognitive qui montrent que les informations basses fréquences prévalent dans le système humain de classification. Plus précisément, nous découvrons des conditions pour la robustesse de modèles forcés à tirer parti d'informations dans les basses fréquences pendant la phase d'entrainement.

Considérant la volonté de déployer des modèles de machine learning, des méthodes de quantification sont utilisées pour répondre aux problèmes d'espace mémoire et de rapidité à l'inférence. Dans la littérature, certains travaux semblent montrer que la quantification peut apporter une certaine robustesse. Pour mieux comprendre ce phénomène, nous réalisons une étude sur la robustesse des modèles quantifiés, en considérant divers modèles de menace. Contrairement aux travaux existants, nous montrons que la quantification n'offre en réalité aucune robustesse. Plus précisément, nous révélons que la quantification induit une fausse impression de sécurité via le phénomène de *gradient masking*, qui rend les gradients non-pertinents pour trouver un exemple adverse. Nous constatons une faible transférabilité des perturbations adverses entre des modèles quantifiés avec différents niveaux de quantification, ce que nous expliquons en raisonnant sur la quantification et les gradients. De plus, nous prenons ces résultats comme base d'une défense construite à partir d'un ensemble de modèles.

Nous considérons ensuite le phénomène de transférabilité. Nous développons une manière novatrice de contrer la transférabilité de perturbations adverses: l'*effet de leurrage*, qui est implémenté sur un modèle substitut relativement à un modèle cible. Lors de l'élaboration d'un exemple adverse, l'adversaire est amené à choisir certaines directions, qui ne sont pas celles nécessaires pour tromper le modèle cible. L'effet de leurrage est conceptuellement nouveau : il ne rend pas le modèle cible plus robuste, ni n'implémente de méthode *a posteriori* comme la purification. Nous caractérisons l'effet de leurrage, en montrant que les modèles cible et substitut ne se reposent pas sur les mêmes concepts (et donc ne présentent pas les mêmes vulnérabilités). Enfin, nous montrons comment l'effet de leurrage peut être utilisé dans différents scénarios de défense.

Finalement, nous mettons en lumière un lien entre attaques contre l'intégrité et attaques contre la disponibilité. Dans beaucoup de cas d'usage, la précision n'est pas suffisante pour un système, qui requiert aussi que le modèle sous-jacent renvoie des prédictions de confiance. Nous concevons une nouvelles attaque contre la disponibilité d'un système. Elle vise à faire en sorte que le modèle sous-jacent renvoie des prédictions incertaines, avec par exemple un doute entre des classes. Nous montrons comment cette attaque peut être implémentée en tirant parti des méthodes d'élaboration d'exemples adverses.

Integrity of neural networks: defending against adversarial examples and their transferability

Contents

Preamble

Fundings and Projects

This thesis is funded by CEA-Leti. Works are parts of research activities from several research projects: European ECSEL project InSecTT[1], the French program *Investissements d'avenir* (ANR-10-AIRT-05, irtnanoelec) and ANR project PICTURE (AAPG2020)[2]. This work benefited from the French Jean Zay supercomputer thanks to the *AI dynamic access*[3].

Publications

The thesis, started in November 2018 and ending in November 2021, has led to the publication of research articles, which we list below:

- **Impact of low-bitwidth quantization on the adversarial robustness for embedded neural networks**
  Rémi Bernhard, Pierre-Alain Moellic, Jean-Max Dutertre
  (2019) IEEE International Conference on Cyberworlds
  Code: `https://gitlab.emse.fr/remi.bernhard/Quantization-and-Adversarial-Robustness`
- **Luring of transferable adversarial perturbations in the black-box paradigm**
  Rémi Bernhard, Pierre-Alain Moellic, Jean-Max Dutertre
  (2021) IEEE International Joint Conference on Neural Networks (IJCNN)
  Code: `https://gitlab.emse.fr/remi.bernhard/Luring_Adversarial/`
- **Impact of Spatial Frequency Based Constraints on Adversarial Robustness**
  Rémi Bernhard, Pierre-Alain Moellic, Martial Mermillod, Yannick Bourrier, Romain Cohendet, Miguel Solinas, Marina Reyboz
  (2021) IEEE International Joint Conference on Neural Networks (IJCNN)
  Code: `https://gitlab.emse.fr/remi.bernhard/Frequency_and_Robustness/`
- **A Review of Confidentiality Threats Against Embedded Neural Network Models**
  Raphaël Joud, Pierre-Alain Moellic, Rémi Bernhard, Jean-Baptiste Rigaud
  (2021) Workshop on Wireless Intelligent Secure Trustable Things, IEEE 7th World Forum on Internet of Things
- **An Overview of Laser Injection against Embedded Neural Network Models**
  Mathieu Dumont, Pierre-Alain Moellic, Raphael Viera, Jean-Max Dutertre, Rémi Bernhard
  (2021) Workshop on Wireless Intelligent Secure Trustable Things, IEEE 7th World Forum on Internet of Things

Communications

All along the thesis, advances and results were presented in different venues, in order to foster scientific advances in the field of the robustness of neural networks against adversarial examples:

- **Luring of Transferable Adversarial Perturbations in the Black-Box Paradigm**
  (2021) Research Group ISIS (Information, Signal, Image and viSion), online
- **Adversarial Robustness of Quantized Embedded Neural Networks**
  (2019) C&ESAR IA conference, Artificial Intelligence and Defense, Rennes, France
- **Impact of Low-bitwidth Quantization on the Adversarial Robustness for Embedded Neural Networks**
  (2019) Journée de la recherche, Centre de Microélectronique de Provence, Gardanne, France
- **Security of Neural Networks: Attacks, Defenses and Evaluation methods**
  (2019) Journée de la recherche, Centre de Microélectronique de Provence, Gardanne, France
- **Impact of Quantization for Embedded Neural Network Models on the Adversarial Robustness**
  (2019) Journée thématique Sécurité, fiabilité et tesf de SoC 2 : challenges et opportunités dans l'ère de l'Intelligence Artificielle, Paris, France

[1] `www.insectt.eu`
[2] `https://picture-anr.cea.fr`
[3] `http://www.idris.fr/jean-zay/`

Teaching activities

Along with the research activities, I provided students of the *Ecole Nationale Supérieure des Mines de Saint-Etienne* with courses:

- **Algorithms and programming**. 2019-2020. Practical sessions. Junior students of the engineering school.
  Study and implementation of graph theory algorithms in C: Dijkstra, Bellman-Ford and Chandy-Misra-Haas.

- **Security of machine learning**. 2018-2020. Practical sessions and lectures. Senior students of the engineering school.
  Overview of the existing threats against a machine learning model. Presentation of adversarial examples, attack and defense schemes.

Chapter 1

Introduction

1.1 Context

1.1.1 Ubiquitous Deep Learning

In recent years, machine learning and more specifically deep learning, has revealed to be the most efficient technique to achieve remarkable performances in various domains. Notably, deep neural networks reach state-of-the-art results in image classification [1], image generation [2], real-time object detection [3], speech recognition [4], speech synthesis [5] and text translation [6]. In consequence, deep learning is employed in many and various application domains such as medical diagnosis [7], recommendation systems [8], malware detection [9] or autonomous cars [10].

As an aftermath of this success, models are being more and more widespread in the everyday life. On mobile phones, an ever greater number of applications make recurrent calls to distant servers relying on a machine learning model. In addition, an increasing number of publicly available APIs use deep learning, such as the Google Cloud Vision API[1] or Clarifai[2]. Moreover, the use of machine learning is not reserved to publicly available services. Indeed, deep learning plays an important role in critical applications such as cybersecurity, transport, warfare or medical and healthcare.

In other words, machine learning will soon be omnipresent around us, included in systems available to anyone, in private networks, or in state-owned institutions. Accordingly and reasonably, security issues related to machine learning models become more and more studied, whether it concerns the data at stake or the reliability of the models at use. As a response to these worries, new laws are adopted. At the European scale, these security concerns are important inputs to the *New Regulatory Framework on Artificial Intelligence* presented in April 2021[3]. This initiative has been taken in order to allow the European Union to both coordinate the efforts in Artificial Intelligence (AI), and allow for a well defined and harmonized regulation among the European countries. Notably, it encompasses a clear policy for "high-risk AI". This type of AI refers for example to systems relative to biometrics, responsible for judicial decision making, credit scores, etc. This type of action will have to hinge on existing laws focusing on the privacy of data, such as the GDPR (General Data Protection Regulation)[4], which has taken effect since May 2018.

1.1.2 Machine Learning on Embedded Devices

Deployment of deep learning models

The actual development of Machine Learning is mainly focused on two objectives: (1) increasing the performance of models for highly challenging tasks with an impressive growing of their complexity (and their related data sets), as for the recent *transformer* models for natural language processing; (2) the deployment of models in a large variety of devices and hardware platforms. This large-scale deployment responds to two major concerns. The first one relates to the server capacity. Indeed, scaling a server hosting a machine learning system raises quality and availability issues, such as service interruption, that can be efficiently addressed by deploying the model on client embedded devices. The second concern relates to the dependence to communication networks. When the model is integrated in a device, it offers a better user experience, both in terms of practicality and latency.

However, when considering a neural network based model, millions of parameters are involved. In fact, the rise of deep learning follows the emergence of GPU (Graphics Processing Units) technology, which allows to train models way faster, notably by taking advantage of parallel computing. Therefore, embedding a deep learning model on a device leads to practical issues. First of all, the memory footprint can quickly be a limiting factor for constrained devices. For example, a typical ARM Cortex-M4-based microcontroller such as STM32F4 has up to 384 KBytes RAM and a maximum of 2MBytes of Flash memory[5]. Secondly, inference cost in terms of energy is critical for devices like mobile phones or a large variety of connected objects (e.g., industrial sensors). Thirdly, inference speed is necessary to avoid critical latency issues.

Security consequences of models deployment

Regarding the security constraints which are emerging as machine learning models become more and more ubiquitous, embedded models raise a first issue: it widens the surface attack for an

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

[2] https://www.clarifai.com/

[3] https://digital-strategy.ec.europa.eu/en/library/communication-fostering-european-approach-artificial-intelligence

[4] https://gdpr-info.eu/

[5] https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html

adversary. Therefore, it will be harder to secure a deployed model than it would have been if it was only available via queries to a distant server.

When the model is in a distant fashion, some characteristics of the model can be hidden, in order to obfuscate critical information to weaken the adversary. For example, one may conceal the machine learning technique used, so that the adversary has a restricted knowledge of the system to attack. The information given by the model can also be obfuscated, for example by only outputting the predicted label for a classification task.

When the model is an physical fashion, embedded in a device available to the adversary, it is reasonably provided with more attack vectors than in a distant fashion. Notably, even if hardware obfuscation methods are implemented, an adversary can exploit some flaws related to the physical implementation. Therefore, there is a higher chance that the adversary manages to find vulnerabilities than if the model was kept distant.

Moreover, considering a model used both in a distant and physical public fashion, the adversary can take advantage of the vulnerabilities of the public version to mount an attack against the distant system. By doing so, his attack is more powerful than if only the distant system was known.

1.2 Machine Learning under attack

1.2.1 The rise of threats against Machine Learning

Attacks against computer systems and more particularly information systems have been demonstrated and studied for several decades, for example, with denial of service or intrusion attacks. In recent years, many threats targeting directly a machine learning model have emerged. These new attacks mean that numerous applications, ranging from mobile phone applications to military ones, are now vulnerable to attack vectors related to the use of models. This poses a serious and growing menace to the security of popular services as well as critical infrastructures.

Concretely, these new attacks can occur at every stage of the traditional Machine Learning pipeline: during the training phase, when inputs are supplied to the model to train it, or during the inference phase, when the model is queried to achieve its task. The goals of these attacks are diverse. Firstly, they may be designed to strongly deteriorate the inference process of the model. The goal of such attack is to fool the model by modifying inputs. In this case, the integrity of the model is threatened. Secondly, some attacks aim at stealing information about a hidden model or about data the model was trained with. This can concern some private data as medical or financial ones, or the exact parameters of trained machine learning models. This last type of threat is thus particularly concerning regarding privacy and intellectual properties. Thirdly, these attacks may aim at threatening the availability of an overall system containing a machine learning model. The availability is related to system's requirements, such as the time to access it, or the overall quality of the predictions given by the model.

Considering the danger caused by these new attacks, it appears urgent to develop reliable defenses to counter them. Therefore, these threats are increasingly studied by both the AI and the Security scientific communities with more and more joint efforts.

1.2.2 Adversarial examples and transferability

Adversarial examples

Among existing threats, the phenomenon of *adversarial examples* has received a particular attention [11]. Adversarial examples consist of clean inputs which have been maliciously modified to fool a model. For example, for a classification task, fooling the model means that the model does not predict the same label for the original input and its modified version. Importantly, these inputs are modified in such a way that humans are not influenced by them. This means that for a classification task, the model outputs two different labels for a clean example and its adversarial counterpart, while the human classifies them in the same category. This specificity indicates that adversarial examples can be deployed in the real-world without arousing suspicion. For example, a traffic road sign can be subtly modified, by adding some stickers on it with specific colors at specific locations, so that it does not attract any attention. However, this real-world adversarial example will be able to fool an autonomous car system based on a neural network model, as demonstrated on a Tesla car (Hardware Pack 1 with a MobilEye camera system) by a MacAfee laboratory in 2020[6].

[6] https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/

Adversarial examples is actually the most studied threat among the one targeting machine learning models, for two main reasons. The first one is that it concerns the most well-spread scenario of a already deployed model that an adversary wants to attack. The second reason is that it can have the most harmful consequences among the other threats relating to the effort this attack requires. For example, one modified traffic sign can lead to a deadly motor vehicle accident.

Transferability

The phenomenon of *transferability* of adversarial examples strongly aggravates their dangerousness. The transferability property means that an adversarial example crafted to fool a specific model can be used to fool another one. Transferability is therefore a powerful tool for an adversary which does not have access to a target model. Indeed, in this case, the adversary can train his own substitute model, or use an already available model, to craft an adversarial example on it. In this case, the process of crafting an adversarial example is the most favorable one, as no information of the substitute model is hidden from the adversary. This adversarial example is then later *transferred* to the model the adversary does not have access to, in order to fool it. Moreover, the transferability of adversarial examples is even more dangerous that it exists between different machine learning techniques [12] (e.g., from a SVM to a neural network), and for different domains [13].

With the growing deployment of models, the transferability phenomenon has an increasing importance. Notably, for computer vision tasks, this is a common scenario that the same model is used in the distant system and the public device. The adversary can then extract from the hardware device useful information about the model, which will help him in the choice of his substitute model. As an example, the adversary can leverage SCA (Side Channel Analysis), a method exploiting the implementation of the model, to retrieve the architecture of a neural network [14].

1.3 Objectives and contributions

In this thesis, we focus on better understanding the nature of adversarial examples, how to thwart them, notably how to impede their *transferability*. The motivation behind this research direction is twofold. On one hand, adversarial examples is one of the actual major threats to deep learning. Therefore, better understanding the root causes of their existence can bring fruitful intuition for future efficient defense schemes. On another hand, *transferability* is a growing concern with the deployment of embedded models, which represents an important topic for the upcoming activities of the laboratory in which this work takes place[7].

More precisely, to gain meaningful insight on the cause for vulnerability of a neural network to adversarial examples, we are interested in making new connections between the robustness to adversarial examples, and the way the human vision system processes frequencies. In order to find methods to hinder the transferability of adversarial examples, we study what explains the transferability phenomenon, to derive methods to reduce it. Moreover, regarding the actual landscape of threats and the everyday scenario, we are interested in investigating how actual threats can be linked to existing ones.

In **chapter 2**, we enter into details concerning the existing threats against a machine learning model. Overall, this allows to better understand which are the scenarios and application domains at stake. We present general concepts behind existing threats against confidentiality, integrity and availability.

In **chapter 4**, we review the phenomenon of adversarial examples and their transferability, for a neural network trained for an image classification task. Notably, we detail formally the corresponding threat model, and give examples of both attack and defense schemes.

In **chapter 5**, we investigate the reasons which make a neural network vulnerable to adversarial examples, while seeming benign for a human. More specifically, we tackle this topic at a frequency level, by taking advantage of the fact that humans predominantly use low frequency information to perform classification.

In **chapter 6**, to better evaluate the security risks induced by the large-scale machine learning model deployment, we investigate the specific features of models learned with quantization methods in regards of robustness against adversarial examples. Moreover, we investigate the impact of quantization of the transferability of adversarial perturbations.

In **chapter 7**, we study in-depth the transferability phenomenon, and design an innovative way of hindering the transferability of adversarial examples between two neural network models. This novel concept can then be used in different defense scenarios.

[7]Secure Architectures and Systems Joint Research Team between CEA LETI and Mines Saint-Etienne

In **chapter 8**, we extend the scope of adversarial examples, i.e. attacks against integrity, to an attack that aims at deteriorating the access to a system relying on a machine learning model, i.e. an attack against the availability of a system. Notably, we formalize a new kind of threat against availability, and show how to leverage adversarial perturbations to implement this attack.

Chapter 2

Overview of the Security of Machine Learning

In this chapter, we present an overview of the existing attacks that may target a machine learning model. This presentation is intended at laying out the landscape of existing threats, with a high-level description of the concept for each threat, with illustrative examples.

First, we detail when threats may occur during the deployment process of a machine learning model, and who may realize them by interfering in this process. For each type of existing attack, we describe the underlying concept and present illustrative examples. Second, we explain how these threats can be seen under the classical CIA triad (Confidentiality, Integrity, Availability), and categorize them accordingly. Finally, we enter into details concerning the adversary by presenting the general threat model. The threat model is a proper definition of what can be an adversary's goal, knowledge and capacities. The definition of a threat model is a compulsory step when designing a defense, to formally characterize the adversary. Then, we explain how threat models are established in practice. Notably, we explain how the threat model is strongly influenced by the real-life use-case envisaged for the machine learning model.

Related publications proposed during the thesis:

- [15] Joud, R., Moellic, P. A., Bernhard, R., Rigaud, J. B. *A Review of Confidentiality Threats Against Embedded Neural Network Models.* IEEE World Forum on Internet of Thing (WF-IOT). 2021.
- [16] Dumont, M., Moellic, P. A., Viera, R., Dutertre, J.-M., Bernhard, R. *An Overview of Laser Injection against Embedded Neural Network Models.* IEEE World Forum on Internet of Thing (WF-IOT). 2021.

2.1 Attacks blueprint

2.1.1 Machine learning pipeline

Whatever the underlying algorithm chosen by practitioners, the development of every supervised machine learning model follows the same process. The complete Machine Learning pipeline that leads to a model deployment is illustrated in Figure 2.1. At the first stage, raw data is gathered. This data may have been collected by sensors, which have been specifically set up for that purpose, such as cameras, microphones, multimeter devices, etc. The data may come also from already existing databases, which can be publicly available or private. The second stage consists in preprocessing the data in order to build a data set used to train the model. The preprocessing scheme is vast and covers numerous concerns. It goes from basic cleaning and formatting to statistical investigation to solve problems such as multicollinearity and class-imbalance. The third stage consists in performing the training phase of the model with the data set built in the previous stage. This stage resumes to search for the best model in terms of generalization error. The fourth stage is the deployment of the model in an overall system. The deployed model is now part of a system, and completely operational. Importantly, the deployment of a machine learning model can be a continuous process. For example, in *online learning*, new data is unceasingly obtained and processed to refine the model parameters.

At each stage of this process, from the data collection process to the instant the system is put into operation, different threats are feasible. A brief overview of the different types of attacks that may occur during the machine learning pipeline is presented in Figure 2.2. As described in the next sections, these threats vary relatively to what the adversary is allowed to perform depending on the considered stage.

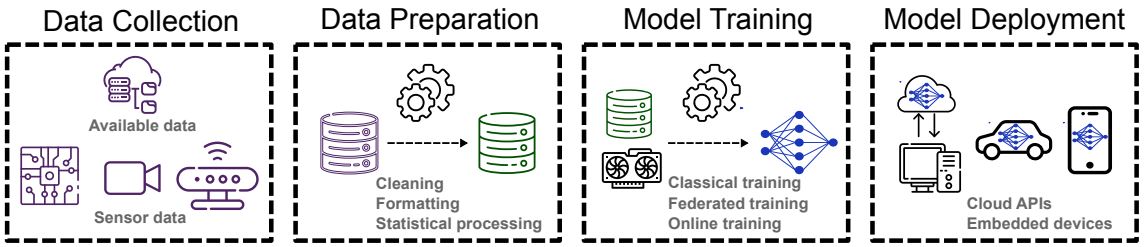

Figure 2.1: Complete machine learning pipeline. Different phases of the development of a model.

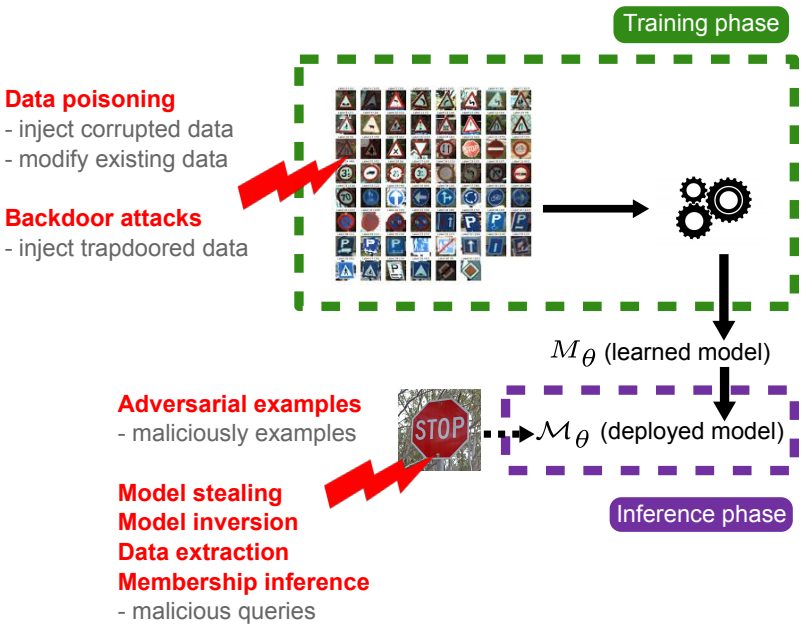

Figure 2.2: Different ways of attacking a machine learning model pipeline.

2.1.2 Attacking at training-time

Training threats encompass all attacks that can be performed during the first, second and third stage of the development of a model, as presented in Figure 2.1. The goal of such attacks is to maliciously influence the training stage, in order to eventually have some specific influence on the final model. This kind of threats thus relates to the modification of the training data set. There are two main scenarios where such attack is feasible.

Firstly, it may occur when the training phase is outsourced on the cloud for example. In this case, an adversary that manages to penetrate the system responsible for the training phase can modify at will the training set, in order to mount an attack.

The second scenario, which takes place when the data gathering and preprocessing steps are kept private, requires a more powerful adversary than in the previous scenario. Indeed, as the adversary is supposed to be able to interfere during the training phase, it indicates that the adversary is one of the stakeholders of the project dedicated to the development of the machine learning model. Depending on his role, and the type of project, he may interfere in two different ways.

The first way is related to *federated learning* which designates a machine learning technique where a model is trained in a decentralized way among different collaborative actors. Each actor participates to the training phase with his own data, training a local model, and then sharing model updates with other actors on a common server to update a global model, which then sends updates to each participant. During this process, the data of each actor is not communicated to the others actors, thus addressing privacy and confidentiality requirements. Considering the privacy gains brought by such a method, federated learning is a well-spread process in various critical domains, such as medical and healthcare or defense [17, 18]. With federated learning, the adversary is therefore one of the collaborative actors, acting as a malicious participant. The main characteristic of this scenario is that the adversary is limited to modify his own training data. After performing the attack on his data, he will update his local model accordingly, and share the compromised updates with other collaborative actors.

The second way is related to *online learning*. In this case, external users are requested (voluntarily or not) to provide meaningful data, which will be later used to fine-tune the model: parameters of the model are continuously updated to allow it to be more efficient for future inputs. For example, users are asked about the pertinence of the output of a recommendation system, to be able to later provide them with more pertinent ones. The adversary can then be one of these users, and provide false feedback on the recommendations.

These threat scenarios concern the *poisoning attacks* and *backdoor attacks*, which are detailed in section 2.2.

2.1.3 Attacking at inference-time

Inference threats encompass all attacks that can be performed during the fourth (final) stage of the development process of a model (cf. Figure 2.1.). The range of potential adversaries performing this type of attack is therefore much wider than for training threats. Indeed, where it was limited to actors having access to the training phase, it now involves every actor having access to the deployed model.

Depending on the type of access, details about the machine learning model may be or not available to the adversary. These details include the type of machine learning algorithm used, knowledge of the model parameters, etc. The information that is available to an adversary is strongly linked with the real-life scenario within which the machine learning model is deployed. Considering an API relying on a machine learning model, everyone who is able to use this API is a potential adversary. However, as it is often the case with distant APIs such as the Google Cloud Vision API [1] or Clarifai[2], internal details of the underlying models are not communicated. Therefore, the adversary, which can be any user of this API, is limited to querying the API to mount an attack. Considering a machine learning model implemented without obfuscation method on a mobile device, the adversary can be any individual having access to the hardware platform (e.g., a 32-bit microcontroller, a typical platform for many IoT domains), and internal details may be easy to reverse engineer.

Relatively to the goal of an adversary, these threats, detailed in the next section, can be divided into five types: *model inversion* attacks, *data extraction* attacks, *model stealing* attacks, *membership inference* attacks and *adversarial examples* attacks.

2.2 Training threats

In this section, we present conceptually *data poisoning* and *backdoor* attacks at a high-level, and present illustrative examples for each type of attack.

2.2.1 Data poisoning

Principle In order to deteriorate the quality of a deployed model, an adversary can mount a *data poisoning* attack by either injecting new poisoned data in the data set used for training the model, or maliciously modifying existing training data. Data poisoning attacks may aim at decreasing the accuracy of the target model on specific test set examples [19, 20], or for the whole test set [21–23]. In the latter case, the deployed model can therefore end to be particularly not accurate, preventing it being used in a system. Indeed, if the model does not satisfy some requirements in terms of task-based performances (e.g. accuracy), defined by requirements of the system it is deployed in, it can be declared as unreliable. The availability of the system relying on this model may then be compromised. Therefore, data poisoning attacks can be particularly damaging in critical application domains such as healthcare, where the system is in charge of managing decisive operations. In the former case, where specific examples are targeted, it allows to fool a model on specific examples, and consequently severely harm the decision process of the model. Indeed, when the model is responsible for identifying anomalies in a process, or detect unauthorized users, data poisoning thus can make an anomaly fall between the cracks, or allow an unauthorized user to bypass the detection system.

Examples Shahafi *et al.* [19] show that 50 poisoned examples are sufficient to mount an attack on the CIFAR10 data set [24], which makes a model misclassify a specific target example, with 70% success rate. On the same data set, and without access to the target model, Zhu *et al.* [20] show that, with only poisoning 1% of the training set, an adversary can mount an attack to make the target model misclassify a test set example.
Biggio *et al.* [21] decrease by up to 15% the accuracy of a Support Vector Machine trained to distinguish two classes of the MNIST data set [25], introducing only one poisoned example. [23] shows that an adversary introducing 3% of poisoned data in the training set can make the test set error increase by 11% for a sentiment analysis task [26]. Muñoz-González *et al.* [22] mount a poisoning attack and show how it can increase the test set error on two data sets, RansomWare [27] and SpamBase [28], for logistic regression, Adaline and a multilayer perceptron. Results are

[1] https://cloud.google.com/vision/
[2] https://www.clarifai.com/

presented in Figure 2.3. Notably, an adversary poisoning 15% of the training set manages to increase the test error of a MLP by 25% on the RamsonWare data set.

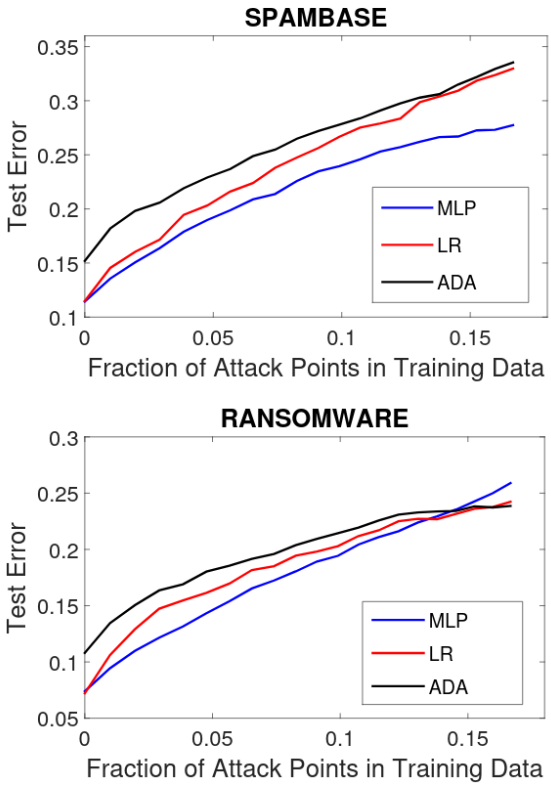

Figure 2.3: Data poisoning attack results (from [22]). For two data sets and different machine learning algorithms is presented the error induced on the test set by perturbing a portion of the training set.

2.2.2 Backdoor attacks

Principle When the purpose of the attack is to trick the model into having specific behaviors on some inputs, the adversary can mount a *backdoor attack*. The adversary intervenes at training time to make the model behave particularly at inference time on inputs in which some specific trigger is included. In order for the model to respond at these specific triggers at inference time, the adversary must include in the training data so-called trapdoored input-label pairs. Trapdoored inputs consist of clean inputs containing specific signals on which the model is trained to produce a malicious behavior. This type of threat is particularly harmful for classification tasks for two main reasons. First, by definition, it allows an adversary to precisely induce misclassification at test time. Second, when the model is deployed, as long as no malicious input is given to the model, its behavior seems standard. As an example, for a detection system, a model may be trained for the classification task of distinguishing between benign and dangerous inputs. With backdoor attacks, an adversary can simply include a trigger in an input to make the model classify it as benign, and therefore bypassing detection. In the case of a user authentication system, this raises severe security breaches, as an unauthorized user can access a guarded facility [29]. In the case of classifying road signs, this can lead to harmful consequences [30].

Examples Gu *et al.* [30] perform a backdoor attack against a U.S. road sign detector, which allows at inference that a simple sticker on a "stop" road sign makes the object detection model to mistake it as a "speed limit" road sign. An illustration of the attack is proposed in Figure 2.4. Turner *et al.* [31] propose a *backdoor attack* method where trapdoor input-label pairs do not raise suspicion when examined by a human. Indeed, many trapdoor attacks require injecting malicious trapdoor inputs with labels in the training set, where the label is inconsistent with the input [30], and would thus easily be flagged as suspicious by a human. Liu *et al.* [32] take advantage of the natural reflection phenomenon to create stealthy backdoor inputs. The malicious cue takes the

form of a reflection on the image. Eventually, Guo *et al.* [29] introduce a backdoor attack, which allows an adversary to impersonate a authorized user, for a face recognition system.

Figure 2.4: Backdoor attack (from [30]). The clean image (left), the clean image with the trigger (middle), and the resulting misclassification (right).

2.3 Inference threats

In this section, we present conceptually inference time attacks at a high-level, and present illustrative examples for each type of attack.

2.3.1 Model Inversion

Principle Considering an already deployed model, the goal of the adversary may be to retrieve hidden information about sensitive features of input data. To this end, he can leverage *model inversion* attacks. Given some target sensitive information of an input sample, the adversary only needs to be able to get the output response of the machine learning model, and/or already have some auxiliary knowledge of some non-sensitive parts of the input data, to retrieve a very good approximation of the target sensitive information. This type of attack constitutes a severe threat to confidentiality and privacy, as a machine learning model may be trained on sensitive data to perform some task, and then be released on the market. The sensitive-features may concern personal privacy, such as information about the identity of a person, or some of his behaviors [33–38].

Examples Fredriskon *et al.* [37] consider a face recognition model, trained on the picture of people under various conditions (exposure, brightness, with/without glasses, etc.) to predict their names. With only the confidence scores corresponding to an input image, they show that an adversary is able to recover a highly resembling image to the one of the input image (see Figure 2.5). Zhao *et al.* [36] also target face models trained on face images. With the knowledge of the target model outputs, as well as gradient-based explanation methods, an adversary can retrieve images highly similar to the true ones. Fredrikson *et al.* [33] consider a linear regression model trained to predict the dose of warfarin based on personal information. Given auxiliary knowledge of the input data, as well as the predicted dose of warfarin, the authors manage to recover sensitive knowledge about genetic markers. Zhang *et al.* [35] leverage a generative adversarial network (GAN), trained with public available data related to the task the machine learning is trained for, to perform a model inversion attack against a face recognition classifier. Given auxiliary knowledge in the form of blurred, or masked input face images, the GAN is used to retrieve the sensitive features and recover the entire input face images. Mehnaz *et al.* [34] show that an adversary can mount a *model inversion* attack with high accuracy and precision to reveal personal privacy related information. With the first data set used in the experiments, the General Social Survey (GSS) data set [39], used to train models that predict the level of happiness of an individual in his marriage, an adversary targets the input information "Have you watched X-rated movies in the last year ?", and the proposed attack reaches 61% accuracy. With the second data set, the Adult data set [28], used to train model that predict if an individual earns more than $50K a year, an adversary targets the marital status of the individual, and the proposed attack reaches more than 70%.

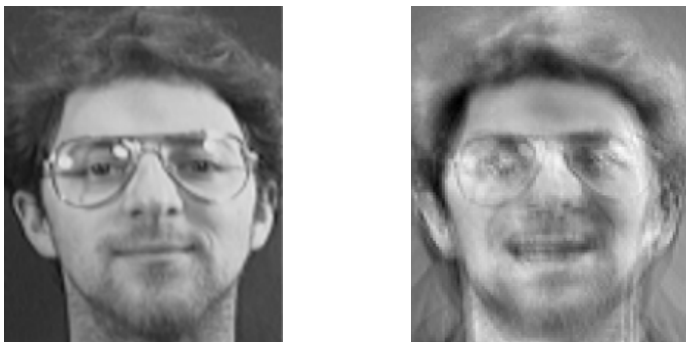

Figure 2.5: Model inversion attack (from [37]). The clean input (left) and the retrieved image with the model inversion attack (right).

2.3.2 Data extraction

Principle Given a deployed model, the adversary may want to retrieve exact parts of the data the model was trained with. To do so, the adversary may leverage *data extraction* attacks. While model inversion attacks allow to recover sensitive features of input data with high fidelity, even if this data is not part of the training set, data extraction attacks allow to recover exact portions of the training data. Data extraction attacks exploit the memorization phenomenon of machine learning models that tend to memorize exact information of the training data. Memorization occurs at an early stage of the training phase, and label memorization, for example, is needed for good generalization [40]. Moreover, this overfitting is not a necessary consequence of memorization, as language models, which do not exhibit overfitting, memorize exact text sequence of the training data [41,42]. Data extraction attacks are therefore a threat to the privacy of information contained in the training data.

Examples. Concerning personal data, Carlini *et al.* [41] show that machine learning tend to memorize infrequent data encountered during training. Exploiting this phenomenon, they manage to extract credit card numbers from a data set of emails sent between employees of the Enron Corporation [43]. Also exploiting memorization, Carlini *et al.* [42] attack the GPT-2 language model, trained on data scraped from the internet [44], and manage to extract the name, the phone number, the fax number, and the physical address of individuals. An illustration of this attack is presented in Figure 2.6.

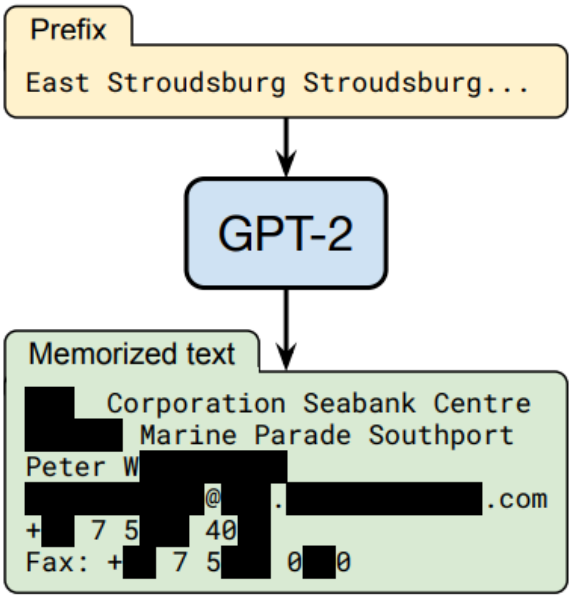

Figure 2.6: Data extraction attack (from [42]). A particular query allows to retrieve the name, the phone number, the fax number, and the physical address of some individual.

2.3.3 Model Stealing

Principle When the goal of an adversary is to reverse engineer a hidden machine learning model, i.e. retrieve a hidden machine learning model with high fidelity, he can mount a *model stealing* attack. This kind of threat is particularly concerning, for two reasons. On one hand, this raises private property and copyright issues. Indeed, model stealing attacks allow to copy an existing model without bearing the same costs as the model issuers, such as research and development costs and training related expenses (some actual models come with hundred of millions of parameters which have to be fine-tuned). An adversary can then either use a model for his own purpose without paying for it, in case of the initial model usage was a paid service, or release his copy on the market, causing intellectual property breaches and unfair competition. On another hand, model stealing attacks ease the mounting of various types of attacks against a hidden model. Whether the adversary wants to perform a model stealing attack, or a membership inference attack or craft adversarial examples (these two last threats are presented thereafter), being able to have a more detailed knowledge of the target model allows to mount more dangerous attack. Indeed, in the case of the target model being hidden to the adversary, model stealing attacks allow him to obtain a local substitute model on which he will fine-tune his attack. Once he has derived the best way to perform some attacks, he can use it against the distant target model. To perform a model stealing attack, the adversary can take advantage of a purely mathematical reasoning [45–47]. He can also exploit leakages related to the physical implementation of a model with Side-Channel Analysis (SCA) [14, 15, 48, 49], which denotes any kind of attack exploiting information collected from the implementation of a system.

Examples Jagielski *et al.* [45] target a two-layer neural network model using the ReLU activation function. Exploiting the way the ReLU function change sign, they manage to completely recover weight values of the first layer of a model. The weights of the second layer can then be determined algebraically. Carlini *et al.* [46] manage to mount a fast and precise *model stealing* attack by seeing the model extraction as a cryptanalytic problem. Oh *et al.* [47] use a metamodel, which predicts the attributes of the target hidden model by exploiting the responses of the target model to specific inputs. Moreover, the metamodel is able to find meaningful input queries regarding the goal of retrieving information about the target model.

Batina *et al.* [14] use SCA methods such as the Correlation Power Analysis (CPA) to exploit electromagnetic emanations from an ARM Cortex-M3 microcontroller on which a neural network model is implemented, in order to retrieve important information such as the activation functions used, the number of layers and the weight values, as 32-bit floating point information. The setup of the attack is presented in Figure 2.7. Maji *et al.* [48] also use SCA to exploit timing, electromagnetic and power leakages to perform a *model stealing* attack for models implemented on two very common micro-processors (ATmega328P, ARM Cortex-M0+) and a RISC-V platform. Hua *et al.* [49] use off-chip memory access information of a hardware accelerator to retrieve information about the architecture and weights of a neural network model.

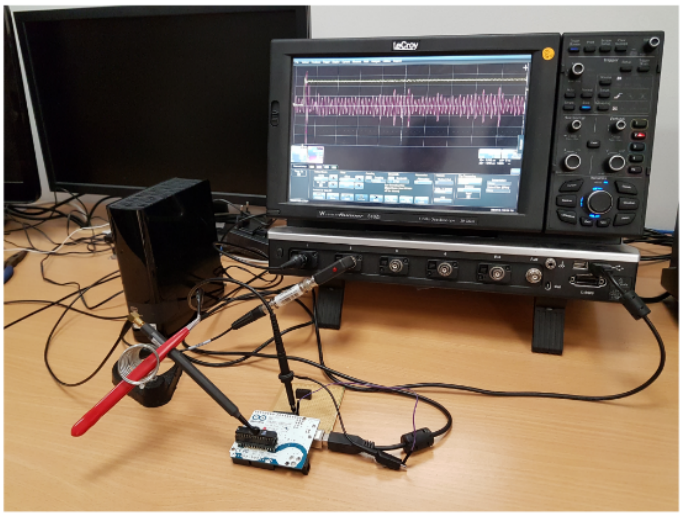

Figure 2.7: Setup of a model stealing attack based on side-channel analysis (from [14]).

2.3.4 Membership Inference

Principle In order to determine if a particular record of data was part or not of the training set, an adversary can mount a *membership inference* attack. This type of threat can lead to severe privacy disclosure. Indeed, knowing that an individual was part of the training set may reveal sensitive information about him. As an example, a model may have been trained to predict some medication dosage value based on samples of medical data of people suffering from a particular disease. Considering a medical data sample, determining that this sample was used to train the model implies that the person is suffering from the disease.

More generally, membership inference attacks are therefore a growing concern for machine learning models related to private data, as the disclosure of this data can have highly undesirable consequences on people's lives. Specifically, medical and healthcare related models are particularly threatened by the nature of the information needed to train these models [50]. Moreover, not only the data used to train a specific target model can be under attack, but also the data used to train a model that is later used via transfer learning to train the target model [51]. Consequently, as transfer learning is a very widespread method, membership inference attacks have therefore a very extended threat surface.

Examples Yeom *et al.* [52] establish a strong link between overfitting and the vulnerability of machine learning models against membership inference attacks. Consequently, for many models, an example well predicted may be sufficient to ascertain that is was part of the training set. Shokri *et al.* [53] perform *membership inference* attacks for a classification task, which are based on the confidence score vectors given by the target model. In fact, this attack is performed by training a machine learning model for each class, which predicts if an example is part of the training set of the target model, by taking as inputs the confidence score vector given by the target model. This attack process is illustrated in Figure 2.8. Leino *et al.* [54] take advantage of the fact that a model learns features which are only predictive of the training data. Considering an adversary having a complete access to the target model, they mount a membership inference attack by investigating if a model relies on such type of features. Sablayrolles *et al.* [55] derive formally the optimal strategy to perform a membership inference attack, and show that the adversary only requires to have access to the loss function value for the target example. Therefore, an adversary with limited information, i.e. only able to access loss function values, is as powerful as an adversary having a complete access to the target model. Choo *et al.* [56] show that, under the assumptions that the adversary knows the target architecture and training setup, as well as data from the same distribution as the one used to train the target model, an adversary can perform efficient membership inference attacks by only having access to the output label.

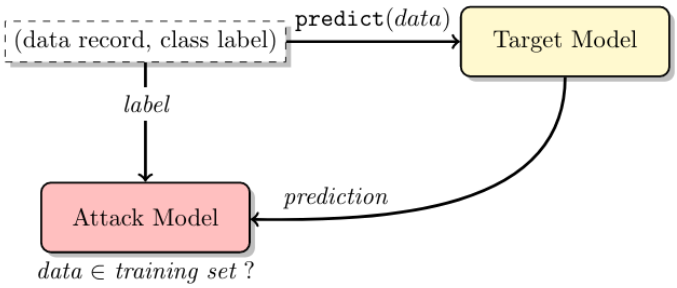

Figure 2.8: Illustration of the attack process from [53]. From the confidence score of the target model, and the ground-truth label of the record, the attack model predicts if the record was part of the training set or not.

2.3.5 Adversarial Examples

Principle When the goal of the adversary is to fool the inference process of a machine learning model, he can use *adversarial examples* attacks. This threat consists in maliciously modifying a clean input, by adding an adversarial perturbation to it, in order to deceive the prediction of a target model. Importantly, this adversarial perturbation has to be imperceptible or appear as benign, in order not to raise suspicion from a human observer. The phenomenon of adversarial examples arises from the fact that machine learning models leverage features different than those used by humans when performing predictions [57]. Notably, models can rely on specific signals

in an image, to which humans are not sensitive to. Perturbing these features therefore allows an adversary to efficiently fool a model in an inconspicuous way. As this type of attack allows to trick a model into making an erroneous prediction for a specific input at inference time, it is a menace to the integrity of the prediction scheme of a wide range of systems. As an example, an adversary managing to include adversarial perturbations on fake glasses can bypass efficiently a face recognition system [58]. Moreover, *adversarial examples* for images have been shown to be feasible in real-world conditions, with various weather conditions and possible points of view [59].

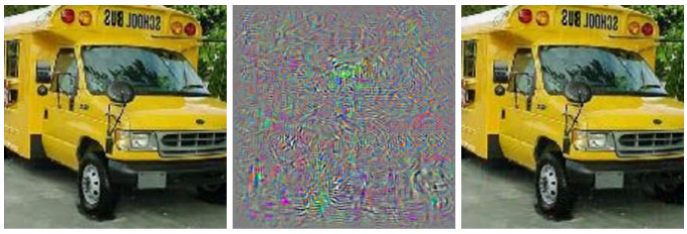

Figure 2.9: Illustration of adversarial examples from [11]. Clean image classified as a "school bus"(left), magnified adversarial perturbation (middle), and resulting adversarial example misclassified by the target model as "ostrich" (right).

Examples Szegedy *et al.* [11] firstly introduce a method to craft *adversarial examples* against a neural network model by searching for a small perturbation maximizing the classification loss. An illustration of adversarial examples crafted with this method is proposed in Figure 2.9. Engstrom *et al.* [60] show that an adversary can exploit malicious rotations and translations in an image classification task to craft *adversarial examples*. Shamsabadi *et al.* [61] propose an attack method where *adversarial examples* are crafted by identifying parts of an image, where color modification does not raise suspicion, and subsequently maliciously modify their colors to fool a machine learning model. Su *et al.* [62] leverage differential evolution strategies to craft *adversarial examples* by only modifying one pixel in an image. Cheng *et al.* [63] show how to craft adversarial examples against a classification model in a very limited setting when the adversary can only query the target model and retrieve the final output label. Interestingly, Papernot *et al.* [64] introduce the phenomenon of *transferability*, which constitutes a powerful tool for an adversary with a constrained access to the target model. In fact, they show that an adversarial example crafted on a substitute (source) model can fool another machine learning model (target). Importantly, the machine learning models do not need to be of the same type (e.g. adversarial examples can be crafted on a support vector machine and fool a neural network model) [12]. Therefore, an adversary can train his own substitute model, craft adversarial examples on it, and transfer them to the target model, on which it would have been more difficult to craft adversarial examples due to a limited access. In Chapter 4, we give a detailed description of the phenomenon of adversarial examples.

2.4 Confidentiality, Integrity and Availability

There is an important distinction to be made for the threats presented above, whether they occur during the training or the inference phases of a model. On one hand, the adversary's goal is to steal information about a model or the input, such as for *model inversion*, *model stealing*, and *membership inference* attacks. On the other hand, the adversary's goal is to tweak the behavior of a model, as for *data poisoning*, *backdoor attacks* and *adversarial examples* attacks. On a higher level, the goal of each of this category of attacks can be viewed under the CIA triad (Confidentiality, Integrity and Availability) that is the classical paradigm in IT (Information Technology) security. These three concepts enable to exhaustively cover the different aspects to be protected for an IT system. Whenever an attack of any type succeeds against a system (e.g. data stealing, denial of service, unauthorized access, etc.), this indicates that one or several of these three principles has been violated. Figure 2.10 displays the different attacks linked with confidentiality, integrity or availability.

Confidentiality refers to the ability to protect confidential or private data from unauthorized access, in order to prevent a malicious disclosure of it. Private data may refer to confidential personal data such as medical or financial records, as well as to data protected by intellectual

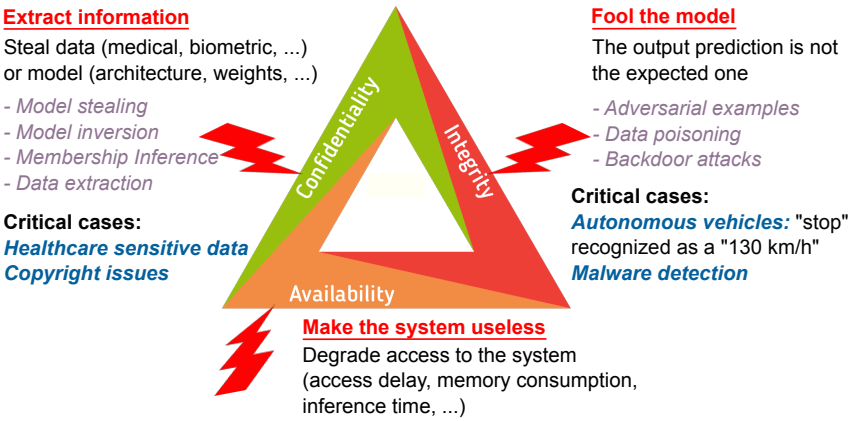

Figure 2.10: CIA (Confidentiality, Integrity, Availability) triad for machine learning models with corresponding possible attacks.

property legislation, such as copyrights or patents. In the case of a machine learning model, the confidentiality of sensitive information is either related to the data at stake or to the model in itself. Relatively to their goals, attacks involving confidentiality threats are *model inversion*, *data extraction*, *membership inference* or *model stealing*. For the first three attacks, the privacy of the data is threatened, whereas for *model stealing*, the intellectual property can be stolen.

Integrity refers to the protection against any malicious modification that aim at compromising the correct achievement of a task. When related to a machine learning model, integrity refers to the integrity of the inference process. An adversary mounting an attack against the integrity of the model targets its inference process, in that he aims at compromising the integrity of the output. *Backdoor attacks* and *adversarial examples* are the attack methods reviewed previously, which aim at targeting the integrity of a machine learning model.

Availability designates the ability to keep a system accessible to authorized users. The goal of an adversary targeting the availability of a system is to prevent access to it, or significantly degrade its quality, with respect to specific requirements. Importantly, availability attacks have to be considered not in terms of a model only, but in terms of the overall system containing the model, contrary to confidentiality and integrity. Indeed, as mentioned in [65]: "Availability is however ill defined for a model in isolation but makes sense for the ML-based system as a whole". For example, considering latency requirements, the availability can be threatened with denial of service attacks, which completely block the access to the system, or other attack methods increasing the response time. Considering quality related requirements of a system, an adversary can target memory constraints or processing time requirements. Task-based performance can also be under attack, in order to jeopardize a reliable usage of a system containing a machine learning model. As an example, the accuracy of a model can be at stake. In this case, attacks against the availability of a system show a link with integrity attacks, but differ conceptually. Indeed, in the case of a poisoning attack, or for adversarial examples, modifying the confidence scores is a goal in itself to reduce the accuracy on specific examples. In the case of an attack against availability focused on task-based performances, the final goal is to impact the behavior of the system in which the model takes place. *Data poisoning* attacks targeting a loss of accuracy on all examples is a form of attack against the availability of a system. Recently, Shumailov *et al.* [66, 67] introduced two availability oriented attack schemes. *Sponge examples* [66] increase the latency and energy consumption of the inference for neural networks implemented on devices. With data ordering attacks [67], an adversary can slow down the training process. These results have also been demonstrated by Grosse *et al.* [68] who target the initialization of the weights by taking advantage of some properties of the ReLU function.

2.5 General threat model

Considering any system, its security is relative to a certain type of attack performed by a specific adversary. When designing a defense or an attack method, in order to properly evaluate its efficiency, it is therefore a mandatory step to define both the objective of the adversary and his

range of action. Considering some malicious goals of an adversary against a machine learning model, the threat model will detail at which step of the machine learning process (from training to inference) the way the adversary interferes, as well as how he is allowed to interfere, and to what extent. The adversary threat model is composed of three components: *adversary goal*, *adversary knowledge* and *adversary capacity*.

Adversary goal

The *adversary goal* formally defines the objective of the adversary when attacking a machine learning model. For example, the goal of an adversary performing *membership inference* attacks is to find if a record was used as part of the training set. In the same way, the goal of a *model stealing* attack can be to retrieve the precise architecture of a hidden neural network model. The goal of *adversarial example* attacks is to fool the machine learning model inference process by crafting an adversarial perturbation that is added to a clean input to form an adversarial example.

Adversary knowledge

The *adversary knowledge* specifies what the adversary is able to know about the model under attack. For example, for a neural network model, an adversary may be in a complete white-box setting, therefore being able to know everything about the model, i.e. its exact architecture, weights values, the loss function used for the training phase, etc. In a more restrictive setting, such as a gray-box or black-box setting, the adversary may not have access to the model, being only able to query it to get output values. Moreover, depending on the setting, the output values the adversary can retrieve may vary. In a classification task, he may be only able to retrieve the final predicted class, or the confidence scores for the top three values. Reasonably, the more an adversary will have an exhaustive knowledge of the model, the more he will be able to mount efficient attacks.

Adversarial examples attacks illustrate this claim well, notably by considering the ability of an adversary to compute the gradients of a machine learning model to craft adversarial perturbations. Indeed, an adversary in a white-box setting will be able to derive formally the gradients of the loss used to train the model with respect to the input. This will allow him to compute precisely the minimum adversarial perturbation needed to fool a model. On the contrary, if the adversary is in a black-box setting, being able to query the target model but only retrieving the output confidence scores, he will have for example to use estimates of these gradients, leading to less optimal perturbations.

Adversary capacity

The *adversary capacity* simply sets a bound on the ability of the attacker to cause harm. For some attacks, given some *adversary goal* and *adversary knowledge*, an unconstrained adversary would result in an attack that would be both unrealistic and without possible way of countering it. For example, in the case of *adversarial examples*, not defining a proper *adversary capacity* results in an adversary which could simply modify weight values at test time to cause misclassification. Such an adversary, able to modify the weights values at each inference, would be unstoppable considering software defense schemes. Therefore, for *adversarial examples*, the mostly common adversary capacity in the literature consists on a bound of the size of the adversarial perturbation (e.g. considering the l_p norm of the adversarial perturbation).

2.5.1 Establishing a threat model

Considering some target machine learning model and an adversary with a defined *adversary goal*, the *adversary capacity* and *adversary knowledge* arise predominantly from real-life use-cases of the target model. Firstly, the way of performing the training phase will allow an adversary to intervene or not. When kept private, no training phase threat is reasonably feasible. When delegated to the cloud or involving many participants, such as for federated learning, exposures to training time attacks are widened. Secondly, the intended use of the model once deployed will greatly influence the access to an adversary willing to perform inference time attacks, as it defines the way users can interact with it. Therefore, depending on practical concerns regarding the training and inference phases, the *adversary capacity* and *adversary knowledge* will be more or less constrained.

For example, considering a model trained to perform a classification task, and an adversary willing to fool it with *adversarial examples*, the real-life scenario regarding its deployment will strongly influence the *adversary knowledge*. If the model is made available through a private API offering to query it and get the label output, the *adversary knowledge* resumes to the final

label only. The adversary does not know the machine learning algorithm used, and is thus in a strong black-box setting. On the contrary, if the model is deployed on local devices such as microcontrollers without any obfuscation scheme, the adversary is now in a complete white-box setting, knowing all details about the machine learning models.

When developing an attack or defense scheme, it can also be very meaningful to consider a strict threat model. On one hand, in the case of a defense scheme, considering an adversary in a complete white-box setting allows to study the worst-case scenario for the defender, against an adversary able to mount adaptive attacks. Demonstrating the efficiency of a defense scheme in this setting justifies its validity for all possibles use-cases. On another hand, when developing an attack method, considering a very restricted access to the target model allows to show its efficiency even in hard black-box setting, where only very limited information is available to the adversary.

2.6 Conclusion

In this chapter, we have considered a general supervised machine learning model, and presented the existing attacks that can target it. Precisely, we introduced the different specific features of the deployment of a machine learning model, in order to better identify which type of adversary can intervene at each moment of the machine learning pipeline. For each existing attack, whether it requires acting at training or inference time, we detailed it at a conceptual level, describing its principle, laying out some examples for each of them. We explained how these current threats can be viewed under the CIA (Confidentiality, Integrity and Availability) triad. Eventually, we specified the threat model, whose definition is mandatory when considering an attack or defense scheme against a machine learning model.

In Chapter 4, we will detail in depth the adversarial examples phenomenon against neural network models used for an image classification task, as it is the core subject of this thesis.

Chapter 3

Notations, formalism and data sets

3.1 Miscellaneous

The range of integers from N (included) to M (included) is denoted by $[\![N, M]\!]$.
The cardinality of a set $\mathcal{A}$ is denoted by $|\mathcal{A}|$.
Considering a condition $\mathcal{D}$, the indicator function of $\mathcal{D}$ is denoted as :

$$\mathbb{1}_{\mathcal{D}} = \begin{cases} 1 \text{ if } \mathcal{D} \text{ is true} \\ 0 \text{ otherwise} \end{cases} \tag{3.1}$$

The sign function for a real value a is defined as follows:

$$\text{sign}(a) = \begin{cases} 1 \text{ if } a > 0 \\ -1 \text{ if } a < 0 \\ 0 \text{ if } a = 0 \end{cases} \tag{3.2}$$

When applied to a vector, the result of the sign function is simply the vector of the sign function applied to each component of this vector.
The Clip function for a real value a, designed to clip a between two values u and v ($u < v$), is defined as follows:

$$\text{Clip}(a, u, v) = \min(\max(a, u), v) \tag{3.3}$$

When applied to a vector, the result of the Clip function is simply the vector of the Clip function applied to each component of this vector.
For a vector $u \in \mathbb{R}^d$, u_i denotes the component of u at index $i \in [\![1, d]\!]$.
The l_p norm of a vector $u \in \mathbb{R}^d$ for $0 < p < \infty$, denoted by $\|u\|_p$, corresponds to:

$$\|u\|_p = \left(\sum_{i=1}^{d} |u_i|^p \right)^{\frac{1}{p}} \tag{3.4}$$

The l_∞ norm of a vector $u \in \mathbb{R}^d$, denoted by $\|u\|_\infty$, corresponds to:

$$\|u\|_\infty = \max_{i=1}^{d} |u_i| \tag{3.5}$$

The l_0 norm of a vector $u \in \mathbb{R}^d$, denoted by $\|u\|_0$, corresponds to:

$$\|u\|_0 = \sum_{i=1}^{d} \mathbb{1}_{\{u_i \neq 0\}} \tag{3.6}$$

The l_p ball of center c and radius r, corresponding to $\left\{ u \mid \|u - c\|_p \leq r \right\}$, is denoted by $B_p(c, r)$. $B_p(r)$ simply denotes the l_p ball of radius r centered at origin.
We denote by $\text{proj}_{B_p(c,r)}(u)$ the projection of a vector u onto the l_p ball of center c and radius r.
Considering two vectors u and v, the dot product between these vectors is denoted as $u \cdot v$.
The cosine similarity between u an v is denoted as follows:

$$\text{cossim}(u, v) = \frac{u \cdot v}{\|u\|_2 \|v\|_2} \tag{3.7}$$

Considering a function f, such that:

$$f \colon \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^{d_3}$$
$$x_1 \times x_2 \mapsto y$$

The gradient of the function f at point (x_1, x_2) with respect to the variable x_1 is denoted as $\nabla_{x_1} f(x_1, x_2)$. Considering two metric spaces (X, d_x) and (Y, d_y), a function $f : X \to Y$ is said lipschitz continuous if there exists $K > 0$, such that $\forall (x_1, x_2) \in X \times X$:

$$d_y(f(x_1), f(x_2)) \leq K \; d_x(x_1, x_2)$$

K is said to be a lipschitz constant of f.
Considering two discrete probability distributions p and q with support A, the cross entropy is given by:

$$CE(p, q) = -\sum_{a \in A} p(a) \; \log \; q(a) \tag{3.8}$$

The Kullback-Leibler divergence between p and q is given by:

$$KL\left(p\|q\right) = \sum_{a \in A} p(a) \frac{\log\ p(a)}{\log\ q(a)} \tag{3.9}$$

$\mathcal{N}(m, \sigma^2)$ denotes the Gaussian distribution with mean m and standard deviation σ.
For a model trained for a classification task and a set of adversarial examples, the adversarial accuracy denotes the accuracy of this model on the set of adversarial examples. For example, an adversarial accuracy of 0.3 for a set of 100 adversarial examples crafted from 100 well-classified examples means that the model still classifies well 30 of these adversarial examples.

3.2 Classification task for a neural network

We consider a classification task where the goal is to map each input from an input space $\mathcal{X} \subset \mathbb{R}^d$ to one of the K possible class labels from the label space $\mathcal{Y} = \{1, 2, \ldots, K\}$. Importantly, for this task, we suppose that input-label pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are sampled independently from an unknown probability distribution $\mathcal{D}$.

3.2.1 Inference process

To solve this classification task, we consider a neural network model M_θ. For the sake of readability, the notation θ for the model parameters is not mentioned in the following notations, except when necessary. This model allows to assign to an input $x \in \mathcal{X}$ a confidence score to each class, in the form of a *confidence score vector* $F(x) = (F_1(x), F_2(x), \ldots, F_K(x)) \in [0,1]^K$, where $\sum_{c=1}^{K} F_c(x) = 1$. The predicted label is then $M(x) = \text{argmax}_i\, F_i(x)$. The function $F : \mathcal{X} \mapsto [0,1]^K$ is the *score function* of M. The *logits*, consisting of the output vector at the final layer, without considering an activation function, is denoted by $h(x) = (h_1(x), h_2(x), \ldots, h_K(x))$. The *confidence score vector* is obtained by applying the softmax function to the logits:

$$F_i(x) = \frac{e^{h_i(x)}}{\sum_{j=1}^{K} e^{h_j(x)}} \tag{3.10}$$

The output vector at the penultimate layer of a neural network model is denoted by $g(x)$. This vector corresponds to what is often called *features* in the deep learning literature such as [57]: the *logits* are a linear combination of the values in $g(x)$, and so the prediction of the model can be seen as learning a linear classifier on these feature values. The pipeline for the prediction scheme for an input x is summarized in Equation 3.11.

$$x \in \mathcal{X} \longrightarrow \cdots \longrightarrow g(x) \longrightarrow h(x) \in \mathbb{R}^K \xrightarrow{softmax} F(x) \in [0,1]^K \xrightarrow{argmax} M(x) \in \mathcal{Y} \tag{3.11}$$

3.2.2 Training process

In order to perform the training phase of the model M_θ, it is needed to first define a cost function $\mathcal{L}(x, y, M_\theta) : \mathcal{X} \times \mathcal{Y} \times [0,1] \mapsto \mathbb{R}$, which indicates how much the predicted label given by M_θ fits the ground-truth label y (the smaller the more accurate the model is). The objective of the training phase of the model M_θ being to obtain a model minimizing the cost function for input-label pairs (x, y) drawn from $\mathcal{D}$, it thus consists in minimizing the error measured by $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}(x, y, M_\theta)\right]$. Therefore, the objective of the training phase of M_θ resumes to the following optimization problem:

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}(x, y, M_\theta)\right] \tag{3.12}$$

However, as the distribution $\mathcal{D}$ is unknown, the training phase of M is performed in practice by minimizing the *empirical error*, which resumes to minimizing the cost function on the available training set $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$. Practically, the training phase of M_θ therefore resumes to the *empirical risk minimization* problem, given as:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(x_i, y_i, M_\theta) \tag{3.13}$$

In practice, as solving directly the empirical risk minimization problem given in Equation 3.13 is intractable, the Stochastic Gradient Descent (SGD) method is performed. SGD consists in

updating iteratively the parameters θ, by considering – at each step – a batch of samples of the train set, instead of the whole train set. More precisely, at each iteration, a batch of $B < n$ input-label pairs is drawn, $(x_1, y_1), (x_2, y_2), \dots (x_B, y_B)$, and the parameters are updated as follows:

$$\theta \leftarrow \theta - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(x_i, y_i, M_\theta) \tag{3.14}$$

where η is the learning rate. After the model M_θ is trained using the train set, its performance is evaluated on the test set, to assess its *generalization* capacity. The difference between the error on the training set and the error on the test set is called the generalization gap.
For a classification task, a typical example of a cost function is the cross-entropy loss. Considering an input-label pair (x, y), the cross-entropy loss measures the adequacy between the model confidence score vector $F(x)$ and the one-hot encoded vector version of y, denoted as y_{hot}. The cross entropy-loss is given by:

$$\mathcal{L}_{CE}(x, y, M_\theta) = CE(y_{hot}, F(x)) \tag{3.15}$$

$$= -\sum_{c=1}^{K} \mathbb{1}_{\{c=y\}} \log F_c(x) \tag{3.16}$$

Considering a neural network model, we will use the term *standard model* to refer to a model trained with the cross-entropy loss.

3.2.3 Data sets

Throughout this thesis, we will mention and use different classification data sets, which are common data sets used in the adversarial examples literature. We present in this part all these data sets.

MNIST

The MNIST (Modified National Institute of Standards and Technology) data set [25] is composed of grayscale images of handwritten digits, each image representing a digit from "0" to "9". The task consists in attributing to each input image of a handwritten image the correct digit (label) among the 10 possible. The train set and test set are composed of $60,000$ and $10,000$ image-label pairs, respectively. Each image is of size 28×28. Sample images from the MNIST data set are presented in Figure 3.1.

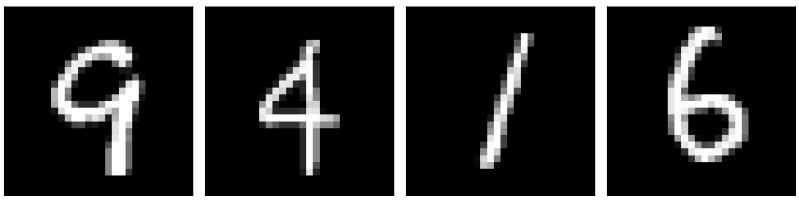

Figure 3.1: Sample images from the MNIST data set [25].

SVHN

The SVHN (Street View House Numbers) data set [69] is composed of real-world color images of numbers taken from Google Street View images. Each image depicts one digit from "0" to "9". The task consists in attributing to each input image of a handwritten digit the correct digit (label) among the 10 possible. The train set and test set are composed of $73,257$ and $26,032$ image-label pairs, respectively. Each color image is of size 32×32. Sample images from the SVHN data set are presented in Figure 3.2.

CIFAR10 and CIFAR100

The CIFAR10 (Canadian Institute For Advanced Research) data set [24] is composed of real-world images taken from the web. The task consists in classifying each image to one of the 10 possible labels: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, *truck*. The train set and test set are composed of $50,000$ and $10,000$ image-label pairs, respectively. Each image is of size 32×32.

Figure 3.2: Sample images from the SVHN data set [69].

Sample images from the CIFAR10 data set are presented in Figure 3.3. The CIFAR100 data set is a finer version of the CIFAR10 data set, consisting in the same images but with 100 classes instead of 10.

Figure 3.3: Sample images from the CIFAR10 data set [24]. From left to right, the correct labels are *cat*, *plane*, *truck* and *horse*.

ImageNet

The ImageNet data set [70] from the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) is a real-world large-scale data set. The task consists in classifying each image to one of the $1,000$ labels. In this thesis, we consider images cropped at size 224×224. The train set and test set are composed of 1.2 million and $50,000$ image-label pairs, respectively. Sample images from the ImageNet data set are presented in Figure 3.4.

Figure 3.4: Sample images from the ImageNet data set [70]. From left to right, the correct labels are *tree frog*, *barber chair* and *brown bear*.

Chapter 4

Adversarial Examples against Neural Network Models

In Chapter 2, we depicted the landscape of the existing threats that may target a general machine learning model. In this chapter, we give an in-depth analysis of the phenomenon of adversarial examples targeting specifically a neural network model, trained for an image classification task.

We begin by presenting a succinct history of adversarial examples in the literature, which is an ever growing concern in the scientific community. Then, in order to better position our work, we focus on the difference between digital and real-world adversarial examples. Secondly, we proceed to detail formally the threat model corresponding to adversarial examples targeting a neural network model trained for an image classification task. Then, we review the main explanations which have been given to analyze the vulnerability of neural network models to adversarial examples. Eventually, we detail various methods that an adversary can use to craft adversarial examples, as well as existing defense schemes. Notably, we present the transferability property, which is a core subject of this thesis.

4.1 The phenomenon of adversarial examples

In this section, we commence by laying out a brief historical review of adversarial examples in the scientific literature. Then, we distinguish between digital (pixel level) and physical (real-world) adversarial examples. Although our work focuses on digital attacks, we give some illustrations of real-world attacks. It allows to better embrace the whole attack surface of adversarial examples, by depicting feasible attacks for every-day life systems.

4.1.1 Brief history of adversarial examples

The seminal work of Szegedy *et al.* [71] in 2013 is the first one recognized to have introduced the vulnerability of neural network models to adversarial examples. It describes them as "counter-intuitive properties" of neural network, misclassifying examples, which are "indistinguishable from the regular examples". It has to be noted that in 2013, a concomitant work from Biggio *et al.* [72] also introduce "evasion attacks" against machine learning classifiers, as examples modified in order to fool a classifier at inference time, but the notion of *indistinguishability* of the perturbation is not at the core of this work.

In 2015, Goodfellow *et al.* [73] extended these works by introducing a very simple method to craft adversarial examples against a neural network for an adversary in a white-box setting. From there, the interest towards adversarial examples in the scientific community has grown exponentially, resulting in an ever increasing number of papers submitted to top-tier conferences over the years. Figure 4.1 illustrates this trend. On one hand, some works were interested in understanding the root cause of this phenomenon. One another hand, some other works focused on developing empirical defense and attack methods.

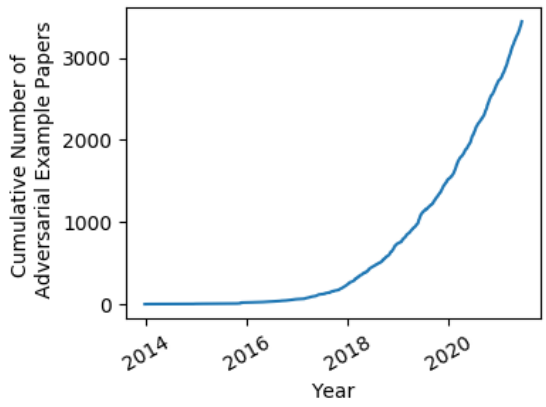

Figure 4.1: From Nicholas Carlini [74]. Number of adversarial examples related papers over the years (extracted from Arxiv).

Regarding the latter type of articles, even if efficient empirical methods were found [75], a major problem appeared: a large number of defense methods submitted to top conferences were defeated quickly after because of a lack of sound evaluation methods [76–80]. In the face of this issue, the

community reasonably understood the need for rigorous adversarial robustness evaluations, for both emerging attack and defense schemes. Consequently, there is an incentive nowadays to properly define the threat model, and meticulously evaluate the robustness with adaptive adversaries [81,82], in order to allow for more rigorous progress in the defense against adversarial examples.

Another type of progress relates to the type of adversarial perturbations, as well as the considered applications domains. First, research does not limit to the l_p norm for the distance between clean and adversarial examples, as introduced in the seminal work of Szegedy *et al.* and Goodfellow *et al.* [71, 73], but has investigated other types of distances such as the Wasserstein distance [83], and perturbations such as rotations and translation [60], color modification [61] and unrestricted but localized perturbations [84]. Second, adversarial examples are not focused anymore only on the image domain, and have spread to other domains such as times series or natural language processing [85, 86].

In parallel to these advances, efficient attack methods continue to be found [87], as well as defense schemes to defend against them [88, 89]. Moreover, the underlying reasons behind the sensitivity of neural networks to adversarial perturbations are increasingly understood [57,90]. Importantly, along the line of empirical defenses are developed provable defenses, providing theoretical guarantees of robustness [91, 92].

4.1.2 Digital and Real-World adversarial examples

Initially focused on image classification tasks [71, 73], adversarial examples have quickly spread to many application domains, such as time series [85, 93], natural language processing [86, 94] and speech recognition [95, 96], and other learning paradigms such as reinforcement learning [97, 98].

Whether this distinction between application domains is illustrative of the vast landscape covered by the threat of adversarial examples, a more important distinction can be made. This distinction revolves around the level at which an adversarial example is considered in a system involving neural networks. Indeed, for such system, adversarial examples can be discriminated whether they occur directly in the digital (pixel) world or in the real (physical) world.

In the former case, adversarial perturbations are considered directly at the model's pixel level input. As an example, for a classification API, the adversary perturbs directly the digital image. In the latter case, physical adversarial perturbations are implemented on real-world objects, and will go through sensors before being digitized and fed as input to the model. Indeed, a system may rely on sensors, such as cameras or microphones, being in charge of capturing real-world data such as images, videos, audio samples, environment feedbacks, etc. and digitizing them according to the neural network input format.

In the case of *physical* adversarial examples, supplementary considerations such as climate variations or environment noises have to be taken into account. As an example, Athalye *et al.* [99] show how to print 3D adversarial objects, which will fool a classifier under various camera distances, lighting conditions, translations, rotations, and solid background colors.

Considering image neural network classifiers, Kurakin *et al.* [100] show that adversarial examples printed on a sheet and then fed to a neural network classifier through a cell phone camera can fool this classifier. Eykholt *et al.* [59] perform real-world adversarial attacks in the form of black and white stickers added to real-world traffic road signs, which fool a neural network classifier recording images at varying distances and angles of view. An illustration of this attack is given in Figure 4.2. Sitawarin *et al.* [101] also propose a method to make advertising road signs, reasonably rejected by the traffic road sign recognition pipeline, being classified as precise traffic road signs. An example is presented in Figure 4.3. In the two works, it has to be noted that the adversarial perturbation, although not imperceptible, is completely benign. Boloor *et al.* [102] manage to deceive an autonomous car driving system by drawing adversarial black lines on the road, which would be considered by a human as simple tread marks.

Against face-recognition systems, Sharif *et al.* [58] design adversarial pair of glasses, which when worn by an individual, allows him to impersonate another individual or not to be detected by the system. Zhou *et al.* [103] design a method for an adversary to fool a face-recognition system, which consists in illuminating the adversary's face with infrared light. The dispositive to diffuse infrared light, composed of very small LEDs, can be hidden in a cap, or a wig. Importantly, as the infrared light is not visible to humans, the infrared light on the face of the adversary does not raise any suspicion. An illustration of this real-world adversarial device is proposed in Figure 4.4.

Considering object detection systems, Thys *et al.* [104] design an adversarial image, which allows an individual not to be recognized as a person. An illustration of this attack is presented in Figure 4.5.

Whether the adversarial examples considered are *digital* or *physical*, crafting them resumes to algorithmic concerns related to neural network internals. The scope of this thesis is the *digital* adversarial examples threat and we will focus on this type of attacks for the next sections of this chapter.

Figure 4.2: Stop sign with adversarial black and white stickers, recognized as a "speed limit 45" road sign, from [59].

Figure 4.3: Clean advertising sign (left), reasonably rejected by the traffic sign recognition system, and its adversarial counterpart (right), recognized as a "stop" road sign, from [101].

Figure 4.4: Physical-world infrared adversarial device from [103].

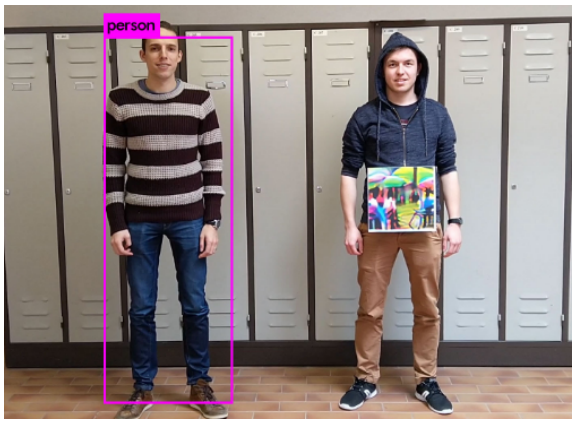

Figure 4.5: Physical attack against an individual detection system, from [104].

4.2 Threat model for adversarial examples against neural networks

In this section, we define the threat model for an adversary wanting to craft digital adversarial examples against a neural network model trained for an image classification task. This allows notably to formally define adversarial examples.

4.2.1 Adversary goal

We consider a target neural network model M_θ, trained for a classification task. As previously presented, adversarial examples, resulting of the malicious addition of an adversarial perturbation to a clean example, are crafted with a twofold objective:

- An oracle $\mathcal{O}$, such as a human, classifies in the same class the clean example and its adversarial counterpart.
- The target model M classifies the adversarial example in another class than the one it classifies the clean example in.

Importantly, these two objectives do not require the clean example to be initially correctly classified by the target model M, they simply require the adversarial perturbation not to raise suspicion.

Targeted attacks

When the adversary wants the adversarial example to be misclassified towards a specific adversarial label y_t, the attack is described as *targeted*. Thus, the goal of the adversary is given by:

$$\text{Find } \delta \quad \text{s.t.} \begin{cases} \mathcal{O}(x+\delta) = \mathcal{O}(x) \\ M(x+\delta) = y_t \end{cases} \tag{4.1}$$

Untargeted attacks

When the adversary does not specify the adversarial label, the attack is described as *untargeted*. Denoting $\mathcal{O}(x)$ the label given by the classification process of an oracle, the goal of the adversary is thus given at a high level by:

$$\text{Find } \delta \quad \text{s.t.} \begin{cases} \mathcal{O}(x+\delta) = \mathcal{O}(x) \\ M(x+\delta) \neq M(x) \end{cases} \tag{4.2}$$

4.2.2 Adversary capacity

In practice, the imperceptibility and benignness characteristics encompassed in the first constraint of the problem given in Equations 4.1 or 4.2 can not be formally expressed. Therefore, in order to account for this, a mathematical proxy $\mathcal{D}$ for this constraint has to be defined, where minimizing this proxy leads to decreased suspiciousness of the adversarial perturbation. This proxy can then be used to define the adversary capacity, which characterizes how much the adversary can alter the clean example x to craft the adversarial example $x' = x + \delta$. The adversary capacity will in fact set an upper bound ϵ on $\mathcal{D}(\delta)$, under which the adversarial perturbation is considered not raising suspicion. The adversary goal described in Equation 4.2 can then be rewritten as the optimization problem given in Equation 4.3.

$$\text{Find } \delta \quad \text{s.t.} \begin{cases} \mathcal{D}(\delta) \leq \epsilon \\ M(x+\delta) \neq M(x) \end{cases} \tag{4.3}$$

The same reasoning applies for Equation 4.1. In Equation 4.3, the proxy $\mathcal{D}$ stays a high-level concept. In the vast majority of works, this proxy is related to the l_p norm. More precisely, it can be related to the l_∞ norm [73], the l_2 norm [105], the l_1 norm [106] and the l_0 norm [107]. In this case, as an example, the adversary capacity is then defined via an upper bound on $\|\delta\|_p$. Some other works have investigated the Wasserstein distance [83]. If the proxy used is relative to

translations and rotations [60], the capacity will consider maximum translations and angle values. In case of localized perturbations such as color changes [61], the adversary capacity specifies which regions of the image can be colored without appearing as unnatural.

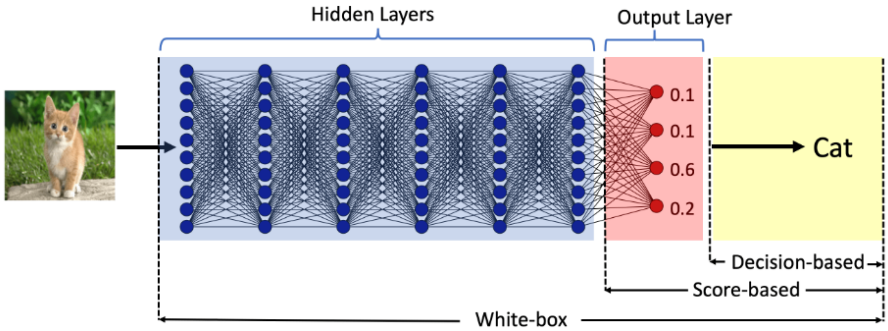

Figure 4.6: Illustration of the adversary capacity in the white and black-box settings, from [108]. In the black-box setting, the attacker may only access to the outputs of the model with different levels of precision (logits, confidence score vector, labels).

4.2.3 Adversarial knowledge

The degree to which the adversary has access to the target model strongly influences his ability to craft adversarial examples against it.

White-box setting. The adversary has a total access to the target model M. He knows the architecture, the parameters values and the cost function $\mathcal{L}$ the model has been trained with. Importantly, this complete knowledge grants him the ability to compute partial derivatives of $\mathcal{L}$, the score function F, the logits function h, and any output function at a hidden layer. This allows him to quantify precisely the variations of these functions with respect to input variations, which is very valuable relatively to his goal. Notably in a white-box setting, the adversary can use so-called *gradient-based* attacks, which are characterized by the ability to derive formally partial derivatives. It is described in more details in Section 4.4.1. Moreover, in a white-box setting no limitation is generally assumed on the number of queries the adversary can make to the target model. The white-box is relative to a worst-case adversary. Notably, when implementing a defense scheme intended at protecting a model against adversarial examples, considering that the adversary also knows the details about the defense scheme allows to ensure that this scheme is efficient against adaptive attacks which are the most powerful feasible attacks.

Black-box setting. The adversary can not compute partial derivatives, which constitutes the main difference with the *white-box* setting. Depending on the information obstruction the adversary faces, many shades of black-box setting exist. In an extreme case, the adversary can only have access to the label output $M(x)$ for an input x. In other cases, only the confidence score vector $F(x)$ or the logits $h(x)$ is available. The adversary can not use *gradient-based* attacks, as he can not derive partial derivatives. In order to craft adversarial examples, the adversary can therefore rely on decision-based or score-based attacks, in the case of an access only to $M(x)$ or $F(x)$, respectively. Details about this type of attack will be presented in Section 4.4.2. Moreover, the adversary may be limited in terms of number of queries. Indeed, when attacking a distant model, each query may be tariffed, and too many queries may also raise suspicion. Instead of using decision-based or score-based attacks, which may be very expensive in terms of computation, the adversary can rely on the transferability phenomenon, detailed in Section 4.4.3. To take advantage of transferability, the adversary needs to train a substitute model, i.e. a model trained for the same classification task as M. Moreover, the closer the substitute model to the target model M in terms of architecture, the better the transferability. Depending on the specificities of the black-box setting, data from the same distribution $\mathcal{D}$ as the one used to train M is available to him or not, as well as the knowledge of the architecture of M.

A high-level summary of the adversary knowledge is proposed in Figure 4.6. Differences between white-box and black-box settings are also detailed in Table 4.1.

Table 4.1: Knowledge of the adversary depending on the white-box or black-box setting. The sign $\sim$ means that the knowledge may vary depending on the tightness of the black-box setting.

Available knowledge of M_θ	*White-box*	*Black-box*
Gradients	✓	X
Predicted label $M(x)$	✓	✓
Confidence score vector $F(x)$	✓	$\sim$
Logits $h(x)$	✓	$\sim$
Architecture	✓	$\sim$
Data from $\mathcal{D}$	✓	$\sim$

4.3 Why do adversarial examples exist?

We defined precisely the threat model corresponding to adversarial examples, where an adversary wants to fool a neural network model trained for an image classification task, while the adversarial perturbation remains benign or imperceptible. This last specificity raises an important point. The feasibility of adversarial examples attacks arises from humans and neural networks not performing image classification in the same way. In this section, we review the main works that aim at explaining this difference, notably the reasons behind the vulnerability of neural network models to adversarial examples.

4.3.1 Early explanations

The first works concerning the origin of adversarial examples explain this phenomenon either as relatively to the distribution of adversarial examples in the input space, or relatively to specificities of the model decision boundary. These investigations are characterized by a low-level appreciation of the problem. They are independent from considerations relative to the high-level concepts learned by the model, specifically the difference of these concepts with the concepts humans rely on to perform classification.

The first explanation was proposed by Szegedy *et al.* [11], which argue that adversarial examples, while being close to training set examples in the input space, are in fact located in low-probability pockets of the input space. During the training phase, the model would then not learn to classify them correctly.

Then, Goodfellow *et al.* [73], hypothesize that the local linear nature of neural networks is at the root of their vulnerability to adversarial examples. The authors explain that given a clean example x and a small perturbation ϵ to x ($x' = x + \epsilon$), if the model score function F is linear in x (i.e $F(x) = w \cdot x$), then ϵ can mean a great perturbation as we have now $F(x') = w \cdot x + w \cdot \epsilon$. The increase is accentuated as the dimensionality increases.

Tanay *et al.* [109] give a geometric explanation: the more the manifold of the data the classifier is trained on is tilted with respect to the classifier decision hyperplane, the smaller the perturbations needed to craft adversarial examples. More precisely, the more a model decision boundary is tilted in the direction of low variance in the data, the smaller the size of perturbations needed to craft adversarial examples. Meng *et al.* [110] make the hypothesis that adversarial examples do not lie on the same manifold as normal examples, while still being close to it. This hypothesis has lead to many defense schemes intended at projecting back adversarial examples on the clean data manifold at inference time, before feeding them to the target model to get the label output [111, 112].

4.3.2 Identifying sensitive features of neural network models

A more recent line of work is interested in identifying the features neural networks are sensitive to. By comparing them to the ones humans are relying on, it helps to explain the vulnerability of models to adversarial examples.

Convolutional Neural Networks are biased towards texture

Geirhos *et al.* [113] show with experiments that Convolutional Neural Networks (CNN) trained on ImageNet tend to rely predominantly on local features such as textures rather than global shapes. On the contrary, humans leverage to a lesser extent textures but rather use various global clues such as silhouettes and edges. An illustration of this phenomenon is presented in Figure 4.7.

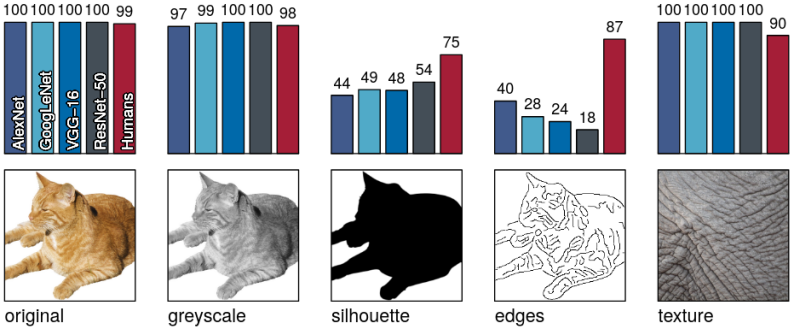

Figure 4.7: Accuracy of humans and models trained on Imagenet on standard and modified images. From left to right images, standard images, grayscale images, silhouette images, edge images and texture images, from [113]. Models seem to rely predominantly on texture to perform classification.

A stylized version of ImageNet is generated with style-transfer in order only to keep shape components: for each image is generated many stylized versions of it such as the shape of the object is conserved while texture varies from one stylized version to another. A CNN classifier trained on this stylized version of ImageNet is shown to rely more on global shape features than a CNN classifier trained on the original ImageNet data set, and performs well on the natural (unmodified) test set. Moreover, the CNN classifier trained on the stylized version of ImageNet is more robust to texture perturbations.

Robust Convolutional Neural Networks rely more on shape and contour

Zhang *et al.* [114] perform several experiments to analyze the concepts on which a CNN model trained with Adversarial Training (a defense scheme intended at making a model more robust against adversarial examples, see Section 4.5.4) relies to perform prediction. The three different concepts considered are shape, contour and texture. To isolate a concept from the two others, three transformations are used:

- *Stylizing* consists in changing the style of an image (with style-transfer), it preserves the shape while destructing textures
- *Saturation* preserves the contours while destructing the textures
- *Patch-shuffling* $k \times k$ splits the image into $k \times k$ parts and shuffles them, it preserves textures while destructing shape.

For each of this three transformations, a data set is generated from the natural one with data augmentation. This allows to separate the concepts models rely on. Indeed, a model showing a little accuracy decrease when evaluated on the stylized data set and a big accuracy decrease when evaluated on the patch-shuffled data set is very likely to rely on the shapes of the images, and not on local textures.

On CIFAR10, Tiny ImageNet (downsized version of ImageNet to size 64×64) and Caltech-256 data sets [115] several CNN models are trained with Adversarial Training, as well as a model trained in the standard way. First, sensitivity maps for the three transformations and all CNNs models are investigated. A sensitivity map here refers to a visualization of the sensitivity of the confidence score for the predicted class to variations of the inputs. The sensitivity maps for adversarially trained CNNs clearly keep shape and contour information, while those for the classical CNN are very noisy. An illustration of this phenomenon is presented in Figure 4.8.

Second, adversarially trained CNNs show higher accuracy than the classical model on the stylized data set. Also, adversarially trained CNNs evaluated on the saturated data set show a smaller accuracy drop when the saturation level increases than the classical model (the more robust the adversarially trained CNN is, the smaller this drop).

Finally, adversarially trained CNNs evaluated on the $k \times k$ patch-shuffled data set show a bigger accuracy drop when k increases (when k equals 8 all models have a 0% accuracy as too much information is lost).

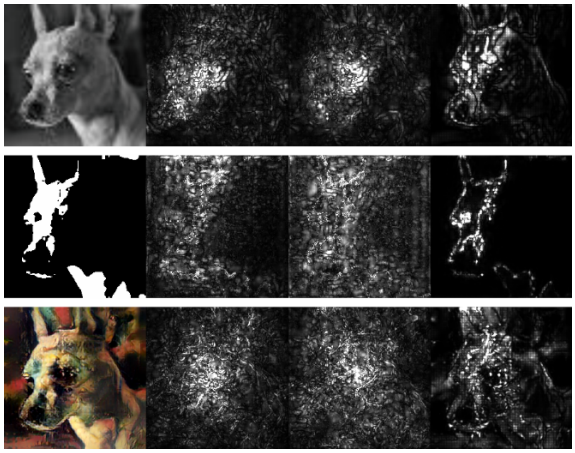

Figure 4.8: Sensitivity maps of different models on a standard image (top row), saturated image to keep contour (middle row) and stylized image to keep shape (bottom row). From left to right, standard model, underfitting model and model trained with Adversarial Training, from [114]. The sensitivity maps for standard models are more noisy than the ones for the model trained with Adversarial Training, more focused on shape and contours.

These experiments show that adversarially trained CNNs rely more on shape and contour information and less on texture, and classical models rely more on textures information than shape.

Moreover, experiments with frequency filters on Caltech-256 show that classical CNNs show equally poor accuracy (around 16%) when evaluated on data set filtered with low-pass or high-pass filters, and an adversarially trained CNN shows poor accuracy when evaluated on data set filtered with high-pass filter but quite good (around 71%) when evaluated on data set filtered with low-pass filter. This may be an indication than adversarially trained CNNs rely more on low-frequency features (textures is a high frequency features while shape is considered a low frequency feature).

Fourier analysis of robustness

Yin *et al.* [90] suggest that the robustness of a classifier to some perturbations depends on the frequency of these perturbations and the frequency of the features the classifier relies on to perform prediction. Experiments are performed to support this idea and illustrate its consequences.

Firstly, a standard model trained on ImageNet achieves 50% accuracy on data preprocessed with the use of a high-pass filter (only high frequencies components are kept). The high frequencies components kept correspond to frequencies invisible to the human eye. This indicates that among all the statistics (a statistic can be color, texture, shape, etc.) that describe the relation between the input and target of the classification task, the classifier may have learned to rely on some of them which are not meaningful to humans. This standard model also achieves above 30% accuracy on low-pass filtered images consisting of indistinguishable colored shapes. An illustration is presented in Figure 4.9.

Secondly, experiments show that models trained with Adversarial Training and gaussian data augmentation become less sensitive to high-frequency perturbations but more sensitive to low-frequency ones. This indicates that these two techniques make the models rely more on low-frequency information. For example, a model trained with gaussian data augmentation will be more robust to high-frequency perturbations such as snow, random noise, etc. but less robust to low-frequency perturbations such as fog and contrast. Similarly, an adversarially trained model has poorer performances when considering the fog perturbation than a naturally trained model. However, adding low-frequency information during training does not improve robustness to low-frequency perturbations. The authors hypothesize that as natural images are more concentrated in the low frequencies, it is harder for the model to become unaware of low-frequency information (thus low-frequency perturbations).

Thirdly, training a model with various perturbation types induce a better robustness, compared to standard and adversarially trained models.

Fourthly, as Adversarial Training makes a model focus more on low-frequency information, adversarial examples crafted on an adversarially trained model tend to present more low-frequency components than for a standard model. This indicates that adversarial examples are not only related to high-frequency components and this explains the failure of defenses relying on high-frequency filtering (JPEG compression for example).

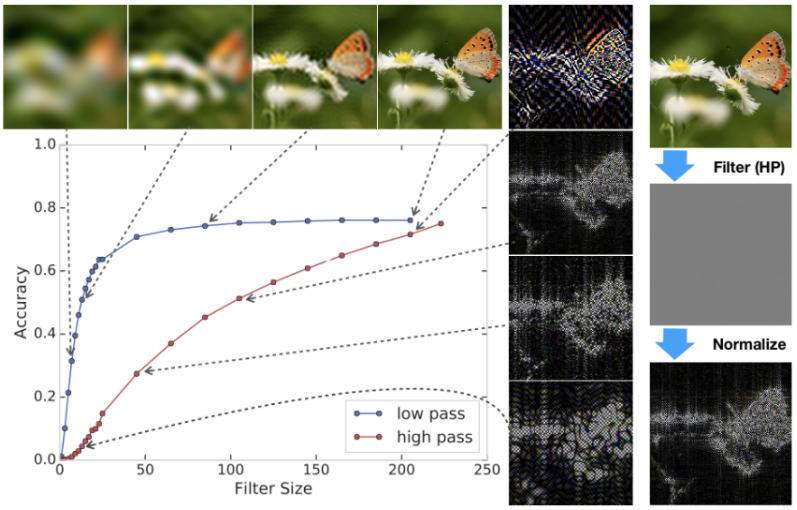

Figure 4.9: Accuracy of a model trained on ImageNet on low-pass and high-pass filtered images, with different filtering intensities, from [90]. The model shows good accuracy even on images that are unrecognizable for humans.

In Chapter 5, we will study the frequency nature of adversarial examples. Notably, we will show that enforcing a model to leverage low frequency information can bring robustness to a model if some conditions are satisfied. Among these conditions, we identify the localization of the information in the frequency spectrum, as well as other factors such as the sensitivity to high frequency noise.

Generalization and frequencies

Wang *et al.* [116] study experimentally the link between the generalization capacity of neural networks and the fact that they rely, or not, on high-frequency components.

Firstly, it is shown that models rely on high-frequency components which are not interpretable by humans, by looking at the recognition accuracy on low-frequency and high-frequency filtered images. Then, it is shown that low-frequency components allow for a better generalization for a correctly labeled data set. Indeed, experiments show that models trained on low-frequency filtered images show higher accuracy on the natural test set than models trained on high-frequency filtered images, meaning that low-frequency components allow for better generalization. On the contrary, models trained on the same data set but with shuffled labels are shown to rely almost equivalently on low-frequency and high-frequency components. The authors claim that this arises from low-frequency components being better aligned with human labels perception.

Secondly, experiments study the influence of different factors. Experiments show that model trained with smaller batch sizes correlate with smaller generalization gaps, and also with invariance to high-frequency components. This agrees with the fact that low-frequency components allow for better generalization. Also, adversarially trained models show invariance to high-frequency components.

Thirdly, experiments study the relation between smoothness of convolution kernels at the input and frequency. Recall that the smoother the convolution kernels at the input, the more the model will ignore high-frequency components. Experiments show that adversarially trained models have smoother convolution kernels than naturally trained ones. However, smoothing convolution kernels only allow for a small robustness increase for naturally trained models, and almost no robustness increase for adversarially trained models.

Intepretability of saliency maps and robustness to adversarial examples

Etmann *et al.* [117] investigate the relation between the robustness to adversarial examples and the interpretability of the saliency maps. Recall that considering an input space of dimension $d = n^2$ (for input images of size $n \times n$ for example), the saliency map is the matrix

$$S(x) = [\nabla_{x_i} F_m(x)]_{i \in [\![i,d]\!]}$$

where $F_m(x)$ is the maximum value of the confidence score vector $F(x)$. The interpretability of the saliency map is represented by the alignment of the saliency map with the input as:

$$\alpha(x) = \frac{|x \cdot S(x)|}{\|S(x)\|_2}$$

The robustness of a model is defined as $\rho(x) = \inf_\epsilon \{\|\epsilon\|_2 \mid M(x+\epsilon) \neq M(x)\}$. Note that this definition of robustness does not require x to be well-classified by the model.

The authors show that for a linear classifier the robustness to adversarial examples and the alignment of the saliency map with the input coincide. For non-linear classifiers, experiments show that this is still verified on average, not necessarily for individual points. Eventually, the more linear the model is locally, the more the relation between robustness and alignment holds.

Kaur *et al.* [118] also notice experimentally that robust models tend to have gradients that are aligned with the human perception. Indeed, an adversarial example x' from a clean example x targeted towards label y_t is crafted with a large perturbation budget on a standard model, a model trained with Adversarial Training (see Section 4.5.4), and a model trained with Randomized Smoothing (see Section 4.5.3). For the two robust models, the adversarial example x' looks like a clean example classified in class y_t, whereas for the standard model, it simply looks like the example x with noisy patterns.

Robust models exploit high-level interpretable features

Engstrom *et al.* [119] show that robust models (here models trained to be robust against adversarial examples with Adversarial Training) learn representations which are more interpretable by humans. A representation that a model has learned for an input x is denoted $g(x)$ and corresponds to the outputs of the penultimate layer. In other words, robust models are shown to extract high-level features which are interpretable by humans.

First, it is shown that for standard models, it is possible to find inputs which are highly visually dissimilar but have close representations. However, for robust models, two images having close representations will be visually similar. This phenomenon is called invertibility of features.

More precisely, considering two source and target images x_1 and x_2, an augmented image is computed as : $x_1' = x_1 + \text{argmin}_\epsilon \|g(x_1+\epsilon) - g(x_2)\|_2$, which corresponds to adding to the source image an ϵ which makes $g(x_1+\epsilon)$ and $g(x_2)$ close. For classical models, x_1' and x_2 will be very visually different, where for robust models x_1' looks like x_2 (even when x_1 is simply random noise). Moreover, when adding a constraint on the l_2 norm of ϵ to force x_1' and x_2 to be visually similar, it is more difficult for robust models to minimize both the distance in the input space (i.e $\|\epsilon\|_2$) and in the representation space (i.e. $\|g(x_1+\epsilon) - g(x_2)\|_2$), than for classical models.

Second, experiments show that robust models allow for easier feature visualization. More precisely, maximizing a component $g_i(x)$ of the representation leads to the same visual modification for different images x (even when x is random noise), and this modification is interpretable by humans (for example maximizing one component of the representation leads to emergence of stripes on different images). However, for standard models, maximizing a component of the representation leads to non-interpretable emerging patterns, which vary depending on the input image x. Therefore, with robust models, one may introduce some characteristic in an image by maximizing the component of the representation corresponding to this characteristic.

Robust and Non-robust features

Ilyas *et al.* [57] show that adversarial examples are the consequence of features derived from patterns with a high predictive power, yet these patterns are meaningless to humans and they can be adversarially modified to fool the target classifier. These features models are inherent to the data set. A feature is a function from the input space $\mathcal{X}$ to $\mathbb{R}$ such that the output value of the target classifier is a linear combination of the feature values. For a neural network, the features values are thus the outputs at the penultimate layer.

More formally, considering a binary classification task, f is said to be *ρ-useful* ($\rho > 0$) if it satisfies Equation 4.4 and is *γ-useful robust* if under the worst perturbation δ chosen in a predefined set of allowed perturbations Δ, f stays γ-useful under this perturbation (Equation 4.5). A ρ-useful feature f is said to be a *non-robust feature* if f is not robust for any $\gamma \geq 0$.

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[y \cdot f(x)\right] > \rho \tag{4.4}$$

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\inf_{\delta\in\Delta} y \cdot f(x+\delta)\right] > \gamma \tag{4.5}$$

When training a classifier in order to minimize the empirical risk, no distinction is made between robust and non-robust features. At inference time an adversary modifying non-robust features can then fool the target classifier.

Experiments are presented on CIFAR10. First, the authors construct a robust data set mainly composed of robust features. More precisely, starting with an adversarially trained model M_{rob}, for each training set point (x, y), x_r is found by solving $\text{argmin}_z \|g(z) - g(x)\|_2$ (where g is the output function at the penultimate layer of M_{rob}). The starting point of the optimization problem is chosen independently from the true label y in order not to introduce features which are not used by M_{rob}. It is shown that a standard model trained on input-label pairs (x_r, y) leads to non-trivial accuracy and robustness. This shows that robustness and accuracy can be reached by discarding non-robust features, i.e. that some non-robust features are responsible for the vulnerability of classifiers to adversarial examples.

Second, the authors construct a data set with mainly non-robust features. More precisely, starting from a clean example x of ground-truth label y, an adversarial example x' targeted towards label y_t is crafted. A model M_{nrob} is then trained using the input-label pairs (x', y_t). This new data set is incorrectly labeled according to humans as x' is close to x, but here the adversarial label y_t is considered as the true label. Thus, when y_t is randomly chosen for each example, the robust features of the initial model are now uncorrelated with the label y_t. Only non-robust features of the initial model are correlated with y_t. It is shown that models trained on this data set reach good accuracy on the normal test set, while having no robustness against adversarial examples. This shows that models rely on non-robust features at training time.

A summary of this finding is presented in Figure 4.10. Eventually, experiments show that the more models learn similar non-robust features, the more transferable adversarial examples are. More precisely, the authors train different models with the precedent data set of points (x, y_t). The models with the highest accuracy on the normal test set are the one with the more transferable adversarial examples.

Figure 4.10: Accuracy of models on the CIFAR10 data set, on standard examples (blue) and adversarial examples (red). $\hat{\mathcal{D}}_R$ denotes the robust data set (i.e. with almost only robust features), and $\hat{\mathcal{D}}_{NR}$ denotes the non-robust data set (i.e. with almost only non-robust features), from [57].

4.4 Attack methods

In the previous sections, we have defined formally adversarial examples, and we have presented relevant work, which gives the intuition about the underlying reasons for their existence. In this section, we present methods intended to allow an adversary to craft adversarial examples against a neural network model trained for image classification. The objective of this section is not to give an exhaustive list of all existing attacks, but rather allow to understand the landscape of attack methods. To this end, we present attack schemes that allow to craft adversarial examples considering different levels of adversary knowledge, and with various ways of searching for an adversarial example.
For all the attacks presented here, we denote by $\mathcal{Q}$ the domain in which inputs are considered. As we study image classification tasks, $\mathcal{Q}$ is typically a range in which values of pixels must fit in. For an image x, we denote by $\text{Clip}(x, \mathcal{Q})$ the clipping operation of every pixel on this image to the domain $\mathcal{Q}$. Unless specified otherwise, the adversary aims at crafting an adversarial example x' considering a clean input-label pair (x, y). The attack schemes are only presented in their untargeted version, unless the attack is specifically designed to craft targeted adversarial examples.

4.4.1 White-box setting

We consider an adversary in a complete white-box setting, i.e. with a full knowledge of the target model's architecture and parameters. The attacks presented in this section are characterized by the fact that, in order to craft advesarial examples, the adversary can explicitly compute derivatives of the score function F, or the logits function h, or the loss $\mathcal{L}$.

L-BFGS algorithm

Szegedy *et al.* [11] are the first to propose a crafting method in a white-box setting. They consider the objective of finding the smallest l_2 adversarial distortion such that the adversarial example is misclassified, as described in Equation 4.6.

$$\begin{aligned} &\underset{\epsilon}{\text{argmin}} \ \|\epsilon\|_2 \\ &\text{s.t.} \begin{cases} M_\theta(x+\epsilon) \neq y \\ x+\epsilon \in \mathcal{Q} \end{cases} \end{aligned} \tag{4.6}$$

To solve this problem, the authors consider the optimization problem given in Equation 4.7, and search for the minimum c (with a box-constrained L-BFGS algorithm) for which the solution ϵ to the problem in Equation 4.7 satisfies $M_\theta(x+\epsilon) \neq y$.

$$\begin{aligned} &\underset{\epsilon}{\text{argmin}} \, c\,\|\epsilon\|_2 - \mathcal{L}(x+\epsilon, y, M_\theta) \\ &\text{s.t. } x+\epsilon \in \mathcal{Q} \end{aligned} \tag{4.7}$$

Fast Gradient Sign Method (FGSM)

Goodfellow *et al.* [73] present the Fast Gradient Sign Method (FGSM). In order to craft adversarial examples, it is searched for the worst-case adversarial perturbation in a l_p ball of radius ϵ relatively to the value of the loss function $\mathcal{L}$ of the target model. This objective is described in Equation 4.8.

$$\begin{aligned} &\underset{\alpha}{\text{argmax}} \ \mathcal{L}(x+\alpha, y, M_\theta) \\ &\text{s.t} \quad \|\alpha\|_p \leq \epsilon \end{aligned} \tag{4.8}$$

By performing a first order approximation of $\mathcal{L}$ around x, the problems turns to:

$$\begin{aligned} &\underset{\alpha}{\text{argmax}} \ \alpha \cdot \nabla_x \mathcal{L}(x, y, M_\theta) \\ &\text{s.t} \quad \|\alpha\|_p \leq \epsilon \end{aligned} \tag{4.9}$$

In their work, the authors only consider the l_∞ norm, and in that case we have $\alpha = \epsilon\,\text{sign}(\nabla_x \mathcal{L}(x, y, M_\theta))$. Therefore, the adversarial example x' is crafted with:

$$x' = x + \epsilon\,\text{sign}(\nabla_x \mathcal{L}(x, y, M_\theta)) \tag{4.10}$$

This attack can be extended to other l_p norms with $1 \leq p < \infty$. In this case, we have

$$\alpha = \epsilon \frac{\nabla_x \mathcal{L}(x, y, M_\theta)}{\|\nabla_x \mathcal{L}(x, y, M_\theta)\|_p}$$

and then x' is given by:

$$x' = x + \epsilon \frac{\nabla_x \mathcal{L}(x, y, M_\theta)}{\|\nabla_x \mathcal{L}(x, y, M_\theta)\|_p} \tag{4.11}$$

Clipping to ensure that $x' \in \mathcal{Q}$ is eventually performed.

Basic Iterative Method (BIM)

Kurakin *et al.* [100] extend the FGSM with a multi-step version of it, the Basic Iterative Method (BIM). This attack scheme consists in iteratively performing the FGSM attack for a given number of iterations, with a step size $\alpha < \epsilon$, projection onto the ball $B_p(x, \epsilon)$ being performed at each step. Clipping to $\mathcal{Q}$ is performed at the end of each step. More precisely, for the l_∞ version of the BIM attack, at step t, the following operations are performed:

$$\begin{aligned} \delta^t &= \alpha \operatorname{sign}(\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta)) \\ x^t &= \operatorname{proj}_{B_p(x,\epsilon)}(x^{t-1} + \delta^t) \\ x^t &= \operatorname{Clip}(x^t, \mathcal{Q}) \end{aligned} \tag{4.12}$$

where x^t denotes the example obtained at step t, and $x^0 = x$. The BIM attack can be seen as the projected gradient descent algorithm applied to the optimization problem given in Equation 4.8, apart from the fact that the sign of $\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta)$ is considered instead of only $\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta)$.

Projected Gradient Descent (PGD)

The Projected Gradient Descent (PGD) [75] attack simply refers to the BIM attack where an initial random step r sampled from a uniform distribution in the range $[-\epsilon, \epsilon]$ is taken. Experimentally, this step has been shown to allow to escape possible local minima around x. Moreover, in order to better explore the output space, and find higher values for $\mathcal{L}(x', y, M_\theta)$, the adversary can perform the PGD attack multiple times, with a different starting point each time (thanks to the initial random sampling), and keep only the example inducing the highest $\mathcal{L}(x', y, M_\theta)$ value.

Importantly, the authors show that the PGD is a strong first-order adversary, as experiments reveal that it would be hard to find a better maximum for the optimization problem of Equation 4.8 with first-order methods.

Sparse l_1 PGD. Tramer *et al.* [120] propose an enhanced version of the l_1 version of the PGD attack. For the initial PGD attack in its l_1 version, the perturbation δ^t computed at each step t results in a vector with *only one* non-null component being $\operatorname{sign} g_s^t$ with g_s^t denoting the component of $g^t = \nabla_x \mathcal{L}(x^{t-1}, y, M)$ for indice $s = \operatorname{argmax}_i |g_i^t|$.

Therefore, the authors propose a modified version of it where δ^t is set to $\operatorname{sign} g^t$ if $|g^t| > P_q(|g^t|)$, and to 0 otherwise, where $P_q(|g^t|)$ is the q^{th} percentile of $|g^t|$. The smaller the parameter q, the more δ components are likely to be modified. The perturbation δ is then normalized to the l_1 unit-ball (i.e divided by $\|\alpha\|_1$).

Wasserstein PGD. Wong *et al.* [83] propose a modification of the PGD attack to consider the Wasserstein distance instead of the distance derived from a l_p norm (see Figure 4.11). A threat model based on the Wasserstein distance encompasses modifications such as translation, rotation and scaling better than a threat model based on the l_p norm. At each step of the PGD attack, the projection is performed on a Wasserstein ball and not on a l_p ball.

Auto-PGD. Croce *et al.* [87] propose Auto-PGD (APGD), an enhanced version of the PGD attack, which only requires to specify the number of iterations and the l_∞ bound for the adversarial perturbation. Specifically, it considers an adaptive step size which takes into account the number of iterations the adversary wants to perform. The main idea is to allow the step size to decrease when no improvement in the optimization objective occurs, and when the step size decreases, to restart at the best solution found so far.

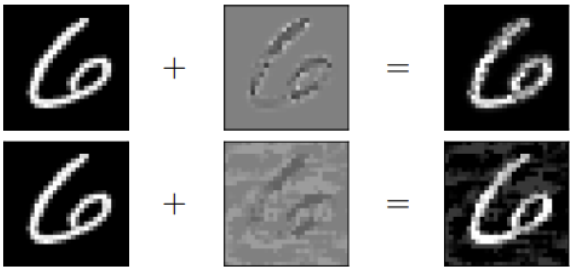

Figure 4.11: Adversarial examples, crafted with Wasserstein PGD (top), and l_∞-PGD (bottom), from [83].

Carlini-Wagner (CW)

Carlini *et al.* [105] consider the joint objective of finding an adversarial example and minimizing the l_2 distortion related to this adversarial example, as described in Equation 4.13.

$$\begin{aligned} &\underset{\epsilon}{\operatorname{argmin}} \ \|\epsilon\|_2 + c\, G(x+\epsilon, y) \\ &\text{s.t.} \quad x + \epsilon \in \mathcal{Q} \end{aligned} \tag{4.13}$$

with:

$$G(x+\epsilon, y) = \max(h_y(x+\epsilon) - \max_{j \neq y} h_j(x+\epsilon), -\kappa) \tag{4.14}$$

where $h_y(x)$ denotes the logit value for the ground-truth indice and κ is a parameter for the desired gap between the logit value for y and the second biggest logit value.

To solve the problem given in Equation 4.13, the change of variable $x + \epsilon = \frac{1}{2}(\tanh(w) + 1)$ is performed to get rid of the box constraint, and the resulting optimization problem is then solved with respect to w with the Adam optimizer. Moreover, this problem is solved for many values of the constant $c \in \mathbb{R}^+$ found with binary search to find the solution with the lowest l_2 distortion. The authors also propose a l_∞ and l_0 version of their attack.

Jacobian-based Saliency Map Method (JSMA)

Papernot *et al.* [121] propose the JSMA method, which is based on the idea of identifying the most sensitive pixels in terms of misclassification, and then modify them.

More precisely, this attack works in an iterative fashion. As long as the maximum number of iterations is not reached, or that the adversarial distortion, in terms of number of modified pixels does not reach a threshold, two steps are performed at each iteration.

Firstly, the adversary begins by measuring the variation of $F_y(x)$ with respect to a variation of each pixel i. To do so is computed the vector of partial derivatives of the confidence score value for the ground-truth label y, for every pixel i, i.e. compute $\nabla_{x_i} F_y(x)\ \forall i \in [\![1, d]\!]$.
Secondly the *saliency map* $S(x, y)$ is computed, which allows to target the pixels which will have the more influence for misclassification.

Decoupling Direction and Norm (DDN)

Rony *et al.* [122] propose the iterative DDN attack. Contrary to previously presented attacks, the goal is not to search for the minimal adversarial distortion as with the CW attack in its l_2 version, neither to fix a maximum allowed adversarial distortion, as with the FGSM or PGD attack. Rather, the idea is to project at each step t the resulting example x^t onto a l_2 ball of some radius ϵ^t, and increase or decrease the radius ϵ^t at next step whether the current x^t was found adversarial or not (i.e. is misclassified or not). More precisely, starting with a radius ϵ^t, a step size α, and an adversarial perturbation of 0 for every pixel, are performed three steps at each iteration t.

DeepFool

Moosavi-Dezfooli *et al.* [123] propose the iterative DeepFool attack method. The idea of this attack is, until no misclassification is induced, to project the current example onto the closest decision boundary, where the projection can be derived thanks to a linear approximation of the decision boundaries. In fact, at each step, the decision boundaries around the current example are

linearized, which allows to derive the equations for the resulting hyperplanes, and then to project the current example onto the closest hyperplane. The resulting adversarial perturbation of this attack is therefore the sum of the perturbations added to the current example at each step to perform the projection. The authors propose a version of the attack where the closest hyperplane is searched relatively to the l_2 distance. However, it can be adapted to any l_p norm.

Sparse Adversarial examples

Croce *et al.* [107] propose a method to craft sparse adversarial examples. Three different ways to modify a single pixel are proposed by the authors, and when modifying one pixel does not suffice to induce misclassification, a non-expensive multi-pixel modification scheme is presented.

An illustration of the adversarial examples crafted are presented in Figure 4.12.

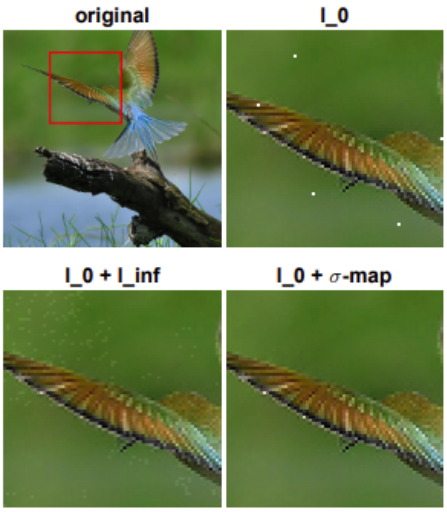

Figure 4.12: Illustration of adversarial examples crafted with the different version of the attack proposed in [107].

AdvGAN

Xiao *et al.* [124] propose AdvGAN, a method which takes advantage of a GAN (Generative Adversarial Network) to craft adversarial examples against a target model M. This scheme is based on a generator model $\mathcal{G}$ and a discriminator model $\mathcal{D}$. Once trained, the generator $\mathcal{G}$ takes as input a clean example x and allows to retrieve an adversarial example as $x' = x + \mathcal{G}(x)$.

Enhancing gradient-based attacks

Some works have investigated ways to improve the efficiency of gradient-based attacks.

Wang *et al.* [125] propose a way to enhance the choice of the initial point for gradient-based attacks. Starting from a point x_0, the idea is to identify a zone around this point and use sound bound propagation methods to over approximate the output variations in this zone. Then, based on this approximation, the best point in terms of desired output variations is found. This point is then used as the initial point for the gradient-based attack.

Tashiro *et al.* [126] propose Output Diversified Initialization (ODI) to improve the initialization step for attacks which use a certain number of restarts, with a different initialization at each restart. These restarts are performed in order to increase diversity among beginning points of the attack, with the aim of better exploring the adversarial subspace of the output space, and then select the best adversarial example. The idea of the ODI scheme is to leverage diversity in the output space rather than in the input space. The motivation comes from the fact that as models tend to be very non-linear, considering diversity in the input space does not necessarily induce diversity in the output space.

4.4.2 Black-box setting (score and decision-based attacks)

We now present attack schemes allowing an adversary to craft adversarial examples against a target model in a black-box setting. A black-box setting may present different variations, relating to the available output (logits, confidence score vector, predicted label, loss value, etc.). However, as the adversary does not have access to both architecture and parameters, he can not formally derive the gradients of any output or loss with respect to the inputs, which constitutes the main characteristic of the black-box setting.
In a first time, we present attack methods intended to directly attack the target model, without using a substitute model. In a second time (Section 4.4.3), we will present in details how an adversary can take advantage of a substitute model and transferability to attack a model in a black-box setting.

Zeroth-Order Optimization (ZOO)

Chen *et al.* [127] propose a score-based attack (i.e. for an adversary supposed only to know the confidence score vector $F(x)$), which uses a gradient estimation method.
The loss to be optimized to craft adversarial examples is the one used for the CW attack (given in Equation 4.13), where the logits function h is replaced by the score function F, as given in Equation 4.15.

$$\begin{aligned} \underset{\epsilon}{\operatorname{argmin}} \ & \|\epsilon\|_2 + c\, G_F(x+\epsilon, y) \\ \text{s.t.} \quad & x + \epsilon \in \mathcal{Q} \end{aligned} \tag{4.15}$$

with:

$$G_F(x+\epsilon, y) = \max(F_y(x+\epsilon) - \max_{j \neq y} F_j(x+\epsilon), -\kappa) \tag{4.16}$$

The optimization procedure is done with the Adam optimizer where the following approximation is used for the derivative of F with respect to x_i (the i^{th} element of x):

$$\nabla_{x_i} F(x) \approx \frac{F(x + he_i) - F(x - he_i)}{2h} \tag{4.17}$$

where h is set to 0.0001 and e_i is a vector with the i^{th} component set to 1, and all other components valuing 0. As the computation of the derivative of F with respect to x_i has to be done d times for each iteration of the optimization algorithm, and requires two evaluations of the score function F at each time, it can quickly become very expensive in terms of computation costs to reach convergence. The authors thus propose to solve the optimization problem in a stochastic way, only updating one pixel at each minimization step (instead of all of them) until convergence. To accelerate the minimization process, the authors also propose to focus on pixels near the main objects on the pictures or to use dimension reduction methods to fasten the optimization.

Simultaneous Perturbation Stochastic Approximation (SPSA)

Uesato *et al.* [79] also propose a score-based attack scheme exploiting a gradient estimation method. The adversary is supposed to have access to the logits $h(x)$, and the adversarial example is searched by solving the optimization problem presented in Equation 4.18.

$$\begin{aligned} \underset{x'}{\operatorname{argmin}} \ & h_y(x') - \max_{j \neq y} h_j(x') \\ \text{s.t.} \ & \|x' - x\|_\infty \leq \epsilon \end{aligned} \tag{4.18}$$

This optimization problem is solved with the SPSA (Simultaneous Perturbation Stochastic Approximation) method [128], which works in an iterative fashion. At each step t with current example x^t, n vectors $v_1, \ldots, v_n$ of dimension d are considered. For each vector v_j ($j \in [\![1, n]\!]$), its i^{th} component $v_{j,i}$ is drawn from a Rademacher distribution (a discrete probability distribution: $v_{i,j}$ has 50% chance of being $+1$, and 50% chance of being -1). Then is computed $g_j = \frac{(f(x^t + \delta v_j) - f(x^t - \delta v_j))}{2\delta}$, where δ is a perturbation size. The example is then updated as $x^{t+1} = \text{proj}_{B_\infty(x,\epsilon)}(x^t - \alpha \sum_{j=1}^n g_j)$, where α is a step size.

Simple Black-box Attack (SimBA)

Guo *et al.* [129] propose a gradient estimation free score-based attack to craft adversarial examples, where the adversary only has access to the confidence score vector. The idea is to pick orthogonal directions in a set without replacement, updating the adversarial perturbation by one of these directions if it lowers the confidence score for the predicted label. The orthogonality of potential directions allows not to search in directions cancelling each other or making the adversarial perturbation grow too much.

Boundary Attack

Brendel *et al.* [130] propose a decision-based attack method (the adversary is only allowed to retrieve the output label of the target model M). The idea is to consider at start an example x^0 classified in a different class than x (i.e. $M(x^0) \neq M(x)$), and to make steps towards x while staying adversarial, adapting the step procedure at each iteration.

In order to approach x while staying adversarial, the concept of the attack is to perform a random walk along the decision boundary. The intuition of the Boundary attack is presented in Figure 4.13

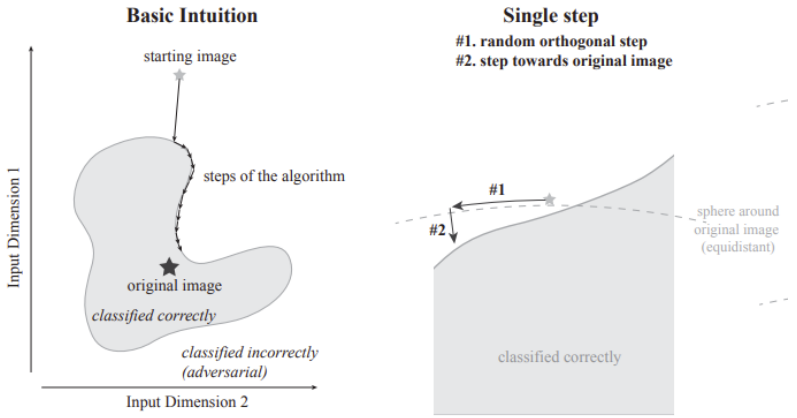

Figure 4.13: Illustration of the intuition of the Boundary Attack, from [130].

Hop Skip Jump Attack (HSJA)

Chen *et al.* [108] also propose a decision-based attack method, which relies on a gradient estimation which is only valid on the decision boundary. Considering a clean example x^0, classified in a different class than x, this attack scheme works in an iterative manner to get closer to x while staying adversarial. Each step involves three main components. Firstly, binary-search is performed to reach the decision boundary. Secondly, the gradient estimation at the decision boundary is performed. Thirdly, it is searched for a step size, so that taking a step in the estimated gradient direction keeps the current example adversarial.

The intuition of the Hop Skip Jump Attack is presented in Figure 4.14

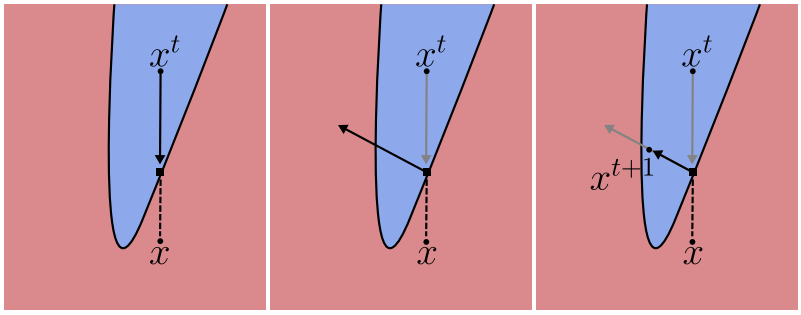

Figure 4.14: Illustration of the intuition of the Hop Skip Jump Attack, from [108]. (Left) Line Search to find the boundary. (Middle) Gradient estimation. (Right) Step size search.

SurFree attack

Maho *et al.* [131] also propose a decision-based attack (the adversary only knows the output label), whose main goal is to get the smallest number of queries made to the target model given some adversarial distortion. Considering a starting point x^0 classified in a different class than x, the attack works in an iterative manner, to get closer to x at each step while staying adversarial. The central idea of this attack is to replace gradients estimates made at the boundary at each step (as for the HSJA attack [108]), which require queries to the model, by the study of multiple directions. For each direction, the maximum distortion decrease is obtained with a geometrical rationale. The motivation behind this choice is that investigating multiple directions and obtaining the biggest distortion decrease for each direction is hoped to produce a faster distortion decrease in terms of number of queries than performing a gradient estimate.

Optimization-based Approach

Cheng *et al.* [132] propose a decision-based (the adversary only has access to the output label) attack that formulates the adversarial example search with an optimization problem which can later be solved with a gradient-free optimization method. This optimization problem is given in Equation 4.19.

$$\begin{aligned} &\underset{\theta}{\text{argmin}}\ g(\theta) \\ &\text{s.t. } g(\theta) = \underset{\lambda}{\text{argmin}} \left(M(x + \lambda \frac{\theta}{\|\theta\|_2}) \neq y \right) \end{aligned} \tag{4.19}$$

Denoting θ^* the solution to the problem given in Equation 4.19, θ^* is the direction towards the closest adversarial example, and $g(\theta^*)$ is thus the shortest distance towards an adversarial example. We then have $x' = x + g(\theta^*)\frac{\theta}{\|\theta\|_2}$.

It is shown experimentally that $g(\theta)$ is continuous even if M is not continuous, an can then be solved by a zeroth-order optimization method.

Efficient Combinatorial Optimization (ECO)

Moon *et al.* [133] also propose a method to craft adversarial examples for an adversary in a black-box setting having access to the values of the loss function. The search for an adversarial example is considered as a discrete optimization problem, which is then solved in a greedy way without any gradient estimation. The motivation comes from the experimental evidence than the adversarial perturbations computed with the l_∞ version of the PGD attack are predominantly found on the vertices of the l_∞ ball. This means that for a l_∞ budget of ϵ, adversarial perturbations mostly value $-\epsilon$ or ϵ. The authors therefore propose to replace the optimization objective used for the FGSM, BIM or PGD method, given in Equation 4.20, by the one given in Equation 4.21.

$$\begin{aligned} &\max_{x'} \mathcal{L}(x, y, M) \\ &\text{s.t. } \|x' - x\|_\infty \leq \epsilon \end{aligned} \tag{4.20}$$

$$\begin{aligned} &\max_{x'} \mathcal{L}(x, y, M) \\ &\text{s.t. } x' - x \in \{-\epsilon, \epsilon\}^d \end{aligned} \tag{4.21}$$

This problem can be reformulated as the one given in Equation 4.22.

$$\begin{aligned} &\max_{\mathcal{S} \subset \mathcal{V}} F(\mathcal{S}) \\ &\text{s.t. } F(\mathcal{S}) = \mathcal{L}\left(x + \epsilon \sum_{i \in \mathcal{S}} e_i - \epsilon \sum_{i \in \mathcal{V} \setminus \mathcal{S}} e_i, y, M \right) \end{aligned} \tag{4.22}$$

where $\mathcal{V}$ denotes the set of all pixel locations, $\mathcal{S}$ denotes the set of pixels modified by $+\epsilon$, and $\mathcal{V}\setminus\mathcal{S}$ denotes the set of pixels modified by $-\epsilon$. e_i denotes the i^{th} standard basis vector. The main challenge in solving the problem given in Equation 4.22 is the fact that finding the set $\mathcal{S}$ is NP-hard. However, the authors show that one can obtain an approximate solution with greedy search algorithms.

The authors choose the Lazy-Greedy algorithm [134], a local-search optimization procedure, which consists in alternatively adding pixels of $\mathcal{S}$ and removing pixels from $\mathcal{S}$ based on a defined rule, until convergence. Moreover, to accelerate the solving, it is chosen to take advantage of the fact that images exhibit locally regular structures. Therefore, the image is divided into blocks at the beginning of the optimization, and the Lazy-Greedy algorithm is applied on each block. The image is then divided in more blocks and the procedure is repeated. It is shown experimentally that the attack converges often before the image being divided at the pixel level.

Limited Queries and Information

Ilyas *et al.* [135] propose attack schemes, in three different settings where the adversary is either limited to a certain amount of queries, has only access to the top k confidence sore values or label outputs.

- Query Limited setting. The adversary is limited in terms of number of queries to the target model, and has only access to the confidence score values.
- Partial Information setting. The adversary is limited to the top k confidence score values.
- Label Only setting. The adversary is limited to the top k label outputs.

Rotations and Translations

Engstrom *et al.* [60] propose attack procedures to craft adversarial examples exploiting rotations and translations of clean examples, where the adversary only needs to know the confidence score vector. Considering the value $x_{i,j}$ of a clean example for pixel at position (i, j), the pixel value of the adversarial example at the same position, $x'_{i,j}$ is built by performing a rotation of angle θ and then a translation of $delta_u$ (resp. δ_v) in the direction for the x-axis (resp. y-axis).

An illustration of the attack is presented in Figure 4.15.

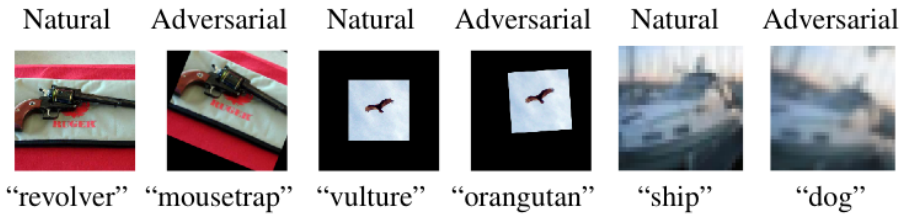

Figure 4.15: Illustration of adversarial rotations and translations, from [60].

Colorfool: Adversarial Colorization

Shamsabadi *et al.* [61] propose a decision-based attack, where the adversary modifies colors of a clean example to turn it into an adversarial one. The specificity of this attack lies in the fact that it separates sensitive regions, where large color changes seem unnatural (e.g. face color, sky, etc.), with non-sensitive regions, which can undergo larger color changes (e.g. walls, ground, etc.).

An illustration of the attack is presented in Figure 4.16. The color of the background is largely changed, as it corresponds to a non-sensitive region, while the color of the skin is only subtly modified.

4.4.3 Black-box setting (transferability)

When the adversary is in a black-box setting, besides decision-based and score-based attacks, he can leverage transferability, to attack the target model. Given that the adversary can train his own substitute (source) model, using transferability is particularly relevant for the adversary when he is limited in terms of number of queries made to the target model. Indeed, score-based and decision-based attacks can be very expensive in terms of computation and queries made to the target model.

The transferability phenomenon constitutes an important part of our work. In the following, we introduce it, and present attack schemes intended at crafting transferable adversarial examples.

Figure 4.16: Illustration of the colorfool attack, with the clean image (left), and its adversarial counterpart (right), from [61]. The background color (non-sensitive region) is largely modified, while the skin color (very sensitive region) is very subtly changed.

The transferability phenomenon

The phenomenon of transferability is firstly introduced by Papernot *et al.* [64]. It is shown than an adversary training a substitute (source) model M_S for the same classification task can take advantage of this model to attack the target model M. In fact, an adversarial example x' crafted on M_S ($M_S(x') \neq M_S(x)$) can also fools M ($M(x') \neq M(x)$). The transferability is therefore a powerful tool for an adversary in a black-box setting. Indeed, the adversary is in a white-box setting relatively to his substitute model M_S, and does not suffer from the limitations inherent to the black-box setting of the target model M. In their work, Papernot *et al.* even propose a method to generate a synthetic data set with a limited number of queries to the target model. As an example, it is shown that with transferability, an adversary can successfully attack a model hosted on Amazon[1] and Google Vertex AI [2].

Importantly, the transferability phenomenon still holds between different machine learning algorithms [12]. As an example, an adversarial example crafted against a decision tree model can successfully fool a support vector machine model, or a neural network model.

In the scope of the threat of adversarial examples, transferability is reasonably the most concerning issue regarding the security of machine learning models. Indeed, on one hand, in a real-life scenario, the target model is almost never in a white-box setting. On another hand, decision-based and score-based attacks require thousands of queries only to craft one adversarial example. This is not tractable in practice, with most machine learning systems providing customer with a paid service. Moreover, these malicious queries can easily be flagged as suspicious [136].

In the following, we present different attack schemes that allow an adversary to increase the transferability of adversarial examples.

Momentum Iterative Method (MIM)

Dong *et al.* [137] propose a method to increase the transferability of adversarial examples crafted with gradient-based attacks, where the main idea is to add a momentum term in the optimization procedure. In fact, remembering at optimization step the precedent update allows to have more consistent (i.e. in the same direction) updates throughout the attack.

The authors illustrate this scheme with the optimization problem considered for the FGSM, BIM and PGD attacks, given in Equation 4.23.

$$\begin{aligned} &\max_{x'} \mathcal{L}(x, y, M) \\ &\text{s.t. } \|x' - x\|_\infty \leq \epsilon \end{aligned} \tag{4.23}$$

For the FGSM, BIM and PGD methods, the optimization problem is solved at each step t by performing $x^t = x^{t-1} + \alpha \operatorname{sign} g^t$, with $g^t = \nabla_x \mathcal{L}(x^{t-1}, y, M)$. With the MIM method, a momentum

[1] https://aws.amazon.com/fr/machine-learning/
[2] https://cloud.google.com/vertex-ai

term to each gradient computation is added, as

$$g^t = \mu g^{t-1} + \frac{\nabla_x \mathcal{L}(x^{t-1}, y, M)}{\|\nabla_x \mathcal{L}(x^{t-1}, y, M)\|_1}$$

where μ is a decay factor. The higher μ, the more the previous gradients are remembered throughout the iterations. The current example is then obtained with $x^t = x^{t-1} + \alpha \operatorname{sign} g^t$. This method is applicable to any gradient-based attack not already including a momentum term in the optimization procedure.

Translation-Invariant Attack and Input Diversity

Dong *et al.* [138] propose to increase the transferability of adversarial examples crafted with a gradient-based attack by accounting for many translated versions of the input during the optimization procedure. The underlying idea is the hypothesis that the source and target models do not rely on the same region of the image to predict the label, and therefore that crafted adversarial examples tend to overfit to the source model's gradients and discriminative regions. Considering many translated version of an input for gradient computation could then allow to alleviate this issue.

More precisely, the authors consider the same optimization problem as for the MIM attack (Equation 4.23), and extend it with the one given in Equation 4.24.

$$\begin{aligned} &\operatorname*{argmax}_{x'} \sum_{i,j} w_{i,j} L(\theta, T_{i,j}(x), y) \\ &\text{s.t. } \|x' - x\|_\infty \leq \epsilon \end{aligned} \tag{4.24}$$

where $i, j \in [\![-k, k]\!]$ with k a maximum number of pixels to shift, $T_{i,j}(x)$ the shifted version of x by i (resp. j) pixels along the x (resp. y) axis, and $w_{i,j}$ coefficients. Solving this optimization problem requires performing $(2k+1)^2$ gradient calculations at each step. To alleviate this burden, the authors propose to use the approximation that $\nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=T_{i,j}(\hat{x})} \simeq \nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=\hat{x}}$, which allows to have:

$$\nabla_x \sum_{i,j} w_{i,j} L(\theta, T_{i,j}(x), y)\Big|_{x=\hat{x}} = W \circledast \nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=\hat{x}}$$

where $\circledast$ denotes the convolution operation and W is a kernel matrix of size $(2k+1) \times (2k+1)$. Therefore, the optimization problem given in Equation 4.24 can be solved efficiently, only performing one gradient computation and convolving it with the matrix W.

This method can be efficiently combined with other methods, such as MIM for example. The combination of the MIM attack with the translation-invariant method is denoted by MIM-TI.
Xie *et al.* [139] also propose a method to decrease the overfitting of adversarial examples to the source network they are crafted on. To do so, the authors propose to use data augmentation with spatially transformed inputs during the optimization procedure. More precisely, at each gradient computation, the input is transformed with probability p with image resizing or random padding. The combination of the MIM attack with the input diversity method is denoted by DIM. Moreover, the combination of the DIM attack with the translation-invariant method is denoted by DIM-TI.

Intermediate Level Attack (ILA)

Huang *et al.* [140] propose the Intermediate Level Attack (ILA) to increase the transferability of an already crafted adversarial example.

Given a hidden layer l of the source model with its output function F_l, and an existing adversarial example x', the attack method aims at maximizing the perturbation at layer l, while keeping the original adversarial direction. Therefore, the attack aims at maximizing $\Delta(y_l^{''}) = F_l(x'') - F_l(x)$ with respect to x'', while not straying too far from the original adversarial direction given by $\Delta(y_l^{'}) = F_l(x') - F_l(x)$. In order to do that, the authors propose to maximize $-\Delta(y_l^{''}) \cdot \Delta(y_l^{'})$ with respect to x''.

The authors choose the layer l experimentally. Notably, they show that as the index of l increases, the linearity of the target model decision boundary increases, which increases transferability. However, above some index, the linearity of the source model becomes more linear than the one of the target model, decreasing transferability.

Attention guided Transfer Attack (ATA)

Wu *et al.* [141] propose a ATA (Attention guided Transfer Attack), a method to enhance the transferability of adversarial examples. This method is based on the observation that some features (internal representations in a neural network model) are commonly used by different models with different architectures, and that therefore focusing more on these critical features would lead to a better transferability.

By denoting by A_k^c the c^{th} feature map in layer k, the idea is to craft an adversarial example (x', y_t) from a clean input-label pair (x, y) such that the features the most exploited by the source model M_S to classify x are the same for x'. With the hypothesis that the target model M exploits predominantly the same features than M_S to classify x, it will make M even more vulnerable to x'.

Direction Aggregated Attack (DA)

Huang *et al.* [142] propose DA (Direction Aggregated), an improvement to existing black-box transfer attacks to craft more transferable adversarial examples. The idea is to replace each direction computation (i.e. each gradient computation) with an aggregation of directions for examples perturbed with gaussian noise. By doing so, it is hoped to relieve overfitting to the source network, allowing to find more meaningful directions to fool the target model. More precisely, considering a source model M_S in a white-box setting, each gradient computation $\nabla_x \mathcal{L}(x, y, M_S)$ is replaced by $\sum_{i=1}^{T} \nabla_x \mathcal{L}(x + \epsilon_i, y, M_S)$, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

Transferable Adversarial Perturbations (TAP)

Zhou *et al.* [143] use the feature representation in intermediate layers, and the reducing of the variations of the adversarial perturbation. More precisely, the TAP attack considers three goals to craft an adversarial example x' from a clean example x on a source model M_S:

- Inducing misclassification for M_S.
- Maximizing the distance between the feature space representation of x and the feature space representation of x' for all layers of the source model M_S. The hypothesis behind this maximization, which is verified experimentally, is that it allows to craft adversarial perturbations that impact the inference process of M_S more strongly and deeply than simply considering misclassification, and improve the black-box transfer rate.
- Removing high frequency components of the adversarial perturbation. Indeed, the authors hypothesize that high-frequency components of the adversarial perturbation will be smoothed by convolution kernels of the target model M.

Skip Gradient Method (SGM)

Wu *et al.* [144] propose SGM (Skip Gradient Method), a method to enhance the transferability of adversarial examples crafted on a source model with residual blocks (a combination of a residual module and a skip connection). The main idea is that backpropagating using more skip connections than residual modules when crafting the adversarial example on the source model allows to rely more on low level information, which would be more transferable across different model architectures. Indeed, skip connections pass directly the information from one layer to another passing through the residual model. The SGM method is therefore designed to downweight the importance of the gradient flowing through the residual module.

Activation Attack (AA)

Inkawhich *et al.* [145] propose Activation Attack (AA), a method to craft transferable targeted adversarial examples. The main idea is to perturb the feature space representation of the source model, relying on the hypothesis that, as the source and distant models have been trained for the same task, they have learned similar features.

More precisely, considering a clean input-label pair (x, y), and a target label y_t, a target input-label pair (x_0, y_t) is chosen, and the idea is to craft an adversarial example x' from x such that the feature space representation of x' at some layer of the source model is close to the feature space representation of x_0 at the same layer. The crafted adversarial example x' is then transferred to the target model, to have it misclassifying it in class y_t. The adversarial example x', whose feature

representation is close to the one of an example of class y_t for the source model, will also have a feature representation close to the one of an example of class y_t on the target model, which will thus misclassify it in class y_t.

Using robust models

Springer *et al.* [146] propose a method to enhance the transferability of targeted adversarial examples. The idea is to use slightly robust models (models trained to be robust to adversarial perturbations with small perturbation budget) as source models. The underlying hypothesis is that slightly robust models learn more universal features [114, 117, 119], i.e. features, which are also used by models trained for the same classification task, as the target model.

This hypothesis is verified with experiments on ImageNet, where models trained to be robust to small l_2 perturbations are shown to be better source models than standard models. For a target model trained with Adversarial Training with perturbation budget ϵ, even targeted adversarial examples with perturbation greater than ϵ hardly transfer with a standard model as the source model. This is not the case when using slightly robust models as source models, which allows significant transferability.

4.5 Defense methods

We propose an overview of the schemes intended at protecting a model against adversarial examples. This section is not intended to give an exhaustive presentation of all existing defense schemes, we simply detail some of them, which are relevant to give intuition about the different underlying ideas.

4.5.1 Overview

The defense schemes against adversarial examples can be divided into two types: reactive and proactive schemes.

Proactive methods aim at making the model more robust in itself against adversarial examples: a model defended with a proactive defense scheme will suffer less from adversarial examples than a standard model. Proactive methods can be either empirical [75, 147] or provable [89, 91]. Provable schemes provide guarantees about the robustness of a model. They certify that no adversarial example can be found at a certain distance of a clean example, if some condition is satisfied. Empirical methods do not give such guarantees but make it harder (or impossible, even if not proven) for an adversary to find an adversarial example at some distance of a clean example.

On the contrary, reactive methods do not aim at making the model more robust against adversarial examples, but rather act as prevention. Reactive methods may be classified in two categories: detection of adversarial examples [148, 149] and purification schemes [111, 150]. Detection aims at recognizing adversarial examples, in order to treat them differently (e.g. discarding them, banning a user, etc.). Purification aims at denoising an adversarial example, so that the model outputs the correct label on the denoised example.

We begin by presenting few examples of reactive methods. Then, we show the concept of provable defense methods, as well as some examples of them. Eventually, we detail more in-depth various empirical schemes, as this is the most common type of defense proposed.

4.5.2 Reactive methods

On the multiple detection schemes that have been proposed in the literature, the vast majority of them have been show to be bypassable [76–80]. This issue arises from two main reasons, which are increasingly addressed as good practices for defense evaluation are continuously fostered [81, 82]. The first issue concerns the setting considered for the evaluation of these defenses. This setting, often refers as a gray-box setting, hypothesizes that the adversary is not aware of the detection/purification mechanism. Under this hypothesis, the defense is shown to be efficient. However, simply relieving this hypothesis, as it is a good practice with adaptive adversaries, makes the defense useless. The second issue is specific to detection schemes, which tend to overfit to the type of adversarial examples they are designed for.

In consequence of these issues, reactive defense schemes are increasingly less proposed in the defense literature. However, this type of defense is still exploited, and detection schemes have a clear advantage over provable and empirical proactive defense schemes. Indeed, in the vast majority of cases, this type of method is easily implemented. More precisely, on one hand it does not need to

retrain the target model, as for empirical proactive schemes and some provable proactive schemes. On another hand, it scales better to any type of architecture than some provable proactive schemes.

We now proceed to illustrate one denoising-based defenses, and one detection-based defense. Xie *et al.* [151] propose a denoising scheme, based on the observation that features maps for adversarial examples present some noise. Under the hypothesis that this noise is in fact the adversarial perturbation at the pixel level which is increased as it goes through layers of the target network, the authors propose to add denoising blocks in the target network architecture after feature maps. For the denoising operation is tested non-local means, bilateral filter, mean filter and median filter. The network is trained with l_∞-PGD adversarial training to train the denoising blocks to denoise the adversarial perturbation in feature maps. Li *et al.* [136] develop a method to detect the process of crafting black-box adversarial examples. In fact, the detection schemes exploits the fact that to craft an adversarial examples against a target model in the black-box setting, an adversary needs to submit similar consecutive queries. Once a sequence of malicious query is detected, all subsequent queries are rejected.

4.5.3 Provable proactive methods

As provable methods are not studied in this thesis, we focus on giving a high-level presentation of them. The objective of provable methods is to certify that a model is robust (or not) against adversarial examples. More precisely, for some input, the robustness certificate is provided in the form of a *certified radius*. This certified radius is a real value indicating that no adversarial example can be found in a ball of this radius centered on the input. The metric used for this ball is usually the distance from the l_p norm. The goal of a provable method is therefore to provide the biggest certified radius for every input.

On one hand, some provable schemes use various formal methods to formally verify the robustness of a model. Among these schemes, *complete* ones verify exactly this robustness. In order to do so, they leverage the Satisfiability Modulo Theory (SMT) [152], or mixed-integer programming [153]. These schemes, as computationally expensive, are only valid for small models, with few layers and specific activation functions (e.g. ReLU [152]), and provide small certified radius.

On another hand, other provable schemes certify the robustness of models by leveraging not exact methods. These methods provide looser certificates than complete ones, but can scale to larger models. Kolter *et al.* [91] propose a provable method based on a linear relaxation of the ReLU activation function. Gowal *et al.* [154] compute an upper bound on the worst case loss for an adversarial perturbation by propagating a worst-case bound of the activation function values throughout the layers. Hein *et al.* [155] derive a lower bound for the l_p norm of the adversarial perturbation required to craft an adversarial example x' from a clean observation x, and derive from it a loss function to minimize during training to increase the robustness of a model. Cohen *et al.* [89] introduce Randomized Smoothing to formally verify the robustness of a model. Randomized Smoothing is based on the idea of turning any model into a *smoothed* version of it, on which we can derive l_2 robustness guarantees. The main advantage of Randomized Smoothing is that contrary to other provable methods, it scales better to large architectures, as it does not depend on the architecture of the initial model.

4.5.4 Empirical proactive methods

Empirical proactive defense methods encompass all the defense schemes which do not provide a formal guarantee for the robustness of a model for some input. With empirical methods, no certified radius around a clean example is given, and thus the robustness of the model for this example can not be proven. Consequently, it is mandatory to evaluate carefully its true efficiency. Many works have emphasized this need, in order to foster a faster advance in the state of the art. Notably, some works have analyzed defense schemes proposed in scientific conferences to demonstrate some recurring flaws that lead to an over-evaluation of their efficiency [78, 79, 82].

Among these flaws, an important one is *gradient masking*, pointed out by Athalye *et al.* [78], which may lead to a false sense of security. Indeed, gradient masking is a phenomenon where gradients of the model (loss, score function) with respect to the inputs are obfuscated. Attack methods which rely on these gradients (gradient-based attacks) could therefore not find an adversarial example at some distance of a clean example, even if this adversarial example exists. Attacking a defended model with a gradient-based attack can therefore lead to the false assumption that the model is well-protected against adversarial example, whereas it is not robust at all, and only prevents some type of attacks from finding adversarial examples. Gradients can be obfuscated in three ways:

- Shattered gradients: the defense scheme uses components that are either non-differentiable, introduce numeric instability or make the gradients non-existent or incorrect. In the case of differentiable gradients, following these gradients does not indicate the right direction to find adversarial examples.
- Stochastic gradients: the defense scheme uses randomization, which makes the gradient being not the same at each computation. Using only one gradient computation can therefore be not sufficient to estimate the right direction to find adversarial examples.
- Exploding and vanishing gradients.

Developing a defense scheme therefore requires paying attention to gradient masking, which has been shown to be present in many empirical defenses [78]. In order to do so, one may use various methods. Firstly, against shattered, exploding or vanishing gradients, one may simply not attack directly the target with gradient-based attacks, but rather use decision-based or score-based attacks, or leverage the transferability phenomenon. In this case, this can allow to circumvent existing gradient instability of the target model, as the gradient of the target model is not directly used. Secondly, for shattered gradient, when the gradient is non-existent or incorrect, Athalye *et al.* [78] propose the BPDA (Backward Pass Differentiable Approximation) that consists in approximating a non-differentiable operation with a differentiable approximation of it. Eventually, for stochastic gradients, they propose the EOT (Expectation over Transformation) method: instead of relying on one computation of the gradient, one may average many values of it, to reduce the impact of the randomization component included in the defense scheme.

Eventually, Carlini *et al.* [81] have laid down a non-exhaustive, yet quite complete, list to properly evaluate the efficiency of an empirical defense scheme.

In the following, we detail some proactive empirical defenses, to give intuition about various ways of defending a neural network against adversarial examples.

Gradient regularization

Ross *et al.* [156] intuitively propose a method to constraint the variations of the usual loss function $\mathcal{L}$ (e.g. $\mathcal{L}_{CE}$) with respect to varations of the input.

More precisely, the authors propose to consider the loss

$$\mathcal{L}_{GREG}(x, y, M) = \mathcal{L}_{CE}(x, y, M) + \lambda \left\| \nabla_x \mathcal{L}_{CE}(x, y, M) \right\|_2^2 \tag{4.25}$$

Experiments show that a model trained with the loss $\mathcal{L}_{GREG}$ is more robust to adversarial examples than a standard model. Moreover, adversarial examples crafted on a model trained with $\mathcal{L}_{GREG}$ are more transferable than the ones crafted on a standard model.

Lipschitz constant

Cisse *et al.* [157] propose to minimize the Lipschitz constant $Lip(M)$ of the target model M, in order to minimize the phenomeonon of the adversarial perturbation being amplified as it goes through the model layers.

More precisely, the goal of this method is to have $Lip(M) \leq 1$, so that the model does not amplify any perturbation of the input. An overall Lipschitz constant being the product of the Lipschitz constant at each layer, it resumes to ensures that $Lip^l(M) \leq 1$ for each layer l.

Adversarial Training

Madry *et al.* [75] propose Adversarial Training, an efficient empirical defense scheme which has lead to many variants of it. Defense schemes consisting in improvements of Adversarial Training are currently the ones providing the best empirical results on various data sets (e.g. [158]).

In the standard training procedure, the training phase of a model M_θ considers the following optimization problem:

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(x, y, M_\theta) \right] \tag{4.26}$$

Initially presented for the l_∞ norm, the idea of Adversarial Training is to perform training to make the model robust to adversarial examples in a l_∞ ball of radius ϵ around each example. To this end, the training phase is performed considering the worst-case input x' in a l_p ball of radius ϵ

around each clean example x, instead of x. The optimization problem considered with Adversarial Training is thus the following:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in B_{\infty}(x,\epsilon)} \mathcal{L}(x', y, M_{\theta}) \right] \tag{4.27}$$

In order to compute the inner maximization problem, the authors propose to use the l_{∞}-PGD attack. Practically, during training, for each clean example x, an adversarial example x' is crafted with the l_{∞}-PGD attack with adversary budget ϵ, and the model is trained to classify x' in the ground-truth class y. Eventually, Adversarial Training can be performed with any l_p norm. For the rest of this thesis, we will refer as an adversarially trained model for a model trained with Adversarial Training.

An illustration of the principle of Adversarial Training is proposed in Figure 4.17. On the left is represented the decision boundary (blue and green colors) for standard training. Around each example, the black square represents the l_{∞} ball of radius ϵ, and thus the range for possible adversarial examples for the l_{∞} threat model considered. As a black square can intersect the decision boundary, the model is not robust to this type of adversarial example, . On the right is represented the idealized decision boundary resulting from Adversarial Training. In this case, no black square intersects the decision boundary, which means that the model is robust to adversarial examples that can be crafted in a l_{∞} ball of radius ϵ around each example.

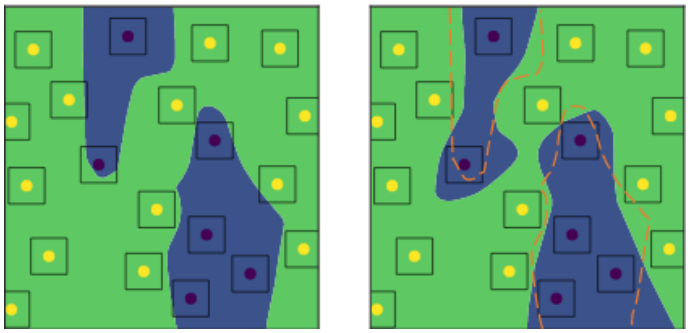

Figure 4.17: Illustration of Adversarial Training, from [147]. On the left is presented the decision boundary resulting from standard training. On the right is presented the decision boundary resulting from Adversarial Training. The black square around each example represents the l_{∞} ball of radius ϵ. The orange dotted line represents the decision boundary of standard training.

Accelerating Adversarial Training. The training phase with Adversarial Training is much longer than with standard training, as for each example, if the l_{∞}-PGD is run for T iterations, T gradients computations are required. Consequently, some works have investigated ways to speed up Adversarial Training [159, 160].

Shafahi *et al.* [159] propose a first way intended at performing Adversarial Training with much smaller training time. The idea of Free Adversarial Training is to consider repeating m times the same procedure for each batch, for $\frac{N}{m}$ epochs (where Adversarial Training is performed for N epochs). For each batch, for each example encountered, it resumes to repeat m times firstly the update of model parameters, then to compute only one step of the l_{∞}-PGD attack (equivalent to the FGSM attack).

Wong *et al.* [160] show experimentally that performing adversarial training using the FGSM attack with non-zero initial perturbation (adding a perturbation to the clean example before performing the gradient step) to craft adversarial examples the model is trained with, can be as efficient as the classical adversarial training where the PGD attack is used.

Robust Overfitting in Adversarial Training. It is known that overfitting in classical training by augmenting the model complexity and training for long periods can help for generalization. Rice *et al.* [161] show experimentally that when considering adversarial training instead of classical training, this is not observed anymore. Indeed, when performing adversarial training, overfitting may cause the robust test set error to increase while the robust train set error continues to decrease. This phenomenon is called robust overfitting. An illustration of the robust overfitting phenomenon is presented in Figure 4.18.

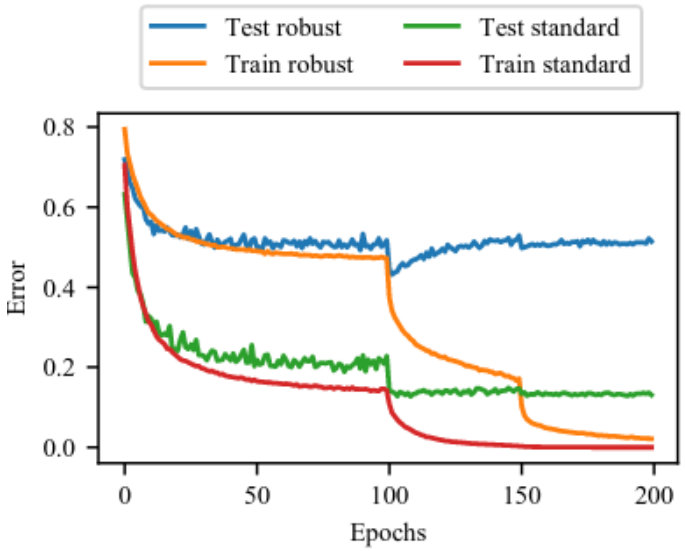

Figure 4.18: Illustration of the robust overfitting phenomenon, from [161]. While train and test standard errors decrease, train robust error decreases but test robust error increases.

Improvements of Adversarial Training.

Maini *et al.* [162] propose an Adversarial Training method which allows to train models which are robust to l_∞, l_2 and l_1 adversarial examples. It consists in performing a single step of the PGD attack for each $p \in \{1, 2, \infty\}$, and keeping the adversarial perturbation resulting in the biggest loss value as the initial adversarial perturbation for the next step. Croce *et al.* [163] also propose a method to perform Adversarial Training for all l_p threat models.

Wang *et al.* [164] propose MART (Misclassification Aware adveRsarial Training), a defense scheme based on the fact of treating differently correctly classified and misclassified examples during Adversarial Training.

Zhang *et al.* [165] propose GAIRAT (Geometry-Aware Instance-Reweighted Adversarial Training), an enhanced version of Adversarial Training, where the main idea is that adversarial examples crafted from examples close to the decision boundary should be treated with more importance than those crafted from examples further from the decision boundary, as they are more responsible for the final shape of a robust decision boundary.

Naseer *et al.* [166] propose SAT (Stylized Adversarial Training), where an adversarial example is crafted leveraging class-boundary information (as with Adversarial Training) but also style and content information, to consider an even stronger adversarial examples. The main idea, considering a clean example, is 1) to craft an adversarial perturbation using feature-level information such as style and content of another example, called target example, from another class than the clean example, and then 2) design another positive example from the same class as the clean example with this perturbation, to eventually minimize a contrastive loss which encompasses maximizing a distance between the clean and target example, and minimizing a distance between the clean and positive example.

Goldblum *et al.* [167] propose ARD (Adversarially Robust Distillation), a method based on distillation to transfer robustness from an adversarially trained teacher model T to a student model S.

Adversarial Feature Stacking (AFS)

Liu *et al.* [168] propose AFS (Adversarial Feature Stacking), a defense method against adversarial examples designed to reduce the trade-off between accuracy and robustness in Adversarial Training.

The main idea is to consider many models trained with Adversarial Training with different adversarial budgets ($\|\epsilon\|$ for the l_∞ threat model), to take advantage of features with various levels of usefulness and robustness. More precisely, it is known that considering $\epsilon_1 > \epsilon_2$, models trained with AT with the ϵ_1 value will learn features which are more robust but less useful than models trained with AT with the ϵ_2 value. This results in the model trained with ϵ_1 showing a greater robust accuracy but a smaller clean accuracy than the model trained with ϵ_2.

Robust Local Features for Adversarial Training (RFLAT)

Song *et al.* [169] propose RLFAT, an enhancement to Adversarial Training which allows the trained model to rely more on local features. This method is based on three observations:

- Adversarially trained models rely more on global features than classically trained models
- Adversarially trained models generalize with difficulty for clean and adversarial examples (i.e there is a smaller test accuracy than for standard models, and there is a consequent gap between adversarial robustness on training and test set)
- Standard models rely more on local features than adversarially trained models

The authors hypothesize than performing Adversarial Training while accounting more for local features would allow better test set accuracy and test set robustness.

To force Adversarial Training to rely more on local features, the authors propose the Random Block Shuffle (RBS) transform which consists of splitting an image into many blocks and shuffling these blocks. At each step of adversarial training, the adversarial example x' crafted from x is passed through the RBS transform before the weight update. As the RBS transform breaks the global structure of the image, it forces training to focus on local features.

Confidence Calibrated Adversarial Training (CCAT)

Stutz *et al.* [170] propose CCAT, a defense method against adversarial examples crafted in the white-box setting, which includes a rejection mechanism. The main idea is to perform Adversarial Training with the l_∞ threat model, while training the model to output a combination of the ground-truth label and the uniform distribution, which becomes more uniform as the distance between the clean example and its adversarial counterpart increases. This procedure aims at making the model output a uniform distribution for adversarial examples crafted considering other threat models, or with bigger l_∞ perturbations, making all type of adversarial examples detectable afterwards.

Jacobian Adversarially Regularized Networks (JARN)

Chan *et al.* [171] show that forcing a model to have salient jacobian matrices for the loss induces better robustness against adversarial examples. A jacobian matrice being salient means that it is interpretable by a human: it shows features relevant to the human. It has already been shown that robust models (trained with adversarial training or randomized smoothing for example) exhibit more salient jacobian matrices than naturally trained models [114, 117].

Bit Planes Feature Consistency (BPFC)

Addepalli *et al.* [172] propose the Bit Plane Feature Consistency (BPFC) defense method against adversarial examples, which consists in training a model to behave similarly for an example and its high bit plane version (i.e. the example where only the most important bits are kept). The motivation is that as human brains are not fooled by adversarial examples and rely on large magnitude components to make a decision (then slightly refines it by looking at details), by enforcing that a model behaves similarly on an example and its high bit plane version, it is hoped that it does not rely on low bit plane information where adversarial perturbation is located. An illustration of an image and various modifications of it, keeping only some bits, is presented in Figure 4.19.

Figure 4.19: Illustration from [172]. (a) Original 8-bit image (b) Weighted sum of (higher) bit planes 7, 6 and 5 (c) Weighted sum of (higher) bit planes 7 and 6 (d) Bit plane 7 - Most significant bit plane (e) Weighted sum of (lower) bit planes 4, 3, 2, 1 and 0

Flatness of the likelihood landscape

Lin *et al.* [173] show that there exists a link between the flatness of the likelihood landscape and the robustness against adversarial examples. Non-robust models naturally show a high likelihood for clean examples (these points are very likely to be points the model has been trained to predict correctly), but a very low likelihood for slightly perturbed clean examples. This means that perturbing the clean examples results in points which are very likely not to have been considered by the model, and thus points for which the models has not been trained to classify correctly. This results in a very non-flat likelihood landscape around clean examples for non-robust models (with a peak at clean examples), whereas robust models show a flatter likelihood landscape around clean examples (this means that slightly perturbed clean examples are also very likely to have been considered by the model and thus very likely to still be well classified). An illustration of this phenomenon is presented in Figure 4.20.

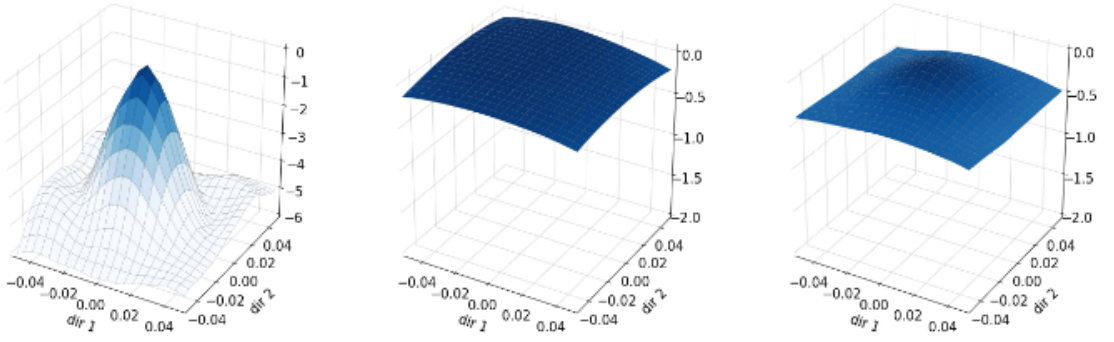

Figure 4.20: Illustration of the log-likelihood around a clean example x for two random directions for various models, from [173]. (Left) standard training. (Middle) Adversarial Training. (Right) Jacobian regularization. The likelihood landscape is strongly not flat around the example for standard training, whereas it is quite flat for Adversarial Training et jacobian regularization.

4.6 Conclusion

In this chapter, we detailed in depth the phenomenon of adversarial examples targeting a neural network model trained for an image classification task. This chapter has allowed to lay out the foundations of the different notions and concepts that will be addressed throughout this thesis.

More precisely, as the study of adversarial examples stays a quite recent field of interest in the scientific community, we began by giving a brief overview of the evolution of the adversarial examples literature. This allows to better tackle the ongoing challenges regarding notably the definition of the threat model and the evaluation of defense schemes. We then proceeded to position our work on *digital* adversarial examples.

We defined formally the threat model for an adversary attacking a neural network for an image classification task. The adversary goal, knowledge and capacity concepts presented here will be specified throughout this work each time it is needed, to property evaluate the efficiency of an attack or a defense scheme. In order to give an intuition of the reasons why neural network are sensitive to adversarial examples, while it is not the case for humans, we reviewed the main explanations for the existence of adversarial examples.

Eventually, to present various ways of finding adversarial examples, as well as various ideas to defend a model against adversarial examples, we detailed some attack and defense schemes. Moreover, this allows to present attack and defense methods that will be used later in this work. For the attack methods, we considered an adversary in a white-box or in a black-box setting. Importantly, we detailed the transferability phenomenon, which is a core subject of our thesis.

Chapter 5

Frequency Analysis of Adversarial Robustness

In the previous chapter, we explained the root cause for the existence of adversarial examples: neural networks rely on features humans are not sensitive to [57]. Notably, we have seen that neural networks classify an image based mostly on local information such as texture [113,114]. Adversarial attacks therefore exploit changes localized to these local features humans do not rely on to perform classification. Indeed, humans make their decision based on various cues, but predominantly global shapes (edges, silhouette, shape, contour, etc.) and to a lesser extent on local information such as texture.

In this chapter, we continue to investigate the reasons for the vulnerability of neural networks to adversarial examples, emphasizing on frequency properties. This allows notably to better understand the sensitivity difference between humans and models. As the starting point of this chapter, cognitive science reports that the process of interpretability for human classification decision relies predominantly on low spatial frequency components.

Consequently, we study the robustness to adversarial perturbations of models trained to leverage information corresponding to different spatial frequency ranges, and more specifically low spatial frequencies. We show that this robustness is tightly linked with the spatial frequency characteristics of the data at stake. Indeed, depending on the data set, the same constraint may results in very different level of robustness (up to 0.41 adversarial accuracy difference).

To explain these results, we conduct several experiments to highlight influential factors. Among these factors, the localization of the information in the frequency spectrum, the sensitivity to high frequency noise, and the transferability of adversarial perturbations between original and low-pass filtered inputs, play an important role.

Related publication proposed during the thesis:

- [174] Bernhard, R., Moellic, P. A., Mermillod , M., Bourrier, Y., Cohendet, R., Solinas, M., Reyboz, M. *Impact of Spatial Frequency Based Constraints on Adversarial Robustness.* IEEE International Joint Conference on Neural Networks (2021). 2021.

5.1 Motivation

The vulnerability of neural network models to adversarial examples arises from models relying on different features or different feature processing from those humans rely on to make their decisions [57, 113, 114, 175]. Interestingly, there is a strong link between robustness and interpretability: the features robust models rely on are interpretable by humans, and vice versa. Indeed, it has been shown that models trained with Adversarial Training or Randomized Smoothing exhibit interpretable gradients [117, 118, 176]: the more robust a model is, the more it will display interpretable features. Inversely, a recent effort demonstrates that a model trained to have explainable jacobian matrices presents adversarial robustness [171]. However, only few works underline this critical link between robustness and the interpretability of the features a model relies on.

The association between robustness and interpretability has been studied by Zhang *et al.* [114] at frequency level. The authors experimentally highlight a link between robustness for models and some frequency properties. For models trained with Adversarial Training, low spatial frequency (hereafter denoted by LSF) information related to interpretable notions such as shape is more important than non-interpretable concepts associated with high spatial frequency (hereafter denoted by HSF) information.

For humans, experimental evidence in neural computation and cognitive psychology suggests the importance of LSF to perform efficient classification [177–179]. Therefore, a natural hypothesis would be that a model trained specifically to rely more on LSF information might present an improved adversarial robustness. This hypothesis has already been indirectly exploited by taking advantage of preprocessing defense schemes based on HSF components filtering [180–182]. Other defenses aim at training a model exploiting more human interpretable information. As an example, Addepalli *et al.* [172] give higher importance to information present in higher bit planes, which is information humans predominantly rely on to make their decisions. However, these methods stay agnostic of the intrinsic spatial frequency characteristics of the data.

In this chapter, we aim at investigating the link between robustness against adversarial examples and frequency properties of the data at stake. More precisely, our objective is twofold. First, we experimentally question some preconceived hypothesis related to adversarial examples, more particularly ones considering adversarial perturbations as a pure HSF phenomenon with data-agnostic spatial frequency characteristics. Second, we aim at investigating the link between the spatial frequency features of the information that a model uses to perform predictions *and* the

robustness against adversarial perturbations offered by spatial frequency-based constraints. Our key contributions are:

- We show that a frequency-based regularization induces very different levels of robustness according to the frequency features of the data. As an example, a low-frequency constrained model on CIFAR10 (that covers a broad frequency spectrum) has no robustness, while reaching a 41 % true robustness for SVHN against the l_∞-PGD attack.
- By analyzing the sensitivity of a standard model as well as adversarial transferability properties, we observe that enforcing a model to rely on LSF information is not a necessary condition to bring adversarial robustness. We identify key factors which condition robustness when training a model to rely on LSF information.
- We notice that, depending on the data set complexity, some models spread over the whole frequency spectrum, and show that constraints spanning different frequency ranges can help improving robustness.
- We discuss combination with Adversarial Training for a future overall defense strategy.

This chapter is organized as follows. After positioning our work in relation to the state-of-the-art in Section 5.2, we present the details about data set and filtering methods that will be used throughout the thesis in Section 5.3. We analyze, in Section 5.4, frequency properties and sensitivity of features learned by a model. Notably, we study to which extent features learned by a model are focused on low or high spatial frequency concepts, and the sensitivity of models to frequency constrained noise. In Section 5.5, we set forth interesting links between transferability of adversarial perturbations and frequency properties of the information models make use of. In Section 5.6, we design loss functions to force a model to extract features in particular frequency ranges, and observe the potential effects on adversarial robustness. We link these effects with analysis performed in Sections 5.4 and 5.5. We also show promising future work direction by binding our findings with Adversarial Training.

5.2 Neural network models and frequency concerns: state-of-the-art

As mentioned in the previous chapter, recent efforts [90, 116] experimentally demonstrate that regular models predominantly exploit non-interpretable HSF components, and that robust models tend to use more concepts, such as shape, associated to LSF [113, 114], in a way similar to that of humans [177–179]. Therefore, a common belief is that the adversarial vulnerability of a model comes from the utilization of HSF components [183]. However, Yin *et al.* [90] show that adversarial perturbations cannot be viewed only as a HSF phenomenon: adversarially trained models, more sensitive to LSF, stay vulnerable to an adversary that optimally exploits this part of the frequency spectrum. Notably, Sharma *et al.* [184] make use of LSF constraints to craft adversarial examples against adversarially trained models on ImageNet and use fewer iterations than classical adversarial attacks.

As adversarial examples are the consequence of models relying on brittle and non-interpretable features [57], some papers exploit the idea of enforcing a model to use features to which humans are sensitive. During training, Yin *et al.* [90] add LSF noise and demonstrate that it does not necessarily improve robustness to LSF perturbations. As a explanation, the authors hypothesize that as natural images are more LSF concentrated, it is harder for a model to become invariant to these spatial frequencies and, then, to LSF perturbations. Before, Geirhos *et al.* [113] interestingly took advantage of stylized images that keep only shape information within a data augmentation process to improve the robustness against common perturbations. Even if not related to frequency concerns, Addepalli *et al.* show in [172] that constraining a model to rely on the information contained in the higher bit planes only (as humans make decisions based on the information of large magnitude) has a positive impact on robustness.

5.3 Preliminaries

5.3.1 Filtering with the Fourier transform

In order to perform low-pass or high-pass filtering for an image, we simply delete the undesired spatial frequencies in the Fourier domain, similarly to almost all works in the adversarial examples

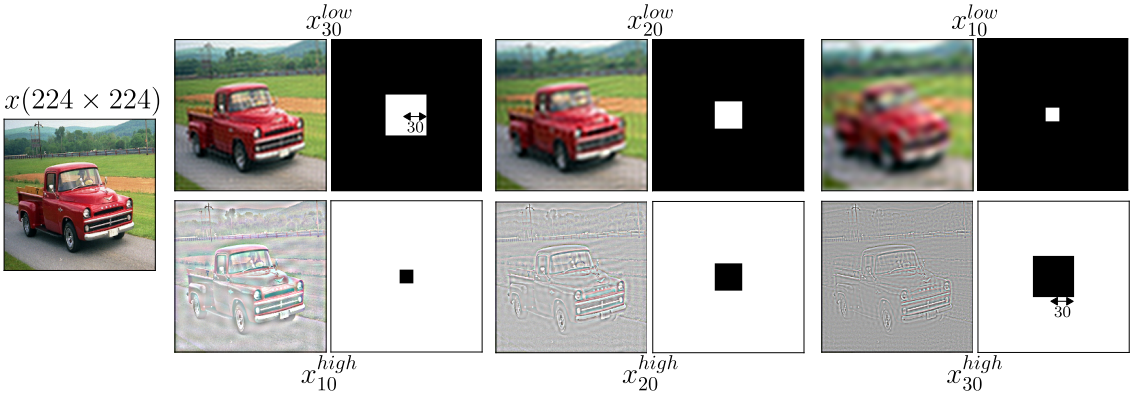

Figure 5.1: Original image (left). Low-pass filtered images with corresponding mask (first row). High-pass filtered images (magnified for visualization) with corresponding mask (bottom row). For the Fourier domain masks, white denotes value 1 and black value 0. **For LSF, low filtering intensity means restricted low-pass filtering. For HSF, high filtering intensity means restricted high-pass filtering.**

literature [90, 116, 183, 184]. We note $\mathcal{F}$ and $\mathcal{F}^{-1}$ the Discrete Fourier Transform (DFT), and its inverse function, respectively. $\mathcal{C}$ and $\mathcal{C}^{-1}$ denote a specific operator which centers coefficients of the Fourier transform, and its inverse, respectively. More precisely, for a gray-scale image $a \in \mathbb{R}^{n \times n}$, $\mathcal{CF}(a) \in \mathbb{C}^{n \times n}$ denotes the centered variant of $\mathcal{F}(a) \in \mathbb{C}^{n \times n}$, where coefficients for low frequencies are located at the center, and those for high-frequencies at the corners. The filtered version of an image a is obtained in the following way:

1. First, the centered DFT $\mathcal{F}_c(a) = \mathcal{CF}(a)$ is computed.
2. Second, the Hadamard product of $\mathcal{F}_c(a)$ with a mask Ω is performed, as $\mathcal{F}_c(a) \odot \Omega$, to select the desired frequencies.
3. Third, the result is off-centered with the operator $\mathcal{C}^{-1}$.
4. Finally, the resulting filtered image is obtained with the inverse $\mathcal{F}^{-1}$ of the Fourier transform.

In summary, the filtered image is obtained as: $a^{freq} = \mathcal{F}^{-1}\mathcal{C}^{-1}\mathcal{F}_c(a) \odot \Omega$.
For a color image, the procedure is applied to each of the channels. We denote x_i^{low} and x_i^{high} respectively the low-pass and high-pass filtered versions of x at intensity i. The filtering intensity is encompassed in the form of the binary mask Ω:

- Low-pass filtering. Ω corresponds to 1's *only in* the $2i \times 2i$ square in the middle.
- High-pass filtering. Ω corresponds to 1's *outside* the $2i \times 2i$ square in the middle

In consequence, for low-pass (resp. high-pass) filtering, the smaller (resp. the higher) the intensity i, the stronger the filter is. We illustrate this process in Figure 5.1.

The *LSF task* (resp. *HSF task*) refers to the classification task where inputs have been low-pass (resp. high-pass) filtered at some intensity, i.e. where input-label pairs correspond to (x_i^{low}, y) (resp. (x_i^{high}, y)) for some i. M_i^{low} (resp. M_i^{high}) denotes a model trained for the LSF (resp. HSF) task.

5.3.2 Data sets and models

We consider CIFAR10, SVHN and a custom-built data set, named *Small ImageNet*, built by extracting 10 meta classes from the ImageNet ILSVRC2012 benchmark. For each meta class, we extracted 3,000 images from the original training set and 300 images from the non-blacklisted validation set. More details about the Small ImageNet data set are presented hereafter. All input images are scaled to $[0, 1]$.

CIFAR10 and SVHN, both involving color images of size 32×32 allow to make comparisons and draw conclusions about the observed phenomenon. *Small ImageNet*, composed of 224×224 color images, enables to extrapolate results on images with higher definition.

SVHN and CIFAR10

For SVHN we use a model inspired from VGG [185], whose architecture is presented in Table 5.1. The model is trained for 25 epochs, a batch size of 28, and Adam optimizer with a learning rate starting at 0.01 and decreasing to 0.001 after 15 epochs. A natural models reaches a test set accuracy of 0.96.

Table 5.1: SVHN classifier architecture. BN, MaxPool(u,v) and Conv(f,k,k) denote respectively batch normalization, max pooling with window size (u,v), and 2D convolution with f filters and kernel of size (k,k).

Architecture
Conv(64,3,3) + BN + ReLU
Conv(64,3,3) + MaxPool(2,2) + BN + ReLU
Conv(128,3,3) + BN + ReLU
Conv(128,3,3) + MaxPool(2,2) + BN + ReLU
Conv(256,3,3) + BN + ReLU
Conv(256,3,3) + MaxPool(2,2) + BN + ReLU
Dense(1024) + BN + ReLU
Dense(1024) + BN + ReLU
Dense(10) + softmax

For CIFAR10 we use a WideResNet28-8 model [186]. The models is trained for 100 epochs, a batch size of 128, and stochastic gradient descent with a momentum of 0.9, with a learning rate starting at 0.1 and decreasing to 0.02, 0.004 and 0.0008 after respectively 40, 75 and 90 epochs. This model reaches a test set accuracy of 0.93.

Small ImageNet

The *Small ImageNet* data set is a downscaled version of the ImageNet data set [70]. The chosen resolution for images is 224×224. To build this data set, we created 10 meta classes from the original ImageNet classes. For each meta class, we extracted $3,000$ images from the original training set and 300 images from the non-blacklisted validation set.
More precisely, each meta class corresponds to three original classes. The final meta classes and the indexes of the ImageNet original classes are detailed in Table 5.2 [1].

Table 5.2: Small ImageNet. Matching between each Small ImageNet class and the original ImageNet class indexes. The human readable labels corresponding to indexes of ImageNet can be found at `https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a`

Meta class	Original ImageNet class index
shark	2, 3 and 4
bird	85, 86 and 87
frog	30, 31 and 32
dog	153, 154 and 155
cat	281, 282 and 283
bear	294, 295 and 296
monkey	367, 368 and 369
ball	768, 805 and 852
chair	423, 559 and 765
truck	569, 717 and 867

For *Small ImageNet* we consider a MobileNetV2 model [187]. The model is trained for 100 epochs, a batch size of 128, and stochastic gradient descent with a momentum of 0.9, with a

[1] The human readable labels corresponding to indexes of ImageNet can be found at `https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a`

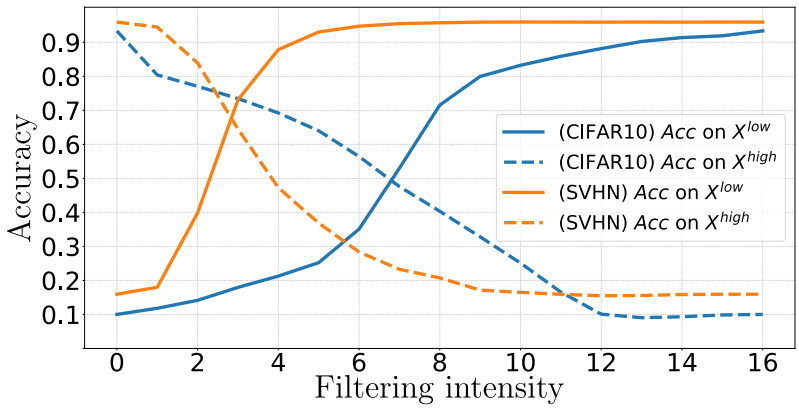

Figure 5.2: CIFAR10 and SVHN. Accuracy of a regular model on low-pass and high-pass filtered data set, for different filtering intensities.

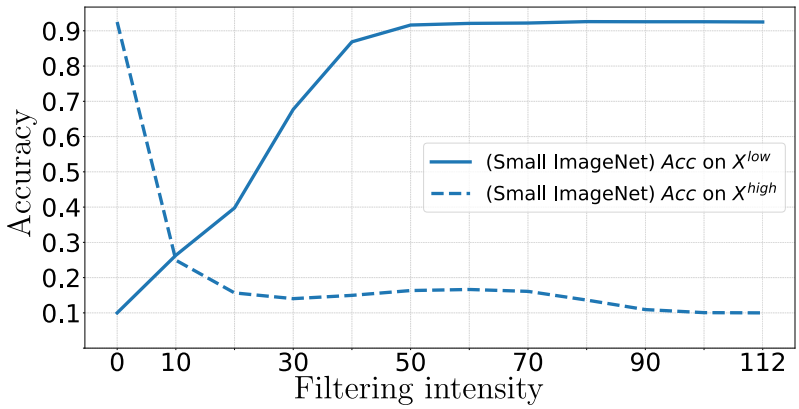

Figure 5.3: Small ImageNet. Accuracy of a regular model on low-pass and high-pass filtered data set, for different filtering intensities.

learning rate starting at 0.1 and decreasing to 0.02, 0.004 and 0.0008 after respectively 40, 75 and 90 epochs. A natural models reaches a test set accuracy of 0.93.

5.4 Frequency properties of data and models

In this section we aim at gaining insight on the way information learned by a classifier trained to solve the regular classification task contains information for the LSF and HSF tasks. These experiments allow to gain intuition on the way models leverage information with respect to data sets frequencies. The conclusions draws in this section will help to better understand robustness dissimilarities which occur between different models trained with the same frequency-based constraints (later in Section 5.6).

5.4.1 Impact of filtered data sets

We begin by evaluating CIFAR10 and SVHN regular models on low-pass and high-pass filtered images. Results are presented in Figure 5.2. For CIFAR10, we notice that the accuracy of a model decreases much slower than SVHN when evaluated on more and more high-pass filtered images. We obtain an opposite result for SVHN with low-pass filtered data. We can therefore assume that the informative features learned by the regular model are more focused on the LSF task for SVHN and are more spread between LSF and HSF tasks for CIFAR10. This analysis is consistent with the intrinsic frequency features of the data sets that is revealed by a classical Fourier analysis (presented in Figure 5.4, top row) that actually shows a quite narrow spectrum for SVHN (towards LSF) and a spread spectrum for CIFAR10.
In Figure 5.3, we present the accuracy of a Small ImageNet regular model on low-pass and high-pass filtered data sets. The accuracy for low-pass filtered images decreases quite slowly considering

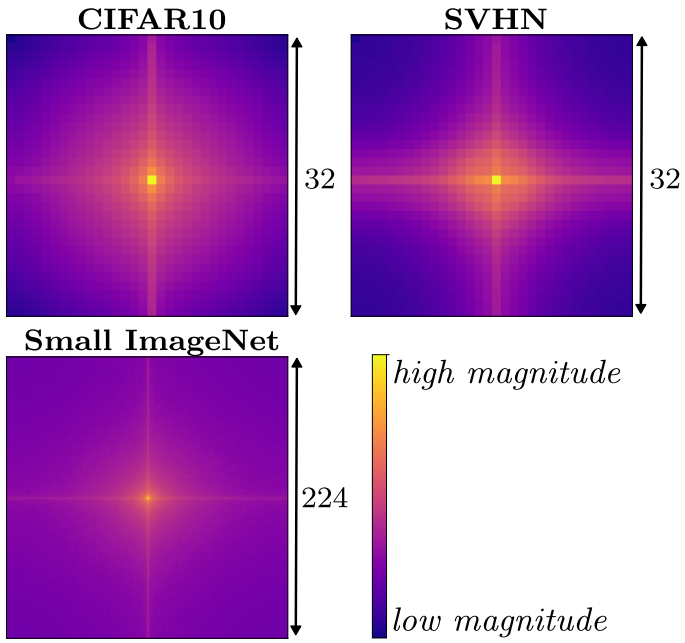

Figure 5.4: CIFAR10, SVHN and Small ImageNet data sets. Fourier spectrum for clean images. Low frequencies are at the center, and high frequencies at the corners.

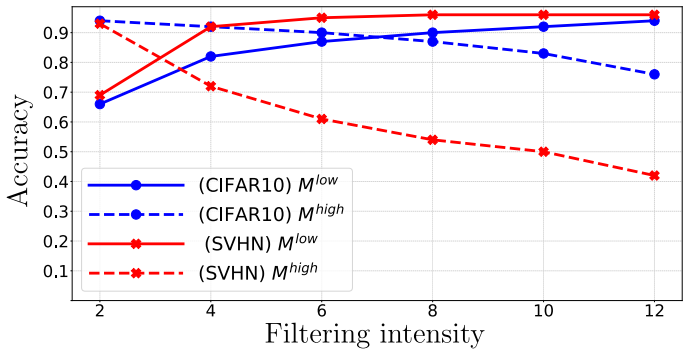

Figure 5.5: CIFAR10 and SVHN. Accuracy of M^{low} (resp. M^{high}) models on LSF (resp. HSF) filtered data sets. (See Figure 5.1 for filtering intensity effect).

the range of the spectrum. This is an indication towards the fact that informative features learned by the regular model are more spread between LSF and HSF tasks for Small ImageNet. This observation is strenghtened by the Fourier analysis in Figure 5.4 (bottom row).

The accuracy of models M_i^{low} (resp. M_i^{high}) for various i for the LSF (resp. HSF) test set filtered at intensity i, presented in Figure 5.5, further highlights this phenomenon. The accuracy reached by models M_i^{low} decreases much slower for SVHN than for CIFAR10, as the filtering intensity increases (and the inverse phenomenon is observable for high-pass filtering). This agrees with previous results (Figure 5.2), i.e. the useful information for the classification task is more distributed in the frequency spectrum for CIFAR10 compared to SVHN for which the information is predominantly concentrated in the low frequencies.

5.4.2 Sensitivity to high spatial frequency noise

We investigate the sensitivity of regular models to perturbations in specific frequencies. For that purpose, we use a similar procedure as in [90]: the sensitivity is measured as the error rate of the model on a set of examples perturbed with noise located only in those spatial frequencies. More precisely, for a clean input image $x \in \mathbb{R}^{n \times n \times c}$, each channel is perturbed independently by the addition of a noise $rvU_{i,j}$, where $U_{i,j}$ is a Fourier basis matrix for coordinates (i, j) in the Fourier domain, r is chosen randomly in $\{-1, 1\}$, and v controls the magnitude of the perturbation. We then measure the error rate of a model on $1,000$ well-classified test set examples perturbed with

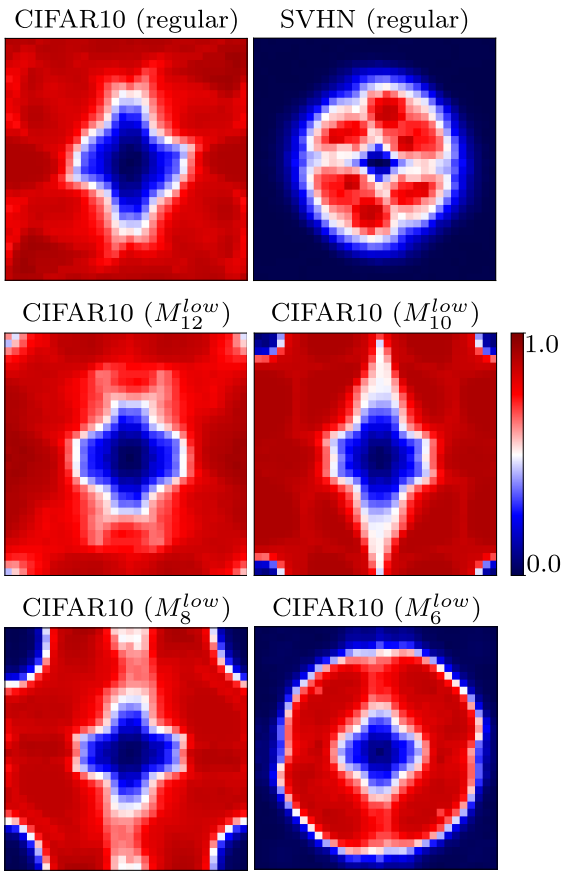

Figure 5.6: CIFAR10 and SVHN. Error rate of models on data perturbed with Fourier constrained noise. Value at (i, j) measures the error rate on data perturbed with noise along Fourier basis matrice $U_{i,j}$. Low frequencies are located at the center, and high frequencies at the corner. High values (red color) denote a high sensibility, and low values (blue color) denote a low sensibility.

this type of noise, as a function of (i, j). We focus our analysis on CIFAR10 and SVHN, two data sets with the same image size, and we experimentally set $v = 4$. Results are presented in Figure 5.6 (top row). We notice that a CIFAR10 regular model is more sensitive to LSF and HSF noise, while a SVHN regular model is more sensitive to LSF and not to high and very high frequencies.

For CIFAR10, we also investigate the sensitivity of models M_i^{low}. Results are presented in Figure 5.6 (middle and bottom rows). Interestingly, we notice that the models M_i^{low} become less sensitive to HSF as i decreases (i.e. as the low-pass filtering intensity increases). Therefore, for CIFAR10, training a model on low-pass filtered images brings robustness against HSF perturbations. However, this robustness would be useless if adversarial perturbations were to rely on a broader spectrum than only HSF. We investigate this in the following section.

5.5 Transferability analysis

The transferability of adversarial perturbations between two models has been explained by shared non-robust useful features [57] (see Section 4.3.2), that is features sensitive to adversarial perturbations and exploited for the prediction by both models. Therefore, to gain a better understanding of the nature of features at stake as well as frequency properties of adversarial perturbations, we study the transferability of adversarial examples between a regular model and models trained for the LSF or HSF task (i.e. M^{low} and M^{high}). For these M^{low} and M^{high} models, a pre-processing layer consisting of the filtering operation is added before the model to evaluate it on non-filtered inputs. We use the l_∞ DIM attack [139] (a state-of-the-art gradient-based attack tuned for transferability), with 40 iterations, a probability $p = 0.8$ and a l_∞ perturbation budget of 0.03.

We first remind that for models M_i^{low}, a low filtering intensity i means that the model is trained on a strict low-pass filtered data set (see Figure 5.1) and as the filtering intensity i increases, more and more HSF are considered. Thus, for example, adversarial examples crafted on M_2^{low} exploit

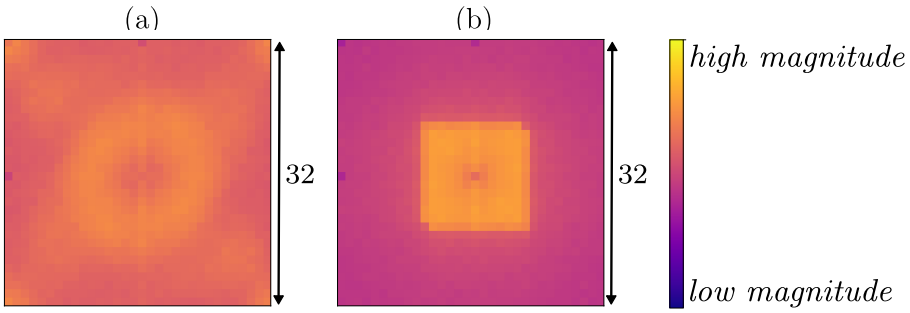

Figure 5.7: CIFAR10. Fourier spectrum for the adversarial perturbation (a) Regular model, (b) Model trained on low-pass filtered data set at intensity 6.

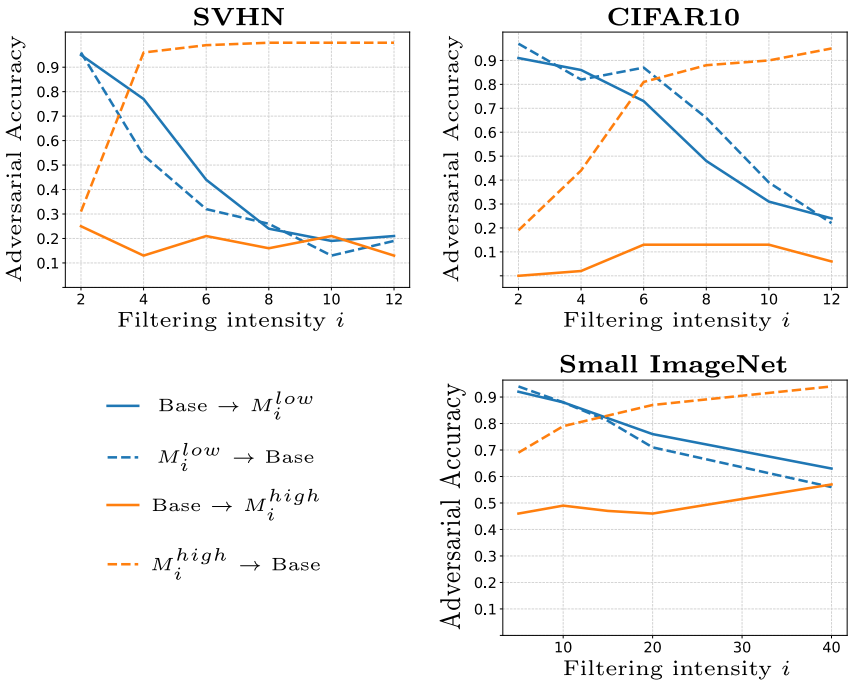

Figure 5.8: SVHN (top left), CIFAR10 (top right), Small ImageNet (bottom right). Transferability analysis between base model and models trained on a filtered data set. (See Figure 5.1 for filtering intensity effect).

non-robust features learned with only LSF information and adversarial examples crafted on M_{12}^{low} will have the possibility to take advantage of non-robust features determined on a broader range of frequencies. On the contrary, for models M_i^{high}, a high filtering intensity i means that only HSF are kept (see Figure 5.1), and a low intensity represents a larger spectrum, gathering more LSF. Here, adversarial examples crafted on M_{12}^{high} will exploit non-robust features learned exclusively on HSF information and adversarial examples crafted on M_2^{high} will rely on non-robust features from a broader spectrum. We illustrate this in Figure 5.7 *(a)* and *(b)*, by presenting the magnitude of the Fourier spectrum of adversarial perturbation crafted on the regular model and on M_6^{low}. We observe that the perturbation is quite uniformly distributed along the spectrum (confirming a result from [90]) for the regular model and predominantly focused on the low frequencies for M_6^{low}. This is an important point, as it shows that adversarial examples are not HSF phenomena but may rely on a large range of spatial frequencies to efficiently fool a model.

Furthermore, analyzing the transferability results of Figure 5.8, we can provide interesting information about the nature of the robust and non robust features of a model as well as the properties of adversarial perturbations. Importantly, we observe similar behaviors for CIFAR10, SVHN and Small ImageNet.

A first conclusion comes from the two-way transferability between the regular model M (noted *Base*) and models M_i^{low} (blue curves). Indeed, we notice that the stronger the low-pass filtering (smaller i), the lower the transferability of adversarial perturbations. This indicates that the

regular classification task and the LSF task share predominantly robust useful features.

Secondly, an important outcome arises from the dissimilarity between the two orange curves for each data set. On one hand, the solid orange curve attests the impact of non-robust features exploiting HSF, highlighted by an almost constant success of adversarial examples crafted from the base model against models M_i^{high} (about 0.1, 0.2 and 0.5 of adversarial accuracy for CIFAR10, SVHN and Small ImageNet respectively). On another hand, the dotted orange curve indicates that, as the high-pass filtering becomes more restrictive (i.e. as the i value increases), the transferability of adversarial examples crafted on M_i^{high} to the regular model decreases. These observations support the claim that, to be efficient, adversarial perturbations must exploit a wide part of the spectrum, and therefore cannot be only focused on HSF. This is particularly the case for Small ImageNet, as *i)* the accuracy from a regular to a M^{high} model is higher than for CIFAR10 and SVHN, and *ii)* the transferability of adversarial examples crafted on M_i^{high} is already poor for $i = 5$.

We can summarize our conclusions as follows:

- Robustness is strongly related to features that rely on LSF information.
- Adversarial perturbation are not efficient when focused only on HSF.

In the next section, we propose to look deeper into the impact of frequency-based constraints when training a regular model in order to investigate if it can help to increase its robustness.

5.6 Adversarial robustness of frequency-constrained models

Following the precedent observations, we investigate loss functions designed to make a model to leverage informative features for both the regular task and the LSF and/or HSF task and evaluate how it impacts the adversarial robustness.

5.6.1 Frequency-based regularization

To enforce a model M_θ to rely on information relative to the LSF or HSF task, we define the following loss function L^{freq}, which acts on the logits (with h being the logits function).

$$\begin{aligned}\mathcal{L}_{i,j}^{freq}(x, y, M_\theta) =& \mathcal{L}_{CE}(x, y, M_\theta) + \lambda_1 \left\| h(x) - h(x_i^{low}) \right\|_2^2 \\ &+ \lambda_2 \left\| h(x) - h(x_j^{high}) \right\|_2^2\end{aligned} \tag{5.1}$$

For readability, when $\lambda_1 = 0$ (i.e. constraint is only focused on the HSF task) the loss is simply noted as $\mathcal{L}_i^{high}$, and for $\lambda_2 = 0$ as $\mathcal{L}_i^{low}$. During training, the cross-entropy part $\mathcal{L}_{CE}$ makes the model to learn useful features to solve the regular classification task. The second part (moderated by λ_1 and λ_2) constraints the model to extract useful features coherently to the LSF and HSF tasks.

5.6.2 Training setup

For SVHN, we consider the loss functions $\mathcal{L}_i^{low}$ and $\mathcal{L}_i^{high}$ for $i \in \{2, 4, 6, 8, 10\}$, and $\mathcal{L}_{i,j}^{freq}$.

We train models for 85 epochs, a batch size of 64, and Adam optimizer with a learning rate starting at 0.01 and decreasing to 0.001 and 0.0001 after 42 and 64 epochs. The constants λ_1 and λ_2 used to perform the trade-off between the cross-entropy loss and the constraints are increased during training. The value starts at 0.1, increasing at 0.5, 1.0, 2.5 and 3.0 after respectively 10, 20, 40 and 75 epochs.

For CIFAR10, we consider the loss functions $\mathcal{L}_i^{low}$ and $\mathcal{L}_i^{high}$ for $i \in \{2, 4, 6, 8, 10\}$, and $\mathcal{L}_{i,j}^{freq}$.

We train models for 200 epochs, a batch size of 128, and stochastic gradient descent with a momentum of 0.9, with a learning rate starting at 0.1 and decreasing to 0.01, 0.001 after respectively 100, and 160 epochs. For the constants λ_1 and λ_2, the value starts at 0.1, increasing at 0.5, 1.0, 1.75 and 2.5 after respectively 20, 57, 114 and 170 epochs.

For Small ImageNet, as each model training is costly, we consider the loss functions $\mathcal{L}_i^{low}$, $\mathcal{L}_i^{high}$ and $\mathcal{L}_{i,j}^{freq}$ for representatives values in $\{10, 20, 40, 60\}$.

We train models for 200 epochs, a batch size of 128, and stochastic gradient descent with a momentum of 0.9, with a learning rate starting at 0.1 and decreasing to 0.01, 0.001 after respectively 110, and 180 epochs. For the constants λ_1 and λ_2, the value starts at 0.01, increasing at 0.1, 0.5, 1.0 and 2.0 after respectively 24, 71, 142 and 190 epochs.

5.6.3 Robustness evaluation

In this section, the robustness against adversarial examples is evaluated with the adversarial accuracy (Acc_{add}), considering an adversary in the white-box setting under the common l_∞ threat model. Adversarial examples are crafted with the l_∞-PGD attack [75], with a perturbation budget of $\epsilon = 0.03$. To provide the more accurate evaluation of robustness as possible, notably to assess that no gradient masking occurs, we follow state-of-the-art guidelines from [81]. Notably, we report results for $5,000$ iterations (we check that increasing this number does not lower robustness) against $1,000$ well-classified test set examples, when the following sanity checks are passed:

- increasing the perturbation budget ϵ does decrease the adversarial accuracy,
- the gradient-free attack SPSA [79] does not perform better than l_∞-PGD,
- the robustness against black-box transfer attacks [64] from a model with the same architecture and the MIM attack [137] does not perform better than PGD.

Table 5.3: CIFAR10, SVHN and Small ImageNet. Best and worst accuracy on clean (Acc) and adversarial examples (Acc_{adv}) of models trained with the constrained loss $\mathcal{L}_i^{low}$ or $\mathcal{L}_i^{high}$.

SVHN				
	Acc	$Constraints$	Acc_{adv}	$Constraints$
Best	0.96	$\mathcal{L}_{i>2}^{low}$	0.41	$\mathcal{L}_6^{low}$
Worst	0.93	$\mathcal{L}_2^{low}$	0.0	$\mathcal{L}_{10}^{low}$
CIFAR10				
	Acc	$Constraints$	Acc_{adv}	$Constraints$
Best	0.92	$\mathcal{L}_2^{low}$, $\mathcal{L}_{10}^{high}$	0.0	$\mathcal{L}_*^{low}$, L_*^{high}
Worst	0.74	$\mathcal{L}_2^{low}$	0.0	$\mathcal{L}_*^{low}$, $\mathcal{L}_*^{high}$
Small ImageNet				
	Acc	$Constraints$	Acc_{adv}	$Constraints$
Best	0.92	$\mathcal{L}_{40}^{low}$	0.0	$\mathcal{L}_*^{low}$, $\mathcal{L}_*^{high}$
Worst	0.88	$\mathcal{L}_{40}^{high}$	0.0	$\mathcal{L}_*^{low}$, $\mathcal{L}_*^{high}$

Table 5.4: CIFAR10. Accuracy on clean (Acc) and adversarial (Acc_{adv}) examples of models trained with loss L_i^{low} on a low-pass version of CIFAR10. The value l denotes the filtering intensity for the data set.

Data filtering: $l =$	*6*	*6*	*8*	*8*	*10*	*10*	*12*	*12*
Loss constraint: $i =$	*2*	*4*	*2*	*4*	*2*	*4*	*4*	*10*
Acc	0.75	0.82	0.76	0.83	0.73	0.84	0.83	0.92
Acc_{adv}	0.09	0.08	0.06	0.05	0.09	0.11	0.08	0.0

5.6.4 Influence of intrinsic frequency properties of the data bias on the level of adversarial robustness

Results for $\mathcal{L}^{low}$ and $\mathcal{L}^{high}$ are presented in Table 5.3, and allow for a first important observation. Indeed, we see that the same constraint induces very different effects on the robustness, depending

on the data set at stake. For CIFAR10 and Small ImageNet we observe no robustness when considering separate losses $\mathcal{L}_*^{low}$ or $\mathcal{L}_*^{high}$. On the contrary, models trained on SVHN with $\mathcal{L}^{low}$ present an interesting level of adversarial robustness depending on the intensity level.

This observation links with different frequency properties of the data sets highlighted in Section 5.4: the information learned by models trained on CIFAR10 and Small ImageNet are spread over the whole frequency spectrum, and – on the contrary – focused on LSF for SVHN. To further investigate this link, we proceed to check if a CIFAR10 model trained with $\mathcal{L}^{low}$ would still reach some robustness on low-pass filtered data (i.e. train with $\mathcal{L}_i^{low}(\theta, x^{low}, y)$). In other words, we try to exclude the high spatial frequencies, which are predominantly non robust, and that we assume to explain this complete lack of robustness.

To that purpose, we train models with $\mathcal{L}_i^{low}$ $(i = 6, 8, 10, 12)$ on low-pass filtered versions of CIFAR10 $(i = 2, 4, 10)$. Thus, we ensure that the model learns features strictly focused in the LSF. Results are presented in Table 5.4. Interestingly, we measure a slight but true and non-negligible robustness for some models with loss $\mathcal{L}^{low}$ (up to an adversarial accuracy of 0.11).

As robustness is noticed for SVHN with $\mathcal{L}^{low}$ or for CIFAR10 with $\mathcal{L}^{low}$ on low-pass filtered data, a first conclusion is that a model relying predominantly on useful features of the LSF task can be made more robust with low-frequency based constraint ($\mathcal{L}^{low}$). Moreover, these models also share a non-sensitivity to high and very high frequencies (cf. Section 5.4), which highlights another influential factor on the robustness of a model trained with the loss $\mathcal{L}^{low}$.

However, is a frequency-based regularization suitable for complex data sets such as CIFAR10 or Small ImageNet? In Table 5.5, we present results with a constraint spanning a wider spectrum ($\lambda_1 > 0$, $\lambda_2 > 0$). For SVHN, as expected, the combined constraints does not enable to reach better robustness compared to the loss $\mathcal{L}^{low}$, as the combination of the two constraints is not compatible with the intrinsic frequency properties of the data set: information is predominantly concentrated in the low frequencies. For Small ImageNet and CIFAR10 the combined constraint forces the model to learn informative features for the LSF and HSF tasks, which are mainly robust as features informative of the LSF task are mostly robust, as shown in Section 5.5 for CIFAR10. Interestingly, stronger level of robustness is observed for Small ImageNet and we hypothesize that the impact of the combined constraint is all the more efficient that the frequency spectrum is wider.

Table 5.5: CIFAR10, SVHN and Small ImageNet. Best and worst accuracy on clean (Acc) and adversarial examples (Acc_{adv}) of models trained with the constrained loss $\mathcal{L}_{i,j}^{freq}$.

SVHN				
	Acc	$Constraints$	Acc_{adv}	$Constraints$
Best	0.96	$\mathcal{L}_{10,4}^{freq}$, $\mathcal{L}_{6,*}^{freq}$, $\mathcal{L}_{8,6}^{freq}$	0.30	$\mathcal{L}_{6,3}^{freq}$
Worst	0.95	$\mathcal{L}_{4,12}^{freq}$	0.0	$\mathcal{L}_{10,4}^{freq}$, $\mathcal{L}_{6,10}^{freq}$, $\mathcal{L}_{6,8}^{freq}$
CIFAR10				
	Acc	$Constraints$	Acc_{adv}	$Constraints$
Best	0.95	$\mathcal{L}_{5,3}^{freq}$, $\mathcal{L}_{8,6}^{freq}$	0.12	$\mathcal{L}_{5,3}^{freq}$, $\mathcal{L}_{*}^{high}$
Worst	0.92	$\mathcal{L}_{4,12}^{freq}$	0.03	$\mathcal{L}_{4,12}^{freq}$, $\mathcal{L}_{6,8}^{freq}$
Small ImageNet				
	Acc	$Constraints$	Acc_{adv}	$Constraints$
Best	0.92	$\mathcal{L}_{60,*}^{freq}$	0.36	$\mathcal{L}_{40,20}^{freq}$
Worst	0.89	$\mathcal{L}_{10,60}^{freq}$	0.0	$\mathcal{L}_{60,*}^{freq}$

5.6.5 Compatibility of frequency-based regularization with Adversarial Training

Motivated by the true robustness observed when training some models with loss functions $\mathcal{L}^{low}$, $\mathcal{L}^{high}$ or $\mathcal{L}^{freq}$, we study the combination of these losses with Adversarial Training (hereafter AT) [75], a common and widely used approach when defending against an adversary in the white-box setting. This study is of particular interest as AT is shown to be related to frequency concern as

it makes a model rely more on low-frequency concepts and being less sensitive to perturbations in the high-frequencies [113, 114]. AT consists of training a model on adversarial examples generated online and designed to induce a worst-case loss. Considering a l_∞ bound ϵ for adversarial perturbations and an input-label pair (x, y), an adversarial example $x' = x + \delta$ is generated following:

$$\delta = \underset{\|\delta\|_\infty \leq \epsilon}{\operatorname{argmax}} \quad \mathcal{L}_{CE}(x + \delta, y, M_\theta) \tag{5.2}$$

The model is then trained minimizing the loss $\mathcal{L}_{AT} = \mathcal{L}_{CE}(\theta, x + \delta, y)$. For the combination of Adversarial Training with frequency-based constraints, we define the loss functions $\mathcal{L}^{AT,freq}$:

$$\mathcal{L}^{AT,freq}_{i,j}(x, y, M_\theta) = \mathcal{L}^{freq}_{i,j}(x + \delta, y, M_\theta) \tag{5.3}$$

Training setup

As AT can be prone to robust overfitting [161], we perform early-stopping relatively to the error on an hold-out set of test set examples when training models with these loss functions.

For SVHN, models are trained for $80,000$ steps, a batch size of 64, and stochastic gradient descent with a momentum of 0.9, with a learning rate starting at 0.01 and decreasing to 0.001, 0.0001 after respectively $40,000$, and $60,000$ steps. For the constants λ_1 and λ_2, the value starts at 0.05, increasing at 0.1, 0.2, 0.25, 0.3 and 0.4 after respectively 1000, $10,000$, $20,000$ $40,000$ and $70,000$ steps.

For CIFAR10, models are trained for $80,000$ steps, a batch size of 128, and stochastic gradient descent with a momentum of 0.9, with a learning rate starting at 0.1 and decreasing to 0.01, 0.001 after respectively $40,000$, and $60,000$ steps. For the constants λ_1 and λ_2, the value starts at 0.1, increasing at 0.3, 0.5, 0.7 and 1.0 after respectively $10,000$, $20,000$ $40,000$ and $70,000$ steps.

We evaluate the robustness against the l_∞ PGD attack with $\epsilon = 0.03$ for CIFAR10 and SVHN, considering the same sanity checks and attack parameters as the ones used in Section 5.6, to ensure the tightest evaluation of robustness as possible.

Results

Results are presented in Tables 5.6 and 5.7 for CIFAR10 and SVHN, respectively. We highlight the fact that, when dealing with Adversarial Training, there is an inherent trade-off between accuracy and robustness [147]. Therefore, for a fair comparison, we focus on the benefits of the addition of frequency-based constraints, only for no (or up to 0.1%) loss of natural accuracy (highlighted in bold in Tables 5.6 and 5.7). Particularly, for CIFAR10, a model trained with loss $\mathcal{L}^{AT,freq}_{2,10}$ outperforms AT by 0.06 on adversarial accuracy. For SVHN, this improvement reaches 0.12 for a model trained with loss $\mathcal{L}^{AT,freq}_{10,4}$.

From these results, two important and dependent observations can be made. In first place, it shows that existing defense scheme can in fact benefit from constraints related to frequency properties. However, in second place, it shows that the intrinsic effects of the defense scheme (AT) has an influence on the effectiveness of such frequency-based regularization. Indeed, relatively to each data set, there is no connection between the frequency constraints which allow the loss function $\mathcal{L}^{AT,freq}$ to outperform Adversarial Training, and the ones which bring robustness when considered alone (5.6.4). Indeed, as an example, for CIFAR10, no robustness was brought by constraint on the HSF task (Table 5.3), whereas this is the loss $\mathcal{L}^{AT,high}_i$ which outperforms Adversarial Training. A possible explanation is very likely to come from Adversarial Training biasing the sensitivity of models towards the low frequencies. Therefore, when combining frequency related constraint with a defense scheme, not only do the frequency properties of the data at stake play an important role, but frequency related effects of the defense itself has a strong influence.

5.7 Conclusion and future work

In this chapter, we studied the link between frequency concerns and adversarial robustness. This allows to better tackle the reasons for the sensitivity difference between humans and neural network models to adversarial perturbations. Indeed, the existing literature is focused on explaining this difference either with low-level features [57] or high-level concepts such as shape, contour, texture, etc. [113, 114]. Particularly, this work allows to gain insight on the strong influence of frequency properties of the data at stake that may be very specific according to the application domain.

Table 5.6: CIFAR10. Accuracy on clean (Acc) and adversarial (Acc_{adv}) examples of models trained with loss functions $\mathcal{L}_i^{AT,low}$ ($\lambda_2 = 0$), $\mathcal{L}_i^{AT,high}$ ($\lambda_1 = 0$) and $\mathcal{L}_{i,j}^{AT,freq}$.

Loss $\mathcal{L}_{AT}$: $Acc = 0.84$, $Acc_{adv} = 0.55$				
Loss $\mathcal{L}_i^{AT,low}$				
$i =$	*2*	*4*	*8*	*10*
Acc	0.80	0.79	0.81	0.82
Acc_{adv}	0.59	0.60	0.57	0.56
Loss $\mathcal{L}_i^{AT,high}$				
$i =$	*4*	***6***	*8*	*10*
Acc	0.84	**0.84**	0.85	0.83
Acc_{adv}	0.58	**0.6**	0.55	0.60
Loss $\mathcal{L}_{i,j}^{AT,freq}$				
$(i,j) =$	*(10,4)*	*(2,10)*	*(4,8)*	*(8,6)*
Acc	0.84	0.8	0.81	0.82
Acc_{adv}	0.58	0.61	0.58	0.6

Table 5.7: SVHN. Accuracy on clean (Acc) and adversarial (Acc_{adv}) examples of models trained with loss functions $\mathcal{L}_i^{AT,low}$ ($\lambda_2 = 0$), $\mathcal{L}_i^{AT,high}$ ($\lambda_1 = 0$) and $\mathcal{L}_{i,j}^{AT,freq}$.

Loss $\mathcal{L}_{AT}$: $Acc = 0.93$, $Ac_{adv} = 0.54$				
Loss $\mathcal{L}_i^{AT,low}$				
$i =$	*2*	*4*	*8*	*10*
Acc	0.87	0.9	0.90	0.90
Acc_{adv}	0.59	0.59	0.60	0.61
Loss $\mathcal{L}_i^{AT,high}$				
$i =$	*4*	*6*	*8*	*10*
Acc	0.87	0.8	0.88	0.87
Acc_{adv}	0.56	0.58	0.57	0.57
Loss $\mathcal{L}_{i,j}^{AT,freq}$				
$(i,j) =$	***(10,4)***	*(2,10)*	*(4,8)*	*(8,6)*
Acc	**0.92**	0.92	0.91	0.91
Acc_{adv}	**0.66**	0.6	0.64	0.66

We investigated through experiments the robustness of models enforced during training to leverage information contained in specific frequency ranges, based on the result from cognitive psychology that humans predominantly rely on low-frequency information to make their decisions. Importantly, we found that models constrained to rely on useful information for the LSF task do not necessarily show robustness against adversarial examples. We found that this is the case only for models relying predominantly on useful features for the LSF task, and with a non-sensitivity to high frequency noise. Interestingly, when the information encompassed in the images is spread over the whole frequency spectrum, a constraint spanning a wide frequency spectrum is a viable solution. Moreover, the efficiency of frequency-based regularization when combined with existing defense schemes is strongly dependent of the nature of these schemes.

These experiences as well as the conclusions of previous efforts in neural computation and cognitive psychology highlight the fact that the intrinsic frequency characteristics of data must be necessarily considered when designing robust defense strategies against integrity-based attacks of

supervised models.

In this chapter, we tackled the intrinsic nature of adversarial examples, and enhanced the understanding of robustness by leveraging frequency properties of the data. In the next chapters, we will be interested in understanding which factors can influence adversarial attacks. Notably, a big part of this work will tackle the powerful transfer attacks for an adversary in the black-box setting. In chapter 6, we study the impact of model quantization on the strength of adversarial examples and their transferability. In chapter 7, we explicitly design a method to thwart black-box transfer attacks.

Future Work. In terms of leveraging frequency property to increase the robustness of a model, we identify two possible improvements. The first one is related to the frequency filtering method used. Notably, one could perform experiments with smoother filtering methods than the one used in this work, e.g. gaussian filtering. The second one is related to the combination of frequency constraints with Adversarial Training. Better understanding the interactions between the constraints and the natural biasing of Adversarial Training towards LSF could allow to derive rules about which constraint to choose.

Chapter 6

Quantization for robustness and transferability for quantized models

In the previous chapter, we gained insight on the link between data frequency properties and the vulnerability of models to adversarial perturbations. We studied the frequency nature of adversarial perturbations, and investigated how to bring robustness to models by studying frequency properties specific to the data at stake.

In this chapter, we investigate another crucial aspect regarding the robustness of neural networks, which has to do with the growing will to deploy neural networks models on embedded systems. In fact, with this intent comes memory footprint and energy consumption issues related to the devices on which these models are implemented. Finding lighter solutions to store neural networks has therefore become a major research topic. Consequently, some methods such as model quantization, coupled with efficient inference methods, are becoming increasingly used in deep learning. In this chapter, we study how quantization schemes affects adversarial robustness.

More precisely, in this chapter, we investigate the adversarial robustness of quantized neural networks under different threat models. Contrary to previous conclusions in the literature, we find that quantization does not offer any protection, and results in a severe form of gradient masking [78]. To explain the source of this gradient masking issue, we advance some hypotheses related notably to theoretical concerns of the quantization methods used. Moreover, we study how quantization may impact transferability, which is at the core of this thesis. We experimentally observe poor transferability capacities between models with different quantization levels (number of bits used), which we explain by the *quantization value shift* phenomenon and gradient misalignment. Eventually, we explore how these results can be exploited with a defense based on an ensemble of models.

Related publication proposed during the thesis:

- [188] Bernhard, R., Moellic, P. A., Dutertre, J.-M. *Impact of low-bitwidth quantization on the adversarial robustness for embedded neural networks.* IEEE International Conference on Cyberworlds (CW). 2019.

6.1 Motivation

The outstanding performances of neural networks in various domains have been allowed – among others – by the tremendous computation power offered by the popularization of GPU. Also, the resulting trained architectures come with thousands or even millions parameters. As the desire to run pre-trained neural network based application (e.g image recognition) on embedded or mobile systems grows, one must investigate the ways to solve the practical issues involved. Firstly, inference cost in terms of energy is critical for devices like mobile phones or a large variety of connected objects (e.g., industrial sensors). Secondly, inference speed is necessary to avoid critical latency issues. Last but not least, the memory footprint can quickly be a limiting factor for constrained devices. For example, a typical ARM Cortex-M7-based microcontroller such as STM32H7 has 2 MByte of Flash memory and 512 KByte of SRAM memory[1]. A. Capotondi and M. Rusci (University of Bologna[2]) illustrate (Figure 6.1) how different MobileNet (v1) models can fit in a STM32H7. Typically, most of the MobileNet architectures cannot fit the memory constraints of a STM32H7 with 32-bit floating point parameters. With the 160-MobileNet-0.25 a device should have at least $0.47 * 4 = 1.88$ MBytes of Flash memory to only store the model parameters. This issue raises the question of the necessity of full-precision parameters for DNN models. Figure 6.1 (right) show that with a 8-bit integer representation of the parameters, seven MobileNet architectures fit the memory constraints (red rectangle).

On a software related side, some APIs like the Android Neural Network API (NNAPI[3]) have been already developed to allow to run efficiently trained models with famous frameworks (TensorFlow[4], etc.) on Android systems. Tensorflow Lite (TFLite[5]) allows to transfer pre-trained model to mobile or embedded devices thanks to model compression techniques and 8-bit post-training weights quantization. ARM-NN[6] is another SDK that does the link for applications between various machine learning frameworks and diverse Cortex CPUs or Mali GPUs types (note that CMSIS-NN[7] is dedicated to Cortex-M MCU). STMicroelectronics also proposes an AI expansion

[1] https://www.st.com/en/microcontrollers-microprocessors/stm32h7-series.html
[2] Source: `https://github.com/EEESlab/mobilenet_v1_stm32_cmsis_nn`
[3] https://developer.android.com/ndk/guides/neuralnetworks
[4] https://www.tensorflow.org/
[5] https://www.tensorflow.org/lite
[6] https://developer.arm.com/products/processors/machine-learning/arm-nn
[7] https://github.com/ARM-software/CMSIS_5

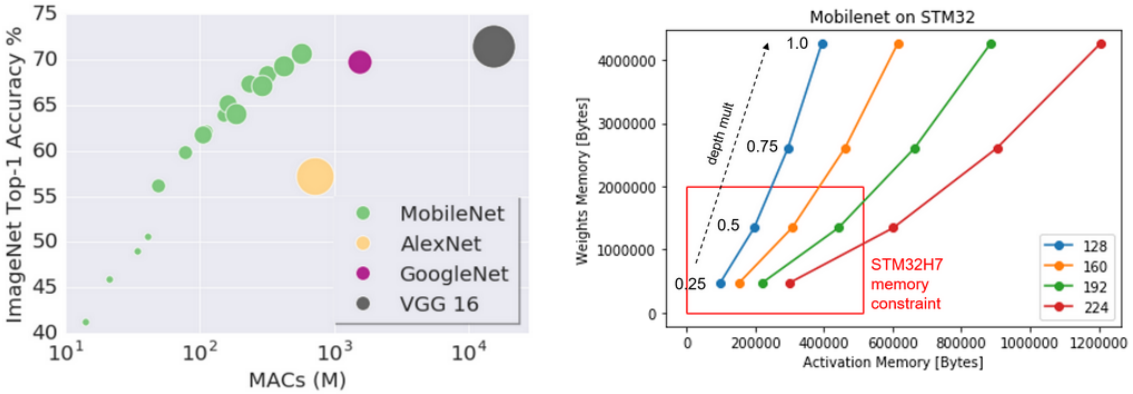

Figure 6.1: Left: MAC complexity of several state-of-the-art models. MAC = Multiply-accumulate operations: the basic linear operation between the input and the weights (and bias) of a layer ($x \leftarrow x + w.x$). Right: The Flash and SRAM memory consumption for different MobileNet (v1) models with 8-bit weights. The memory requirements for a STM32H7 are represented with the red rectangle (parameters for Flash and activation computations for SRAM).

pack for the STM32CubeMX[8], to map pre-trained neural network models into different STM32 microcontroller series thanks to 8-bit post-training quantization and other optimization tricks related to the specificity of these platforms.

On a more theoretical side, the scientific literature has investigated two different paths to address the memory footprint, inference speed and energy consumption. On one hand, methods to directly reduce the number of parameters have been explored [189–193]. These methods are related mostly to pruning. On another hand, quantization schemes have been developed [194–203]. These schemes reduce the memory footprint and are often coupled with efficient computation methods, which allow to reduce inference time and thus energy consumption.

In terms of security, as embedded systems with neural networks models become ubiquitous, it is a particularly interesting topic to evaluate the robustness of state-of-the-art quantization methods under different threat models. Moreover, studying the transferability of adversarial examples between original (i.e. full precision) and quantized neural networks may highlight weaknesses or strengths of future embedded systems. This may allow to better understand if quantization in itself could be a relevant defense against adversarial examples or, on the contrary, exacerbates these flaws.

In this chapter, we study the robustness of natural and quantized models against adversarial examples under different threat models and for various types of attacks. Our key contributions are:

- We show experimentally that quantization in itself offers poor protection against various adversarial attack methods. More precisely, we explain why activation quantization can lead to severe gradient masking [78]. This phenomenon leads to non-useful gradients to craft adversarial examples, and therefore leads to a false sense of security.
- We show very poor transferability capacities of adversarial examples between full-precision and quantized models and between quantized models with different bitwidths. In order to explain it, we advance many hypothesis, including a *quantization shift* phenomenon and gradient misalignment.
- Based on the transferability results, we investigate a defense based on an ensemble of models with different quantization levels. The results are promising and pave the way towards a possible perspective for future work.

This chapter is organized as follows. After reviewing some common quantization methods for neural networks in Section 6.2, we position our work in relation to the state-of-the-art in Section 6.3. In Section 6.4, we bring details about the experiments which will be performed throughout this work. Notably, we specify the threat models studied, the data sets and the attacks used. In Section 6.5 we present results for the evaluation of the robustness of quantized models for an adversary in a white-box or a black-box setting using gradient-based and score-based attacks. In Section 6.6, we detail the results for the transferability of adversarial perturbations between quantized models.

[8]https://www.st.com/en/embedded-software/x-cube-ai.html

Eventually, in Section 6.7, we take advantage of the previous transferability results to develop a defense based on an ensemble of quantized models.

6.2 Quantization of neural networks

During inference, energy consumption grows with memory access, which itself grows with memory footprint. Quantization is one of the existing methods to reduce memory footprint and increase inference speed, and other approaches are extensively studied to compress as well as to speed up inference. Notably, reducing the number of parameters has been reasonably investigated. As an example, Denil *et al.* [189] show, for specific architectures and data sets, that some of the parameters are predictable from the others. Han *et al.* [192] develop a three-step method (pruning, clustering, tuning) to efficiently compress a neural network, achieving a reduction of AlexNet memory footprint by a factor of 35. Hacene *et al.* [190] propose a method to reduce the memory size of a convolutional neural network by pruning connections based on a deterministic rule. This method is also coupled with weight binarization (see [194]) and an efficient hardware architecture on a FPGA in order to increase inference speed.

Quantization schemes are based on the rationale of mapping floating point weight values (usually 32 bits) to a precise format (e.g. 4 bits), to reduce memory footprint. Moreover, This is often coupled with efficient computation methods, which take advantage of this precise format, and allow for a reduced inference time. Considering the various advantages of quantization schemes, this is an important field of investigation. Quantization can be performed as a post-training process or during training.

6.2.1 Post-training quantization

For post-training quantization, as previously described in introduction, several tools have been recently proposed to map full precision pre-trained models for inference purpose (TFLite, ARM-NN, STMCubeMX AI) by coarsely quantizing some weights into – usually – no more than 8-bit integers. More advanced methods propose clustering methods [193] or information theoretical vector quantization methods (inspired by the work from Denil *et al.* [189]) such as the work from Gong *et al.* [191], which achieves about 20 times compression of the model with only 1% loss of classification accuracy on the Imagenet benchmark.

6.2.2 Quantization aware training

Quantization at training time consists in training models to perform inference with weight and/or activation quantized. These approaches enable to reach state-of-the-art performance with lower bitwidth precision. Hereunder, we detail some of the most popular works on that field that we consider for our experimentations.

Binary Connect and Binary Net.

Courbariaux *et al.* [194] present a method to train neural networks with weights w binarized to $w_b \in \{-1, 1\}$. During training, weights are binarized for the forward pass, and as the binarization operation can be not differentiable or lead to the vanishing gradient problem, a *Straight Through Estimator* (STE) is used. A STE is a common technique to compute gradients when quantization operations lead to differentiability issues. In this case, the STE given in Equation 6.1 is used for the backward pass:

$$\frac{\partial \mathcal{C}(w)}{\partial w} \approx \left.\frac{\partial C}{\partial w}\right|_{w=w_b} \mathbf{1}_{|w|\leq 1} \tag{6.1}$$

Where C is the cost function used for training . Courbariaux *et al.* [195] pursue this idea by training binary networks (BNN) with weight values w and activation function a^k values binarized to $(w_b, a_b^k) \in \{-1, 1\}^2$. During the backward pass, the authors used the same STE principle for activations as in Equation 6.1 above. Darabi *et al.* [204] propose an improvement of this method by adding regularization and a more complex approximation of the derivative on the backward pass.

Xnor Net.

Rastegari *et al.* [197] binarize weigths and activation values no more to $\{-1, 1\}$ but to $\{-\alpha, \alpha\}$ with $\alpha \in \mathbb{R}^{*,+}$. They formalize the search of the best binarization approximation of the real-valued weights as the following optimization problem.:

$$\underset{B,\alpha}{\operatorname{argmin}} \|W - \alpha B\|_2 \tag{6.2}$$

Where W is the weight matrix and B is a matrix with only -1 and 1. During the backward pass, a STE is used.

Ternarization.

Li *et al.* [198] propose a method to train a neural network with weight values ternarized to $\{-1, 0, 1\}$ during the forward pass. Zhu *et al.* [199] also propose a method to train a neural network with weight values ternarized to $\{-\alpha_l, 0, \alpha_u\}$ during the forward pass, where (α_l, α_u) $\mathbb{R}^2$, $\alpha_l > 0$, $\alpha_u > 0$, and α_l and α_u are learnable parameters updated during training.

Low bitwidth quantization.

Gupta *et al.* [200] propose to limit the bitwidth of the weights values to 16 bits and to use stochastic rounding. Dorefa *et al.* [201] propose a method to train neural networks with low-bitwidth weight values, gradients and activation function values. They claim that taking advantage of this technique during the forward pass could help speed up the training of neural network on resource-limited hardware, and naturally speed up the inference. More precisely, for a real value $x \in [0, 1]$ and a number of bits n, the quantization of x to n bits is given in Equation 6.3.

$$Q(x, n) = \frac{round((2^n - 1)x)}{2^n - 1} \tag{6.3}$$

This quantization function is used for weights, activation values and gradients. The weight and activation values are quantized on the forward pass only. The authors also found that quantizing gradients on the backward pass (with a STE) was requisite. Moreover, this quantization format allows to implement en efficient inference procedure. Indeed, $Q(x, n)(2^n - 1)$ results in 2^n values, each of them representable with a n bits integer. During the forward pass, one can thus take advantage of a bit convolution kernel method (see [201] for details) with respect to the $Q(x, n)(2^n - 1)$ values, and then scale afterwards with the $2^n - 1$ value.

Ding *et al.* [202] propose another weight quantization method with a constraint on the number of "1" in the binary representation of weights, along with some efficient computation method.

Polino *et al.* [203] propose a method which involves distillation and quantization of the weight values to decrease the storage size of a model. For the quantization part, given some weight value w, it is first scaled to a value v in $[0, 1]$, then mapped with a function Q to the nearest mapping points among the $s + 1$ points in $[0, 1]$ and then scaled back to the original scale.

Throughout this chapter, we will use the Binary Net method and Binary Connect respectively from [195] and [194], and the quantization method (Dorefa-Net) from [201].

6.3 Robustness and quantization

For neural networks, some authors have already investigated the link between quantization schemes and robustness against adversarial examples. Galloway *et al.* [205] claim that neural networks trained with weights and activation values binarized to $\{-1, 1\}$ have an interesting robustness against adversarial examples. However, this robustness was only demonstrated for the MNIST data set, and use stochastic quantization. This quantization scheme induces the stochastic gradient phenomenon [78], which can mislead to the true efficiency of this defense by causing what Uesato *et al.* [79] called *obscurity*.

Lin *et al.* [206] bring some hypotheses about the weaknesses of quantization-based defense methods against adversarial examples. They show experimentally that these defense methods can, in fact, denoise an adversarial example or enlarge its perturbation, depending on the size of the perturbation in the input space and the number of bits used for quantization. Thus, quantization can participate in an error amplification or attenuation effect. However, they only apply the FGSM attack [73] in a white-box setting against simple activation quantization. Although focused on model compression (pruning), Zhao *et al.* [207] studied the robustness of quantized neural networks

against adversarial examples with a fixed-point quantization scheme. However, this scheme does not differentiate between weight and activation quantization, does not consider less than 4 bits. Moreover, the experiments are limited to a restricted set of gradient-based attacks.

Rakin *et al.* [208] propose a defense method based on activation quantization coupled with Adversarial Training [75]. However, this defense scheme has been later shown by Lin *et al.* [206] to introduce gradient masking.

Interestingly, Khalil *et al.* [209] note that gradients derived via the use of a *Straight Through Estimator* may not be representative of the true gradients. This observation leads to questions about the efficiency of gradient-based attacks against quantized neural networks, and strengthens up the motivation to study gradient masking issues and black-box attacks against such models. The authors propose a Mixed Integer Linear Programming (MILP) based attack, which shows good results on the MNIST data set but is not scalable to large neural networks, due to computation cost issues.

6.4 Evaluating the Robustness of Quantized Models

In this section, we present all the details of the experiments we carried out to evaluate the robustness of quantized neural networks.

6.4.1 Threat model

We study the impact of quantization on robustness for two different threat models. First, we consider a complete white-box setting. The adversary has therefore a full access to the target model, and can use gradient-based methods to attack it. Second, we consider a black-box setting where the adversary knows the architecture of the target model, and is allowed to retrieve the logits $h(x)$ for an input x.

In this black-box setting, we will look at two attack schemes. On one hand, the adversary will attack the target model with score-based methods (detailed hereafter). On another hand, the adversary will leverage the transferability phenomenon to attack the target model. The substitute model and the target model having the same architecture guarantees the worst-case transferability scenario for the defender [210]. This is also a realistic scenario as we use classical image collections. Considering the global context of embedded neural networks for inference, numerous *popular* and proven architectures (such as the ResNet networks) are directly applied for a large scope of applications. Then, a scenario where an adversary craft malicious inputs from a known full precision models to attack an optimized (i.e. quantized) model with the same architecture is a realistic scenario.

Attacking the target model with gradient-based and score-based methods will provide some insights on the robustness of quantized neural networks. Notably, it will allow to identify some gradient masking issues. Attacking the target model by leveraging transferability will allow to study the transferability of adversarial perturbations between quantized models. As for the adversary capacity, we consider a perturbation budget defined with the l_∞ norm, and set it to 0.03 for the two data sets.

However, we must highlight an important characteristic of the threat models when dealing with an attacker who aims to target an embedded machine learning model. In that case, both the *architecture* of the model itself *and* its *implementation* are important. That means we need to consider a twofold white/black box paradigm: on one hand, the adversary can have – classically – full or no knowledge of the model architecture (abstraction level), and, on the other hand, he may also have full or no knowledge of the model implementation within the *target device* (physical level).

As previously said, in this work and more particularly in the section dealing with the use of quantized networks as a defense mechanism, we mainly focus on a threat model where an attacker has a white-box access to the model architecture but not for its implementation in the target device. Then, the most natural scenario corresponds to an attacker that tries to directly transfer the adversarial examples crafted from a full precision model to the embedded system.

We are conscious of the limitation of such a scenario since, obviously, the attacker may guess – thanks to information about the hardware platform (i.e. memory, precision constraints, etc.) – relevant optimization methods applied to the model (weights and activations quantization, pruning, etc.). That means an advanced adversary could try to craft adversarial examples from a quantized model of its own (without knowing the quantization method used for the target device neither if additional optimizations have been performed).

6.4.2 Data sets and models

All of the experiments throughout this work are conducted on the SVHN and CIFAR10 data sets. For each data set, we train a full-precision (32-bit floating point) neural network (hereafter called "float model" in tables), and various quantized neural networks.

More precisely, for each data set, the neural network architecture is based on the one presented in [195]. It consists of convolutional blocks, each of them being the stack of a convolution layer, a batch-normalization layer and the ReLU activation function, followed by dense blocks being a stack of a dense layer, a batch-normalization layer and the ReLU activation function. At the top of the network, we chose a dense layer with the softmax activation function, contrary to [195], where there is no activation function but a final batch normalization layer. Contrary to [195] where the hinge-loss is used, we use the cross-entropy loss as we found it to converge faster. The optimization is done with Adam [211], using a staircase decay for the learning rate. Models architecture are detailed in Table 6.1.

Table 6.1: SVHN and CIFAR10 classifier architecture. BN, MaxPool and Conv denote respectively batch normalization, max pooling and convolution.

Layer type	CIFAR10	SVHN
Conv + BN+ ReLU	(128,3,3)	(128,3,3)
Conv + MaxPool + BN + ReLU	(128,3,3), (2,2)	(128,3,3), (2,2)
Conv + BN + ReLU	(256,3,3)	(256,3,3)
Conv + MaxPool + BN + ReLU	(256,3,3), (2,2)	(128,3,3), (2,2)
Conv + BN + ReLU	(512,3,3)	(512,3,3)
Conv + MaxPool + BN + ReLU	(512,3,3), (2,2)	(512,3,3), (2,2)
Dense + BN + ReLU	1024, (2,2)	1024, (2,2)
Dense + BN + ReLU	1024, (2,2)	1024, (2,2)
Dense + softmax	10	10

For the quantized models, four different quantization bitwidth are considered: 1,2,3 and 4 bits. For each bitwidth, we consider quantization on the weights only (weight quantization) or the weights and the output of each convolutional or dense block (full quantization). For the weight binarization, the full binarization and the 2,3,4-bit quantization, we use respectively the Binary Connect method [194], the Binary Net method [195] and the Dorefa Net method [201] described in Section 6.2.2. The input layer and the last dense layer are never quantized, to allow an efficient training [195].

The accuracy of each model on the test sets is presented in Table 6.2. Quantization does not affect significantly the accuracy, except for fully binarized models which achieve only 0.79 and 0.89 accuracy on CIFAR10 and SVHN respectively, which represents a non negligible drop of performance. For quantized models with more than 1 bit, the test set accuracy is comparable to the one obtained for full-precision models. These results are consistent with [195] and [201]. Note that in [195] the authors explain the performance of binarized networks with a regularization effect brought by quantization and [201] show that the architecture as well as the size of the data set can have an impact on the performance of quantized networks.

Table 6.2: Models accuracy on test sets. Full quantization means that both weights and activation values are quantized.

	CIFAR10				SVHN			
Full-precision	0.89				0.96			
Bitwidth	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Full quantization	0.79	0.87	0.88	0.88	0.89	0.95	0.95	0.95
Weight quantization	0.88	0.88	0.88	0.88	0.96	0.95	0.96	0.95

6.4.3 Attack methods

For the white-box setting, we consider the FGSM [73], the BIM [100], and the CWl2 attack [105]. For the black-box setting the score-based attacks ZOO [127] and SPSA [79] are used. These attacks are described in Section 4.4. For the FGSM attack, the perturbation budget is set to $\epsilon = 0.03$. For the BIM attack, we set the perturbation budget to $\epsilon = 0.03$, the number of iterations to 500, and the step size to 0.008. For the CWl2, we set the number of iterations to $1,000$, the learning rate for the Adam optimizer to 0.1, the initial constant c to 0.9, 10 binary search steps. For the white-box setting, the trade-off constant κ is set to 0. When considering transferability, this parameter is tuned. More precisely, we consider $\kappa > 0$ to build adversarial examples on the source model more likely to transfer. We test $\kappa = 5, 10, 15, 20, 25$ and report the best transferability results for each source model.

For the SPSA attack, we set the perturbation budget to $\epsilon = 0.03$, the number of iterations to 500, the number of evaluation per update to 20, the learning rate for the Adam optimizer to 0.1, and the perturbation size δ for the gradient approximation to 0.01. For the ZOO attack, we set the same parameters as for the CWl2 attack. This will allow a clear comparison between the gradient-based, and score-based version of the same attack concept. Notably, this will allow to detect some gradient masking issue (see Section 6.5).

For FGSM, BIM, CWl2 and SPSA we use the Cleverhans library [212], and for ZOO we use the original code provided by the authors[9].

We sum up the main characteristics of these attacks in table 6.3.

Table 6.3: Main characteristics of the considered adversarial examples crafting methods

	FGSM	BIM	CWl2	SPSA	ZOO
Gradient-based	✓	✓	✓		
Gradient-free				✓	✓
one-step	✓				
iterative		✓	✓	✓	✓
$\boldsymbol{l_\infty}$	✓	✓		✓	
$\boldsymbol{l_2}$			✓		✓

6.4.4 Evaluation metrics

For each attack, we report two evaluation metrics (see [81] for an extended review of the adversarial robustness evaluation):

- The adversarial accuracy, which is the accuracy of the model on adversarial examples (denoted *Acc* in the result tables). The crafting method generates an adversarial example x' from each input x of the test set X. Hereafter we note X' the adversarial test set on which is computed the adversarial accuracy. The higher the adversarial accuracy, the less the model is fooled by adversarial examples, i.e. the more the model is robust against the attack. The adversarial accuracy is measured on $1,000$ clean examples well-classified by the target model.
- The average minimum-distance of the adversarial perturbation, i.e. in our case, the average l_2 norm and l_∞ norm of the difference between clean and adversarial examples which succeed to fool the target model (simply noted l_2 and l_∞ in the result tables). This quantifies the average distortion needed by the attacker to fool the model.

6.5 Robustness against gradient-based and gradient-free attacks

We start by performing adversarial robustness experiments for full-precision and quantized models with gradient-based and gradient-free attacks that may be used in a black-box setting where the gradient computation is unfeasible.

[9] https://github.com/huanzhang12/ZOO-Attack

In both case, we find that quantization does not provide reliable protection. More precisely, we notice three phenomenons:

- Quantization of activation outputs causes gradient masking. This has the consequence of hindering all gradient-based attacks and preventing some gradient-free attacks, relying on the approximation of the gradient of the output function (ZOO), to perform well.
- Properly tuning existing gradient-based attacks (CWl2) allows for an adversary in the white-box setting to efficiently attack a quantized model.
- The gradient estimation scheme of some gradient-free attacks (SPSA) overrides gradient masking and offers efficiency for an adversary in a black-box setting.

In the following experiments, all the attacks are performed on $1,000$ well-classified samples from the test set.

Results of direct attacks against fully quantized and weight-only quantized models are presented respectively in tables 6.4 and 6.5. For these tables and the following ones dealing with quantized models, first row of the results is for 1-bit model (binarized model), second row for the 2-bit model, third row for the 3-bit model and fourth row for the 4-bit model.

Table 6.4: Adversarial accuracy and distortions for gradient-based and gradient-free attacks against full-precision (32-bit) and fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.12	1.65	0.03	**0.66**	1.65	0.03	0.29	1.66	0.03	**0.78**	1.64	0.03
				0.19	1.65	0.03				0.39	1.66	0.03
				0.17	1.65	0.03				0.37	1.66	0.03
				0.18	1.65	0.03				0.4	1.66	0.03
BIM	0.0	1.17	0.03	**0.66**	1.01	0.03	0.0	1.16	0.03	**0.79**	1.0	0.03
				0.06	1.14	0.03				0.11	1.13	0.03
				0.11	1.17	0.03				0.11	1.13	0.03
				0.06	1.14	0.03				0.1	1.13	0.03
CWl2	0.0	0.58	0.02	**0.05**	0.78	0.03	0.0	0.64	0.03	**0.04**	1.02	0.03
				0.03	0.6	0.03				0.01	0.67	0.03
				0.02	0.55	0.03				0.02	0.66	0.03
				0.04	0.6	0.03				0.01	0.68	0.03
SPSA	0.0	1.37	0.03	**0.16**	1.31	0.03	0.0	1.38	0.03	**0.4**	1.32	0.03
				0.0	1.34	0.03				0.14	1.34	0.03
				0.0	1.36	0.03				0.07	1.35	0.03
				0.0	1.36	0.03				0.04	1.37	0.03
ZOO	0.0	0.72	0.03	0.56	0.1	0.03	0.0	0.91	0.03	0.82	0.07	0.03
				0.83	0.13	0.03				0.93	0.1	0.03
				0.76	0.24	0.03				**0.94**	0.11	0.03
				0.73	1.09	0.03				0.93	0.38	0.03

Robustness of binarized neural networks

A first observation from the comparison of Tables 6.4 and 6.5 is that the weight-only quantization has no impact on the robustness.

In Table 6.4, we notice that fully binarized models are far more robust to FGSM and BIM than their full-precision counterparts, as noted by Galloway *et al.* [205]. However, these models achieve only 0.79 and 0.89 accuracy on the CIFAR10 and SVHN test data sets, respectively (see table 6.2). This represents a non negligible drop of performance.

However, the CWl2 attack is almost as efficient against fully binarized neural networks as against full-precision models. This attack enables to reach 0% adversarial accuracy on a full-precision model, and only 5% adversarial accuracy against a fully-binarized model. A first conclusion is that fully-binarized neural networks do not bring much robustness improvement compared

Table 6.5: Adversarial accuracy and distortions for gradient-based and gradient-free attacks against full-precision (32-bit) and weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.12	1.65	0.03	0.11	1.65	0.03	0.29	1.66	0.03	0.28	1.66	0.03
				0.18	1.65	0.03				0.38	1.66	0.03
				0.18	1.65	0.03				**0.4**	1.66	0.03
				0.19	1.65	0.03				0.39	1.66	0.03
BIM	0.0	1.17	0.03	0.0	1.19	0.03	0.0	1.16	0.03	0.0	1.16	0.03
				0.0	1.15	0.03				0.0	1.14	0.03
				0.0	1.15	0.03				0.0	1.14	0.03
				0.0	1.15	0.03				0.0	1.13	0.03
CWl2	0.0	0.58	0.04	0.0	0.57	0.03	0.0	0.64	0.06	0.0	0.64	0.03
				0.0	0.6	0.03				0.0	0.66	0.03
				0.0	0.61	0.03				0.0	0.67	0.03
				0.0	0.62	0.03				0.0	0.68	0.03
SPSA	0.0	1.37	0.03	0.0	1.38	0.03	0.0	1.38	0.03	0.0	1.38	0.03
				0.0	1.37	0.03				0.0	1.37	0.03
				0.0	1.36	0.03				0.0	1.37	0.03
				0.0	1.36	0.03				0.0	1.37	0.03
ZOO	0.0	0.72	0.03	0.0	0.75	0.03	0.0	0.91	0.03	0.0	0.92	0.03
				0.0	0.74	0.03				0.0	0.92	0.03
				0.0	0.72	0.03				0.0	0.95	0.03
				0.0	0.73	0.03				0.0	0.93	0.03

to full-precison models against gradient-based attacks, contrary to what is claimed by Galloway *et al.* [205].

We also note that fully binarized models are relatively more robust to SPSA as well compared to full-precision models. This combined with the slightly poorer performance of CWl2 on binarized models indicates that the loss surface for binarized models is difficult to optimize over.

For full quantization with more than 1 bit, the gradient-based attacks are almost as efficient as against a full-precision model, except for FGSM on SVHN only with a 10% gain of accuracy.

Activation quantization causes gradient masking

Interestingly, we see that the ZOO attack fails to produce efficient adversarial examples a lot when attacking fully quantized neural networks. More precisely, we have two different observations:

- When adversarial examples are crafted on a full-precision model, ZOO and CWl2 reach almost the same adversarial accuracy, with slightly higher l_2 distortion for ZOO.
- When adversarial examples are crafted on a model with quantized weights and activations, we note that the adversarial accuracy is higher with ZOO than with CWl2, but that successful adversarial examples crafted with ZOO have much lower l_2 distortion than the ones crafted with CWl2 as well as the one observed for the full-precision model ($l_2 = 0.72$).

We claim that these observations reveal some form of gradient masking caused by the quantization of activation values: relying on the gradients computed thanks to the STE is not a viable solution to find adversarial examples. Firstly, the almost equal performance of ZOO compared to CWl2 for a full-precision model is expected as gradient-free attacks are supposed to perform worse than their gradient-based counterparts when no gradient masking occurs. Secondly, we argue that the phenomenon observed on full quantization models is due predominantly to gradient masking.

For the ZOO attack, we remind that the following approximation is used for the derivative of F with respect to x_i (the i^{th} element of x):

$$\nabla_{x_i} F(x) \approx \frac{F(x + he_i) - F(x - he_i)}{2h} \tag{6.4}$$

where h is set to 0.0001 and e_i is a vector with the i^{th} component set to 1, and all other components valuing 0.

We support our claim by distinguishing two cases:

- ZOO fails to produce successful adversarial examples and CWl2 succeeds: (1) because of the activation quantization, a little change (he_i) may switch the activation value from one quantization bucket to another, inducing a big change in the predicted softmax values, causing the discrete derivative $F^d_{ZOO}(x)$ to explode; (2) on the contrary, this change can also have no impact (keep values in the same bucket), which results in $F_{ZOO}(x-he_i, y) = F_{ZOO}(x+he_i, y)$, causing the discrete derivative $F^d_{ZOO}(x)$ to be null. To sum up, F_{ZOO} presents some sharp curvatures or flatness around some points, caused by activation quantization, which prevents ZOO to build successful adversarial examples. The CWl2 attack avoids this problem as it computes gradients thanks to a STE, even if the gradient computed may not be exactly the same as the true gradient [209].

- Both ZOO and CWl2 succeed to produce successful adversarial examples: the l_2 distortion for the successful adversarial examples produced by ZOO is smaller than the one produced by CWl2. Around these points, the surface of the objective function to optimize does not present any sharp curvature or flatness. However, as noted by [209], the gradients computed by the CWl2 attack are not representative of the true gradient. The gradient are therefore being better estimated by the ZOO attack, which explains the lower l_2 distortion observed.

When checking for gradient masking, a crucial step is to verify that gradient-based attacks perform worse than gradient-free ones. In our case, for the quantization of activation values, we notice that the SPSA attack (a gradient-free attack) performs quite better in terms of adversarial accuracy than the BIM attack, against fully binarized models. This indicates gradient masking issues. We also make the hypothesis that SPSA avoids the sharp curvatures or flatness observed around some points, where ZOO fails to produce adversarial examples, because of the more efficient gradient estimation method suited to noisy objective functions [79].

For the weight quantized models results presented in Table 6.5, we do not observe the same phenomenon as for the full quantization models. No apparent robustness is noticeable for the weight-quantized models. No sharp variation or flatness is induced for the objective function of the ZOO attack of the weight-quantized models, as it originated from the quantization of activation values. It has to be noted that we also measure that the variance of the logits values between full-precision and weight-only quantized models is almost the same.

6.6 Black-box transfer attacks

In this section, we perform transferability experiments between full-precision and quantized models and show poor transferability capacities, which we explain with the *quantization value shift* phenomenon and gradient misalignment.

We present results of transferability when the source network (i.e. the models the adversarial examples are crafted from) is full-precision, fully or weight-only quantized models, and the adversarial examples are transferred to (target models) full-precision, fully or weight-only quantized networks, in figures 6.2 and 6.3. The results for the cases where the source networks are 2-bit or 4-bit quantized models are not presented here for the sake of conciseness and as these results can be interpolated from the one we present. More complete tables can be found in Appendix 6.9.

Weak transferability

A first observation is that transferability results are quite poor for attacks FGSM, BIM and SPSA. The CWl2 attack, given tuning the parameter κ, suffers less from transferability issues, at the cost of increased l_2 and l_∞ distortion, except when the source or target network is a fully binarized network. Indeed, for the κ values tested (for fully binarized models, the results reported are for $\kappa = 5$), when the source network is a fully binarized model, CWl2 struggles to find adversarial examples having both misclassification and a little l_2 distortion. This results in adversarial examples being missclassified but not imperceptible by a human. Our hypothesis is that this comes from the loss function that is hard to optimize, as noted in Section 6.5. We also note that, as already noticed by Wu *et al.* [213], and contrary to what was initially found by Kurakin *et al.* [100], that BIM – as it is the case here – may produce more transferable adversarial examples than FGSM .

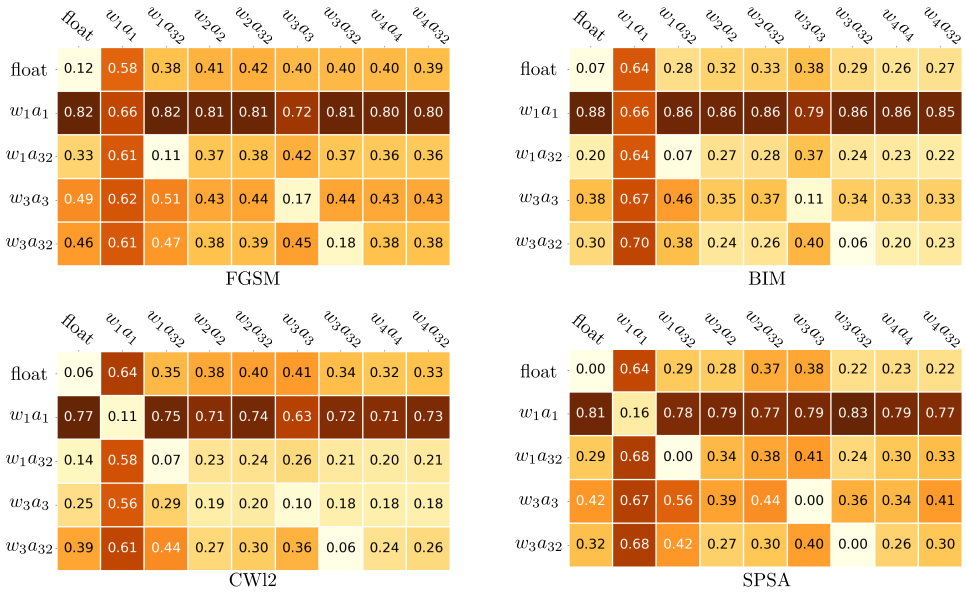

Figure 6.2: Adversarial transferability results for CIFAR10. Rows are relative to source networks and columns to target networks. Values correspond to adversarial accuracy. The lower the value, the more transferability occurs. $w_i a_j$ refers to a model where weights are quantized to i bits and activations to j bits.

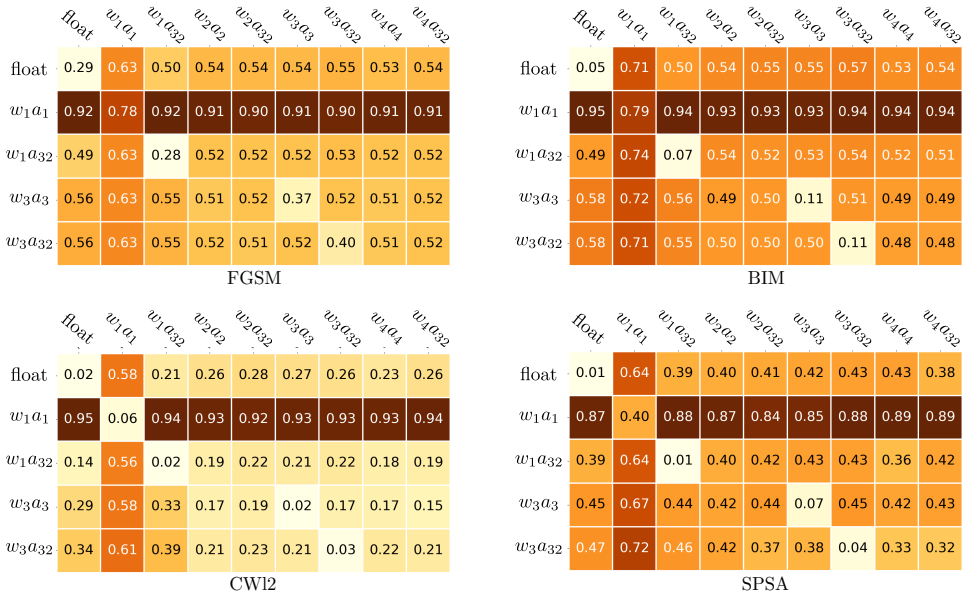

Figure 6.3: Adversarial transferability results for SVHN. Rows are relative to source network and columns to target networks. Values correspond to adversarial accuracy. The lower the value, the more transferability occurs. $w_i a_j$ refers to a model where weights are quantized to i bits and activations to j bits.

Quantization shift phenomenon

These poor transferability results can be explained by the *quantization value shift* phenomenon which takes places when quantization ruins the adversarial effect by mapping two different values to the same quantization bucket. This phenomenon is noticeable for both weight-quantized models and fully-quantized models. Hereafter, we give two simple examples of this. These examples are based on two assumptions: (1) on the source model, the adversarial perturbation is effective because

it induces some relation between two output values, and (2) this same adversarial perturbation is effective on the target model also if it causes the same two output values to share the same relation. Then, considering activation quantization or weight quantization:

- Activation quantization. The adversarial perturbation is effective because it causes two activation values to be different for the source model. In this case, on the target model, the same two activation values for the same adversarial perturbation can be mapped to the same value, because of different quantization levels between the source and target model.
- Weight quantization. The adversarial perturbation is effective because it causes two output values to have some precise order relation (e.g. one is greater than the other). In this case, two different quantization levels on the source and target models can also tamper the adversarial perturbation. In Figure 6.4, we show a toy example of the impact of weight quantization on adversarial effect: in this example, the adversarial effect is canceled.

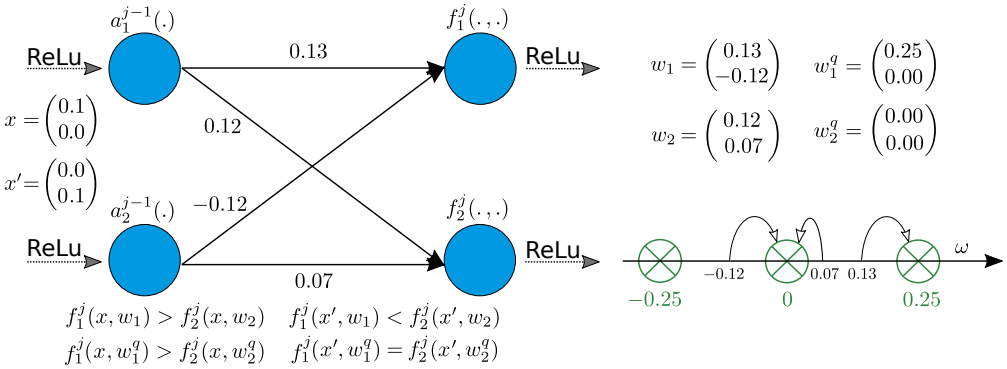

Figure 6.4: A toy example to illustrate the *quantization value shift* phenomenon. Quantization of the weights cancels the adversarial effect.

Consequently, whatever the quantization level of the source model adversarial examples are crafted on, evaluating them on a target model with a different quantization level may hinder their efficiency because of this phenomenon.

Gradient misalignment

Regarding the transferability results, we may also hypothesize that the gradient direction between float models and quantized models and between models with different bitwidths is quite different. Indeed, different gradient directions between models indicate that the way to fool the source model is different than the one it should be used to fool the target model.

This gradient misalignment may be noticeable for the gradient computed with the *Straight Through Estimator*, as poor transferability is observed for the white-box attacks (FGSM, BIM, CWl2), and for the real gradient, as poor transferability is observed for SPSA. To assess the alignment of gradients, we measure the mean cosine similarity between the gradient of the loss function with respect to the input between models with different bitwidth and show the results in Figures 6.5 and 6.6 for CIFAR10 and SVHN, respectively.

We remind that for two vectors a and b, $\text{cossim}(a, b) = 0$ indicates orthogonal vectors for the usual scalar product, $\text{cossim}(a, b) = 1$ indicates aligned vectors in the same direction and $\text{cossim}(a, b) = -1$ indicates aligned vectors in opposite directions.

In Figures 6.5 and 6.6, we first observe that the cosine similarity values for gradients of the loss function between full-precision and quantized models and between quantized models with different bitwidths, are relatively close to 0, indicating nearly orthogonal gradient directions. We observe that the cosine similarity for the gradient between fully-binarized models and others models is the closest to 0. This is in accordance for example with results presented in Fig. 6.2 where transferability capacities for FGSM, BIM, CWl2 and SPSA are the poorest when fully-binarized models are involved. Moreover, this may explain the fact that adversarial accuracy is much higher in tables 6.12 and 6.13 (Appendix 6.9), where adversarial examples are crafted on fully-binarized models, than in the other tables (see Appendix 6.9).

To conclude, transferability results show that quantization strongly alters the chances of success for an adversary who has only access to a full-precision (or quantized) version of a model and wants to attack a quantized (respectively, a full-precision) version of a model, assuming this adversary can not use a black-box attack such as the SPSA one.

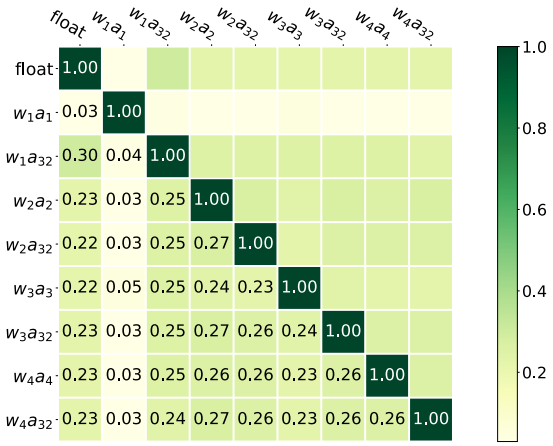

Figure 6.5: CIFAR10. Cosine similarity values between the gradient of the loss function with respect to the input (computed with the STE), for full-precision and quantized models. w_ia_j designates a model with a i-bit weight quantization and a j-bit activation quantization.

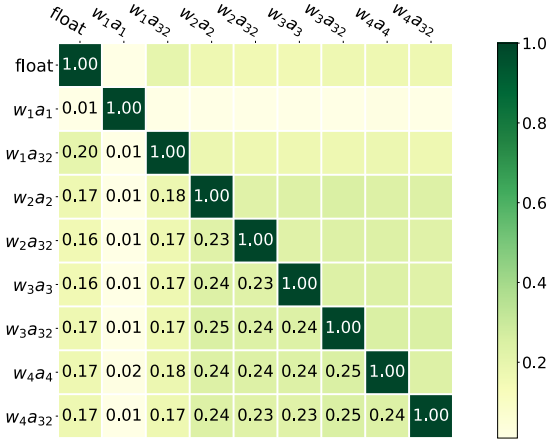

Figure 6.6: SVHN. Cosine similarity values between the gradient of the loss function with respect to the input (computed with the STE), for full-precision and quantized models. w_ia_j designates a model with a i-bit weight quantization and a j-bit activation quantization.

6.7 Ensemble of quantized models

6.7.1 Motivation

Regarding the transferability results, a logical consequence and a natural assumption is to consider an ensemble of quantized models to filter out adversarial examples. In this section, we analyze the relevance of this defense strategy.

We consider an ensemble of quantized models, $\mathcal{M} = \left(M^i\right)_{i\in[\![1,5]\!]}$ consisting in a full-precision model and four fully-quantized models (1,2,3 and 4 bits). As a first step, we analyze statistically how the models agree on clean and adversarial examples crafted using FGSM.

More precisely, we consider an adversarial example crafted on a source model M^s and we look at how the five models agree, given that this adversarial example is successful or not on M^s. Similarly, we look at how the five models agree on clean test set examples, given that this test set example is correctly classified in the true class or, on the contrary, misclassified by M^s.

In Table 6.6, we show for CIFAR10 and SVHN, given the source model M^s, how the four other models agree on test set and adversarial examples crafted with FGSM. For example, for CIFAR10, with the 2-bitwidth model, 77% of test set examples well-recognized by M^s are also correctly classified by the other four models, compared to 33% for misclassified test set examples. Moreover, 37% of successful adversarial examples (i.e. examples that effectively fool M^s) also fool the other four models, and 76% of unsuccessful adversarial examples (on M^s) are also unsuccessful on the other four models.

These results lead to the following observations:

1. The models are more likely to agree on clean samples than on adversarial examples (successful or not).
2. The models are much more likely to agree on unsuccessful (on M^s) adversarial examples.

Table 6.6: Rate of examples for which the other four models agree, depending on the source model M_s and its prediction results (*correctly classified*, *misclassified*, *successful*, *unsuccessful*).

	CIFAR10					SVHN				
	Source model									
	float	w_1a_1	w_2a_2	w_3a_3	w_4a_4	*float*	w_1a_1	w_2a_2	w_3a_3	w_4a_4
	Test set examples									
Correctly classified	0.75	0.85	0.77	0.82	0.76	0.9	0.97	0.91	0.91	0.9
Misclassified	0.39	0.19	0.33	0.33	0.34	0.46	0.16	0.38	0.37	0.4
	Adversarial examples (FGSM)									
Successful	0.31	0.09	0.37	0.31	0.33	0.33	0.09	0.41	0.43	0.39
Unsuccessful	0.88	0.80	0.76	0.78	0.79	0.89	0.94	0.81	0.82	0.84

Based on this preliminary results, we design an ensemble-based defense method for a distant system: the prediction for an upcoming example is done only if m or more models agree, and the final label is the one predicted by these m models. Considering the remarks made above, this approach encourages prediction for clean examples rather than for adversarial examples, and when the prediction is performed on adversarial examples, it would be predominantly unsuccessful ones.

More precisely, we note $valid_{m,\mathcal{M}}(X)$ the set of samples from the input data set X which respect this criterion:

$$valid_{m,\mathcal{M}}(X) = \left\{ x \in X \;\middle|\; \max_{j \in [\![1,K]\!]} \sum_i \mathbf{1}_{M^i(x)=j} \geq m \right\} \tag{6.5}$$

with K the number of labels and $M^i(x)$ the output prediction label of x by M^i.

Then, the *prediction rate* (hereafter, PR) quantifying the proportion of examples from X for which prediction is performed is :

$$PR_{m,\mathcal{M}}(X) = \frac{|valid_{m,\mathcal{M}}(X)|}{|X|} \tag{6.6}$$

Where $|X|$ is the cardinality of X.

When evaluating this defense on the adversarial test set (X'), an overall performance metric, hereafter called *defense accuracy* ($d\_\text{acc}_{m,\mathcal{M}}(X')$) is simply defined as the proportion of adversarial examples which have been filtered out or which are unsuccessful. Practically, the defense accuracy is defined as:

$$d\_\text{acc}_{m,\mathcal{M}}(X') = 1 - \frac{|valid_{m,\mathcal{M}}(X')^s|}{|X'|} \tag{6.7}$$

Where $valid_{m,\mathcal{M}}(X')^s$ denotes the set of successful adversarial examples which thwart the filtering process (i.e. the *error rate* of the defense).

Importantly, the number m has to be decided following a trade-off between the number of test set examples for which prediction is performed (which has to remain high) and the defense error rates (which has to be low) when facing adversarial examples. We experimentally set $m = 4$ for CIFAR10 and $m = 5$ for SVHN to reach a good trade-off. As presented in Table 6.7, the prediction is performed for more than 87% of the clean test set examples for both CIFAR10 and SVHN, with an accuracy of 90% and 98% respectively for CIFAR10 and SVHN on clean test set examples for which prediction is performed.

6.7.2 Results

We present the results for this defense on CIFAR10 and SVHN in table 6.8. For each source model and each attack method, we report the defense accuracy d_acc against four attacks (FGSM,

Table 6.7: Prediction rate and accuracy with the ensemble of models on CIFAR10 and SVHN.

	CIFAR10		SVHN	
	PR	Accuracy	PR	Accuracy
Test set	0.87	0.90	0.87	0.98

Table 6.8: Defense accuracy (d_acc) and adversarial prediction rate (PR) for gradient-based and gradient-free attacks against an ensemble of quantized models, depending of the source model. For example, for adversarial examples crafted with the FGSM attack on a fully binarized model, the d_acc value equals 0.9 against the ensemble of models.

	CIFAR10				SVHN			
	Float model *(32-bit)*		Quantized models *(1,2,3,4-bit)*		Float model *(32-bit)*		Quantized models *(1,2,3,4-bit)*	
	PR	d_acc	PR	d_acc	PR	d_acc	PR	d_acc
FGSM	0.58	0.63	0.73	**0.9**	0.39	0.65	0.80	**0.98**
			0.58	0.53			0.47	0.86
			0.45	0.63			0.47	0.84
			0.53	0.57			0.46	0.87
BIM	0.65	0.44	0.71	**0.88**	0.20	0.85	0.80	**0.99**
			0.62	0.38			0.28	0.81
			0.57	0.44			0.26	0.81
			0.52	0.48			0.25	0.82
CWl2	0.54	0.60	0.17	**0.84**	0.15	0.84	0.18	**0.84**
			0.47	0.53			0.23	0.77
			0.58	0.42			0.2	0.8
			0.36	0.64			0.17	0.83
SPSA	0.68	0.41	0.32	**0.82**	0.22	0.8	0.5	**0.97**
			0.48	0.54			0.32	0.82
			0.44	0.57			0.29	0.79
			0.58	0.42			0.31	0.75

BIM, CWl2 and SPSA), along with the prediction rate PR. For the CWl2 attack, we test $\kappa = 5, 10, 15, 20, 25$ and report the best transferability results.

We notice that results are interesting for adversarial examples crafted from the full precision model, especially for SVHN. Indeed, if an attacker tries to directly transfer the adversarial examples crafted from the full precision model with, for example, the CWl2 attack, 60% of the adversarial examples are harmless (i.e. filtered out or unsuccessful) for CIFAR10 and this robustness is even stronger for SVHN with a defense accuracy superior to 80% for BIM, CWl2 or SPSA. Considering an hidden system composed of a float model and quantized models, these results are particularly promising. Indeed, when mounting a black-box transfer attack, the adversary is much more likely to use a float model as for the substitute model than a quantized version of it.

Moreover, coherently with the transferability results (see Figures 6.2, 6.3 and additional tables in Appendix 6.9), the highest robustness is reached when adversarial examples are crafted from a fully binarized network with a defense accuracy superior to 0.8 – whatever the crafting method – particularly for SVHN. However, there is no significant gain compare to the transferability results obtained by taking each quantized model separately (see figures 6.2 and 6.3). But, except for this case of the fully binarized network, the ensemble of quantized models shows better robustness to transferred adversarial examples than all single models. The more relevant gain is reached with CWl2 attack with a mean (over the 2, 3 and 4-bitwidth networks) defense accuracy of 0.53 and 0.8 respectively for CIFAR10 and SVHN.

6.8 Conclusion and future work

In this chapter, we showed experimentally that quantization in itself offers very poor protection against adversarial examples crafted in a white-box or black-box setting, contrary to what was previously found in the literature. Notably, we found that the quantization of the output of activation functions can lead to gradient masking. In fact, we verified experimentally that the efficiency of some gradient-based and gradient-free attacks can be tampered but other gradient-based or gradient-free attacks do not suffer from gradient masking. As the cause of these issues, we identified the usage of a STE (Straight Through Estimator) to approximate gradients, or the optimization procedure being non well-suited for noisy functions. Eventually, we demonstrate poor transferability capacities between classical models and quantized models, and between quantized models with different bitwidths. We explain this by the *quantization shift phenomenon* which ruins adversarial effects, and gradients misalignment.

As an exploratory work and logical consequence of the transferability results, we analyzed the impact of considering an ensemble of quantized models in order to filter out adversarial examples with a minimum impact on the standard accuracy. Such an ensemble method, like any other detection-based approach, suffers from a narrow threat model since the defense is useless with an attacker aware of the implementation details of the model in the target device [76,82]. However, for black-box paradigms, the use of an ensemble of quantized models may have an interesting impact on the transferability when associated to other and complementary defense mechanisms.

As an important outcome of this chapter, we believe that the characteristics of embedded models, particularly those induced by quantization approaches (weights or activation outputs), have to be taken into consideration in order to design suitable and efficient protection schemes. Indeed, robustness requirements will obviously become more and more compulsory as critical tasks (as well as processed data) will be performed thanks to a growing variety of devices.

In this chapter, we have studied a particular case of the transferability of adversarial perturbations, for quantized models. Notably, we investigated how quantized models can be used in an ensemble to thwart an adversary leveraging black-box transfer attacks. Black-box transfer attacks, because of their efficiency in the realistic black-box setting, are the core of this thesis. Therefore, we will focus in the next chapter on studying more in-depth this phenomenon. More precisely, in chapter 7, we will design an innovative way of thwarting the transferability of adversarial perturbations between specific models in various different real-life use-cases.

Future Work. We identify a room for improvement for the ensemble defense method, in the diversity of quantization methods used. Indeed, the robustness of the ensemble of quantized models is due to the poor transferability of adversarial perturbations between models, caused by the misalignement of gradients. For a bidwidth of more than than 1 bit, the same quantization method is used (Dorefa-Net), with the same STE. Therefore, investigating different quantization methods, with diverse STE could increase the gradient misalignment, and lead to a robustness increase.

6.9 Appendix: Complete transferability results

In the following tables, "–" denotes a value which can not be computed. For example, the l_2 distortion of successful adversarial examples for an attack can not be computed when the adversarial accuracy of the target model against this attack equals 1.

We summarize the reference of the transferability tables where $w_i a_j$ designates a model with a i-bit quantization of the weights and a j-bit quantization of the activation values.

Table 6.9: Summary of the references for the transferability results between full precision, fully quantized and weight only models.

From \ To	full	w_1a_1	w_2a_2	w_3a_3	w_4a_4	w_1a_{32}	w_2a_{32}	w_3a_{32}	w_4a_{32}
full	Table 6.10					Table 6.11			
w_1a_1	Table 6.12					Table 6.13			
w_1a_{32}	Table 6.14					Table 6.15			
w_2a_2	Table 6.16					Table 6.17			
w_2a_{32}	Table 6.18					Table 6.19			
w_3a_3	Table 6.20					Table 6.21			
w_3a_{32}	Table 6.22					Table 6.23			
w_4a_4	Table 6.24					Table 6.25			
w_4a_{32}	Table 6.26					Table 6.27			

Table 6.10: Transferability from full-precision model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.12	1.65	0.03	0.58	1.65	0.03	0.29	1.66	0.03	0.63	1.66	0.03
				0.41	1.65	0.03				0.54	1.66	0.03
				0.4	1.65	0.03				0.54	1.66	0.03
				0.4	1.65	0.03				0.53	1.66	0.03
BIM	0.00	1.17	0.03	0.64	1.18	0.03	0.0	1.16	0.03	0.71	1.16	0.03
				0.32	1.17	0.03				0.54	1.16	0.03
				0.38	1.18	0.03				0.55	1.16	0.03
				0.26	1.17	0.03				0.53	1.16	0.03
CWl2	0.03	0.58	0.04	0.64	0.80	0.07	0.02	0.64	0.06	0.59	0.88	0.1
				0.38	0.82	0.07				0.26	0.91	0.11
				0.41	0.82	0.07				0.27	0.91	0.11
				0.32	0.82	0.07				0.23	0.92	0.11
SPSA	0.0	1.37	0.03	0.64	1.37	0.03	0.0	1.38	0.03	0.64	1.38	0.03
				0.28	1.37	0.03				0.4	1.38	0.03
				0.38	1.37	0.03				0.42	1.38	0.03
				0.23	1.37	0.03				0.43	1.38	0.03
ZOO	0.0	0.72	0.03	0.77	0.56	0.03	0.0	0.91	0.03	0.86	0.68	0.03
				0.84	0.55	0.03				0.91	0.63	0.03
				0.76	0.63	0.03				0.91	0.65	0.03
				0.83	0.6	0.03				0.92	0.67	0.03

Table 6.11: Transferability from full-precision model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.12	1.65	0.03	0.38	1.65	0.03	0.29	1.66	0.03	0.5	1.66	0.03
				0.42	1.65	0.03				0.54	1.66	0.03
				0.4	1.65	0.03				0.55	1.66	0.03
				0.39	1.65	0.03				0.54	1.66	0.03
BIM	0.0	1.17	0.03	0.28	1.17	0.03	0.0	1.16	0.03	0.5	1.16	0.03
				0.33	1.17	0.03				0.55	1.16	0.03
				0.29	1.18	0.03				0.57	1.15	0.03
				0.27	1.17	0.03				0.54	1.16	0.03
CWl2	0.03	0.58	0.04	0.35	0.82	0.07	0.02	0.64	0.06	0.21	0.92	0.11
				0.4	0.82	0.07				0.28	0.91	0.11
				0.34	0.82	0.07				0.26	0.91	0.11
				0.32	0.82	0.07				0.26	0.91	0.11
SPSA	0.0	1.37	0.03	0.29	1.37	0.03	0.0	1.38	0.03	0.39	1.38	0.03
				0.37	1.37	0.03				0.41	1.38	0.03
				0.22	1.37	0.03				0.43	1.38	0.03
				0.22	1.37	0.03				0.38	1.38	0.03
ZOO	0.0	0.72	0.09	0.84	0.56	0.08	0.0	0.91	0.11	0.93	0.51	0.08
				0.83	0.56	0.08				0.9	0.62	0.09
				0.84	0.61	0.09				0.92	0.69	0.1
				0.83	0.58	0.09				0.92	0.64	0.1

Table 6.12: Transferability from fully binarized model to 1,2,3,4-bit fully quantized models and full-precision models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.82	1.65	0.03	0.66	1.65	0.03	0.92	1.64	0.03	0.78	1.64	0.03
				0.81	1.65	0.03				0.91	1.64	0.03
				0.72	1.65	0.03				0.91	1.64	0.03
				0.8	1.65	0.03				0.91	1.64	0.03
BIM	0.88	1.01	0.03	0.66	1.01	0.03	0.95	1.01	0.03	0.79	1.0	0.03
				0.86	1.01	0.03				0.93	1.0	0.03
				0.79	1.02	0.03				0.93	1.0	0.03
				0.86	1.01	0.03				0.94	1.0	0.03
CWl2	0.77	1.95	0.23	0.11	0.78	0.08	0.95	1.28	0.13	0.06	1.02	0.1
				0.71	2.22	0.21				0.93	0.94	0.09
				0.63	2.22	0.21				0.93	0.79	0.08
				0.71	2.3	0.22				0.93	1.29	0.11
SPSA	0.81	1.31	0.03	0.16	1.31	0.03	0.87	1.32	0.03	0.4	1.32	0.03
				0.79	1.32	0.03				0.87	1.32	0.03
				0.76	1.31	0.03				0.85	1.32	0.03
				0.79	1.32	0.03				0.89	1.33	0.03
ZOO	1.00	–	–	0.56	0.1	0.05	1.00	–	–	0.82	0.07	0.05
				0.88	0.1	0.06				0.95	0.09	0.08
				0.82	0.12	0.06				0.94	0.06	0.04
				0.88	0.24	0.1				1.00	–	–

Table 6.13: Transferability from fully binarized model to 1,2,3,4-bit weight-only quantized models and full-precision models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.82	1.65	0.03	0.82	1.65	0.03	0.92	1.64	0.03	0.92	1.64	0.03
				0.81	1.65	0.03				0.9	1.64	0.03
				0.81	1.65	0.03				0.91	1.64	0.03
				0.8	1.65	0.03				0.91	1.64	0.03
BIM	0.88	1.01	0.03	0.88	1.01	0.03	0.95	1.01	0.03	0.94	1.0	0.03
				0.86	1.01	0.03				0.93	1.01	0.03
				0.86	1.01	0.03				0.94	1.01	0.03
				0.85	1.02	0.03				0.94	1.0	0.03
CWl2	0.77	1.95	0.23	0.874	2.21	0.23	0.95	1.28	0.13	0.94	1.07	0.1
				0.74	2.22	0.22				0.92	1.02	0.1
				0.72	2.36	0.22				0.93	1.23	0.12
				0.73	2.26	0.22				0.94	1.19	0.12
SPSA	0.81	1.31	0.03	0.78	1.32	0.03	0.87	1.32	0.03	0.88	1.32	0.03
				0.77	1.32	0.03				0.84	1.32	0.03
				0.83	1.32	0.03				0.88	1.32	0.03
				0.77	1.32	0.03				0.89	1.32	0.03
ZOO	1.00	–	–	0.89	–	–	1.00	–	–	1.00	–	–
				1.00	–	–				1.00	–	–
				1.00	–	–				1.00	–	–
				0.88	0.15	0.08				1.00	–	–

Table 6.14: Transferability from weight-only binarized model to 1,2,3,4-bit fully quantized models and full-precision models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.33	1.65	0.03	0.61	1.65	0.03	0.49	1.66	0.03	0.63	1.66	0.03
				0.37	1.65	0.03				0.52	1.66	0.03
				0.42	1.65	0.03				0.52	1.66	0.03
				0.36	1.65	0.03				0.52	1.66	0.03
BIM	0.2	1.19	0.03	0.64	1.19	0.03	0.49	1.16	0.03	0.74	1.16	0.03
				0.27	1.19	0.03				0.54	1.16	0.03
				0.37	1.19	0.03				0.53	1.16	0.03
				0.23	1.19	0.03				0.52	1.16	0.03
CWl2	0.14	0.84	0.09	0.58	0.83	0.08	0.14	0.97	0.10	0.56	0.93	0.09
				0.23	0.84	0.08				0.19	0.96	0.1
				0.26	0.84	0.08				0.21	0.95	0.09
				0.20	0.84	0.08				0.19	0.95	0.09
SPSA	0.29	1.38	0.03	0.68	1.38	0.03	0.39	1.39	0.03	0.64	1.39	0.03
				0.34	1.38	0.03				0.4	1.39	0.03
				0.41	1.38	0.03				0.43	1.39	0.03
				0.3	1.38	0.03				0.36	1.39	0.03
ZOO	0.93	0.7	0.1	0.88	0.59	0.09	0.97	0.73	0.1	0.9	0.68	0.1
				0.93	0.63	0.09				0.95	0.65	0.09
				0.9	0.71	0.1				0.96	0.65	0.1
				0.94	0.65	0.09				0.96	0.64	0.09

Table 6.15: Transferability from weight-only binarized model to 1,2,3,4-bit weight-only quantized models and full-precision models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.33	1.65	0.03	0.11	1.65	0.03	0.49	1.66	0.03	0.28	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
				0.37	1.65	0.03				0.53	1.66	0.03
				0.36	1.65	0.03				0.52	1.66	0.03
BIM	0.2	1.19	0.03	0.0	1.19	0.03	0.49	1.16	0.03	0.0	1.16	0.03
				0.28	1.19	0.03				0.52	1.16	0.03
				0.24	1.19	0.03				0.54	1.16	0.03
				0.22	1.19	0.03				0.51	1.16	0.03
CWl2	0.14	0.84	0.09	0.07	0.84	0.08	0.14	0.97	0.10	0.02	1.00	0.10
				0.24	0.84	0.08				0.22	0.95	0.10
				0.21	0.84	0.09				0.22	0.95	0.10
				0.21	0.84	0.08				0.19	0.95	0.10
SPSA	0.29	1.38	0.03	0.00	1.38	0.03	0.39	1.39	0.03	0.00	1.38	0.03
				0.38	1.38	0.03				0.42	1.39	0.03
				0.24	1.38	0.03				0.43	1.39	0.03
				0.33	1.38	0.03				0.42	1.39	0.03
ZOO	0.93	0.7	0.1	0.00	0.75	0.1	0.97	0.73	0.1	0.0	0.92	0.1
				0.94	0.65	0.1				0.96	0.63	0.1
				0.94	0.62	0.09				0.97	0.7	0.1
				0.93	0.63	0.09				0.97	0.7	0.1

Table 6.16: Transferability from 2-bit fully quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.45	1.65	0.03	0.6	1.65	0.03	0.56	1.66	0.03	0.62	1.66	0.03
				0.19	1.65	0.03				0.39	1.66	0.03
				0.44	1.65	0.03				0.52	1.66	0.03
				0.37	1.65	0.03				0.51	1.66	0.03
BIM	0.3	1.14	0.03	0.68	1.14	0.03	0.55	1.12	0.03	0.73	1.13	0.03
				0.06	1.14	0.03				0.11	1.13	0.03
				0.39	1.14	0.03				0.50	1.13	0.03
				0.20	1.14	0.03				0.49	1.13	0.03
CWl2	0.29	0.93	0.08	0.54	0.90	0.08	0.36	0.65	0.09	0.66	0.61	0.08
				0.06	0.6	0.04				0.03	0.67	0.07
				0.25	0.94	0.08				0.21	0.69	0.09
				0.15	0.92	0.08				0.18	0.70	0.09
SPSA	0.57	1.34	0.03	0.72	1.33	0.03	0.55	1.34	0.03	0.68	1.33	0.03
				0.0	1.34	0.03				0.14	1.34	0.03
				0.57	1.34	0.03				0.55	1.34	0.03
				0.39	1.34	0.03				0.56	1.34	0.03
ZOO	1.0	–	–	0.98	0.13	0.06	1.0	–	–	0.99	0.12	0.06
				0.83	0.13	0.06				0.93	0.1	0.06
				0.99	0.2	0.09				1.0	–	–
				0.99	0.24	0.09				1.0	–	–

Table 6.17: Transferability from 2-bit fully quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.45	1.65	0.03	0.46	1.65	0.03	0.56	1.66	0.03	0.54	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
				0.37	1.65	0.03				0.53	1.66	0.03
				0.37	1.65	0.03				0.51	1.66	0.03
BIM	0.3	1.14	0.03	0.35	1.14	0.03	0.55	1.12	0.03	0.55	1.13	0.03
				0.25	1.14	0.03				0.50	1.12	0.03
				0.21	1.14	0.03				0.52	1.13	0.03
				0.21	1.14	0.03				0.49	1.13	0.03
CWl2	0.29	0.92	0.08	0.34	0.92	0.08	0.36	0.65	0.09	0.42	0.64	0.09
				0.19	0.93	0.08				0.22	0.7	0.09
				0.15	0.92	0.08				0.21	0.69	0.09
				0.18	0.93	0.08				0.19	0.7	0.09
SPSA	0.57	1.34	0.03	0.61	1.34	0.03	0.55	1.34	0.03	0.58	1.34	0.03
				0.44	1.34	0.03				0.56	1.33	0.03
				0.37	1.34	0.03				0.56	1.34	0.03
				0.42	1.34	0.03				0.53	1.34	0.03
ZOO	1.0	–	–	0.99	0.15	0.07	1.0	–	–	1.0	–	–
				1.0	–	–				0.98	0.08	0.04
				1.0	–	–				1.0	–	–
				1.0	–	–				1.0	–	–

Table 6.18: Transferability from 2-bit weight-only quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.47	1.65	0.03	0.61	1.65	0.03	0.57	1.66	0.03	0.64	1.66	0.03
				0.39	1.65	0.03				0.53	1.66	0.03
				0.44	1.65	0.03				0.54	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
BIM	0.34	1.15	0.03	0.68	1.14	0.03	0.58	1.13	0.03	0.72	1.13	0.03
				0.27	1.15	0.03				0.51	1.14	0.03
				0.24	1.15	0.03				0.52	1.13	0.03
				0.20	1.14	0.03				0.51	1.13	0.03
CWl2	0.32	0.90	0.08	0.6	0.86	0.07	0.31	0.84	0.10	0.57	0.81	0.1
				0.2	0.91	0.08				0.18	0.87	0.11
				0.29	0.91	0.08				0.21	0.85	0.10
				0.19	0.91	0.08				0.18	0.86	0.11
SPSA	0.41	1.36	0.03	0.64	1.36	0.03	0.49	1.37	0.03	0.59	1.36	0.03
				0.26	1.36	0.03				0.36	1.36	0.03
				0.43	1.36	0.03				0.41	1.37	0.03
				0.25	1.36	0.03				0.35	1.37	0.03
ZOO	0.96	0.56	0.08	0.86	0.57	0.08	0.97	0.69	0.1	0.90	0.66	0.09
				0.94	0.59	0.08				0.96	0.62	0.08
				0.93	0.62	0.08				0.96	0.61	0.09
				0.95	0.58	0.08				1.0	–	–

Table 6.19: Transferability from 2-bit weight-only quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.47	1.65	0.03	0.47	1.65	0.03	0.57	1.66	0.03	0.56	1.66	0.03
				0.18	1.65	0.03				0.38	1.66	0.03
				0.38	1.65	0.03				0.53	1.66	0.03
				0.39	1.65	0.03				0.53	1.66	0.03
BIM	0.34	1.15	0.03	0.39	1.15	0.03	0.58	1.13	0.03	0.57	1.13	0.03
				0.0	1.15	0.03				0.0	1.16	0.03
				0.25	1.15	0.03				0.53	1.13	0.03
				0.20	1.14	0.03				0.53	1.13	0.03
CWl2	0.32	0.90	0.07	0.38	0.89	0.07	0.31	0.84	0.10	0.35	0.84	0.10
				0.06	0.6	0.04				0.02	0.66	0.06
				0.19	0.91	0.08				0.17	0.86	0.11
				0.19	0.91	0.08				0.19	0.86	0.10
SPSA	0.41	1.36	0.03	0.45	1.36	0.03	0.49	1.37	0.03	0.46	1.36	0.03
				0.0	1.37	0.03				0.0	1.37	0.03
				0.43	1.36	0.03				0.4	1.37	0.03
				0.29	1.36	0.03				0.42	1.37	0.03
ZOO	0.96	0.56	0.08	0.66	0.55	0.08	0.97	0.69	0.1	0.98	0.64	0.09
				0.0	0.74	0.1				0.0	0.92	0.1
				0.95	0.61	0.08				0.97	0.7	0.09
				0.95	0.55	0.08				0.97	0.7	0.1

Table 6.20: Transferability from 3-bit fully quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.49	1.65	0.03	0.62	1.65	0.03	0.56	1.66	0.03	0.63	1.66	0.03
				0.43	1.65	0.03				0.51	1.66	0.03
				0.17	1.65	0.03				0.37	1.66	0.03
				0.43	1.65	0.03				0.51	1.66	0.03
BIM	0.38	1.17	0.03	0.67	1.17	0.03	0.58	1.13	0.03	0.72	1.13	0.03
				0.35	1.17	0.03				0.49	1.13	0.03
				0.11	1.17	0.03				0.11	1.13	0.03
				0.33	1.17	0.03				0.49	1.13	0.03
CWl2	0.25	0.98	0.10	0.56	0.97	0.10	0.28	0.92	0.11	0.58	0.91	0.10
				0.19	0.99	0.09				0.17	0.92	0.11
				0.10	0.97	0.09				0.02	0.96	0.11
				0.18	0.98	0.10				0.15	0.93	0.11
SPSA	0.42	1.36	0.03	0.67	1.36	0.03	0.45	1.36	0.03	0.67	1.35	0.03
				0.39	1.37	0.03				0.42	1.36	0.03
				0.00	1.36	0.03				0.07	1.35	0.03
				0.34	1.36	0.03				0.42	1.36	0.03
ZOO	1.0	–	–	0.99	0.23	0.07	1.0	–	–	1.0	–	–
				1.0	–	–				1.0	–	–
				0.76	0.24	0.07				0.94	0.11	0.05
				1.0	–	–				1.0	–	–

Table 6.21: Transferability from 3-bit fully quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.49	1.65	0.03	0.51	1.65	0.03	0.56	1.66	0.03	0.55	1.66	0.03
				0.44	1.65	0.03				0.52	1.66	0.03
				0.44	1.65	0.03				0.52	1.66	0.03
				0.43	1.65	0.03				0.52	1.66	0.03
BIM	0.38	1.17	0.03	0.46	1.17	0.03	0.58	1.13	0.03	0.56	1.13	0.03
				0.37	1.17	0.03				0.5	1.13	0.03
				0.34	1.17	0.03				0.51	1.13	0.03
				0.33	1.17	0.03				0.49	1.13	0.03
CWl2	0.25	0.98	0.10	0.29	0.98	0.10	0.28	0.92	0.11	0.33	0.92	0.11
				0.2	0.92	0.1				0.19	0.93	0.11
				0.18	0.98	0.09				0.17	0.92	0.11
				0.18	0.98	0.09				0.15	0.94	0.11
SPSA	0.42	1.36	0.03	0.56	1.37	0.03	0.45	1.36	0.03	0.44	1.36	0.03
				0.44	1.36	0.03				0.44	1.36	0.03
				0.36	1.36	0.03				0.45	1.36	0.03
				0.41	1.36	0.03				0.43	1.36	0.03
ZOO	1.0	–	–	1.0	–	–	1.0	–	–	1.0	–	–
				1.0	–	–				1.0	–	–
				1.0	–	–				1.0	–	–
				1.0	–	–				1.0	–	–

Table 6.22: Transferability from 3-bit weight-only quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.46	1.65	0.03	0.61	1.65	0.03	0.56	1.66	0.03	0.63	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
				0.45	1.65	0.03				0.52	1.66	0.03
				0.38	1.65	0.03				0.51	1.66	0.03
BIM	0.3	1.15	0.03	0.7	1.15	0.03	0.58	1.13	0.03	0.71	1.14	0.03
				0.24	1.15	0.03				0.5	1.14	0.03
				0.4	1.15	0.03				0.5	1.14	0.03
				0.2	1.15	0.03				0.48	1.14	0.03
CWl2	0.39	0.83	0.07	0.61	0.81	0.07	0.34	0.87	0.1	0.61	0.86	0.1
				0.27	0.83	0.07				0.22	0.89	0.11
				0.36	0.83	0.07				0.21	0.88	0.11
				0.24	0.82	0.07				0.21	0.89	0.11
SPSA	0.32	1.36	0.03	0.68	1.36	0.03	0.47	1.37	0.03	0.72	1.36	0.03
				0.27	1.36	0.03				0.42	1.37	0.03
				0.4	1.36	0.03				0.38	1.37	0.03
				0.26	1.36	0.03				0.33	1.37	0.03
ZOO	0.96	0.59	0.08	0.86	0.56	0.09	0.96	0.7	0.11	0.9	0.68	0.1
				0.94	0.56	0.09				0.95	0.63	0.11
				0.91	0.65	0.09				0.94	0.72	0.11
				0.94	0.52	0.08				0.94	0.66	0.11

Table 6.23: Transferability from 3-bit weight-only quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.46	1.65	0.03	0.47	1.65	0.03	0.56	1.66	0.03	0.55	1.66	0.03
				0.39	1.65	0.03				0.51	1.66	0.03
				0.18	1.65	0.03				0.4	1.66	0.03
				0.38	1.65	0.03				0.52	1.66	0.03
BIM	0.3	1.15	0.03	0.38	1.15	0.03	0.58	1.13	0.03	0.55	1.13	0.03
				0.26	1.15	0.03				0.5	1.14	0.03
				0.0	1.15	0.03				0.0	1.14	0.03
				0.23	1.15	0.03				0.48	1.14	0.03
CWl2	0.39	0.83	0.07	0.44	0.83	0.07	0.34	0.87	0.1	0.38	0.87	0.10
				0.30	0.83	0.07				0.23	0.88	0.11
				0.06	0.82	0.07				0.03	0.90	0.11
				0.26	0.83	0.07				0.21	0.89	0.11
SPSA	0.32	1.36	0.03	0.42	1.36	0.03	0.47	1.37	0.03	0.46	1.37	0.03
				0.3	1.36	0.03				0.37	1.37	0.03
				0.0	1.36	0.03				0.0	1.37	0.03
				0.3	1.36	0.03				0.32	1.37	0.03
ZOO	0.96	0.59	0.08	0.96	0.6	0.08	0.96	0.7	0.11	0.97	0.57	0.1
				0.94	0.57	0.08				0.95	0.68	0.1
				0.0	0.72	0.09				0.0	0.95	0.11
				0.95	0.55	0.09				0.96	0.72	0.11

Table 6.24: Transferability from 4-bit fully quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.46	1.65	0.03	0.62	1.65	0.03	0.56	1.66	0.03	0.63	1.66	0.03
				0.4	1.65	0.03				0.53	1.66	0.03
				0.47	1.65	0.03				0.53	1.66	0.03
				0.18	1.65	0.03				0.4	1.66	0.03
BIM	0.32	1.15	0.03	0.69	1.15	0.03	0.55	1.13	0.03	0.73	1.13	0.03
				0.26	1.15	0.03				0.50	1.13	0.03
				0.42	1.15	0.03				0.48	1.13	0.03
				0.06	1.14	0.03				0.1	1.13	0.03
CWl2	0.36	0.86	0.07	0.63	0.82	0.06	0.33	0.81	0.10	0.62	0.79	0.09
				0.25	0.86	0.07				0.23	0.82	0.10
				0.36	0.87	0.07				0.321	0.82	0.10
				0.05	0.6	0.04				0.02	0.68	0.07
SPSA	0.33	1.36	0.03	0.64	1.36	0.03	0.45	1.37	0.03	0.66	1.36	0.03
				0.3	1.36	0.03				0.37	1.36	0.03
				0.44	1.36	0.03				0.41	1.37	0.03
				0.0	1.36	0.03				0.04	1.37	0.03
ZOO	0.97	1.45	0.18	0.95	0.89	0.13	0.99	0.7	0.14	0.99	0.31	0.09
				0.96	1.07	0.14				0.99	0.61	0.13
				0.97	1.45	0.16				0.99	0.8	0.09
				0.73	1.09	0.14				0.93	0.38	0.1

Table 6.25: Transferability from 4-bit fully quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.46	1.65	0.03	0.47	1.65	0.03	0.56	1.66	0.03	0.55	1.66	0.03
				0.41	1.65	0.03				0.52	1.66	0.03
				0.4	1.65	0.03				0.53	1.66	0.03
				0.39	1.65	0.03				0.53	1.66	0.03
BIM	0.32	1.15	0.03	0.4	1.15	0.03	0.55	1.13	0.03	0.53	1.13	0.03
				0.29	1.15	0.03				0.50	1.13	0.03
				0.25	1.15	0.03				0.49	1.13	0.03
				0.26	1.15	0.03				0.47	1.13	0.03
CWl2	0.37	0.86	0.07	0.45	0.84	0.07	0.33	0.82	0.10	0.37	0.81	0.09
				0.29	0.87	0.07				0.22	0.82	0.10
				0.25	0.86	0.07				0.22	0.82	0.10
				0.25	0.86	0.07				0.21	0.82	0.10
SPSA	0.33	1.36	0.03	0.43	1.36	0.03	0.45	1.37	0.03	0.44	1.36	0.03
				0.34	1.36	0.03				0.39	1.36	0.03
				0.27	1.36	0.03				0.38	1.37	0.03
				0.3	1.36	0.03				0.39	1.37	0.03
ZOO	0.97	1.45	0.18	0.98	1.40	0.17	0.99	0.7	0.14	0.99	0.26	0.11
				0.97	1.42	0.17				0.99	0.11	0.05
				0.97	1.45	0.16				1.0	–	–
				0.97	1.45	0.17				0.99	0.27	0.8

Table 6.26: Transferability from 4-bit weight-only quantized model to 1,2,3,4-bit fully quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.47	1.65	0.03	0.63	1.65	0.03	0.56	1.66	0.03	0.63	1.66	0.03
				0.4	1.65	0.03				0.52	1.66	0.03
				0.46	1.65	0.03				0.53	1.66	0.03
				0.4	1.65	0.03				0.51	1.66	0.03
BIM	0.31	1.15	0.03	0.71	1.15	0.03	0.56	1.13	0.03	0.74	1.13	0.03
				0.27	1.15	0.03				0.50	1.13	0.03
				0.42	1.15	0.03				0.5	1.13	0.03
				0.24	1.15	0.03				0.48	1.13	0.03
CWl2	0.39	0.85	0.07	0.65	0.83	0.06	0.37	0.80	0.09	0.61	0.79	0.09
				0.28	0.86	0.07				0.26	0.81	0.10
				0.39	0.87	0.07				0.26	0.81	0.10
				0.26	0.86	0.07				0.24	0.80	0.10
SPSA	0.34	1.36	0.03	0.71	1.36	0.03	0.46	1.37	0.03	0.66	1.36	0.03
				0.31	1.36	0.03				0.44	1.37	0.03
				0.43	1.36	0.03				0.38	1.37	0.03
				0.24	1.36	0.03				0.36	1.37	0.03
ZOO	0.96	0.6	0.08	0.86	0.56	0.08	0.96	0.69	0.1	0.88	0.69	0.09
				0.95	0.56	0.09				0.96	0.61	0.1
				0.93	0.65	0.09				0.95	0.66	0.09
				0.95	0.6	0.09				0.94	0.61	0.09

Table 6.27: Transferability from 4-bit weight-only quantized model to 1,2,3,4-bit weight-only quantized models.

	CIFAR10						SVHN					
	Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*			Float model *(32-bit)*			Quantized models *(1,2,3,4-bit)*		
	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞	Acc	l_2	l_∞
FGSM	0.47	1.65	0.03	0.48	1.65	0.03	0.56	1.66	0.03	0.55	1.66	0.03
				0.41	1.65	0.03				0.52	1.66	0.03
				0.4	1.65	0.03				0.52	1.66	0.03
				0.19	1.65	0.03				0.39	1.66	0.03
BIM	0.31	1.15	0.03	0.39	1.15	0.03	0.56	1.13	0.03	0.55	1.13	0.03
				0.28	1.15	0.03				0.50	1.13	0.03
				0.24	1.15	0.03				0.51	1.13	0.03
				0.0	1.15	0.03				0.0	1.13	0.03
CWl2	0.39	0.85	0.07	0.47	0.984	0.07	0.37	0.80	0.09	0.40	0.79	0.09
				0.31	0.87	0.07				0.26	0.81	0.10
				0.29	0.87	0.07				0.26	0.81	0.10
				0.06	0.62	0.04				0.02	0.68	0.07
SPSA	0.34	1.36	0.03	0.48	1.36	0.03	0.46	1.37	0.03	0.44	1.36	0.03
				0.37	1.36	0.03				0.36	1.37	0.03
				0.26	1.36	0.03				0.42	1.37	0.03
				0.0	1.36	0.03				0.0	1.37	0.03
ZOO	0.96	0.6	0.08	0.96	0.58	0.08	0.96	0.69	0.1	0.97	0.61	0.09
				0.95	0.54	0.08				0.94	0.7	0.1
				0.96	0.57	0.09				0.96	0.66	0.1
				0.0	0.73	0.09				0.0	0.93	0.11

Chapter 7

Luring of transferable adversarial perturbations

In the previous chapter, we studied the impact of model quantization on the robustness of neural networks. We observed that the transferability of adversarial perturbations is tampered between models with different quantization levels. From these results, we derived an ensemble-based defense against an adversary in the black-box setting leveraging transfer black-box attacks. Transfer black-box attacks are a very powerful tool for an adversary in the black-box setting. For this reason, and as black-box settings are the most realistic threat model, transfer black-box attacks are at the core of this thesis.

In this chapter, we pave the way towards a new approach to improve the robustness of a model against black-box transfer attacks. We design the *luring effect*, which tricks the adversary into choosing false directions to fool the target model.

The *luring effect* is initially designed with a removable additional neural network being included in the target model. Training the additional model is achieved thanks to a loss function acting on the logits sequence order. Our deception-based method only needs to have access to the predictions of the target model and does not require a labeled data set. We explain the luring effect thanks to the notion of robust and non-robust useful features and perform experiments on MNIST, SVHN and CIFAR10 to characterize and evaluate this phenomenon.

Then, we identify three possible implementations of the *luring effect* as part of a defense mechanism. Two scenarios consider a black-box or gray-box setting and implement purification/detection schemes. Eventually, we show how the *luring effect* can be leveraged to detect the crafting process of adversarial perturbations.

Related publication proposed during the thesis:

- [214] Bernhard, R., Moellic, P. A., Dutertre, J.-M. *Luring of transferable adversarial perturbations in the black-box paradigm.* IEEE International Joint Conference on Neural Networks (2021). 2021.

7.1 Motivation

Many defenses have been proposed to protect the integrity of ML systems, predominantly focused on an adversary in the white-box setting [75, 89, 147, 215, 216]. However, only few works have considered the much more realistic black-box settings, which constitute increasingly existing use-cases since many models are deployed within secure environments and accessible only through open or restrictive API. Contrary to the white-box paradigm where the adversary is allowed to use existing gradient-based attacks [73,75,105,106,125,137], an attacker in a black-box setting can only access the output label, confidence scores or logits from the target model. While an adversary in such setting can still take advantage of gradient-free methods [62, 79, 108, 129, 130, 135], he faces two major drawbacks. First, the number of queries requires to mount the attack can be prohibitive. In the scenario of a pay-per-query system, using gradient-free methods can then be very costly. Second, the numerous queries (thousands) submitted to the target model can be easily flagged as suspicious. To avoid these potential pitfalls, the adversary can thus take advantage of the transferability property [64] by crafting adversarial examples on a *substitute model* and then transfering them to the target model. The transferability of adversarial perturbations, studied in a particular case in the previous chapter, is therefore a main concern for the security of machine learning models.

In this chapter, we design the *luring effect*, an innovative way to limit the transferability of adversarial perturbation towards a model, opening a new direction for robustness in the realistic black-box setting. As ML-based online API are likely to become increasingly widespread, and regarding the massive deployment of edge models in a large variety of devices, several instances of a model may be deployed in systems with different environment and security properties. Thus, the black-box paradigm needs to be extensively studied to efficiently protect systems in many critical domains. Considering a target model M that a defender aims at protecting against adversarial examples, we first propose a method which allows to build a model T, that is an augmented or modified version of M, such that adversarial examples do not transfer from T to M. Importantly, training T only requires to have access to M, meaning that no labeled data set is required, so that our approach can be implemented at a low cost for any already trained model. T is built by augmenting M with an additional component P (with $T = M \circ P$) taking the form of a neural network trained with a specific loss function with logit-based constraints. From the observation that transferability of adversarial perturbations between two models occurs because they rely on similar non-robust features [57], we design P such that (1) the augmented network exploits useful features of M and that (2) non-robust features of T and M are either different or require different

perturbations to reach misclassification towards the same class. Our deception-based method is conceptually new as it does not aim at making M relying more on robust-features as with proactive schemes [75,147], nor tries to anticipate perturbations which directly target the non-robust features of M as with reactive strategies [110, 217]. We then leverage the *luring effect* in other defense scenarios.

Our key contributions are:

- We present an innovative approach to thwart transferability between two models, which we name the *luring effect*. This phenomenon, as conceptually novel, opens a new direction for adversarial research.
- We propose an implementation of the luring effect which fits any pre-trained model and does not require a label data set. An additional neural network is pasted to the target model and trained with a specific loss function that acts on the logits sequence order.
- We experimentally characterize the luring effect, and discuss its potentiality for three different black-box defense strategies on MNIST, SVHN and CIFAR10.

7.2 Luring Adversarial Perturbations

7.2.1 Objectives and design

Our objective is to find a novel way to make models more robust against transferable black-box adversarial perturbation without expensive (and sometimes prohibitive) training cost required by many white-box defense methods. Our main idea is based on classical deception-based approaches for network security (e.g. honeypots) and can be summarized as follow: *rather than try to prevent an attack, let's fool the attacker*. Our approach relies on a network $P : \mathcal{X} \to \mathcal{X}$, pasted to the already trained target network M before its input layer, such as the resulting augmented model will answer $T(x) = M \circ P(x)$ when fed with input x. The additional component P is designed and trained to reach a twofold objective:

- *Prediction neutrality:* adding P does not alter the decision for a clean example x, i.e. $T(x) = M \circ P(x) = M(x)$;
- *Adversarial luring:* according to an adversarial example x' crafted to fool T, M does not output the same label as T (i.e. $M \circ P(x') \neq M(x')$) and, in the best case, x' is inefficient (i.e. $M(x') = y$, where y is the ground-truth label).

To explain the intuition of our method, we follow the feature-based framework proposed in [57] where a *feature* f is a function from $\mathcal{X}$ to $\mathbb{R}$, that M has learned to perform its predictions. Considering a binary classification task whre $y = 1$ or $y = -1$, f is said to be ρ-*useful* ($\rho > 0$) if it satisfies Equation 7.1 and is γ-*useful robust* if under the worst perturbation δ chosen in a predefined set of allowed perturbations Δ, f stays γ-useful under this perturbation (Equation 7.2). Eventually, a ρ-useful feature f is said to be a *non-robust feature* if f is not robust for any $\gamma \geq 0$.

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[y \cdot f(x)\right] > \rho \tag{7.1}$$

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\inf_{\delta\in\Delta} y \cdot f(x+\delta)\right] > \gamma \tag{7.2}$$

An adversary which aims at fooling a model will thus perform perturbations to the inputs to influence the useful features which are not robust with respect to the perturbation he is allowed to apply. We denote $\mathcal{F}_M^*$ the set of ρ-useful features learned by M. We consider a set of allowed perturbations Δ and $\gamma > 0$ such that we note $\mathcal{F}_M^{*,R}$ and $\mathcal{F}_M^{*,NR}$ respectively the set of γ-robust and non-robust features learned by M (relatively to Δ). An adversary that aims at fooling $M \circ P$ will alter function compositions of the form $f \circ P$ with $f \in \mathcal{F}_M^*$. These function compositions are the non-robust useful features of $M \circ P$, whose set is denoted $\mathcal{F}_{M\circ P}^{*,NR}$.

Based on the observations that transferability of adversarial perturbations between two models occurs because these models rely on similar non-robust features [57], we consider $f \circ P \in \mathcal{F}_{M\circ P}^{*,NR}$, and we derive two possibilities which ensure the lowest transferability between $M \circ P$ and M, regarding the robustness of f:

- f is robust ($f \in \mathcal{F}_M^{*,R}$). This case means that adversarial perturbations from Δ is sufficient to *flip* the augmented feature $f \circ P$ but is not efficient to directly impact f. This case is the optimal one, since the adversarial example is unsuccessful on the target model ($M \circ P(x') \neq y$ and $M(x') = y$)

- f is non-robust ($f \in \mathcal{F}_M^{*,NR}$). As $f \circ P$ is non robust, in this case, thwarting transferability means that the additional model P impacts the way useful features vary with respect to input alterations so that the adversarial perturbation lead to two different labels.

We encompass these two cases (illustrated in Figure 7.1) within what we name the luring effect. The adversary is tricked into modifying input values in some way to flip useful and non-robust features of $M \circ P$ *and* these modifications are either without effect on the useful features of M, or flip the non-robust features of M in a different way (and therefore are detectable, as presented in Sections 7.4 and 7.5).

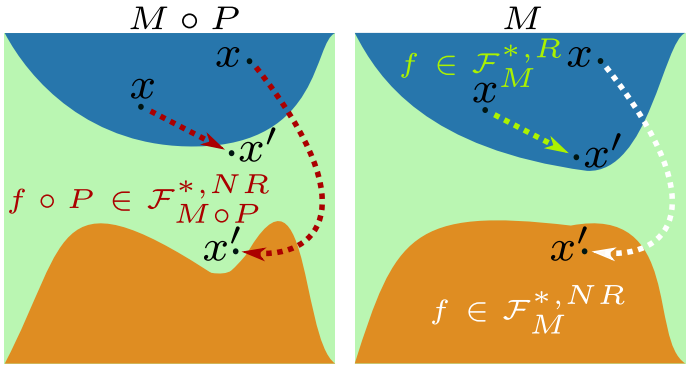

Figure 7.1: Luring effect. (left) x' fools $M \circ P$ by flipping a non-robust feature $f \circ P$ (toward the green class). (right) However, f can be a robust feature, then M is not fooled (still in the blue class), or a non-robust feature but switched differently than $f \circ P$ (towards the orange class).

7.2.2 Training the luring component

To reach our two objectives (*prediction neutrality* and *adversarial luring*), we propose to train P with constraints based on the predicted labels order. For $x \in \mathcal{X}$, let α and β be the labels corresponding respectively to the first and second highest confidence score given to x by M. The training of P is achieved with a new loss function that constraints α to (still) be the first class predicted by $M \circ P$ (*prediction neutrality*) and that makes the logits gap between α and β the highest as possible for $M \circ P$ (*adversarial luring*).

To understand the intuition behind this loss function, let's formalize the concepts learned by M and $M \circ P$. The prediction given by M corresponds to "class α is predicted, class β is the second possible class". Once P has been trained following this new loss function, the prediction given by $M \circ P$ corresponds to "class α is predicted, the higher confidence given to class α, the smaller confidence given to class β". Concepts learned by M and $M \circ P$ share the same goal of prediction, i.e. "class α" is predicted, but the relation between class α and class β is forced to be the most different as possible. As learned concepts are essentially different, then useful features learned by M and $M \circ P$ to reach these concepts are necessarily different, and consequently display different types of sensitivity to the same input pixel modifications. We designate by *direction* of confidence towards classes the way logits are distributed relatively to classes, and the gap between logits values. As an example, it can designate the fact that the biggest logit value is relative to some classe, the second biggest to another class, and the gap between these two logit values. Therefore, we can assume here that the *direction* of confidence towards classes is forced to be structurally different for $M \circ P$ and M, and we hypothesize that useful features of the two classifiers should be different and behave differently to adversarial perturbations.

The *luring loss* $\mathcal{L}_{lur}$, designed to induce this behavior, is given in Equation 7.3 and the complete training procedure is detailed in Algorithm 1.

For an input $x \in \mathcal{X}$, the logits for the target model M and the augmented model $M \circ P$ are denoted $h^M(x)$ and $h^{M \circ P}(x)$, respectively. The parameters of P are denoted by Φ. $h_i^M(x)$ and $h_i^{M \circ P}(x)$ correspond respectively to the values of $h^M(x)$ and $H^{M \circ P}(x)$ for class i. The classes a

and b correspond to the second maximum value of h^M and $h^{M \circ P}$ respectively. M has already been trained and its parameters are frozen during the process.

$$\mathcal{L}_{lur}\left(x, M(x)\right) = -\lambda \left(h^{M\circ P}_{M(x)}(x) - h^{M\circ P}_{a}(x)\right) + \max\left(0, h^{M\circ P}_{b}(x) - h^{M\circ P}_{M(x)}(x)\right) \tag{7.3}$$

The first term of Equation 7.3 optimizes the gap between the logits of $M \circ P$ corresponding to the first and second biggest unscaled confidence score (logits) given by M (i.e. $M(x)$ and a). This part formalizes the goal of changing the direction of confidence between $M \circ P$ and M. The second term is compulsory to reach a good classification since the first part alone does not ensure that $h^{M\circ P}_{M(x)}(x)$ is the highest logit value (*prediction neutrality*). The parameter $\lambda > 0$, called the *luring coefficient*, allows to control the trade-off between ensuring good accuracy and shifting confidence direction.

Algorithm 1 Training of the luring component

Input: trained model M, training steps K, learning rate η, batch size B, luring coefficient λ
Output: luring component P, with parameters Φ

$h^{M\circ P}_{c}(x)$ denotes the logits of $M \circ P$ for input x and class c
Randomly initialize the parameters Φ of P
for step $= 1 \dots K$: **do**
    $\{x_1, x_2, \dots, x_B\}$: batch of training set examples
    for $i = 1 \dots B$: **do**
        (a,b) $\leftarrow$ (class of the 2^{nd} max value of $h^M(x_i)$, class of the 2^{nd} max value of $h^{M\circ P}(x_i)$)
        $\mathcal{L}_{lur}\left(x_i, M(x_i)\right) \leftarrow -\lambda \left(h^{M\circ P}_{M(x_i)}(x_i) - h^{M\circ P}_{a}(x_i)\right) + \max\left((0, h^{M\circ P}_{b}(x_i) - h^{M\circ P}_{M(x_i)}(x_i)\right)$
    end for
    $\Phi \leftarrow \Phi - \eta \sum_{i=1}^{B} \nabla_{\Phi} \mathcal{L}_{lur}(x_i, M(x_i))/B$
end for
return P

7.3 Characterization of the luring effect

7.3.1 Objective

Our first experiments are dedicated to the characterization of the luring effect by (1) evaluating our objectives in term of transferability and (2) isolating it from other factors that may influence transferability. For that purpose, we compare our approach (*Luring*) to the following approaches, which differ from ours in the training procedures and loss functions:

- ***Stack* model**: $M \circ P$ is retrained as a whole with the cross-entropy loss. *Stack* serves as a first baseline to measure the transferability between the two architectures of $M \circ P$ and M.
- ***Auto* model**: P is an auto-encoder trained separately with binary cross-entropy loss. *Auto* serves as a second baseline of a component resulting in a *neutral* mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$.
- ***C_E* model**: P is trained with the cross-entropy loss between the confidence score vectors $M \circ P(x)$ and $M(x)$ in order to mimic the decision of the target model M. This model serves as a comparison between our loss and a loss function which does not aim at maximizing the gap between the confidence scores.

7.3.2 Data sets and models

We perform experiments on MNIST, SVHN and CIFAR10. For MNIST, M has the same architecture as in [75]. For SVHN and CIFAR10, we follow an architecture inspired from VGG [185]. Architectures and training setup for M are detailed for MNIST, SVHN and CIFAR10 in Tables 7.1, 7.2 and 7.3.

Architectures and training setup for P are detailed for MNIST, SVHN and CIFAR10 in Tables 7.4, 7.5 and 7.6.

Table 7.1: MNIST base classifier architecture. Epochs: 5. Batch size: 28. Optimizer: Adam with learning rate 0.01. MaxPool(u,v) and Conv(f,k,k) denote max pooling with window size (u,v) and 2D convolution with f filters and kernel of size (k,k), respectively.

Architecture
Conv(32,5,5) + ReLU
Maxpool(2,2)
Conv(64,5,5) + ReLU
MaxPool(2,2)
Dense(1024) + ReLU
Dense(10) + softmax

Table 7.2: SVHN base classifier architecture. Epochs: 50. Batch size: 28. Optimizer: Adam with learning rate 0.01. BN, MaxPool(u,v) and Conv(f,k,k) denote respectively batch normalization, max pooling with window size (u,v) and 2D convolution with f filters and kernel of size (k,k), respectively.

Architecture
Conv(64,3,3) + BN + ReLU
Conv(64,3,3) + MaxPool(2,2) + BN + ReLU
Conv(128,3,3) + BN + ReLU
Conv(128,3,3) + MaxPool(2,2) + BN + ReLU
Conv(256,3,3) + BN + ReLU
Conv(256,3,3) + MaxPool(2,2) + BN + ReLU
Dense(1024) + BN + ReLU
Dense(1024) + BN + ReLU
Dense(10) + softmax

Table 7.3: CIFAR10 base classifier architecture. Epochs: 200. Batch size: 32. Optimizer: Adam with learning rate starting at 0.1, decreasing to 0.01 and 0.001 respectively after 80 and 120 epochs. BN, MaxPool(u,v) and Conv(f,k,k) denote respectively batch normalization, max pooling with window size (u,v) and 2D convolution with f filters and kernel of size (k,k), respectively.

Architecture
Conv(128,3,3) + BN + ReLU
Conv(128,3,3) + MaxPool(2,2) + BN + ReLU
Conv(256,3,3) + BN + ReLU
Conv(256,3,3) + MaxPool(2,2) + BN + ReLU
Conv(512,3,3) + BN + ReLU
Conv(512,3,3) + MaxPool(2,2) + BN + ReLU
Dense(1024) + BN + ReLU
Dense(1024) + BN + ReLU
Dense(10) + softmax

Table 7.4: Defense component architecture for MNIST. MaxPool(u,v), UpSampling(u,v) and Conv(f,k,k) denote max pooling with window size (u,v), upsampling with sampling factor (u,v) and 2D convolution with f filters and kernel of size (k,k), respectively.

Architecture
Conv(16,3,3) + ReLU
MaxPool(2,2)
Conv(8,3,3) + ReLU
MaxPool(2,2)
Conv(8,3,3) + ReLU
Conv(8,3,3) + ReLU
UpSampling(2,2)
Conv(8,3,3) + ReLU
UpSampling(2,2)
Conv(16,3,3)
UpSampling(2,2)
Conv(1,3,3) + sigmoid

- *Stack* Epochs: 5. Batch size: 28. Optimizer: Adam with learning rate 0.001
- *Auto* Epochs: 50. Batch size: 128. Optimizer: Adam with learning rate 0.001
- *C_E* Epochs: 64. Batch size: 64. Optimizer: Adam with learning rate starting at 0.001, decreasing to 0.0002 and 0.0004 respectively after 45 and 58 epochs
- *Luring* Epochs: 64. Batch size: 64. Optimizer: Adam with learning rate starting at 0.001, decreasing to 0.0002 and 0.0004 respectively after 45 and 58 epochs

Table 7.5: Defense component architecture for SVHN. BN, MaxPool(u,v), UpSampling(u,v) and Conv(f,k,k) denote batch normalization, max pooling with window size (u,v), upsampling with sampling factor (u,v) and 2D convolution with f filters and kernel of size (k,k), respectively.

Architecture
Conv(128,3,3) + BN + ReLU
MaxPool(2,2)
Conv(256,3,3) + BN + ReLU
MaxPool(2,2)
Conv(512,3,3) + BN + ReLU
Conv(1024,3,3) + BN + ReLU
MaxPool(2,2)
Conv(512,3,3) + BN + ReLU
Conv(512,3,3) + BN + ReLU
UpSampling(2,2)
Conv(256,3,3) + BN + ReLU
UpSampling(2,2)
Conv(128,3,3) + BN + ReLU
UpSampling(2,2)
Conv(64,3,3) + BN + ReLU
Conv(3,3,3) + BN + sigmoid

- *Stack* Epochs: 20. Batch size: 256. Optimizer: Adam with learning rate 0.001
- *Auto* Epochs: 5. Batch size: 128. Optimizer: Adam with learning rate 0.001
- *C_E* Epochs: 210. Batch size: 256. Optimizer: Adam with learning rate starting at 0.0001, decreasing to 0.00001 and 0.000008 respectively after 126 and 168 epochs
- *Luring* Epochs: 210. Batch size: 256. Optimizer: Adam with learning rate starting at 0.0001, decreasing to 0.00001 and 0.000008 respectively after 126 and 168 epochs. Dropout is used

Table 7.6: Defense component architecture for CIFAR10. BN, MaxPool(u,v), UpSampling(u,v) and Conv(f,k,k) denote batch normalization, max pooling with window size (u,v), upsampling with sampling factor (u,v) and 2D convolution with f filters and kernel of size (k,k), respectively.

Architecture
Conv(128,3,3) + BN + ReLU
Conv(256,3,3) + BN + ReLU
MaxPool(2,2)
Conv(512,3,3) + BN + ReLU
Conv(1024,3,3) + BN + ReLU
Conv(512,3,3) + BN + ReLU
Conv(512,3,3) + BN + ReLU
Conv(256,3,3) + BN + ReLU
UpSampling(2,2)
Conv(128,3,3) + BN + ReLU
Conv(64,3,3) + BN + ReLU
Conv(3,3,3) + BN + sigmoid

- *Stack* Epochs: 200. Batch size: 32. Optimizer: Adam with learning rate starting at 0.1, decreasing to 0.01 and 0.001 respectively after 80 and 120 epochs
- *C_E* Epochs: 216. Batch size: 256. Optimizer: Adam with learning rate starting at 0.00001, decreasing to 0.000005 and 0.0000008 respectively after 154 and 185 epochs. Dropout is used
- *Luring* Epochs: 216. Batch size: 256. Optimizer: Adam with learning rate starting at 0.00001, decreasing to 0.000005 and 0.0000008 respectively after 154 and 185 epochs. Dropout is used

Table 7.7 gathers the test set accuracy and agreement rate (on the ground-truth label) between each augmented model and M. For the luring parameter, we set $\lambda = 1$ for MNIST and SVHN, and $\lambda = 0.15$ for CIFAR10. For CIFAR10, since no autoencoder we tried reaches correct test set accuracy, we exclude the *Auto* model. We observe that our approach has a limited impact on the test accuracy with a relative decrease of 1.71%, 4.26% and 4.48% for MNIST, SVHN and CIFAR10 respectively.

Table 7.7: Test set accuracy and agreement (augmented and target model agree on the ground-truth label) between each augmented model and the target model M, noted BASE.

Model	Data set					
	MNIST		SVHN		CIFAR10	
	Test	Agree	Test	Agree	Test	Agree
Base	0.991	–	0.961	–	0.893	–
Stack	0.980	0.976	0.925	0.913	0.902	0.842
Auto	0.971	0.969	0.950	0.943	–	–
C_E	0.982	0.977	0.919	0.907	0.860	0.834
Luring	0.974	0.969	0.920	0.917	0.853	0.822

7.3.3 Attack setup and metrics

For the characterization of the luring effect, we attack the model $M \circ P$ of the four approaches and transfer to M *only* the adversarial examples that are successful for these four models.

We define the *disagreement rate*, noted $DR(X')$, that represents the rate of successful adversarial examples crafted on $M \circ P$ for which M and $M \circ P$ do not agree. The *disagreement rate* is defined formally in Equation 7.4. To measure the best case where the luring effect leads to unsuccessful adversarial examples when transferred to M, we note $IAR(X')$ an *inefficient adversarial examples rate* that represents the proportion of successful adversarial examples on $M \circ P$ but not on M. The *inefficient adversarial examples rate* is defined formally in Equation 7.5. For both metrics, the higher, the better the luring effect to limit transferable attacks on the target model M.

$$DR(X') = \frac{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y, M \circ P(x') \neq M(x')}}{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y}} \tag{7.4}$$

$$IAR(X') = \frac{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y, M(x') = y}}{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y}} \tag{7.5}$$

We use the gradient-based attacks FGSM [73], PGD [75], and MIM [137] in its l_∞ (MIM) and l_2 (MIML2) versions. We used three l_∞ perturbation budgets (ϵ values): 0.2, 0.3 and 0.4 for MNIST, 0.03, 0.06 and 0.08 for SVHN, and 0.02, 0.03 and 0.04 for CIFAR10. For MIML2, we report results when adversarial examples are clipped to respect the threat model with regards to ϵ. For PGD, MIM and MIML2, the number of iterations is set to $1,000$, the step size to 0.01 and μ to 1.0 (MIM and MIML2). For MIML2, the l_2 bound is set to 30 on MNIST and 2 on SVHN and CIFAR10. An illustration of a clean image and its adversarial counterpart for the maximum perturbation allowed is presented in Figure 7.2. We note that the ground-truth label is still clearly recognizable.

7.3.4 Results and further analysis of the luring effect

Results

We report results for the DR and IAR in Figure 7.3. The highest performances reached with our loss (for every ϵ) show that our training method is efficient at inducing the luring effect. More precisely, we claim that the fact that both metrics decrease much slower as ϵ increases compared to the other architectures, brings additional confirmation that non-robust features of M and $M \circ P$ tend to behave more differently than with the three other considered approaches.

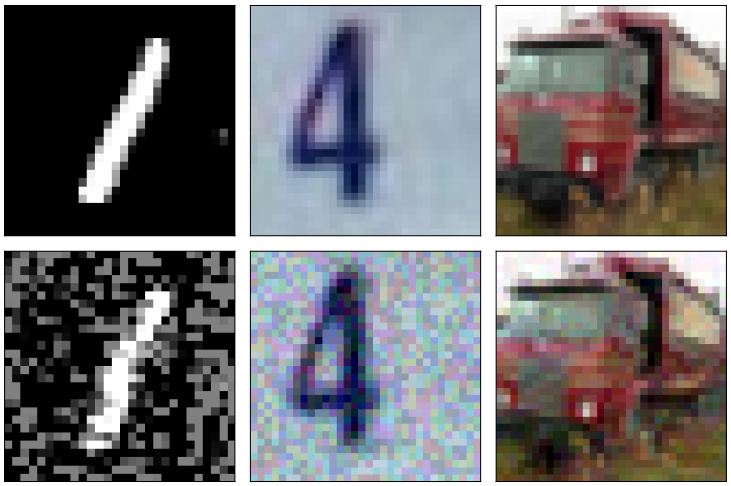

Figure 7.2: (top) Clean image, (bottom) adversarial example for the maximum perturbation allowed (left to right: MNIST, SVHN, CIFAR10; $\epsilon = 0.4, 0.08, 0.04$).

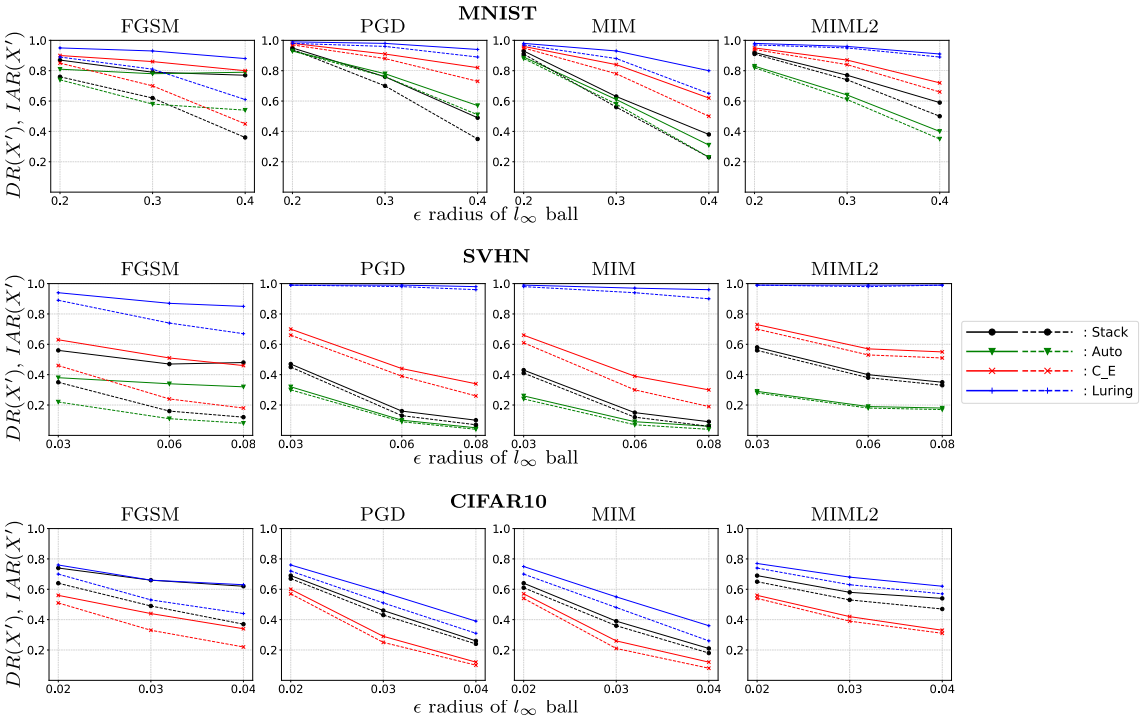

Figure 7.3: Disagreement Rate (solid line) and Inefficient Adversarial examples Rate (dashed line) for different attacks.

Distance to an adversarial example

As a complementary analysis, we investigate the impact of our additional component compared to the other approaches on the distance between a clean and an adversarial example. This will allow to assess that the better results observed previously with the luring loss are due to the *luring effect* being induced and not simply different distortion needed to fool M and $M \circ P$. More precisely, this analysis will allow to verify one crucial issue: adversarial examples are found at approximately the same distance for $M \circ P$ and M, for all approaches. Indeed, if the distance between a clean and an adversarial input is much lower for $M \circ P$ with our approach than the other approaches (and lower than the distance for the target model M), the better results could then be explained as the adversary simply not perturbing enough the clean example when attacking $M \circ P$. The target model M could then show robustness but this would not be caused by the

Table 7.8: Average l_2 and l_∞ distortions between clean and adversarial examples generated with the CWL2 attack for the target model M and the augmented models $M \circ P$

	MNIST		SVHN		CIFAR10	
Model	Mean l_2	Mean l_∞	Mean l_2	Mean l_∞	Mean l_2	Mean l_∞
Target	0.96	0.27	0.32	0.04	0.42	0.04
Stack	0.87	0.25	0.34	0.05	0.4	0.03
Auto	0.94	0.35	0.29	0.04	–	–
C_E	0.92	0.34	0.29	0.04	0.41	0.04
Luring	0.94	0.31	0.31	0.04	0.39	0.04

luring effect, but simply minimal adversarial distortion concerns. To this end, we measure the l_2 and l_∞ means of the adversarial perturbations needed to find an adversarial example, and verify that our approach is globally at the same level than the other approaches. For that purpose, an appropriate and recommended attack is CWl2 attack [105] that searches for the closest adversarial examples in terms of the l_2 distortion. We run CWl2 with the following parameters: 10 binary search steps, learning rate of 0.1, initial constant of 0.5 and 500 iterations for MNIST, SVHN and CIFAR10. These parameters have been chosen such that increasing the number of iterations does not lower the final l_2 distortion. For each data set, we run the attack on $1,000$ test set examples correctly classified for the four approaches for both M and $M \circ P$. All the adversarial examples fool the models. The average l_2 and l_∞ distortions are reported in Table 7.8: each perturbation size corresponds to the lowest perturbation size among the 10 constant values of the CWl2 attack objective function (see Equation 4.13) tested. Results show that there is no significant difference between our approach and the other approaches. This is an additional observation that our *luring loss* allows to train an augmented model which causes the *luring effect* by predominantly targeting different useful non-robust features rather than artificially modifying the scale of the adversarial distortion needed to cause misclassification. In other words, this indicates that our approach truly impacts the underlying useful features and does not only imply different distortions scales needed to fool both M and $M \circ P$.

Investigating the l_0 distortion

Moreover, we also investigate the l_0 distortion. For MNIST, as presented in Figure 7.4 – left, we notice that the l_0 distortion (i.e. the cardinality of the pixels impacted by the adversarial attack) is significantly higher with our method. This points out that P leverages more different useful features, which are easily identifiable for MNIST because of the basic structure of the images. This effect can also be demonstrated thanks to saliency maps (i.e. analyzing $\nabla_x P(x)$), as illustrated in Figure 7.5. In Figure 7.5, if we denote by $P_1(x), P_2(x), ..., P_d(x)$ the output of the component P, each pixel represents $\nabla_x(P_1 + P_2 + \cdots + P_d)(x)$. Indeed, for the *Luring* model, P is sensitive to many additional *useless* background pixels than M, compared with the other approaches. Note that for *Auto*, modifying P's mapping consists almost exclusively on modifying pixels correctly correlated with the true label.

For more complex inputs from SVHN or CIFAR10, without a uniform background as for MNIST, we observe that the l_0 distortion is approximately the same for all the architectures (Figure 7.4 – right). However, for these two data sets, we analyze the influence of P's mapping thanks to the logits variations. More precisely, we analyze how P acts on the variation of the logits with respect to the input features ($\nabla_{x_i} h(x)$). For that purpose, we first pick $1,000$ test set examples correctly classified by all the models (the base model M and the augmented models of the four approaches: *Auto*, *Stack*, *C_E* and *Luring*). Then, for each augmented model and each class c, we compute the number Ω_c of pixels x_d that lead to an opposite effect on prediction between M and $M \circ P$. This quantity is defined in Equation 7.6 for an input x.

$$\Omega_c(x) = \left|\left\{d \mid \nabla_{x_d} h_c(x) \cdot \nabla_{x_d} h_c^{M \circ P}(x) < 0\right\}\right| \tag{7.6}$$

Next, we look at the proportion of these examples for which Ω_c is higher for the Luring approach than for the other approaches. Results for each class are presented in Table 7.9. For both SVHN and CIFAR10, these proportions are strictly higher than 80% whatever the class. This is a supplementary indication toward the fact that – with respect to the allowed adversarial perturbation

altering x – our approach is more relevant than other approaches (that also perform a mapping within $\mathcal{X}$) to *flip* useful and non-robust features of $M \circ P$ and M towards different labels.

Table 7.9: Rate of examples (over $1,000$) for which logits vary more differently between M and $M \circ P$, for the *Luring* approach against the other approaches.

	Class									
	1	2	3	4	5	6	7	8	9	10
SVHN	0.81	0.85	0.86	0.89	0.86	0.82	0.83	0.84	0.83	0.82
CIFAR	0.86	0.88	0.88	0.87	0.88	0.87	0.89	0.86	0.88	0.85

7.4 Using the luring effect as a black-box defense

By fooling an adversary on the way to target a black-box model, the phenomenon that we characterized in the previous section can be seen as a defense mechanism on its own or as a complement of state-of-the-art approaches. In this section we discuss one way to use our approach as a protection in a strict black-box setting.

7.4.1 Threat model

In the classical black-box transfer setting, the adversary uses a surrogate model M' to craft adversarial examples, and he then transfers them to the target model M. Therefore, a way to take advantage of the luring effect in this context is to consider an adversary provided with the model $T = M \circ P$ as for the surrogate model. We assume that the adversary can query T without any limit on the number of queries to get the logits outputs, but does not have access to the architecture of T (black-box setting), as he could infer M directly.

We assume that the adversary has no access to the inner parameters and architecture, neither of the target model M nor the augmented model $M \circ P$. This setting is classical when dealing with cloud-based API or edge neural networks in secure embedded systems where users have only access to input/output information with different querying abilities and output precision (i.e. scores or labels).

Related real-world transferability scenarios may be linked to the increasingly widespread deployment of models in a large variety of devices (*edge AI*) or services (*cloud AI*) as an integral part of complex systems with different security requirements. For example, it is classically the case with the IoT domain where a model may be the part of a critical system and, simultaneously, be deployed and used in more mainstream connected objects with looser security requirements. The model M is deployed for the critical system with strong security access, not accessible to an attacker, and the augmented protected model will be deployed in the others black-box systems with looser accessibility settings.

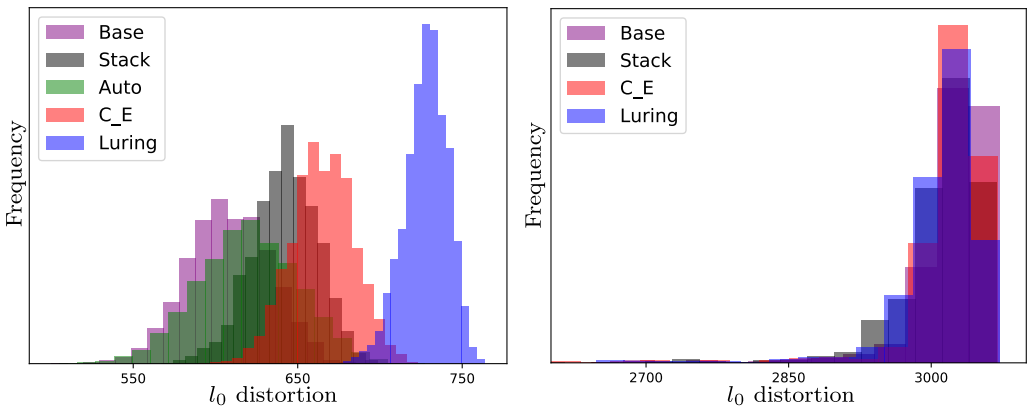

Figure 7.4: l_0 adversarial distortion for MNIST (left) and CIFAR10 (right).

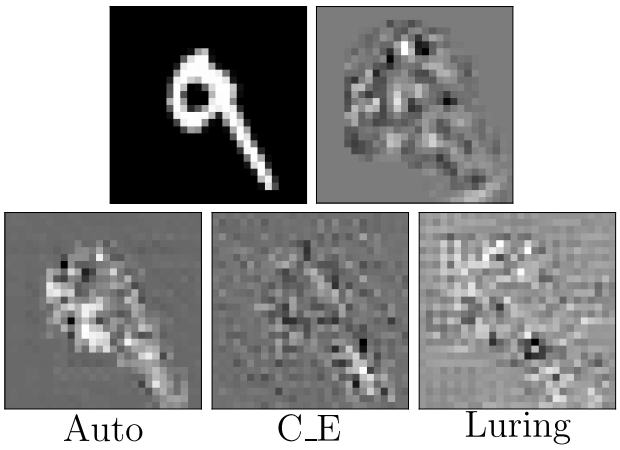

Figure 7.5: MNIST. (top) Clean image and gradient of the cross-entropy loss with respect to input; (bottom) mapping gradients $\nabla_x P(x)$ for 3 augmented models.

7.4.2 Metrics

Considering a secure black-box system with limited access, that only contains the target model M, the goal of the defender is to thwart attacks crafted from $T = M \circ P$. The defender can rely on the ability of the luring effect to lead to unsuccessful adversarial by looking at $M(x)$ only. In this case, the appropriate metric is the classical adversarial accuracy (*Acc*), which is the standard accuracy measured on the adversarial set X' and noted Acc_{MoP} and Acc_M respectively for $M \circ P$ and M. The defender can also detect adversarial examples by comparing the two inference outputs $M(x)$ and $M \circ P(x)$. This scenario assumes that M is completely hidden (the output is not available to the adversary), as it would allow him to mount a black-box attack on the binary classifier given by $M(x)/T(x)$. An appropriate metric is the rate of adversarial examples which are either detected or well-predicted by M, as expressed in Equation 7.7 and noted DAC for *Detection Adversarial Accuracy*.

$$\text{DAC} = 1 - \frac{\sum_{x \in X'} \mathbf{1}_{M \circ P(x') = M(x'), M(x') \neq y}}{|X'|} \tag{7.7}$$

Classically, the adversarial capability is an upper bound ϵ of the perturbation $\|x' - x\|_\infty$.

7.4.3 Attacks

In order to evaluate our defense with respect to the threat model, we attack $M \circ P$ with strong gradient-free attacks (described in Section 4.4):

- SPSA attack [79] ensures to consider the strongest adversary in our black-box setting as it considers that the adversary has access to the logits of $M \circ P$.
- Coherently with an adversary that has no querying limitation, ECO [133] is a strong score-based gradient-estimation free attack.

To perform an even more strict evaluation, and to anticipate future gradient-free attacks, we report the best results[1] obtained with the state-of-the-art transferability designed gradient-based attacks MIM, DIM, MIM-TI and DIM-TI [137–139], under the name MIM-W.

All the attacks (SPSA, ECO, MIM, DIM, MIM-TI, DIM-TI) are performed on $1,000$ correctly classified test set examples, with parameter values tuned for the highest transferability results. More precisely, we searched for parameters leading to the lowest Acc_M and DAC values.

Gradient-free attacks For the SPSA attack, the learning rate is set to 0.1, the number of iterations is set to 100 and the batch size to 128. For the ECO attack, for the three data sets, the number of queries is set to $20,000$. We did not perform early-stopping as it results in less transferable adversarial examples. The block size is set to 2, 8 and 4 respectively for MNIST, SVHN and CIFAR10.

[1] Here *best* is from the adversary point of view.

Gradient-based attacks For DIM and DIM-TI, the optimal p value was searched in the set $\{0.1, 0.2, \cdots, 1.0\}$. We obtained the optimal values of 1.0, 1.0 and 0.6 respectively on MNIST, SVHN and CIFAR10. For MIM and its variants, the number of iterations is set to $1,000$, 500 and 100 respectively on MNIST, SVHN and CIFAR10, and $\mu = 1.0$. The optimal kernel size required by MIM-TI and DIM-TI was searched in $\{(3,3),(5,5),(10,10),(15,15)\}$. The kernel size for the MIM-TI and DIM-TI attacks resulting in the lowest Acc_M and DAC values are reported in Tables 7.10, 7.11 and 7.12 for MNIST, SVHN and CIFAR10, respectively.

Table 7.10: MNIST. Kernel size for the MIM-TI and DIM-TI attacks.

Attack	Architecture	ϵ value (l_∞)		
		0.3	0.4	0.5
MIM-TI	Stack	5×5	5×5	5×5
	Auto	10×10	5×5	5×5
	C_E	10×10	10×10	5×5
	Luring	5×5	5×5	10×10
DIM-TI	Stack	5×5	5×5	5×5
	Auto	5×5	5×5	5×5
	C_E	10×10	10×10	10×10
	Luring	5×5	5×5	5×5

Table 7.11: SVHN. Kernel size for the MIM-TI and DIM-TI attacks.

Attack	Architecture	ϵ value (l_∞)		
		0.03	0.06	0.08
MIM-TI	Stack	5×5	5×5	5×5
	Auto	5×5	5×5	5×5
	C_E	5×5	10×10	5×5
	Luring	10×10	10×10	10×10
DIM-TI	Stack	5×5	5×5	5×5
	Auto	5×5	5×5	5×5
	C_E	5×5	5×5	5×5
	Luring	10×10	10×10	10×10

Table 7.12: CIFAR10. Kernel size for the MIM-TI and DIM-TI attacks.

Attack	Architecture	ϵ value (l_∞)		
		0.02	0.03	0.04
MIM-TI	Stack	3×3	3×3	3×3
	Auto	3×3	3×3	3×3
	C_E	3×3	10×10	3×3
	Luring	3×3	3×3	3×3
DIM-TI	Stack	3×3	3×3	3×3
	Auto	3×3	3×3	3×3
	C_E	3×3	3×3	3×3
	Luring	3×3	3×3	3×3

7.4.4 Results

The results are presented in Table 7.13 for MNIST, SVHN and CIFAR10. For CIFAR10, since no autoencoder we tried allows to reach a correct test set accuracy for the *Auto* approach, we do not consider it. First, we notice that Acc_{MoP} almost always equal 0 with gradient-based iterative

Table 7.13: Adversarial accuracy for $M \circ P$ (AC_{MoP}), M (AC_M), and Detection Adversarial Accuracy (DAC) for different architectures. MNIST(top), SVHN (middle), CIFAR10 (bottom).

MNIST		Stack			Auto			C_E			Luring		
	ϵ	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC
SPSA	0.2	0.0	0.96	0.97	0.03	0.95	0.95	0.0	0.97	0.98	0.14	**0.99**	**0.99**
	0.3	0.0	0.86	0.89	0.03	0.90	0.92	0.0	0.94	0.95	0.05	**0.96**	**0.97**
	0.4	0.0	0.72	0.77	0.0	0.85	0.87	0.0	0.88	0.91	0.02	**0.95**	**0.96**
ECO	0.2	0.03	0.86	0.92	0.03	0.86	0.88	0.01	0.91	0.93	0.05	**0.99**	**1.0**
	0.3	0.02	0.56	0.65	0.03	0.68	0.7	0.01	0.8	0.87	0.02	**0.91**	**0.96**
	0.4	0.01	0.35	0.54	0.03	0.36	0.46	0.01	0.45	0.48	0.03	**0.77**	**0.79**
MIM-W	0.2	0.0	0.79	0.82	0.0	0.81	0.82	0.0	0.85	0.88	0.19	**0.92**	**0.93**
	0.3	0.0	0.31	0.45	0.0	0.35	0.45	0.0	0.43	0.57	0.13	**0.69**	**0.75**
	0.4	0.0	0.07	0.24	0.0	0.07	0.17	0.0	0.13	0.31	0.07	**0.34**	**0.45**

SVHN		Stack			Auto			C_E			Luring		
	ϵ	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC
SPSA	0.03	0.10	0.54	0.56	0.06	0.37	0.38	0.06	0.67	0.68	0.0	**0.96**	**0.97**
	0.06	0.01	0.21	0.24	0.0	0.10	0.11	0.0	0.37	0.42	0.0	**0.96**	**0.96**
	0.08	0.0	0.13	0.15	0.0	0.06	0.06	0.0	0.23	0.28	0.0	**0.94**	**0.96**
ECO	0.03	0.06	0.42	0.44	0.14	0.48	0.49	0.18	0.66	0.68	0.20	**0.97**	**0.98**
	0.06	0.0	0.11	0.12	0.06	0.09	0.11	0.1	0.35	0.39	0.1	**0.86**	**0.88**
	0.08	0.0	0.03	0.07	0.06	0.09	0.09	0.08	0.29	0.32	0.09	**0.84**	**0.86**
MIM-W	0.03	0.04	0.32	0.35	0.01	0.20	0.21	0.03	0.41	0.45	0.11	**0.81**	**0.87**
	0.06	0.0	0.06	0.09	0.0	0.03	0.05	0.0	0.10	0.18	0.0	**0.58**	**0.71**
	0.08	0.0	0.03	0.06	0.0	0.01	0.02	0.0	0.06	0.13	0.0	**0.48**	**0.67**

CIFAR10		Stack			C_E			Luring		
	ϵ	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC
SPSA	0.02	0.01	0.75	0.78	0.06	0.68	0.71	0.12	**0.78**	**0.82**
	0.03	0.0	0.57	0.62	0.01	0.45	0.49	0.02	**0.59**	**0.64**
	0.04	0.0	0.38	0.42	0.0	0.22	0.24	0.01	**0.42**	**0.50**
ECO	0.02	0.15	0.7	0.71	0.14	0.6	0.63	0.16	**0.81**	**0.87**
	0.03	0.09	0.44	0.59	0.09	0.36	0.42	0.2	**0.65**	**0.67**
	0.04	0.05	0.29	0.29	0.04	0.28	0.34	0.09	**0.35**	**0.44**
MIM-W	0.02	0.0	0.49	0.52	0.02	0.4	0.43	0.03	**0.58**	**0.64**
	0.03	0.0	0.24	0.28	0.0	0.15	0.19	0.0	**0.30**	**0.39**
	0.04	0.0	0.13	0.16	0.0	0.05	0.1	0.0	**0.18**	**0.28**

attacks[2]. This is an indication that P does not induce a form of gradient masking [78].
Remarkably on SVHN, for $\epsilon = 0.08$ (the largest perturbation), and for the worst-case attacks, adversarial examples tuned to achieve the best transferability only reduce Acc_M to 0.48 against our approach, compared to almost 0 for the other architectures. The robustness benefits are more observable on SVHN and MNIST but the results on CIFAR10 are particularly promising in the scope of a defense scheme that only requires a pre-trained model. Indeed, for the common l_∞ perturbation value of 0.03, the worst DAC value for the *C_E* and *Luring* approaches are respectively 0.19 (against DIM attack) and 0.39 (against DIM-TI attack).

7.4.5 Compatibility with ImageNet and Adversarial Training

We scale our approach to ImageNet (ILSVRC2012). For our experiments, the target model M consists of a MobileNetV2 model [187], reaching 71.3% of accuracy on the validation set. The architecture for the defense component is described in Table 7.14. We choose to present our approach along with the *C_E* approach as it represents well the purpose of the method we present in this chapter. Indeed, the *C_E* approach only requires training the additional component P, which fits into the scope of a defense which can be implemented on top of any already trained model M. For both the *Luring* and *C_E* approaches, the additional component P is trained for 100 epochs, with a bath size of 256, using a momentum of 0.9 with learning rate starting at 0.01, decreasing to 0.001 after 80 epochs. The λ parameter of the loss is set to 1.0. The accuracy of the augmented model as well as the agreement rate (the model M and the augmented model $M \circ P$)

[2]Parameters are tuned relatively to the adversarial goal: lowering our defense efficiency. Thus, the lowest performances sometimes do not correspond to the lowest accuracy of $M \circ P$ on the adversarial examples.

are presented in Table 7.15. The attacks are performed on $1,000$ well-classified images from the validation set, and three common l_∞ perturbation budgets are considered : $4/255$, $5/255$ and $6/255$. Results are presented in Table 7.16. The highest results observed in terms of both Acc_M and DAC values with our approach confirm the scalability of our method on large-scale data sets. For a common l_∞ perturbation budget of $4/255$, the smaller Acc_M and DAC observed against the strong MIM-W attack equal 0.4 and 0.55 with our approach, while they equal 0.23 and 0.35 with the C_E approach. Following the results previously observed on the benchmarks used for characterization, these results validate the scalability of our approach to large-scale data sets.

Table 7.14: Defense component architecture for ImageNet.

Architecture
Conv(128,3,3) + BN + ReLU
MaxPool(2,2)
Conv(256,3,3) + BN + ReLU
MaxPool(2,2)
Conv(512,3,3) + BN + ReLU
MaxPool(2,2)
Conv(512,3,3) + BN + ReLU
Conv(256,3,3) + BN + ReLU
UpSampling(2,2)
Conv(128,3,3) + BN + ReLU
UpSampling(2,2)
Conv(64,3,3) + BN + ReLU
Conv(3,3,3) + BN + sigmoid

Table 7.15: Test set accuracy and agreement (augmented and target model agree on the ground-truth label) between each augmented model and the target model M, noted Base.

Model	Test	Agree
Base	0.713	–
C_E	0.679	0.647
Luring	0.653	0.614

Table 7.16: ImageNet. Acc_{MoP}, Acc_M and DAC for different source model architectures.

		C_E			Luring		
	ϵ	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC
MIM-W	$4/255$	0.0	0.23	0.35	0.00	**0.4**	**0.55**
	$5/255$	0.0	0.15	0.25	0.00	**0.28**	**0.43**
	$6/255$	0.0	0.08	0.18	0.00	**0.18**	**0.33**

Even if very few efforts tackle the issue of thwarting adversarial perturbations transferred from a source model to a target model (transfer black-box attacks), there exists numerous detection and defense schemes intended to protect a target model in a white-box or gray-box setting against adversarial perturbations. As the *luring effect* is conceptually new, it can be combined to any of these existing methods, to provide an enhanced protection. Indeed, our deceiving-based approach acts on the way $M \circ P$ performs inference relatively to M. As the effort is focused on the design of P, another defense method, focusing on M and designed to protect M, can be combined with our scheme. We choose here to consider the combination with adversarial training [75], a state-of-the-art approach for robustness in the white-box setting. The model M is already trained with

adversarial training. The l_∞ bound constraint for PGD is set to 0.3 for MNIST and 0.03 for both SVHN and CIFAR10. During training, the PGD is performed for 40 steps, and a step size of 0.1 for MNIST. The PGD is performed for 10 steps with a step size of 0.008 for both SVHN and CIFAR10. We report in Table 7.17 the accuracy and agreement (augmented and target models agree on the ground-truth label) between the target model M and its augmented version T, whether M is trained classically or with adversarial training.

Table 7.17: Test set accuracy and agreement (augmented and target model agree on the ground-truth label) between an augmented luring model T and a target model M. Base and Base–Luring denote respectively a target model trained classically and the augmented model trained with the luring defense. AdvTrain and AdvTrain–Luring denote respectively a target model trained with adversarial training and the augmented model trained with the luring defense.

Model	Data set					
	MNIST		SVHN		CIFAR10	
	Test	Agree	Test	Agree	Test	Agree
Base	0.991	–	0.961	–	0.893	–
Base–Luring	0.970	0.970	0.920	0.920	0.850	0.820
AdvTrain	0.990	–	0.930	–	0.790	–
AdvTrain–Luring	0.960	0.960	0.870	0.850	0.780	0.760

In Table 7.18 we report the Acc_M and DAC values corresponding to the same threat model as the one used throughout this section, against the strong **MIM-W** attack. Moroever, we add the accuracy in a white-box setting of the model M against the PGD attack with 40 iterations and a step size of 0.01, denoted as $AC_{M,wb}$. Interestingly, we observe a productive interaction between the two defenses with DAC values that have greatly increased for the three data sets, as well as the Acc_M values for MNIST and CIFAR10. Interestingly, we note that the joint use of these defenses clearly improves the detection performance, with DAC values superior to 0.8 for the three data sets as well a strong improvement of the Acc_M metric for MNIST (0.97) and CIFAR10 (0.85).

Table 7.18: $\text{Acc}_{M,wb}$, Acc_M and DAC values (among FGSM, MIM, DIM, MIM-TI and DIM-TI) for the luring approach. Base–Luring and AdvTrain–luring denote the cases where the model M is trained respectively classically and with adversarial training.

		Luring			AdvTrain–Luring		
	ϵ	$\text{Acc}_{M,wb}$	Acc_M	DAC	$\text{Acc}_{M,wb}$	Acc_M	DAC
MNIST	0.3	0.0	0.69	0.75	0.82	0.97	0.98
SVHN	0.03	0.0	0.81	0.87	0.45	0.898	0.911
CIFAR10	0.03	0.0	0.30	0.39	0.41	0.85	0.88

7.4.6 Discussion and Related Work

A good practice in the field of adversarial robustness is to consider an adaptive adversary [81]: an attacker using some knowledge of the defense method to bypass it more efficiently. In our case, the only information that could be given to an adversary without breaking the black-box threat model is the supplementary knowledge that the augmented model has been trained with the luring loss. Indeed, M has to stay hidden, otherwise the adversary could use a decision-based attack to thwart the defense [63, 130]. Moreover, with a more permissive access to T, for example a white-box access to T, the adversary could simply extract M from the augmented model, and then defeat the defense. The additional information of the luring loss, would not bring any advantage to the adversary in order to craft adversarial perturbations on T which transfer to M. Indeed, it would imply for the attacker to inverse the effect of the luring loss on the prediction output vector of T, which appears as a prohibitive effort. Therefore, respecting the threat model, an adaptive adversary would not be more efficient than the adversary considered here.

Defenses in the black-box context are weakly covered in the literature as compared to the numerous approaches focused on white-box attacks. More particularly, very few approaches deal with query-based black-box attacks, such as the recent BlackLight [136]. A honeypot-based approach has been recently proposed [218] with a *trapdoored model* to detect adversarial examples. Even if the threat model (white-box) is different from ours, this approach is conceptually close to our deception approach, and it is interestingly mentioned that transferability between the target and the trapdoor-protected model is very poor. More precisely, Shan *et al.* [218] introduce trapdoors to detect white-box adversarial examples attacks in order to trick the adversary's attack into converging towards specific adversarial directions, which later enables a straightforward detection. To this end, to defend a model against targeted adversarial examples towards label y_t, the clean data set of input-label pairs (x, y) is augmented with *trapdoored* pairs $(x + \Delta, y_t)$, where Δ is a specific trigger. Then, the target model is trained by minimizing the cross-entropy between the clean and trapdoored input-label pairs to obtain the *trapdoored model*. An input x is detected as a targeted adversarial example thanks to a threshold-based test, that compares a "*neuron activation signature*" of this input against that of the embedded trapdoors. Note that the authors extend their approach to untargeted adversarial examples. We refer to the article [218] for more details.

The authors mention that transferability between the target and the trapdoored model is very poor. Based on the provided code[3], we train trapdoored models with the architectures we used for M. We observe that the transferability is hight from M to the trapdoor version but, on the contrary, the transferability is weak when adversarial examples are crafted on the trapdoored model and transferred to the target model. For the latter scenario, we report in Table 7.19 the worst Acc_M and DAC values for FGSM, MIM, DIM, MIM-TI and DIM-TI along with those of our method. The parameters used to run the attacks are the same as described in Section 7.4.3. These results would correspond to a defense similar to ours where the public model is not the augmented model $M \circ P$, but the trapdoored model trained from M. The threat model stays the same, corresponding to an adversary which wants to transfer adversarial examples from the public model to the target model M. Results in Table 7.19 on MNIST and CIFAR10 for DAC indicate that using trapdoors could allow for a more efficient detection of adversarial examples, especially for large adversarial perturbations. Results for the Acc_M values on the three data sets also corroborate the idea that both methods could benefit from each other. By fitting the trapdoor approach in our black-box threat model we highlight complementary performances in terms of AC_M and DAC metrics, meaning that an overall deception-based strategy is a promising direction for future work.

Table 7.19: Optimal Acc_{MoP}, Acc_M and DAC values (among FGSM, MIM, DIM, MIM-TI and DIM-TI) for the Trapdoor and Luring approaches.

		Trapdoor			Luring		
	ϵ	Acc_{MoP}	Acc_M	DAC	Acc_{MoP}	Acc_M	DAC
MNIST	0.2	0.06	0.82	**0.97**	0.19	**0.92**	0.93
	0.3	0.0	0.54	**0.94**	0.13	**0.69**	0.75
	0.4	0.0	0.27	**0.87**	0.07	**0.34**	0.45
SVHN	0.03	0.1	0.48	0.58	0.11	**0.81**	**0.87**
	0.06	0.07	0.38	0.48	0.0	**0.58**	**0.71**
	0.08	0.01	0.23	0.38	0.0	**0.48**	**0.67**
CIFAR10	0.02	0.0	**0.71**	**0.84**	0.03	0.58	0.64
	0.03	0.0	**0.58**	**0.77**	0.0	0.30	0.39
	0.04	0.0	**0.45**	**0.78**	0.0	0.16	0.27

7.5 Using the luring effect as a gray-box defense

In this section, we design another way of taking advantage of the *luring effect*, to be able to consider a more permissive threat model, especially relatively to the setting in which the deployed model is considered.

[3] https://github.com/Shawn-Shan/dnnbackdoor

7.5.1 Threat model

We consider a target model M to protect against transfer black-box attacks. The objective is to train a substitute model M', to which the adversary has a white-box access, and embed on it the luring effect so that adversarial examples targeting M' do not transfer to M. Contrary to Section 7.4, the substitute model M' has to be fully available to the adversary. In this scenario, the defender thus can not leverage an additional component as in Section 7.4 to induce the *luring effect* for an augmented model $T = M \circ P$. Indeed, the adversary would then simply separate P from M, and craft adversarial examples on M.

This threat model corresponds to the real-life scenario where the target model is part of a system with strong security requirements: the model M is thus hidden from potential adversaries. Meanwhile, the ML practitioner that owns the model M want to release a mainstream version of the model M, the model M', and logically wants to guarantee that adversarial examples targeting M' do not transfer to M.

The goal of the defender is to limit the transferability from the public model M' to the hidden target model M. As for Section 7.4, the defender can choose to rely on the luring effect for leading to unsuccessful adversarial examples and look only at $M(x)$. In this case, the adversarial accuracy Acc_M is the appropriate metric. The adversary can also choose to compare the output from M' and M. In this case, the DAC metric (Equation 7.7) is an appropriate metric.

7.5.2 Implementation

The idea is to take advantage of the luring effect demonstrated for a black-box setting in the previous sections, and to implement it on a model M'. In order to implement it, we require M' to have the same architecture of M. The parameters of M are frozen during the training of M', and M' is initialized with the parameters of M. This allows to encompass since the beginning the way M performs prediction in M', which is necessary to induce the luring effect. Then, to induce the luring effect for M', we train only the first layers of M' with a specific loss function intended, and inspired from the loss used in a black-box setting (Equation 7.3). Importantly, the usage of the *luring effect* in this scenario also does not require a labeled data set.

Note that as M' has the same architecture as M, this ensures a strict evaluation of our method. Indeed, adversarial examples transfer better between two models with the same architecture [213].

The loss with which the first layers of M' are trained is described in Equation 7.8, and the whole procedure is described in Algorithm 2 and hereafter.

For an input $x \in \mathcal{X}$, we denote by $h^{M'}(x)$ the logits for the model M'. $h_i^{M'}(x)$ corresponds to the value of $h^{M'}(x)$ for class i. The classes a and b correspond to the second maximum value of h^M and $h^{M'}$ respectively.

To train the substitute model M', our method is as follows:

- **(1)** We initialize the parameters of M' with the parameters of M.
- **(2)** We freeze the last k layers of M' and train only the remaining layers.
- **(3)** We train the remaining layers of M' with the loss described in Equation. 7.8

$$\begin{aligned}\mathcal{L}_{lur-gray}\left(x, M(x)\right) = &-\lambda\left(h^{M'}_{M(x)}(x) - h^{M'}_{a}(x)\right) + \max\left(0, h^{M'}_{b}(x) - h^{M'}_{M(x)}(x)\right) + \\ &\max\left(c_1, \left|h^{M'}_{M(x)}(x) - h^{M'}_{a}(x)\right|\right) + \max\left(c_2, \left|h^{M'}_{M(x)}(x) - h^{M'}_{b}(x)\right|\right)\end{aligned} \tag{7.8}$$

The first two terms correspond directly with the original luring loss adapted to M'. The two other terms, including stability coefficients c_1 and c_2, have been added to respond to practical training issues, more precisely to avoid logits to grow exponentially.

Steps **(1)** and **(2)** are the essential parts for the implementation of the *luring effect* in our setting. Indeed, the main idea behind the *luring effect* is to train a model to rely on different features than another model. The model being trained thus needs to take into account the way the other model performs prediction. In the classical implementation of it (black-box setting), this was implemented as P was trained in a bigger model $T = M \circ P$. As the parameters of M were frozen, it ensured that P was trained upon the way M preforms prediction. In our setting, initializing M' parameters with the M ones and keeping for M' some layers with the weights of M allows for this. Then, the first layers being trained with the loss given in Equation 7.8 play the role of the component P of the black-box setting.

Algorithm 2 Luring effect for a gray-box setting

Input: trained model M, training steps K, learning rate η, batch size B, luring coefficient λ, number of layers k, stability coefficients c_1 and c_2
Output: luring model M', with parameters Φ

$h_c^{M'}(x)$ denotes the logits of M' for input x and class c
Initialize the parameters Φ of M' with the parameters of M
Freeze the last k layers for M'
for step $= 1 \dots K$: **do**
    $\{x_1, x_2, \dots, x_B\}$: batch of training set examples
    for $i = 1 \dots B$: **do**
        $(a,b) \leftarrow$ (class of the 2^{nd} max value of $h^M(x_i)$, class of the 2^{nd} max value of $h^{M'}(x_i)$)
        $\mathcal{L}_{lur-gray}(x_i, M(x_i)) \leftarrow \lambda \left(h^{M'}_{M(x)}(x) - h^{M'}_a(x) \right) + \max \left(0, h^{M'}_b(x) - h^{M'}_{M(x)}(x) \right) +$
        $\max \left(c_1, \left| h^{M'}_{M(x)}(x) - h^{M'}_a(x) \right| \right) + \max \left(c_2, \left| h^{M'}_{M(x)}(x) - h^{M'}_b(x) \right| \right)$
    end for
    $\Phi \leftarrow \Phi - \eta \sum_{i=1}^{B} \nabla_\Phi \mathcal{L}_{lur-gray}(x_i, M(x_i)) / B$
end for
return M'

7.5.3 Experiments of characterization

We train different substitute models M' with different loss functions. This allows to identify the contributions of the steps **(1)**, **(2)** and **(3)** of our method. The different models considered are as follows:

- ***Oth_Init* model**: The last k layers are frozen. The weights of M' are initialized with the weights of M. Then, the non-frozen layers of M' are trained with another weight initialization scheme than M. This allows to evaluate the transferability for the same architecture, the same frozen layers, but a different weight initialization scheme for the first layers.
- ***C_E* model**: The last k layers are frozen. The weights of M' are initialized with the weights of M. Then, the non-frozen layers of M' are trained by minimizing the cross-entropy between the confidence score of M and M'. This allows to evaluate a training scheme not implementing the *luring effect*. Moreover, by training M' this way, we make M and M' very likely to rely on the same features (more details in the results).
- ***Lur_all* model**: No layer is frozen. The weights of M' are initialized with the weights of M. M' is trained (entirely) with the gray luring loss. This allows to evaluate the pertinence of the step **(2)**, i.e. to justify that including the way M performs prediction in the training of M' is meaningful.

Table 7.20: Test set accuracy and agreement (M' and M agree on the ground-truth label) between each M' model and the target model M, noted Base.

Model	Data set					
	MNIST		SVHN		CIFAR10	
	Test	Agree	Test	Agree	Test	Agree
Base	0.992	–	0.948	–	0.899	–
Oth_Init	0.991	0.990	0.950	0.940	0.884	0.840
C_E	0.992	0.991	0.949	0.943	0.898	0.897
Lur_all	0.990	0.987	0.949	0.929	0.899	0.874
Luring	0.961	0.959	0.922	0.904	0.856	0.837

Note that training a model with the luring loss but without weight initialization leads to a model without correct accuracy. Therefore, aside from its usefulness to implement the *luring effect* by taking into account the way M performs prediction, the step **(1)** is mandatory.

We consider the same architecture, training setup, attack methods and parameters as for Section 7.3. For MNIST, all except the first layer are frozen. For SVHN and CIFAR10, all except the first

12 layers are frozen The accuracy and agreement (models M and M' agree on the ground-truth label) for the three data sets are presented in Table 7.20. We note that our method does not induce a significant accuracy loss.

We attack the model M' with FGSM, PGD, MIM and MIMl2. The adversary capacity is an upper-bound on the l_∞ adversarial distortion $\|x' - x\|_\infty$ set to $\epsilon = 0.03$ for SVHN and CIFAR and to $\epsilon = 0.3$ for MNIST. The results for the Detection Rate (DR) and Inefficient Adversarial examples Rate (IAR) are reported in Tables 7.21, 7.22 and 7.23 for MNIST, SVHN and CIFAR10, respectively.

Table 7.21: MNIST. DR and IAR results ($\epsilon = 0.3$).

		Oth_Init	C_E	Lur_all	Lur
FGSM	IAR	0.10	0.01	0.34	**0.71**
	DR	0.17	0.03	0.75	**0.93**
PGD	IAR	0.0	0.0	0.20	**0.81**
	DR	0.0	0.0	0.33	**0.93**
MIM	IAR	0.0	0.02	0.12	**0.78**
	DR	0.0	0.04	0.29	**0.91**
MIMl2	IAR	0.0	0.0	0.27	**0.97**
	DR	0.01	0.0	0.51	**0.99**

Table 7.22: SVHN. DR and IAR results ($\epsilon = 0.03$).

		Oth_Init	C_E	Lur_all	Lur
FGSM	IAR	0.03	0.01	0.19	**0.32**
	DR	0.18	0.09	0.44	**0.61**
PGD	IAR	0.03	0.0	0.21	**0.43**
	DR	0.03	0.0	0.29	**0.55**
MIM	IAR	0.03	0.01	0.23	**0.40**
	DR	0.03	0.01	0.34	**0.55**
MIMl2	IAR	0.0	0.0	0.11	**0.13**
	DR	0.0	0.0	0.30	**0.30**

Table 7.23: CIFAR10. DR and IAR results ($\epsilon = 0.03$).

		Oth_Init	C_E	Lur_all	Lur
FGSM	IAR	0.14	0.0	0.05	**0.26**
	DR	0.42	0.0	0.2	**0.66**
PGD	IAR	0.06	0.0	0.14	**0.19**
	DR	0.06	0.0	0.15	**0.43**
MIM	IAR	0.04	0.0	0.18	**0.20**
	DR	0.05	0.0	0.21	**0.44**
MIMl2	IAR	0.0	0.0	0.06	**0.26**
	DR	0.0	0.0	0.23	**0.28**

For MNIST and SVHN, the IAR and DR results clearly validate our approach. For CIFAR10, the difference for the IAR metric between our approach and the *Lur_all* model is not always very significant. However, this is the case for the DR rate, which characterizes the most the *luring effect*.

For the three data sets, the IAR and DR results are almost 0 for the C_E approach. This was an expected result as M and M' are forced to predict the same confidence score. Indeed, this can be seen as the inverse of the *luring effect*, where we train models to rely on different features by acting on the logits sequence order. The training of model C_E can be seen as forcing M and M' to rely mostly on the same features.

Moreover, we train for each data set a model from scratch and using a different weight initialization scheme. For the FGSM, PGD and MIM attacks, in almost all cases, this model gives better or equivalent results than the model Lur_all. This strengthens the fact that only applying the luring loss to another model, but not retaining the way M performs prediction, allows for little improvement over simply training another model.

7.5.4 Results as a defense

After having characterized the efficiency of our implementation of the luring effect in a gray-box setting, we now present results when attacking the model M' with state-of-the-art *white-box* transferablity tuned attacks.

We consider the MIM-W attack (see Section 7.4) run on $1;000$ correctly classified test set examples, with parameter values tuned for the highest transferability results (same parameters tuning as in Section 7.4).

Results for MNIST, SVHN and CIFAR10 are reported in Tables 7.24, 7.25 and 7.26, respectively. We denote by $Acc_{M'}$ the adversarial accuracy for the model M'.

Table 7.24: MNIST. $Acc_{M'}$, Acc_M and DAC results for MIM-W ($\epsilon = 0.3$).

Oth_Init			C_E			Lur_all			Lur		
$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.21	0.0	**0.52**	**0.79**

Table 7.25: SVHN. $Acc_{M'}$, Acc_M and DAC results for MIM-W ($\epsilon = 0.03$).

Oth_Init			C_E			Lur_all			Lur		
$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC
0.0	0.03	0.03	0.0	0.01	0.01	0.0	0.23	0.31	0.0	**0.35**	**0.50**

Table 7.26: CIFAR10. $Acc_{M'}$, Acc_M and DAC results for MIM-W ($\epsilon = 0.03$).

Oth_Init			C_E			Lur_all			Lur		
$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC	$Acc_{M'}$	Acc_M	DAC
0.0	0.03	0.04	0.0	0.0	0.0	0.0	0.16	0.17	0.0	**0.17**	**0.45**

These results clearly assess the efficiency of using the *luring effect* in a more permissive setting. As already observed in the precedent section, for CIFAR10, using a detection scheme would be more profitable than simply relying on the prediction of M. Indeed, for CIFAR10, DAC results are much higher with our approach than the other approaches considered. This difference is less sharp for the DAC metric.

Comparison with the black-box setting

We compare the results obtained in the gray-box setting with the ones obtained in the black-box setting (Section 7.4, Table 7.13). For more clarity, we present this comparison in Table 7.27, for the MIM-W attack, $\epsilon = 0.3$ for MNIST, and $\epsilon = 0.03$ for SVHN and CIFAR10.

In terms of the DAC metric, we notice no superiority of a particular setting. However, for the black-box scenario, results for the metric Acc_M are clearly higher than for the gray-box scenario. We claim that in the black-box setting, as the whole model M is kept, the luring effect is more strongly induced than for the gray-box scenario, where the way M performs prediction is only kept partially. Moreover, in the gray-box setting, as models M' and M share the same architecture, the transferability is more likely to be higher than for the black-box setting where models T and M do not have the same architecture.

Table 7.27: Comparison of the black-box and gray-box settings in terms of Acc_M and DAC metrics, for $\epsilon = 0.3$ (MNIST) and $\epsilon = 0.03$ (SVHN and CIFAR10).

Setting	Data set					
	MNIST		SVHN		CIFAR10	
	Acc_M	DAC	Acc_M	DAC	Acc_M	DAC
Black-box	**0.69**	0.75	**0.58**	**0.71**	**0.30**	0.39
Gray-box	0.52	**0.79**	0.35	0.50	0.17	**0.45**

7.6 Using the luring effect to detect the crafting process of adversarial examples

In this section, we propose another protection scenario taking advantage of the *luring effect*: detecting and stopping the crafting process of black-box adversarial examples.

7.6.1 Threat model

We consider a system $\mathbb{S}$ to protect against black-box adversarial examples. Considering an adversary leveraging a method to craft a black-box adversarial example x' from a clean input x, this method requires multiple queries to be made to the system $\mathbb{S}$. Our goal is to detect as soon as possible (i.e. after the fewer queries as possible) this type of attack, to mitigate it.

To implement this method, our idea is to rely on the *luring effect* by designing $\mathbb{S}$ to be composed of M and M', where M' is the model trained for the gray-box scenario (Section 7.5). As the *luring effect* causes M' and M to disagree on adversarial examples, we hope that it will allow M and M' to disagree at a very early stage of the crafting process of x'. More precisely, considering $x^1, x^2, \ldots, x^{T-1}$ the inputs submitted to $\mathbb{S}$ as part of the crafting process of $x' = x^T$ from $x^0 = x$, we hope that:

- M and M' will disagree on x^i with the smaller i possible.
- M and M' agree on the ground-truth label y on x^j for $j < i$.

In accordance with these objectives, we choose the system $\mathbb{S}$ to give an output only if M and M' agree (i.e. $M(x) = M'(x)$ for an input x). Moreover, as the *luring effect* is efficient at thwarting transferability from M' to M, the output given in case of label-agreement between M' and M corresponds to the output given by M'. By doing so, firstly, the system $\mathbb{S}$ is expected to answer the correct label for clean examples *and* inoffensive queries of the adversarial example crafting process. Secondly, when M and M' disagree, it will thwart the adversary, as no response is given: in the absence of a output, the adversary can't continue the adversarial crafting process.

We consider here an adversary having access to the logits of M'. Therefore, the adversary is allowed to use score-based black box adversarial attacks needing access to the logits. The adversary capacity is an upper-bound on the l_∞ adversarial distortion $\|x' - x\|_\infty$ set to $\epsilon = 0.03$ for SVHN and CIFAR and to $\epsilon = 0.3$ for MNIST.

7.6.2 Attacks

In order to evaluate the efficiency of the luring effect in this scenario, we consider the SPSA and ECO attacks with the same parameters than in Section 7.4: SPSA and ECO. SPSA is run with 100 iterations. The x^i correspond to the intermediate adversarial example obtained after each of

these iterations. ECO is run with a limit of $20,000$ queries. The x^i correspond to the intermediate adversarial examples obtained after each local search optimization. Note that for a fixed number of queries, the number of intermediate adversarial obtained is not the same for different examples. We choose to keep only the first 100 intermediate adversarial examples.

As the methods to craft black-box adversarial examples are continuously evolving, we decide to also present results resulting from white-box attacks tuned for transferability. This type of attacks computes gradients exactly, and is thus enables to consider a kind of optimal black-box attack. To this end, we consider the DIM attack with parameter $p = 0.9$.

7.6.3 Baselines and metrics

As for baselines for comparison, we consider the following models:

- ***k-Oth_Init* model**: The last k layers are frozen. The weights of M' are initialized with the weights of M. M' is trained with another weight initialization scheme than M.
- ***C_E* model**: The last k layers are frozen. The weights of M' are initialized with the weights of M. M' is trained minimizing the cross entropy between the confidence score of M and M'.
- ***Lur_all* model**: No layer is frozen. The weights of M' are initialized with the weights of M. M' is trained (entirely) with the gray luring loss.

Consequently, we note $\mathbb{S}_{k-Oth\_Init}$, $\mathbb{S}_{C\_E}$, $\mathbb{S}_{Lur\_all}$ and $\mathbb{S}_{Lur}$ the systems composed of M and respectively the models $k-Oth\_Init$, C_E, Lur_all and Lur. As already specified, when querying a system $\mathbb{S}_.$, if M' and M agree, the adversary is provided with the output given by M'. The rate of clean test set examples well predicted by $\mathbb{S}$ is therefore simply the rate of test set examples for which M and M' agree. This accuracy of the different systems on clean examples is presented in Table 7.28

Table 7.28: Accuracy of the different systems (noted $Acc_{\mathbb{S}}$) on clean test set examples.

Model	Data set		
	MNIST	SVHN	CIFAR10
	$Acc_{\mathbb{S}}$	$Acc_{\mathbb{S}}$	$Acc_{\mathbb{S}}$
$\mathbb{S}_{k-Oth\_Init}$	0.990	0.940	0.840
$\mathbb{S}_{C\_E}$	0.991	0.943	0.897
$\mathbb{S}_{Lur\_all}$	0.987	0.929	0.874
$\mathbb{S}_{Lur}$	0.959	0.904	0.837

To measure the efficiency of our scheme, we consider an hypothetical adversarial example x' crafted from a clean input-label pair (x, y) on M'. We note $x^1, x^2, \ldots, x^{T-1}$ the $T-1$ queries, with $x^0 = x$ and $x^T = x'$. We say that an adversarial example is *detected at iteration* i if $M(x^i) \neq M'(x^i)$ while having been correctly predicted for the past $i-1$ iterations (i.e. $M(x^j) = M'(x^j) = y \; \forall j \in [\![1, i-1]\!]$). If during the process, there exits some $j \in [\![1, i-1]\!]$ for which we have $M(x^j) = M'(x^j) \neq y$, the adversarial example is therefore not considered as detected at iteration i. Considering m adversarial examples, we measure:

1. $Det(i)$: how many of them are detected at iteration $i < T$. For a given i, the higher the better.
2. $Mean\_Det$: the mean number of iterations at which adversarial examples are detected. The smaller the better.
3. Det: the proportion of them which are detected before the completion of the attack (i.e. how many of them are detected at some iteration i with $i < T$. The higher the better.
4. $Inof$: the proportion of them which are not detected for $i \leq T$, i.e. well-predicted during all the attack process (i.e. how many of them $M(x'_i) = M'(x'_i) = y \; \forall i \in [\![1, T]\!]$). This indicates the proportion of adversarial examples that are inefficient against M' and M at the same time.

5. *Tot*: the proportion of them which are either detected before the completion of the attack *or* well-predicted during all the attack process. This corresponds to the addition of 3. and 4.

The most important metrics to assess the efficiency of the luring effect in this scenario are reasonably $Det(i)$, $Mean\_Det$, and Det, whereas the $Inof$ metric is more illustrative and the Tot metric an indicator of the global performance of the system against adversarial examples.

We present hereafter the results for MNIST, SVHN and CIFAR10 data sets. For a better visualization of $Det(i)$, we present it with a Figure.

7.6.4 Results

MNIST

The results for the metric $Det(i)$ are presented in Figure 7.6. Results for other metrics ($Mean\_Det$, Det, $Inof$ and Tot) are presented in Table 7.29

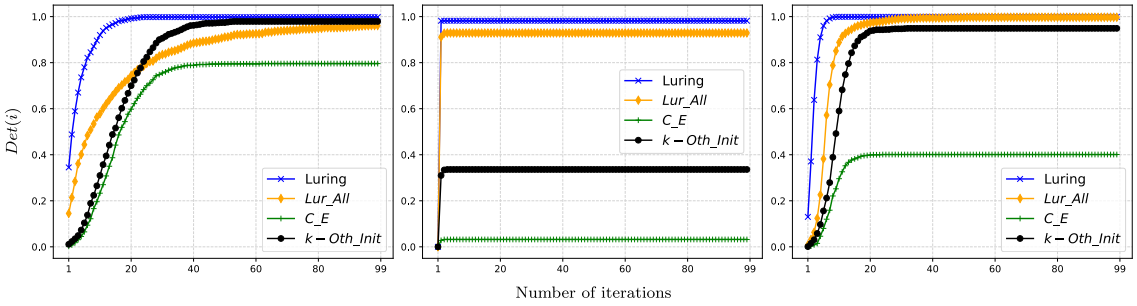

Figure 7.6: MNIST ($\epsilon = 0.3$). $Det(i)$ results. (left) SPSA. (middle) ECO. (right) MIM.

Table 7.29: MNIST. Results for the different metrics in detecting the crafting process of adversarial examples. All attacks are run with $\epsilon = 0.03$.

	Lur			Lur_All			C_E			k-Oth_Init		
	SPSA	ECO	DIM	SPSA	ECO	DIM	SPSA	ECO	DIM	SPSA	ECO	DIM
Mean_Det	**3.5**	**2.04**	**2.2**	17	9.26	7.3	33	132	61	18	93	14.1
Det	**0.998**	**0.982**	**0.999**	0.961	0.929	0.997	0.796	0.032	0.401	0.979	0.336	0.949
Inof	0.0	0.0	0.0	0.038	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Tot	0.998	0.982	0.999	0.999	0.929	0.998	0.796	0.03	0.401	0.979	0.336	0.949

For the $Det(i)$ metric, looking at the Figure 7.6, we notice very clearly that for the three types of attacks, our approach allows to detect way more adversarial examples at a lower iteration. Moreover, we notice that the mean number of iterations required to detect an adversarial example ($Mean\_Det$) is much lower with *Lur* than with the other schemes. We notice that for the three metrics Det, $Inof$ and Tot, values are slightly equivalent between Lur and Lur_All, while significantly higher than C_E and k-Oth_Init.

SVHN

The results for the metric $Det(i)$ are presented in Figure 7.7. Results for other metrics ($Mean\_Det$, Det, $Inof$ and Tot) are presented in Table 7.30

For SVHN, for the four metrics $Mean\_Det$,Det, $Inof$ and Tot, results are significantly better with our luring scheme than with other schemes. Notably, the number of adversarial examples detected before the completion of the adversarial examples crafting process is much greater with our scheme (see results for metric Det), as well as for the number of adversarial examples detected for each each iteration (cf Figure 7.7).

CIFAR10

The results for the metric $Det(i)$ are presented in Figure 7.8. Results for other metrics ($Mean\_Det$, Det, $Inof$ and Tot) are presented in Table 7.31

For CIFAR10, the same observations as for SVHN can be made regarding the results.

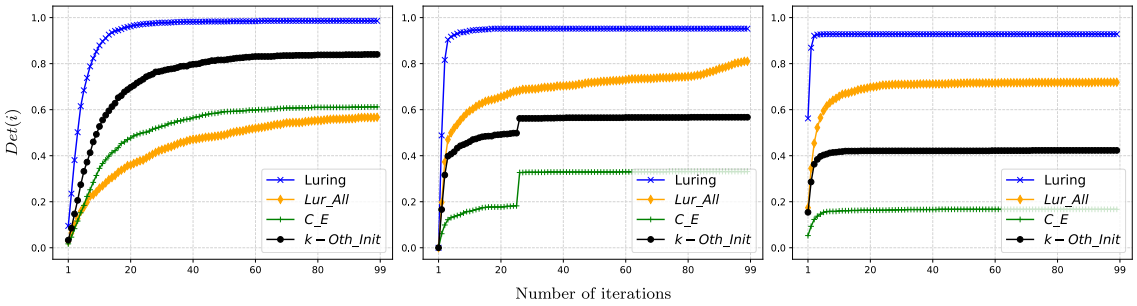

Figure 7.7: SVHN ($\epsilon = 0.03$). $Det(i)$ results. (left) SPSA. (middle) ECO. (right) MIM.

Table 7.30: SVHN. Results for the different metrics in detecting the crafting process of adversarial examples. All attacks are run with $\epsilon = 0.03$.

	Lur			Lur_All			C_E			k-Oth_Init		
	SPSA	ECO	DIM	SPSA	ECO	DIM	SPSA	ECO	DIM	SPSA	ECO	DIM
Mean_Det	**6.3**	**5.3**	**4.6**	55.1	29.3	24.5	47.3	63.1	82.1	41.6	41.6	56.8
Det	**0.986**	**0.952**	**0.93**	0.567	0.81	0.719	0.612	0.331	0.167	0.84	0.567	0.423
Inof	0.002	0.0	0.0	0.401	0.125	0.081	0.042	0.146	0.001	0.046	0.064	0.004
Tot	0.998	0.952	0.928	0.968	0.935	0.8	0.654	0.477	0.168	0.886	0.631	0.427

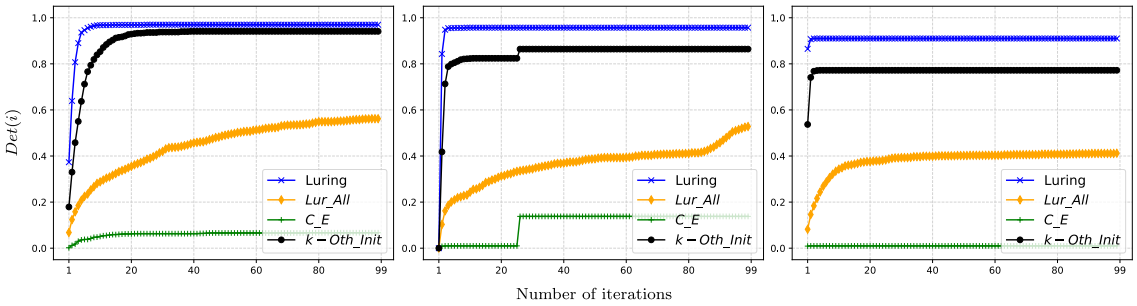

Figure 7.8: CIFAR10 ($\epsilon = 0.03$). $Det(i)$ results. (left) SPSA. (middle) ECO. (right) MIM.

Table 7.31: CIFAR10. Results for the different metrics in detecting the crafting process of adversarial examples. All attacks are run with $\epsilon = 0.03$.

	Lur			Lur_All			C_E			k-Oth_Init		
	SPSA	ECO	DIM	SPSA	ECO	DIM	SPSA	ECO	DIM	SPSA	ECO	DIM
Mean_Det	**4.3**	**3.8**	**5.5**	55.2	63.4	59.5	93.4	81.2	98.9	9.4	15.1	22.1
Det	**0.97**	**0.957**	**0.91**	0.562	0.528	0.412	0.067	0.138	0.01	0.941	0.864	0.772
Inof	0.0	0.0 0.0	0.29	0.194	0.116	0.0	0.128	0.0	0.0	0.04	0.0	
Tot	0.97	0.957	0.91	0.852	0.722	0.528	0.067	0.266	0.01	0.941	0.904	0.772

7.7 Conclusion and future work

In this chapter, we designed and studied the *luring effect*, a conceptually innovative approach to improve the robustness of a model against transfer black-box adversarial perturbations. This concept, which basically relies on a deception strategy, is inspired by the notion of robust and non-robust features for transferability. The *luring effect* is implemented on a decoy model, with a specific loss, which tricks the adversary into targeting on the decoy model different non-robust features than the ones of the target model. Importantly, the *luring effect* only relies on the logits of the target model. It does not require a labeled data set and therefore can by applied to any pre-trained model.

We characterized the *luring effect* on MNIST, SVHN and CIFAR10 and show that it allows to successfully thwart an adversary using state-of-the-art black-box transfer attacks with large allowed adversarial budgets.

We proposed three possible scenarios, with different threat models, where the defender can take advantage of the *luring effect*. Notably, two scenarios relate to thwarting the transferability of adversarial perturbations, and one scenario considers detecting the crafting process of transferable adversarial examples.

Future work. The concept behind the *luring effect* is innovative as it aims at *fooling the adversary*. It does neither try to make the target model more robust in itself, nor implement a reactive strategy. In this work, the *luring effect* has been designed to thwart transferability, a major concern relatively to the well-spread black-box scenario, and a core subject of this thesis. However, we believe that this paradigm of a deception strategy to thwart an adversary could pave the way towards a new type of defense methods, beyond the scope of transferability concerns.

Chapter 8

Leveraging adversarial attacks against the availability of a ML system

In the previous chapters, we focused on the phenomenon of adversarial examples, posing a risk to the integrity of models. More specifically, we studied the link between robustness and data specific frequency properties, analyzed the impact of quantization on the adversarial robustness, and derived a novel way to thwart the transferability of adversarial perturbations.

In this last chapter, we study how the mechanism of adversarial perturbations can be leveraged to attack the availability of a system. This chapter has to be seen as an exploratory work of the link between adversarial examples (integrity) and other threats to a machine learning model.

Whereas threats against integrity and confidentiality can be formalized at a model-level, availability threats require to consider how the model interacts with an overall system with specific quality and performance requirements. More particularly, in most cases, a system relying on a model exploits the model outputs *if and only if* these predictions reach a certain level of confidence. Here, we design an availability-based attack that targets, at inference time, the quality of the model prediction. The goal is not to fool the model but to inject *uncertainty*, which leads to confidence issues for a system about the underlying model. By using the same principle as a state-of-the-art adversarial example crafting method (PGD), we design two strategies to build *uncertain examples*, which trick the model into making uncertain predictions. Interestingly, by their test-time nature, *uncertain examples* allow to gain insight on the efficiency of existing defense schemes against adversarial examples when confronted with this new type of threat.

8.1 Motivation

Although reaching impressive performances on complex and critical tasks, the deployment of deep neural network models suffers from major threats that concern their *integrity*, *confidentiality* and *availability* [65], as presented in Chapter 2. Confidentiality and integrity constitute an important body of works in the last decade, with major and critical breaches and flaws that urge the development of sound evaluation methodologies [82], as well as robust and certified defense schemes. On one side, confidentiality-based attacks mainly aim at recovering information about the training data [37, 53] or reverse-engineering a black-box model [219]. On the other side, the objective of integrity-based attacks is to alter the performance of the model for the task it has been designed for. To this end, *poisoning attacks* [220] act at training time, while *adversarial examples* [11] maliciously modify inputs at inference time.

Importantly, as mentioned in [65], contrary to integrity and confidentiality-based threats which can be viewed at a model-level, availability has a broader definition and has to be regarded at a system-level, i.e. by taking into account the overall system (its requirements and constraints) in which the model takes place.

Here, we discuss and formalize a novel type of inference-time attack against availability, which threatens the quality of predictions of an already trained model, but does not aim at fooling it. Thus, relatively to its expected main performance metric (e.g. the accuracy), the model sill provides the *correct* prediction but with a high level of uncertainty, which deeply impacts how the system will use these outputs. Actually, with this first study, we claim that for many application domains, the *accuracy* of the model is not – and *cannot* be – sufficient to make it *reliable* because its level of confidence is also a major system requirement. Therefore, at inference time, an attacker can choose to target the *accuracy* or the *reliable* nature of the model predictions. In that sense, we introduce *uncertain examples*, which aim at increasing the *uncertainty* of the model predictions. Here, uncertainty is directly related to system requirements about the confidence score vectors.

Our contributions are as follows:

- We discuss the security concept of availability in the case of a ML-based system that takes benefit from a deep neural network, and its close link with integrity. We introduce the notion of *uncertainty* in the light of the system confidence requirements.
- We present methods to craft *uncertain examples*, which are inference-time attacks taking advantage of typical adversarial perturbations, not to fool a model but to inject uncertainty in its predictions.
- We evaluate our attack on CIFAR10 [24] and explore its impact on models trained with Adversarial Training [75], which allows to expose meaningful links with classical integrity-based attack such as adversarial examples.

8.2 Availability of a ML-based system

Model and system level. Integrity and confidentiality threats classically set within a perimeter defined by the model itself and its related training and inference pipeline. On the contrary availability threats make sense at a larger level, i.e. by considering the interactions between the model and the overall system. More particularly, important system features are related to the component(s) that process the model input or the one(s) dedicated to model output in order to make an *operational decision* with regards to the final task of the system. Therefore, the availability of a system is defined through system performance and quality requirements. These requirements concern a large variety of features such as processing time (e.g. inference latency), memory limitations, network bandwidth, confidence / certainty level or overall task-centered performances.

Objective. In consequence of this divergence of scope, the objective of an integrity or confidentiality-based attack is focused and well defined (e.g., fool the model, extract model architecture and internals or training data) whereas an availability threat may have different goals and attack vectors. More precisely, an adversary may target the *direct access* to the system or its *quality*. The first case typically concerns denial-of-service (DoS) attacks that can exploit flaws related to the software or hardware environment in which the model takes place. For example, a server hosting a model may not be scaled to process thousands of simultaneous queries. Regarding the quality of the system, the goal of the attacker is to jeopardize the ability to fit some requirements usually related to memory, energy or processing time constraints such as the recent *sponge examples* [66]. We introduce an attack against the quality of a system, which directly targets the model in terms of task-based performance metrics.
For this task-based oriented availability attacks, there exists a close link with integrity threats, which is crucial to note. In the case of a poisoning attack, or for adversarial examples, modifying the confidence scores is a goal in itself. In the case of an attack against the quality of a system focused on task-based performances, the final goal is to impact the behavior of the system in which the model takes place.

Overall and Relative Confidence Our work concerns an inference time attack, which targets the *quality* of the model in terms of task-based performances, by acting on the *uncertainty* of the predictions. On a high-level, the *uncertainty* of a confidence score vector relates to the fact that the system decides whether or not he can trust the model for this prediction. For the system requirement, we distinguish between *overall* and *relative* confidence. These concepts are both formalized in the Section 4.2. On one hand, *overall confidence* relates to a model being not confident enough in the predicted label. A typical requirement is that the maximum confidence score must be superior to a certain threshold, otherwise the system is not able to make a decision or, to do so, requires additional models or data (or even a human expertise) that may be time consuming. On another hand, *relative confidence* concerns the case when there is too much doubt between several possible labels. Here, the reliability of the prediction is based on the fact that the biggest confidence score must be far ahead of other ones. As an illustration, let's consider a classification task with 5 labels. For an input correctly classified in the first class, a confidence score vector like $(0.51, 0.49, 0, 0)$ or $(0.34, 0.33, 033, 0)$ may raise serious doubt about the reliability of the system.

8.3 Attacks against a ML-based system

Recently, Shumailov *et al.* propose *sponge examples* [66], which target the consumption and latency of a device. The goal of this attack is to alter an input so that the model is more *activated* compared to the inference with the clean sample. To the best of our knowledge, this is the first availability-based attack at inference-time focused on system-based performances (here, energy consumption and inference speed). The same author proposes an attack on the training process by reordering the training data (BRRR: batch reordering, reshuffling, replacing) [67]. Identical conclusions have been drawn in [68] where the training process is fooled by an attack exploiting the initialization of the weights. In both cases, from an integrity point of view, modifying the randomness of the training batches or altering the initialization of the weights are close to what a typical data poisoning aims to achieve. However, both works show that such attacks can also impact the availability of the machine learning system since it can noticeably slow down or even reset the learning progress. These attacks are related to classical *data poisoning* attacks [22] that manipulate the training data to degrade the performance of the learning process only for very specific inputs or at a global-level.

However, data poisoning takes place at training time, and considers a very permissive threat model, where the adversary is supposed to be able to interfere during the training phase of the model. Whereas this can be seen as realistic for federated machine learning, where the adversary can be one of the multiple actors, it assumes a very powerful adversary in the classical scenario where only one actor collects data and trains the model.

8.4 Threat model

8.4.1 Adversary goal

The goal of the adversary is to mount an inference-time attack intended at injecting *uncertainty* in the confidence score vector of a target model. We begin by defining formally the *uncertainty* nature of a confidence score vector with *Overall Confidence (OC)* and *Relative Confidence (RC)*:

- *Overall Confidence (OC)* relates to a requirement that sets that the maximum confidence score must be superior to a certain threshold τ_o: $F_{M(x)}(x) \geq \tau_o$.
- *Relative Confidence (RC)* concerns the case where there is too much doubt between several possible labels. Here, the reliability of the prediction is based on the fact that the biggest confidence score must be far ahead of other ones: $F_{M(x)}(x) - F_j(x) \geq \tau_r$ with $j \neq M(x)$ and τ_r a threshold value.

An important remark is that an output prediction may be flagged as uncertain for one criterion but not the other, and vice versa. From an adversary point of view, considering a maliciously modified version of x, $x' = x + \delta$, increasing the uncertainty of predictions therefore resumes to the two following possible objectives:

Overall Confidence Uncertainty Attack (OC-UA):

$$\min_{x'} ||x - x'||_p \quad \text{s.t. } M(x') = M(x) \text{ and } F_{M(x)}(x') < \tau_o \tag{8.1}$$

Relative Confidence Uncertainty Attack (RC-UA):

$$\min_{x'} ||x - x'||_p \quad \text{s.t. } M(x') = M(x) \text{ and } F_{M(x)}(x') - F_j(x') < \tau_r \; \forall j \in \mathcal{T} \tag{8.2}$$

where $\mathcal{T}$ is a chosen subset of target classes.

Note that, in our threat model, we set that the predicted class is not altered by the attack ($M(x') = M(x)$). Firstly, by investigating if unsuccessful adversarial examples reach the goal of the OC-UA or RC-UA attacks, it allows to gain insight on the vulnerability of integrity defense schemes against *uncertain examples*, as presented in Section 8.6. Secondly, if the condition $M(x') = M(x)$ is not required, *uncertain examples* are a special case of adversarial examples, as they may both change the predicted label and reach the goal of the OC-UA or RC-UA attack. From the adversary point of view, this would result in an unclear goal. If the goal were to mount a OC-UA or RC-UA attack, he would have not aimed at changing the predicted label, as it corresponds to a more difficult objective.

8.4.2 Adversarial capacity and knowledge

The adversarial capacity defines how much the adversary is allowed to alter x. In order to gain insight on the difference between our attack and adversarial examples, we use the same theoretical framework and set an upper bound of the perturbation: $\|x' - x\|_\infty < \epsilon$. We consider an adversary in the white-box setting, with a full knowledge of the target model and a complete access to it. Moreover, the attacker has a relevant knowledge of the system requirements that define the conditions under which the system discard the model output. Therefore, the attacker knows τ_o and/or τ_r.

Algorithm 3 RC-UA attack with n iterations, perturbation budget ϵ, step size α, target classes $\mathcal{T}$ and threshold value τ_r

Input: clean example x, number of iterations n, perturbation budget ϵ, initial step size α, target labels $\mathcal{T}$, threshold value τ_r
Output: adversarial example x'

1: $y_c^t \leftarrow \frac{1}{|\mathcal{T}|+1}$ if $c \in \mathcal{T} \cup \{M(x)\}$, 0 otherwise
2: $i \leftarrow 0$
3: **while** $i < n$ and $F_{M(x)}(x') - F_j(x') \geq \tau_r$, $\forall j \in \mathcal{T}$: **do**
4: $\quad x'_i \leftarrow x'_i - \alpha\, sign\Big(\nabla_x \mathcal{L}\big(x'_i, y_c^t, M_\theta\big)\Big)$
5: $\quad x'_i \leftarrow \text{proj}_{B_\infty(x,\epsilon)}(x'_i)$
6: $\quad$ **if** $M(x'_i) \neq M(x)$ **then**
7: $\quad\quad x'_i \leftarrow x'_{i-1}$
8: $\quad\quad \alpha \leftarrow \frac{\alpha}{2}$
9: $\quad$ **end if**
10: $\quad i \leftarrow i + 1$
11: **end while**
12: **return** x'_n

8.5 Uncertain Examples

We propose an attack that injects uncertainty in the output confidences scores, following the threat model detailed in the previous section, for both the *Overall Confidence* (OC) and *Relative Confidence* (RC) objectives. Interestingly, we show that this attack achieves high performance with a simple adaptation from the state-of-the-art PGD attack [75], used without random initialization.

8.5.1 Relative Confidence Uncertainty Attack

Considering a set $\mathcal{T}$ of target classes, and a threshold value τ_r, we do not target explicitly the gap $F_{M(x)}(x') - F_j(x')\ \forall j \in \mathcal{T}$. Instead, we aim at having the *worst* score vector relatively to the ground-truth label y and the targeted labels, in terms of entropy. Indeed, the optimal uncertainty can be seen as the biggest entropy for $F(x')$, i.e. a uniform distribution for classes $M(x) \cup \mathcal{T}$, with the constant confidence value $\frac{1}{|\mathcal{T}|+1}$ for these classes. However, as the label associated to the highest score has to stay the same, the objective is thus to find the smallest ϵ such that:

$$\begin{aligned} F_{M(x)}(x') &= \frac{1}{|\mathcal{T}|+1} + \epsilon && && (8.3)\\ F_j(x') &= \frac{1}{|\mathcal{T}|+1} - \frac{\epsilon}{|\mathcal{T}|} && \forall j \in \mathcal{T} \\ F_j(x') &= 0 && \forall j \notin \mathcal{T} \cup \{M(x)\} \end{aligned}$$

In practice, we use a modified version of the *targeted version* of l_∞-PGD [75], a state-of-the-art iterative method to craft adversarial examples, used here without random initialization. The target output values are fixed to $\frac{1}{|\mathcal{T}|+1}$ for the classes in $\mathcal{T} \cup \{M(x)\}$. A simple trick is used to enforce that the *uncertain example* x' stays classified as the clean example x, and the process is stopped as soon as $F_{M(x)}(x') - F_j(x') < \tau_r$. More precisely, at the end of each iteration t, the condition $M(x'_t) = M(x)$ is checked. If not, x'_t is not updated and the step size α is halved (lines 7 and 8). This condition ensures that we always move towards the optimal uncertain prediction while ensuring a constant and correct classification. The RC-UA attack is described in Algorithm 3, and an illustration of it is presented in Figure 8.1.

8.5.2 Overall Confidence Uncertainty Attack

Here, the goal is simply to lower the confidence score corresponding to the true label, while ensuring that the example stays well-classified. A simple way to achieve it is to perform the RC-UA attack described above by modifying the condition for the main loop (line 3) to match the OC-UA objective, and replacing line 4 with $x'_i \leftarrow x'_i + \alpha\, sign\Big(\nabla_x \mathcal{L}\big(M(x'_i), y\big)\Big)$, that means to shift to the *untargeted version* of l_∞-PGD attack [100].

t

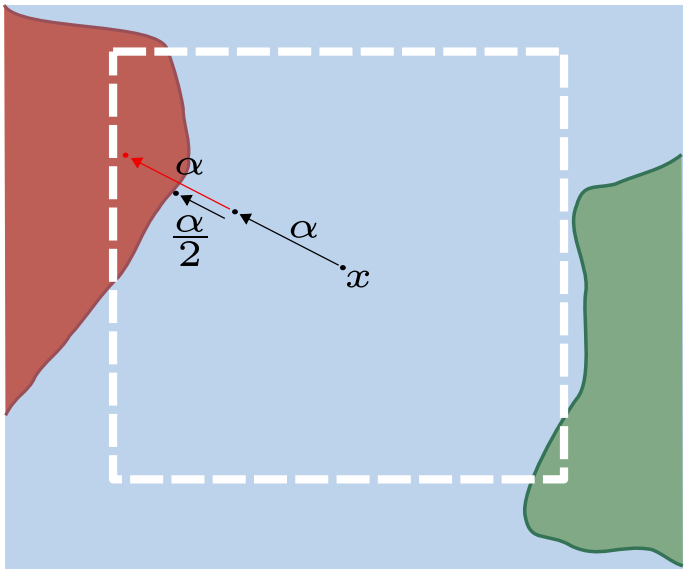

Figure 8.1: Illustration of the RC-UA attack. The white dotted square represents the l_∞ ball of radius ϵ centered at x. Each color denotes a different class.

8.6 Experiments

8.6.1 Data sets and models

We evaluate the uncertain examples on the CIFAR10 data set. For the model, we use the Wide-Resnet 28-8 architecture [186]. Additionally, we train a robust version of the model thanks to Adversarial Training (AT) [75]. The classically trained model achieves 93.3% accuracy on test set examples. The model trained with Adversarial Training achieves 84.1% accuracy on clean test set examples, and 55.1% accuracy against adversarial examples crafted with the l_∞-PGD attack with $\epsilon = 0.03$, $\alpha = 0.008$ and 100 iterations (all the accuracy values measured on adversarial examples presented in this section are computed with these parameters). For our attacks, to allow for comparison with adversarial examples, we also consider $\epsilon = 0.03$ and a starting step size $\alpha = 0.008$.

8.6.2 Uncertainty on standard examples

Depending on the requirements of the system, in our case the τ_o and τ_r values, the accuracy at the system-level is impacted. Indeed, as an example, for Overall Confidence (OC), for a clean example x and some value τ_o, if $F_{M(x)}(x) < \tau_0$, the system does not give a prediction. In Figures 8.2 and 8.3 are presented the accuracy at the system level for different values of confidence thresholds τ_o and τ_r. We notice that for the standard model (Figure 8.2), the confidence scores are such as except for high τ_r and τ_o values (values greater than 0.9), the accuracy at the system level is almost not impacted. For a model trained with Adversarial Training (Figure 8.3), the accuracy is much more impacted, especially for $\tau_r > 0.4$ and $tau_c > 0.6$. This result was expected as training with AT is much more difficult, and therefore the confidence scores are not as high as with standard training for the predicted class, and the difference between confidence scores for different classes is lower than with standard training.

8.6.3 Convergence of uncertain examples

As an undefended model is vulnerable to adversarial examples, uncertain examples, which are a weaker version of adversarial examples as no misclassification is induced, are thus feasible for each clean example. We consider the RC-UA attack, where $\mathcal{T}$ only contains one target class, and illustrate in Table 8.1 the number of iterations needed to perform this attack, for different values of the threshold τ_r. More precisely, for each value of τ_r, we measure the mean number of iterations, for all the possible pair of predicted and target classes, with $1,000$ test set examples for each pair. As expected, as the τ_r value decreases, the number of iterations needed increases. For comparison, adversarial examples crafted with the l_∞-PGD attack are successful at inducing

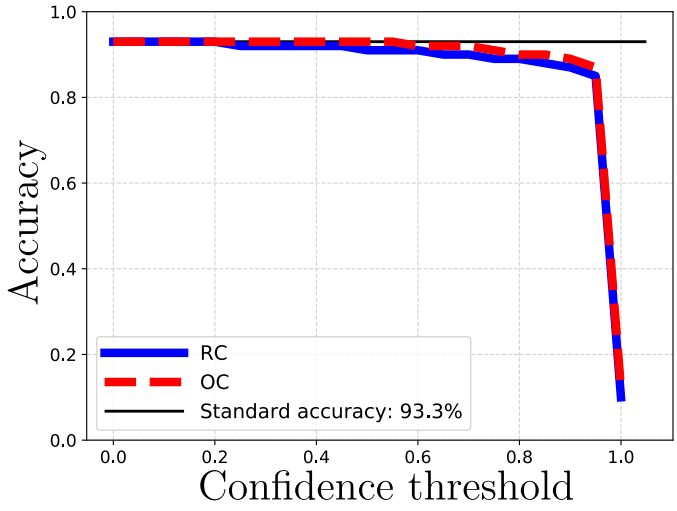

Figure 8.2: CIFAR10. Accuracy at the system-level for a standard model, depending on the τ_o and τ_r values of Overall Confidence (OC) and Relative Confidence (RC), respectively.

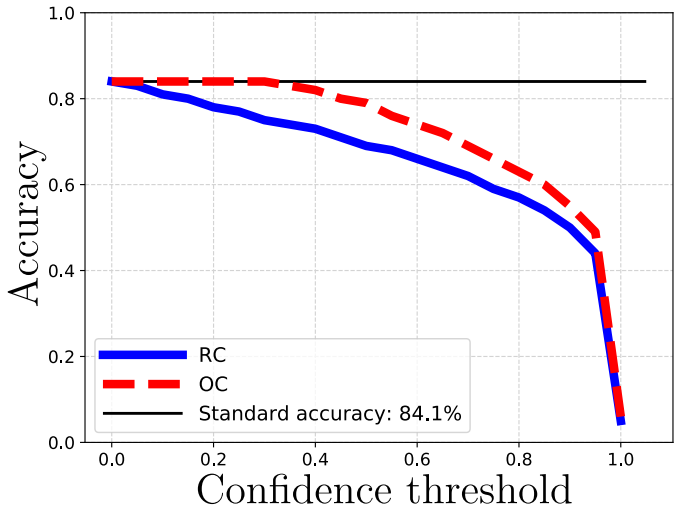

Figure 8.3: CIFAR10. Accuracy at the system-level for a model trained with Adversarial Training, depending on the τ_o and τ_r values of Overall Confidence (OC) and Relative Confidence (RC), respectively.

misclassification with at most 5 iterations. For a reasonable value of $\tau_r = 0.2$, the RC-UA attack needs on average 8.2 iterations, thus inducing only a slight supplementary burden for the adversary. More interestingly, for a small value of $\tau_r = 0.05$ (i.e. the difference between the confidence score of the predicted class and the second biggest confidence score is inferior to 0.05), only 12.2 iterations are required on average.

.

Table 8.1: Mean number of iterations ($Mean\_iters$) for the RC-UA attack, with various τ_r. The mean is performed for all the possible pairs (predicted, target), with $1,000$ test set examples for pair.

τ_r	0.05	0.1	0.15	0.2	0.25	0.3
$Mean\_iters$	12.2	10.2	9.0	8.2	7.5	6.8

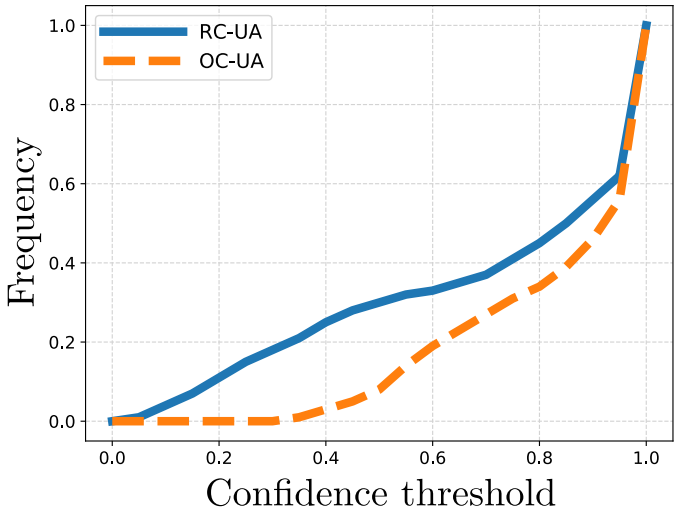

Figure 8.4: Percentage of unsuccessful l_∞-PGD adversarial examples for which the OC-UA or RC-UA is reached, for various confidence threshold values t.

8.6.4 Impact on Adversarial Training

Relatively to our threat model, we only consider the 55.1% of unsuccessful adversarial examples, i.e. the examples which are protected by AT against changes in the predicted label. We investigate the proportion of these adversarial examples for which the OC-UA or RC-UA is successful. To this end, we consider two ways of mounting the OC-UA or RC-UA:

- RC-UA: given a confidence probability vector $F(x'_{PGD})$ resulting from the l_∞ PGD-attack, the adversary chooses, for the target probability vector, the values 0.5 for classes y and the class corresponding to the second maximum value of $F(x'_{PGD})$. The rationale behind this choice is to consider the class corresponding to the confidence score the l_∞-PGD tried to maximize to achieve misclassification, as this corresponds to the *easiest* class to target.
- OC-UA: the initial adversarial examples crafted with the l_∞-PGD attack are kept, as they correspond to the ones where the maximum confidence score $F_y(x')$ is the most decreased.

Figure 8.4 presents the proportion of unsuccessful l_∞-PGD adversarial examples for which the OC-UA or RC-UA attacks are performed, for various confidence thresholds. For example, we observe that among the 55.1% of unsuccessful adversarial examples, we can reach the RC-UA goal with $\tau_r = 0.25$ for almost 15% of them. For this value of τ_r, the accuracy (RC) at the system level equals 78%. Also, for $\tau_r = 0.25$, the accuracy (OC) at the system level equals 84%.

These results show that despite being efficient to protect a model against misclassification, a defense scheme such as Adversarial Training is not able to completely thwart an adversary that targets the confidence of the predictions by injecting uncertainty.

8.7 Conclusion

In this chapter, we defined a conceptually novel threat against the availability of a system relying on a neural network model. This new threat is based on the fact that the prediction of a model can be correct, but may show *uncertainty*. In this case, relatively to some requirements of a system, its availability may be threatened. We define formally the concept of *uncertainty* of a model's output, and design two test-time attacks which allow to craft *uncertain examples*, which make the model produce unreliable predictions. Interestingly, experiments show that existing integrity defense schemes may protect a model against adversarial examples, but leave it vulnerable to *uncertain examples*, which may lead to the whole system being made unavailable.

Beyond the new threat to availability that we studied, we highlight the existing links between robustness to integrity attacks for a model and robustness to availability attacks for a system. Moreover, this shows to which extent the knowledge of mechanisms behind some type of threat (integrity) may be used to design an attack for another type of threat.

Chapter 9

Conclusion

9.1 Contributions of the thesis

The first objective of this thesis was to draw new conclusions about the robustness to adversarial examples and the frequency nature of the data at stake. This objective is part of a more general one: better understanding the simultaneous vulnerability of models and the insensitivity of humans to adversarial perturbations. We chose to tackle this issue under a frequency perspective, with the motivation of cognitive psychology results underlining the prevalence of low frequency components in the human decision process. To this end, in Chapter 5, for a model, we began by investigating different behaviors related to frequency properties specific to the data this model was trained on. This study allowed then to derive conditions under which some frequency constraints added to the cross-entropy loss induced robustness. Notably, in relation with the way humans perform classification, we identified precise factors which condition robustness when the model is enforced to leverage low-frequency information. On a more general level, this work is an incentive towards better taking into account neural computation knowledge when designing defense schemes against adversarial examples.

The second objective of this thesis was relative to the impact of quantization on robustness, and the transferability phenomenon. In fact, evaluating the influence of quantization on adversarial perturbations allows for a better awareness of integrity issues related to the increasingly widespread usage of quantization to embed models on devices. In that way, we showed in Chapter 6 that, contrary to what was believed in some works, quantization does not offer any robustness. This conclusion is quite important considering that critical tasks are more and more performed on embedded devices.

Strongly linked with the large-scale deployment of models on devices, the transferability of adversarial perturbations is a powerful tool for an adversary in the black-box setting. In Chapter 6, we noticed poor transferability properties between models with different quantization levels (bitwidths). These results led to a first possible ensemble-based defense leveraging quantization. In Chapter 7, we studied specifically how to thwart the transferability between two models. To this end, we designed the concept of *luring effect*, an innovative way to reduce the transferability of adversarial perturbations. We showed how to induce the luring effect for two different threat models, supporting its application in various settings. Interestingly, as this concept is novel, it is fully compatible with other existing defense schemes. Moreover, the luring effect can be used to prevent black-box attacks by detecting the crafting process of an adversarial example against a distant system.

Eventually, we opened our work to the frontier between integrity and availability. More precisely, in Chapter 8, we showed how methods intended at crafting adversarial examples, targeting the integrity of a model, can be leveraged for attacks targeting the availability of a system. In many application domains, the accuracy of a model is not sufficient in itself. In order to make a decision, a system needs the underlying model to output scores with some level of certainty. In Chapter 8, we derived a notion of uncertainty related to the doubt between classes. Then, we showed how to take advantage of adversarial perturbations to make a model output uncertain scores. This last exploratory work of the thesis draws a meaningful link between adversarial examples and other types of threat than integrity threats. Moreover, it underlines the fact that in many use-cases, a protection against adversarial perturbations is simply a security feature, and does not mean that the model can be deployed securely.

9.2 General perspective

At inference time, algorithmic threats have been extensively studied in the literature, whether they relate to model inversion, data extraction, model stealing, membership inference or adversarial examples. However, as a direct consequence of the will to deploy machine learning models massively, the surface attack for these threats is severely extended. Indeed, an adversary can take advantage of models being more available because embedded on devices, to mount an attack using physical attacks.

This is particularly worrisome for the integrity of machine learning models, and more specifically in the case of adversarial examples. Indeed, a deployed model may be protected with a software scheme such as Adversarial Training against adversarial perturbations, which mean malicious modifications of the input. However, to fool the model, the adversary can leverage a fault injection method to flip bit values of weight parameters [221], i.e. to cause a a malicious modifi-

cation of the weights. Even if the model was protected against input modification, the embedded characteristic of the target model make it vulnerable to a physical attacks. As another example, considering the same model in a distant system and in a public device, an adversary can use Side Channel Analysis to retrieve the architecture of the model [14]. Then, he can take advantage of this information to mount a black-box transfer attack against the distant model. The evolution of the usage of machine learning models in the everyday life is thus making the surface attack evolve, more specifically widen, leading to models being more vulnerable to new types of attacks.

Against physical threats, some defense schemes from the hardware security field, such as obfuscation, already exist. However, these methods are not specifically designed for embedded machine learning models. In face of the growing deployment of models, taking into account for the simultaneous attacks at a software and hardware level becomes more and more compulsory. Based on these conclusions, we advocate as a perspective for future work for the development of defense schemes mixing physical and algorithmic features, i.e. taking advantage of the best of both worlds. More precisely, we claim that future defense schemes, in order to be efficient, will have to combine protections against software threats and hardware threats. Notably, an algorithmic defense could take into account a possible malicious modification of the weights to counter bit-flipping attacks. In the same vein, a physical protection could be designed to consider the specific characteristics related to the implementation of a neural network. To summarize, we deeply believe that mixing software and hardware defense schemes is the path towards a more secure deep learning in the nowadays environment where models are increasingly deployed.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems, 2012.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, 2014.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal processing magazine, pp. 82–97, 2012.

[5] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, 2014.

[7] J. Fauw, J. Ledsam, B. Romera-Paredes, S. Nikolov, N. Tomasev, S. Blackwell, H. Askham, X. Glorot, B. O'Donoghue, D. Visentin, G. Driessche, B. Lakshminarayanan, C. Meyer, F. Mackinder, S. Bouton, K. Ayoub, R. Chopra, D. King, A. Karthikesalingam, and O. Ronneberger, "Clinically applicable deep learning for diagnosis and referral in retinal disease," Nature Medicine, 2018.

[8] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in Proceedings of the 10th ACM Conference on Recommender Systems, p. 191–198, 2016.

[9] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), 2016.

[10] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," in NIPS Deep Learning Symposium, 2016.

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in International Conference on Learning Representations, 2014.

[12] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," arXiv preprint arXiv:1605.07277, 2016.

[13] M. Naseer, S. Khan, M. H. Khan, F. Khan, and F. Porikli, "Cross-domain transferability of adversarial perturbations," in Advances in Neural Information Processing Systems, 2019.

[14] L. Batina, S. Bhasin, D. Jap, and S. Picek, "Csi nn: Reverse engineering of neural network architectures through electromagnetic side channel," in 28th USENIX Security Symposium (USENIX Security 19), 2019.

[15] R. Joud, P.-A. Moellic, R. Bernhard, and J.-B. Rigaud, "A review of confidentiality threats against embedded neural network models," arXiv preprint arXiv:2105.01401, 2021.

[16] M. Dumont, P.-A. Moellic, R. Viera, J.-M. Dutertre, and R. Bernhard, "An overview of laser injection against embedded neural network models," arXiv preprint arXiv:2105.01403, 2021.

[17] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems," IEEE Access, vol. 8, pp. 53841–53849, 2020.

[18] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, and et al., "The future of digital health with federated learning," npj Digital Medicine, vol. 3.

[19] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018.

[20] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in Proceedings of the 36th International Conference on Machine Learning, 2019.

[21] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in Proceedings of the 29th International Conference on Machine Learning, ICML 2012, 2012.

[22] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 27–38, ACM, 2017.

[23] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017.

[24] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011.

[27] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," arXiv preprint arXiv:1609.03020, 2016.

[28] C. L. Blake and C. J. Merz, "Uci repository of machine learning databases," 1998.

[29] W. Guo, B. Tondi, and M. Barni, "A master key backdoor for universal impersonation attack against dnn-based face verification," Pattern Recognition Letters.

[30] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," in NIPS Workshop on Mach. Learn. and Comp. Sec, 2017.

[31] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," arXiv preprint arXiv:1912.02771, 2019.

[32] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in European Conference on Computer Vision, 2020.

[33] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in Proceedings of the 23rd USENIX Conference on Security Symposium, 2014.

[34] S. Mehnaz, N. Li, and E. Bertino, "Black-box model inversion attribute inference attacks on classification models," arXiv preprint arXiv:2012.03404, 2020.

[35] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[36] X. Zhao, W. Zhang, X. Xiao, and B. Y. Lim, "Exploiting explanations for model inversion attacks," arXiv preprint arXiv:2104.12669, 2021.

[37] M. Fredrikson, S. Jha, and T. 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.

[38] M. Wu, X. Zhang, J. Ding, H. Nguyen, R. Yu, M. Pan, and S. T. Wong, "Evaluation of inference attack models for deep learning on medical data," arXiv preprint arXiv:2011.00177, 2020.

[39] "The general social survey." `https://gss.norc.org/`.

[40] V. Feldman, "Does learning require memorization? a short tale about a long tail," in Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, 2020.

[41] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in 28th USENIX Security Symposium (USENIX Security 19), 2019.

[42] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al., "Extracting training data from large language models," arXiv preprint arXiv:2012.07805, 2020.

[43] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in Proceedings of the 15th European Conference on Machine Learning, 2004.

[44] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., "Language models are unsupervised multitask learners," OpenAI blog, 2019.

[45] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in 29th USENIX Security Symposium (USENIX Security 20), 2020.

[46] N. Carlini, M. Jagielski, and I. Mironov, "Cryptanalytic extraction of neural network models," arXiv:2003.04884 [cs], 2020.

[47] S. J. Oh, B. Schiele, and M. Fritz, "Towards reverse-engineering black-box neural networks," in Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, 2019.

[48] S. Maji, U. Banerjee, and A. P. Chandrakasan, "Leaky nets: Recovering embedded neural network models and inputs through simple power and timing side-channels–attacks and defenses," IEEE Internet of Things Journal, 2021.

[49] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), 2018.

[50] U. Gupta, D. Stripelis, P. K. Lam, P. Thompson, J. L. Ambite, and G. V. Steeg, "Membership inference attacks on deep regression models for neuroimaging," in Medical Imaging with Deep Learning, 2021.

[51] S. P. Liew and T. Takahashi, "Faceleaks: Inference attacks against transfer learning models via black-box queries," arXiv preprint arXiv:2010.14023, 2020.

[52] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," arXiv preprint arXiv:1709.01604, 2017.

[53] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in Security and Privacy (SP), 2017 IEEE Symposium on, pp. 3–18, IEEE, 2017.

[54] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in 29th USENIX Security Symposium (USENIX Security 20), 2020.

[55] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jegou, "White-box vs black-box: Bayes optimal strategies for membership inference," in Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research, pp. 5558–5567, 2019.

[56] C. A. C. Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," arXiv preprint arXiv:2007.14321, 2020.

[57] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in Advances in Neural Information Processing Systems, pp. 125–136, 2019.

[58] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016.

[59] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[60] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "A rotation and a translation suffice: Fooling cnns with simple transformations," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, pp. 1802–1811, 2019.

[61] A. S. Shamsabadi, R. Sanchez-Matilla, and A. Cavallaro, "Colorfool: Semantic adversarial colorization," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[62] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," IEEE Transactions on Evolutionary Computation, 2019.

[63] M. Cheng, S. Singh, P. Chen, P.-Y. Chen, S. Liu, and C.-J. Hsieh, "Sign-opt: A query-efficient hard-label adversarial attack," in International Conference on Learning Representations, 2020.

[64] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519, ACM, 2017.

[65] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," arXiv preprint arXiv:1611.03814, 2016.

[66] I. Shumailov, Y. Zhao, D. Bates, N. Papernot, R. Mullins, and R. Anderson, "Sponge examples: Energy-latency attacks on neural networks," 2020.

[67] I. Shumailov, Z. Shumaylov, D. Kazhdan, Y. Zhao, N. Papernot, M. A. Erdogdu, and R. Anderson, "Manipulating sgd with data ordering attacks," arXiv preprint arXiv:2104.09667, 2021.

[68] K. Grosse, T. A. Trost, M. Mosbach, M. Backes, and D. Klakow, "On the security relevance of initial weights in deep neural networks," in International Conference on Artificial Neural Networks, pp. 3–14, Springer, 2020.

[69] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011.

[70] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[71] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9, 2015.

[72] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in Proceedings of the 2013th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III, (Berlin, Heidelberg), pp. 387–402, Springer-Verlag, 2013.

[73] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in International Conference on Learning Representations, Wiley Online Library, 2015.

[74] "A complete list of all (arxiv) adversarial example papers." https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html.

[75] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in International Conference on Learning Representations, 2018.

[76] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 3–14, ACM, 2017.

[77] N. Carlini and D. Wagner, "Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples," arXiv preprint arXiv:1711.08478, 2017.

[78] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in Proceedings of the 35th International Conference on Machine Learning, ICML 2018, 2018.

[79] J. Uesato, B. O'Donoghue, A. v. d. Oord, and P. Kohli, "Adversarial risk and the dangers of evaluating against weak attacks," in Proceedings of the 35th International Conference on Machine Learning, ICML 2018, pp. 5025–5034, 2018.

[80] A. Athalye and N. Carlini, "On the robustness of the cvpr 2018 white-box adversarial example defenses," arXiv preprint arXiv:1804.03286, 2018.

[81] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," arXiv preprint arXiv:1902.06705, 2019.

[82] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," in Advances in Neural Information Processing Systems, 2020.

[83] E. Wong, F. R. Schmidt, and J. Z. Kolter, "Wasserstein adversarial examples via projected sinkhorn iterations," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, pp. 6808–6817, 2019.

[84] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," in Advances in Neural Information Processing Systems 31, 2017.

[85] F. Karim, S. Majumdar, and H. Darabi, "Adversarial attacks on time series," IEEE transactions on pattern analysis and machine intelligence, 2020.

[86] Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, and M. Sun, "Word-level textual adversarial attacking as combinatorial optimization," Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020.

[87] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," 2020.

[88] S. Gowal, C. Qin, J. Uesato, T. Mann, and P. Kohli, "Uncovering the limits of adversarial training against norm-bounded adversarial examples," arXiv preprint arXiv:2010.03593, 2020.

[89] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, pp. 1310–1320, 2019.

[90] D. Yin, R. Gontijo Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision," in Advances in Neural Information Processing Systems 32, 2019.

[91] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 2019.

[92] S. Singla and S. Feizi, "Second-order provable defenses against adversarial attacks," in Proceedings of the 37th International Conference on Machine Learning, 2020.

[93] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Adversarial attacks on deep neural networks for time series classification," 2019 International Joint Conference on Neural Networks (IJCNN).

[94] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against bert using bert," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020.

[95] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in Proceedings of the 36th International Conference on Machine Learning, 2019.

[96] S. Hussain, P. Neekhara, S. Dubnov, J. McAuley, and F. Koushanfar, "Waveguard: Understanding and mitigating audio adversarial examples," in 30th USENIX Security Symposium (USENIX Security 21), 2021.

[97] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," Workshop of International Conference on Learning Representations (ICLR), 2017.

[98] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," in Advances in Neural Information Processing Systems, 2020.

[99] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in Proceedings of the 35th International Conference on Machine Learning, 2018.

[100] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in International Conference on Learning Representations, 2016.

[101] C. Sitawarin, A. N. Bhagoji, A. Mosenia, P. Mittal, and M. Chiang, "Rogue signs: Deceiving traffic sign recognition with malicious ads and logos," 1st Deep Learning and Security Workshop (IEEE S&P 2018), 2018.

[102] A. Boloor, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Simple physical adversarial examples against end-to-end autonomous driving models," in 2019 IEEE International Conference on Embedded Software and Systems (ICESS), 2019.

[103] Z. Zhou, D. Tang, X. Wang, W. Han, X. Liu, and K. Zhang, "Invisible mask: Practical attacks on face recognition with infrared," arXiv preprint arXiv:1803.04683, 2018.

[104] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019.

[105] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57, IEEE, 2017.

[106] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: elastic-net attacks to deep neural networks via adversarial examples," in The Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[107] F. Croce and M. Hein, "Sparse and imperceivable adversarial attacks," in The IEEE International Conference on Computer Vision (ICCV), 2019.

[108] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in 2020 IEEE Symposium on Security and Privacy (SP), 2020.

[109] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," arXiv preprint arXiv:1608.07690, 2016.

[110] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 135–147, ACM, 2017.

[111] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," in International Conference on Learning Representations, 2018.

[112] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in International Conference on Learning Representations, 2018.

[113] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.," in International Conference on Learning Representations, 2019.

[114] T. Zhang and Z. Zhu, "Interpreting adversarially trained convolutional neural networks," in Proceedings of the 36th International Conference on Machine Learning, 2019.

[115] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[116] H. Wang, X. Wu, Z. Huang, and E. P. Xing, "High-frequency component helps explain the generalization of convolutional neural networks," in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[117] C. Etmann, S. Lunz, P. Maass, and C. Schoenlieb, "On the connection between adversarial robustness and saliency map interpretability," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, pp. 1823–1832, 2019.

[118] S. Kaur, J. Cohen, and Z. C. Lipton, "Are perceptually-aligned gradients a general property of robust classifiers?," in Advances in Neural Information Processing Systems, 2019.

[119] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, B. Tran, and A. Madry, "Adversarial robustness as a prior for learned representations," arXiv preprint arXiv:1906.00945, 2019.

[120] F. Tramèr and D. Boneh, "Adversarial training and robustness for multiple perturbations," in Advances in Neural Information Processing Systems 33, 2019.

[121] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in Security and Privacy (EuroS&P), 2016 IEEE European Symposium on, pp. 372–387, IEEE, 2016.

[122] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[123] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582, 2016.

[124] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," arXiv preprint arXiv:1801.02610, 2018.

[125] S. Wang, Y. Chen, A. Abdou, and S. Jana, "Enhancing gradient-based attacks with symbolic intervals," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 2019.

[126] Y. Tashiro, Y. Song, and S. Ermon, "Diversity can be transferred: Output diversification for white- and black-box attacks," in Advances in Neural Information Processing Systems 33, 2020.

[127] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 15–26, ACM, 2017.

[128] J. C. Spall et al., "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," IEEE transactions on automatic control, vol. 37, no. 3, pp. 332–341, 1992.

[129] C. Guo, J. R. Gardner, Y. You, A. G. Wilson, and K. Q. Weinberger, "Simple black-box adversarial attacks," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 2019.

[130] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in International Conference on Learning Representations, 2018.

[131] T. Maho, T. Furon, and E. Le Merrer, "Surfree: A fast surrogate-free black-box attack," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[132] M. Cheng, T. Le, P.-Y. Chen, H. Zhang, J. Yi, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in International Conference on Learning Representations, 2019.

[133] S. Moon, G. An, and H. O. Song, "Parsimonious black-box adversarial attacks via efficient combinatorial optimization," in Proceedings of the 36th International Conference on Machine Learning, pp. 4636–4645, 2019.

[134] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in Optimization Techniques, 1978.

[135] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in Proceedings of the 35th International Conference on Machine Learning, ICML 2018, 2018.

[136] H. Li, S. Shan, E. Wenger, J. Zhang, H. Zheng, and B. Y. Zhao, "Blacklight: Defending black-box adversarial attacks on deep neural networks," arXiv preprint arXiv:2006.14042, 2020.

[137] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun 2018.

[138] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[139] C. Xie, Z. Zhang, J. Wang, Y. Zhou, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[140] Q. Huang, I. Katsman, H. He, Z. Gu, S. Belongie, and S.-N. Lim, "Enhancing adversarial example transferability with an intermediate level attack," in The IEEE International Conference on Computer Vision (ICCV), 2019.

[141] W. Wu, Y. Su, X. Chen, S. Zhao, I. King, M. R. Lyu, and Y.-W. Tai, "Boosting the transferability of adversarial samples via attention," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[142] T. Huang, V. Menkovski, Y. Pei, Y. Wang, and M. Pechenizkiy, "Direction-aggregated attack for transferable adversarial examples," arXiv preprint arXiv:2104.09172, 2021.

[143] W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, and Y. Yang, "Transferable adversarial perturbations," in Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[144] D. Wu, Y. Wang, S.-T. Xia, J. Bailey, and X. Ma, "Skip connections matter: On the transferability of adversarial examples generated with resnets," in International Conference on Learning Representations, 2020.

[145] N. Inkawhich, W. Wen, H. H. Li, and Y. Chen, "Feature space perturbations yield more transferable adversarial examples," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[146] J. M. Springer, M. Mitchell, and G. T. Kenyon, "Uncovering universal features: How adversarial training improves adversarial transferability," 2021.

[147] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 2019.

[148] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," arXiv preprint arXiv:1703.00410, 2017.

[149] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," arXiv preprint arXiv:1702.06280, 2017.

[150] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1778–1787, 2018.

[151] C. Xie, Y. Wu, L. van der Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[152] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in International Conference on Computer Aided Verification, 2017.

[153] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in International Conference on Learning Representations, 2019.

[154] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," arXiv preprint arXiv:1810.12715, 2018.

[155] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in Advances in Neural Information Processing Systems, pp. 2266–2276, 2017.

[156] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," 2018.

[157] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 854–863, JMLR. org, 2017.

[158] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. Mann, "Fixing data augmentation to improve adversarial robustness," arXiv preprint arXiv:2103.01946, 2021.

[159] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!," in Advances in Neural Information Processing Systems 33, 2019.

[160] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in International Conference on Learning Representations, 2020.

[161] L. Rice, E. Wong, and J. Z. Kolter, "Overfitting in adversarially robust deep learning," in Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 2020.

[162] P. Maini, E. Wong, and J. Z. Kolter, "Adversarial robustness against the union of multiple perturbation models," in Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 2020.

[163] F. Croce and M. Hein, "Adversarial robustness against multiple l_p-threat models at the price of one and how to quickly fine-tune robust models to another threat model," arXiv preprint arXiv:2105.12508, 2021.

[164] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in International Conference on Learning Representations, 2020.

[165] J. Zhang, J. Zhu, G. Niu, B. Han, M. Sugiyama, and M. Kankanhalli, "Geometry-aware instance-reweighted adversarial training," in International Conference on Learning Representations, 2021.

[166] M. Naseer, S. Khan, M. Hayat, F. S. Khan, and F. Porikli, "Stylized adversarial defense," arXiv preprint arXiv:2007.14672, 2020.

[167] M. Goldblum, L. Fowl, S. Feizi, and T. Goldstein, "Adversarially robust distillation," Proceedings of the AAAI Conference on Artificial Intelligence, p. 3996–4003, 2020.

[168] F. Liu, R. Zhao, and L. Shi, "Adversarial feature stacking for accurate and robust predictions," arXiv preprint arXiv:2103.13124, 2021.

[169] C. Song, K. He, J. Lin, L. Wang, and J. E. Hopcroft, "Robust local features for improving the generalization of adversarial training," in International Conference on Learning Representations, 2020.

[170] D. Stutz, M. Hein, and B. Schiele, "Confidence-calibrated adversarial training: Generalizing to unseen attacks," in Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 2020.

[171] A. Chan, Y. Tay, Y. S. Ong, and J. Fu, "Jacobian adversarially regularized networks for robustness," in International Conference on Learning Representations, 2020.

[172] S. Addepalli, B. Vivek, A. Baburaj, G. Sriramanan, and R. Venkatesh Babu, "Towards achieving adversarial robustness by enforcing feature consistency across bit planes," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[173] F. Lin, R. Mittapali, P. Chattopadhyay, D. Bolya, and J. Hoffman, "Likelihood landscapes: A unifying principle behind many adversarial defenses," in Adversarial Robustness in the Real World (AROW), ECCV, 2020.

[174] R. Bernhard, P.-A. Moellic, M. Mermillod, Y. Bourrier, R. Cohendet, M. Solinas, and M. Reyboz, "Impact of spatial frequency based constraints on adversarial robustness," in 2021 International Joint Conference on Neural Networks (IJCNN), 2021.

[175] J. Jo and Y. Bengio, "Measuring the tendency of cnns to learn surface statistical regularities," arXiv preprint arXiv:1711.11561, 2017.

[176] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in International Conference on Learning Representations, 2019.

[177] P. G. Schyns and A. Oliva, "From blobs to boundary edges: Evidence for time-and spatial-scale-dependent scene recognition," Psychological science, vol. 5, no. 4, pp. 195–200, 1994.

[178] M. Mermillod, P. Bonin, L. Mondillon, D. Alleysson, and N. Vermeulen, "Coarse scales are sufficient for efficient categorization of emotional facial expressions: Evidence from neural computation," Neurocomputing, vol. 73, no. 13-15, pp. 2522–2531, 2010.

[179] R. M. French, M. Mermillod, A. Chauvin, P. C. Quinn, and D. Mareschal, "The importance of starting blurry: Simulating improved basic-level category learning in infants due to weak visual acuity," in Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 24, 2002.

[180] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, "Shield: Fast, practical defense and vaccination for deep learning using jpeg compression," Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018.

[181] Z. Liu, Q. Liu, T. Liu, N. Xu, X. Lin, Y. Wang, and W. Wen, "Feature distillation: Dnn-oriented jpeg compression against adversarial examples," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[182] Z. Zhang, C. Jung, and X. Liang, "Adversarial defense by suppressing high-frequency components," arXiv preprint arXiv:1908.06566, 2019.

[183] Z. Wang, Y. Yang, A. Shrivastava, V. Rawal, and Z. Ding, "Towards frequency-based explanation for robust cnn," arXiv preprint arXiv:2005.03141, 2020.

[184] Y. Sharma, G. W. Ding, and M. A. Brubaker, "On the effectiveness of low frequency perturbations," Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, 2019.

[185] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in International Conference on Learning Representations, 2015.

[186] S. Zagoruyko and N. Komodakis, "Wide residual networks," in British Machine Vision Conference (BVCM), 2016.

[187] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[188] R. Bernhard, P.-A. Moellic, and J.-M. Dutertre, "Impact of low-bitwidth quantization on the adversarial robustness for embedded neural networks," in 2019 International Conference on Cyberworlds (CW), 2019.

[189] M. Denil, B. Shakibi, L. Dinh, N. De Freitas, et al., "Predicting parameters in deep learning," in Advances in neural information processing systems, pp. 2148–2156, 2013.

[190] G. B. Hacene, V. Gripon, M. Arzel, N. Farrugia, and Y. Bengio, "Quantized guided pruning for efficient hardware implementations of convolutional neural networks," arXiv preprint arXiv:1812.11337, 2018.

[191] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," arXiv preprint arXiv:1412.6115, 2014.

[192] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[193] Y. Choi, M. El-Khamy, and J. Lee, "Towards the limit of network quantization," arXiv preprint arXiv:1612.01543, 2016.

[194] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in Advances in neural information processing systems, pp. 3123–3131, 2015.

[195] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," arXiv preprint arXiv:1602.02830, 2016.

[196] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations.," Journal of Machine Learning Research, vol. 18, no. 187, pp. 1–30, 2017.

[197] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in European Conference on Computer Vision, pp. 525–542, Springer, 2016.

[198] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," 2016.

[199] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," arXiv preprint arXiv:1612.01064, 2016.

[200] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in International Conference on Machine Learning, pp. 1737–1746, 2015.

[201] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," arXiv preprint arXiv:1606.06160, 2016.

[202] R. Ding, Z. Liu, R. Shi, D. Marculescu, and R. Blanton, "Lightnn: Filling the gap between conventional deep neural networks and binarized networks," in Proceedings of the on Great Lakes Symposium on VLSI 2017, pp. 35–40, ACM, 2017.

[203] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," arXiv preprint arXiv:1802.05668, 2018.

[204] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "Bnn+: Improved binary network training," arXiv preprint arXiv:1812.11800, 2018.

[205] A. Galloway, G. W. Taylor, and M. Moussa, "Attacking binarized neural networks," in International Conference on Learning Representations, 2018.

[206] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," in International Conference on Learning Representations, 2019.

[207] Y. Zhao, I. Shumailov, R. Mullins, and R. Anderson, "To compress or not to compress: Understanding the interactions between adversarial attacks and neural network compression," Conference on Systems and Machine Learning (SysML), 2019.

[208] A. S. Rakin, J. Yi, B. Gong, and D. Fan, "Defend deep neural networks against adversarial examples via fixed anddynamic quantized activation functions," arXiv preprint arXiv:1807.06714, 2018.

[209] E. B. Khalil, A. Gupta, and B. Dilkina, "Combinatorial attacks on binarized neural networks," arXiv preprint arXiv:1810.03538, 2018.

[210] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in Proceedings of the 34th International Conference on Machine Learning, ICML 2017, 2017.

[211] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[212] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," arXiv preprint arXiv:1610.00768, 2018.

[213] L. Wu, Z. Zhu, C. Tai, et al., "Towards understanding and improving the transferability of adversarial examples in deep neural networks," in Proceedings of The 12th Asian Conference on Machine Learning, 2020.

[214] R. Bernhard, P.-A. Moellic, and J.-M. Dutertre, "Luring of transferable adversarial perturbations in the black-box paradigm," in 2021 International Joint Conference on Neural Networks (IJCNN), 2021.

[215] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 2019.

[216] Y. Carmon, A. Raghunathan, L. Schmidt, P. Liang, and J. C. Duchi, "Unlabeled data improves adversarial robustness," in Advances in Neural Information Processing Systems, 2019.

[217] U. Hwang, J. Park, H. Jang, S. Yoon, and N. I. Cho, "Puvae: A variational autoencoder to purify adversarial examples," IEEE Access, vol. 7, pp. 126582–126593, 2019.

[218] S. Shan, E. Wenger, B. Wang, B. Li, H. Zheng, and B. Y. Zhao, "Using honeypots to catch adversarial attacks on neural networks," in Proceedings of ACM Conference on Computer and Communications Security (CCS), 2019.

[219] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in 29th {USENIX} Security Symposium ({USENIX} Security 20), pp. 1345–1362, 2020.

[220] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 6106–6116, 2018.

[221] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

List of Figures

List of Tables

Appendix A

Résumé de la thèse

Cette thèse a pour objectif de mieux comprendre le phénomène des exemples adverses (*adversarial examples*) et de proposer des mécanismes de défense, plus particulièrement pour lutter contre leur *transférabilité*. La motivation est double. D'une part, les attaques contre l'intégrité des modèles de *Machine Learning* à l'inférence sont reconnues pour leur criticité et il est crucial d'analyser finement les phénomènes sous-jacents pour améliorer le développement de protections robustes. D'autre part, parmi les nombreuses méthodologies et vecteurs d'attaques des exemples adverses (boite blanche, boite noire), ceux tirant profit de leur transférabilité apparaissent comme particulièrement critiques dans un contexte de déploiement des modèles à très large-échelle (par exemple les systèmes embarqués pour l'IoT).

A.1 Introduction

A.1.1 L'omniprésence du machine learning

Depuis quelques années, le *machine learning*, et plus spécifiquement les approches par *representation learning* avec les réseaux de neurones profonds (deep learning), a démontré des performances remarquables pour de nombreuses tâches complexes: classification d'images [1], synthèse d'images [2], détection et reconnaissance d'objets en temps réel [3], reconnaissance de la parole [4], synthèse de la voix [5] ou traduction automatique de texte [6]. Aujourd'hui, les réseaux de neurones profonds sont employés dans de multiple domaines d'applications comme le diagnostic médical [7], les systèmes de recommandation [8], la détection de malwares [9] ou encore les véhicules autonomes [10].

Etant donné ce succès, des modèles sont de plus en plus déployés dans la vie de tous les jours. Par exemple, un nombre croissant d'applications sur les téléphones mobiles fonctionnent en faisant des appels répétés à des serveurs distants qui se reposent sur des modèles de machine learning. En outre, de plus en plus d'API utilisent le deep learning, comme l'API Google Cloud Vision API[1] ou l'API Clarifai[2]. De plus, l'utilisation du machine learning n'est pas réservée à des services qui sont disponibles à tout le monde. En effet, le deep learning joue un rôle important dans de nombreuses applications critiques telles que la cybersécurité, le transport, la défense ou encore la santé.

En d'autres termes, on parle de machine learning *omniprésent* (*ubiquitous artificial intelligence*) avec l'inclusion de modèles dans une multitude de systèmes publics ou privés. Ce déploiement à large-échelle soulève d'importantes questions de sécurité que cela concerne les données en jeu ou la fiabilité intrinsèque des modèles. En réponse à ces inquiétudes, de nouvelles règlementations sont adoptées. A l'échelle européenne, la question de la sécurité des données et des modèles d'IA fait partie intégrante du nouveau *framework* de régulation de l'intelligence artificielle, présenté en Avril 2021 [3] centré sur les intelligences artificielles *à risque* et qui devra compléter et s'articuler avec les législations existantes pour la protection des données personnelles (RGPD, 2018)[4].

A.1.2 Machine learning et systèmes embarqués

Le déploiement de modèles de machine learning

Le développement actuel du machine learning se concentre principalement sur deux objectifs : (1) augmenter les performances des modèles pour des tâches complexes, ce qui s'accompagne généralement d'un accroissement de leur complexité, comme pour les récent modèles *transformers* en traitement automatique de la langue ; (2) le déploiement de modèles vers un vaste panel de plateformes matérielles. Ce déploiement à grande échelle répond aux problèmes classiques du paradigme client/serveur pour les applications d'IA et la trop grande dépendance aux capacités des serveurs ainsi qu'aux réseaux de communication: passage à l'échelle, qualité, disponibilité... Déployer et distribuer *l'intelligence* vers les *clients* permet de réduire cette dépendance. Cependant, les réseaux de neurones profonds peuvent regrouper plusieurs millions de paramètres. Cette complexité est généralement absorbée par des architectures de calcul *GPU* (Graphics Processing Units). De ce fait, embarquer un modèle de deep learning sur une plateforme matérielle contrainte conduit à plusieurs problèmes majeurs: la capacité mémoire de certaines plateformes (typiquement, un microcontrôleur dans l'IoT) peut vite devenir un facteur limitant tout comme la consommation

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

[2] https://www.clarifai.com/

[3] https://digital-strategy.ec.europa.eu/en/library/communication-fostering-european-approach-artificial-intelligence

[4] `https://www.cnil.fr/fr/intelligence-artificielle-lavis-de-la-cnil-et-de-ses-homologues-sur-le-futur-reglement-europeen`

énergétique d'une inférence pour de nombreux domaines d'applications ou le temps de traitement d'une inférence pour des applications temps réel.

Les conséquences du déploiement de modèles en termes de sécurité

Au vu des contraintes de sécurité qui émergent alors que les modèles de machine learning deviennent de plus en plus omniprésents, les modèles embarqués soulèvent un premier problème : ils élargissent la surface d'attaque disponible pour un adversaire. En conséquence, il sera plus difficile de sécuriser un modèle déployé qu'un modèle implanté sur un serveur distant, uniquement disponible via des requêtes à ce serveur.

Pour un modèle distant, certaines de ses caractéristiques peuvent être cachées pour obfusquer des informations critiques et ainsi affaiblir un potentiel adversaire. Par exemple, la nature du modèle (arbre de décision, machine à vecteurs de support, etc.) peut être caché, pour restreindre la connaissance qu'a l'adversaire du système à attaquer. L'information donnée par le modèle en inférence peut aussi être obstruée, par exemple en ne dévoilant que la classe prédite pour une tâche de classification, au lieu du score pour chaque classe possible.

Quand le modèle est physiquement accessible par un attaquant, ce dernier se voit offrir une plus grande surface d'attaque. Notamment, même si des méthodes d'obfuscation sont employées, un adversaire peut exploiter certains défauts liés directement à l'implémentation physique.

De plus, considérant un modèle utilisé à la fois à distance et en rendu public en physique, l'adversaire peut tirer parti des vulnérabilités de la version publique pour monter une attaque contre la version distante. En faisant cela, son attaque est bien plus puissante que si seulement le modèle distant était connu.

A.1.3 Les attaques contre le machine learning

La montée des menaces contre le machine learning

Les attaques contre les systèmes informatiques, et plus particulièrement les systèmes d'information, sont particulièrement étudiées avec, par exemple, les attaques par déni de service. Depuis quelques années, de nombreuses menaces visant directement un modèle de machine learning, et plus particulièrement les réseaux de neurones, ont été démontrées. L'existence de ces nouvelles attaques implique que de nombreuses applications reposant sur des modules de machine learning, dont certaines particulièrement critiques, sont aussi vulnérables vis-à-vis de menaces contre leur intégrité, confidentialité ou disponibilité.

Concrètement, ces nouvelles attaques peuvent survenir à toutes les étapes du traditionnel *machine learning pipeline* : pendant la phase d'apprentissage (quand le modèle est entrainé sur un jeu de données), ou pendant la phase d'inférence, quand le modèle est utilisé réellement pour effectuer sa tâche. Les objectifs de ces nouvelles attaques sont assez divers. Premièrement, elles peuvent être conçues pour altérer sérieusement le processus d'inférence du modèle. Il s'agit de tromper le modèle en modifiant les entrées. Dans ce cas, l'intégrité du système est menacée. Deuxièmement, certaines attaques visent à voler les informations concernant un modèle caché (par exemple, les poids d'un réseau de neurones) ou les données avec lesquelles ce modèle a été entrainé. Cela peut concerner des données privées ou confidentielles (des données médicales, financières, ...). Ce type de menace est donc particulièrement préoccupant au regard de la vie privée, confidentialité et de la propriété intellectuelle. Troisièmement, ces attaques peuvent viser la disponibilité d'un système utilisant un modèle de machine learning. La disponibilité se réfère aux exigences du système (en termes de performances et de qualité), comme le temps d'accès, les temps de traitement (apprentissage ou inférence), ou la qualité moyenne des prédictions délivrées par le modèle.

Considérant les dangers causés par ces nouvelles attaques, il apparait donc comme urgent de développer des méthodes de défense fiables et robustes. En conséquence, ces menaces sont de plus en plus étudiées par les communautés de la sécurité et de l'intelligence artificielle, avec de plus en plus d'efforts communs réalisés.

Exemples adverses

Parmi les menaces existantes, le phénomène des exemples adverses (*adversarial examples*) a reçu une attention particulière [11]. Les exemples adverses sont des données d'inférence modifiées de manière malicieuse (en leur ajoutant une *perturbation adverse*) pour tromper un modèle de machine learning pendant la phase d'inférence. Par exemple, pour une tâche de classification, tromper le modèle revient à ce que le modèle ne prédise pas la même classe pour l'exemple original et sa version modifiée. Il est aussi important que les exemples soient modifiés de manière à ce

qu'un expert humain ne soit pas influencé par ces modifications (i.e. qu'il ne détecte pas de phénomène malveillant). Le modèle prédit deux classes différentes pour l'exemple original et sa version modifiée, tandis que l'humain prédit la même classe pour les deux. Cette spécificité indique donc que les exemples adverses peuvent être déployés dans de nombreux scenarios sans susciter de suspicion. Par exemple, un panneau de signalisation routière peut être modifié subtilement en lui ajoutant des autocollants dessus de manière bien précise, sans que cela attire l'attention car le panneau reste largement reconnaissable. Cependant, ces autocollants vont tromper le système d'une voiture autonome qui se repose sur un modèle de machine learning, comme ça a été montré pour une voiture Tesla par un laboratoire MacAfee en 2020[5].

Les exemples adverses constituent actuellement la menace la plus étudiée parmi celles ciblant les modèles de machine learning, et ce pour deux raisons. La première concerne le scénario le plus répandu d'un adversaire qui veut attaquer un modèle qui est déjà déployé. La seconde raison est que ce type d'attaque a les conséquences les plus graves parmi toutes les attaques existantes, relativement à l'effort qu'elles requièrent. Par exemple, un simple panneau de signalisation adverse peut entrainer un accident de la voie publique mortel.

Transferabilité

Le phénomène de *transférabilité* des exemples adverses aggrave sévèrement leur dangerosité. La transférabilité des perturbations adverses désigne le fait qu'un exemple adverse élaboré pour tromper un modèle peut être utilisé pour tromper un autre modèle entrainé pour réaliser la même tâche. La transférabilité est donc un outil puissant pour un adversaire qui n'a pas un accès direct au modèle ciblé. En effet, dans ce cas, l'adversaire peut entrainer son propre modèle de substitution (appelé modèle substitut), ou utiliser un autre modèle déjà disponible, pour construire un exemple adverse. Dans ce cas, le processus d'élaboration d'un exemple adverse est le plus favorable, car toutes les informations du modèle substitut sont connues par l'adversaire. Cet exemple adverse est ensuite *transféré* au modèle auquel l'adversaire n'a pas accès, pour le tromper. De plus, la transférabilité des exemples adverses est d'autant plus dangereuse qu'elle existe entre différentes techniques de machine learning [12] (par exemple d'une machine à vecteurs de support vers un arbre de décision), et pour différents domaines (i.e. différentes tâches) [13].

Avec le déploiement croissant de modèles, le phénomène de transférabilité est en conséquence de plus en plus préoccupant. Notamment, pour les tâches de vision par ordinateur, il est souvent le cas qu'un même système soit utilisé à la fois pour un système distant et déployé sur des plateformes matérielles. L'adversaire peut donc extraire de l'appareil des informations utiles qui l'aideront dans la fabrication ou le choix du modèle substitut. Par exemple, l'adversaire peut utiliser une attaque par canal auxiliaire, une méthode qui exploite les détails d'implémentation d'un modèle, pour retrouver l'architecture d'un réseau de neurones [14].

A.2 Objectifs et sommaire

A.2.1 Objectifs de la thèse

Dans cette thèse, nous nous intéressons à la robustesse des réseaux de neurones aux exemples adverses, pour une tâche de classification d'images. Plus précisément nous considérons deux principaux axes de recherche. Le premier vise à une meilleure compréhension de la nature intrinsèque des exemples adverses, et à expliquer la vulnérabilité des réseaux de neurones à ces attaques. Ce premier axe, qui s'attache à mieux comprendre l'origine de la sensibilité des modèles à ce type de menace doit permettre la conception de méthodes de défense originales. Le second axe de recherche est orienté vers une réduction de l'impact des attaques adverses, en visant notamment à réduire l'impact de la transférabilité des perturbations adverses. La transférabilite est un outil très puissant pour un adversaire qui se trouve dans le scénario le plus commun où il n'a pas accès directement au modèle ciblé. Etudier comment réduire ce phénomène de transférabilité est de ce fait crucial.

A.2.2 Sommaire

Cette thèse est organisée de la manière suivante :

[5] https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/

- Dans un premier travail, nous étudions la vulnérabilité des réseaux de neurones aux exemples adverses d'un point de vue fréquentiel. Ce travail se base sur des résultats de psychologie cognitive indiquant que les basses fréquences ont une part prédominante dans le système de classification humain. Nous étudions alors comment utiliser à bon escient des contraintes fréquentielles lors de l'entrainement d'un réseau de neurones.

- Dans un second travail, nous considérons les méthodes de quantification, utilisées pour embarquer un réseau de neurones, qui sont de plus en plus employées en conséquence de la volonté croissante de déploiement de modèles. Nous étudions l'impact de ce type de méthodes sur la robustesse des modèles aux exemples adverses, et particulièrement sur leur transférabilité.

- Dans un troisième travail, nous analysons le phénomène de transférabilité, et proposons un concept innovant pour le réduire, l'*effet de leurrage*. L'effet de leurrage vise à tromper l'adversaire, qui va sélectionner certaines caractéristiques du modèles substitut à attaquer, qui ne sont pas celles qu'il aurait fallu choisir pour tromper le modèle cible. L'effet de leurrage est simple d'implémentation, et nous proposons trois cas d'usages concrets pour son utilisation, dont la détection du processus d'élaboration d'exemples adverses.

- Dans un dernier travail, nous montrons comment les perturbations adverses peuvent être utilisées pour attaquer la disponibilité d'un système qui se repose sur un modèle de machine learning. De manière générale, cette étude tisse un lien entre intégrité et disponibilité, et souligne le fait que la précision d'un modèle n'est pas suffisante pour un déploiement sécurisé de celui-ci.

A.3 Sécurité des modèles de machine learning

Dans cette section, nous présentons les différentes menaces qui peuvent cibler un modèle de machine learning.

A.3.1 Processus de développement d'un modèle

Quelque soit la technique de machine learning utilisée, le développement d'un modèle pour une tâche supervisée suit toujours le même processus. Dans un premier temps, des données brutes sont collectées. La seconde étape consiste en leur traitement pour construire un jeu de données, qui sera utilisé pour entrainer le modèle de machine learning. La troisième étape consiste est l'entrainement du modèle avec le jeu de données obtenu suite à l'étape précédente. Cette étape revient à chercher le meilleur modèle en termes d'erreur en généralisation. La quatrième étape est le déploiement du modèle dans un système. Le modèle déployé fait maintenant partie du système, et est totalement opérationnel. Par ailleurs, le déploiement d'un modèle de machine learning peut être un processus continu. Par exemple, dans le cas du *online learning*, de nouvelles données sont obtenues de manière continue pour affiner les paramètres du modèle.

A chaque étape, de la collecte de données au moment où le modèle est mis en service, différentes attaques sont réalisables.

A.3.2 Attaques pendant l'entrainement

Les attaques qui surviennent pendant l'entrainement d'un modèle ont pour but d'influencer de manière malicieuse le processus d'entrainement, pour au final causer des conséquences précises sur le modèle appris. Ce type d'attaque est réalisable selon deux scénarios.

Premièrement, il peut survenir quand la phase d'entrainement est réalisée sur un service *cloud*. Dans ce cas, un adversaire qui arrive à pénétrer le système responsable de l'entrainement peut modifier le jeu de données d'entrainement pour réaliser une attaque.

Le second scenario survient quand la collecte des données et la phase de traitement sont privées. Il nécessite un adversaire bien plus puissant que dans le scénario précédent. En effet, puisque l'adversaire est supposé pouvoir interférer durant la phase d'apprentissage, il implique que l'adversaire est un des acteurs du projet de développement du modèle ciblé.

Data poisoning

Pour détériorer la qualité d'un modèle déployé, un adversaire peut monter une attaque de *data poisoning*, soit en injectant de nouvelles données empoisonnées dans le jeu de données d'entrainement du modèle, soit en modifiant de manière malicieuse des données existantes. Le data poisoning vise

à diminuer la précision d'un modèle pour certains exemples bien précis [19,20], ou pour tout le jeu de données de test [21–23]. Dans ce cas, le modèle déployé peut être particulièrement imprécis, ce qui l'empêche d'être utilisé par un système. En effet, si le modèle ne satisfait pas certains prérequis en termes de performances relatives à la tâche (par exemple la précision), il peut être déclaré comme non-fiable et la disponibilité du système qui se repose sur ce modèle est compromise. En conséquence, le data poisoning peut être particulièrement dommageable dans ces applications critiques telles que la santé, où le système peut être en charge de la prise de décision critiques.

Backdoor attacks

Quand le but de l'adversaire est de faire en sorte que le modèle ciblé ait des comportements spécifiques pour certains exemples, un adversaire peut réaliser une *backdoor attack*. L'adversaire intervient pendant la phase d'entrainement pour faire en sorte que le modèle se comporte d'une façon particulière à l'inférence pour certains exemples dans lesquels sont intégrés des mécanismes de déclenchement appelés *triggers*. Pour que le modèle réagisse quand il rencontre ses mécanismes de déclenchement, l'adversaire doit inclure dans le jeu d'entrainement des données avec ce mécanisme et le comportement désiré. Les exemples avec le mécanisme de déclenchement sont des exemples normaux qui contiennent des signaux spécifiques sur lesquels le modèle cible est entrainé à produire un certain comportement. ce type de menace est particulièrement dangereux pour les tâches de classification, pour deux raisons. Premièrement, par définition, ce genre d'attaque autorise l'adversaire à causer des mauvaises classifications à l'inférence. Deuxièmement, quand le modèle est déployé, tant qu'aucun exemple avec un mécanisme de déclenchement ne lui est présenté, son comportement paraît normal.

A.3.3 Attaques pendant l'inférence

Les attaques à l'inférence comprennent toutes les attaques qui peuvent survenir une fois que le modèle est appris et déployé. La variété d'adversaires qui réalisent ce type d'attaques est bien plus grande que pour les attaques pendant l'entrainement. En effet, alors que pour les attaques pendant l'entrainement, cela était limité aux adversaires ayant accès à la phase d'entrainement du modèle, cela implique maintenant tous les possibles adversaires qui ont accès au modèle déployé. Ce modèle peut être déployé via des APIs distantes, ou embarqué sur une cible matérielle comme un téléphone mobile. En fonction du scénario, l'adversaire aura une connaissance plus ou moins grande des détails du modèle à attaquer : technique de machine learning utilisée, connaissance des paramètres du modèle, etc.

Inversion de modèle

Considérant un modèle déjà déployé, le but d'un adversaire peut être de trouver des informations sensibles des données d'entrée. Pour ce faire, un adversaire peut réaliser une attaque d'*inversion de modèle*, pour retrouver une très bonne approximation de l'information sensible visée. Etant donné une certaine information d'une donnée d'entrée que l'adversaire veut retrouver, avec ce type d'attaque, l'adversaire a uniquement besoin d'être capable d'obtenir la réponse du modèle pour cette entrée, et/ou déjà avoir une connaissance d'informations non-sensibles de cette donnée. Ce type d'attaque constitue une grande menace à la confidentialité et à la vie privée, car un modèle de machine learning peut être entrainé sur des données sensibles pour réaliser une tâche. Ces données sensibles peuvent concerner la vie privée, avec par exemple des informations sur l'identité d'une personne, ou certaines de ses habitudes [35–38].

Extraction de données

Considérant un modèle déployé, l'adversaire peut vouloir retrouver des informations concernant les données avec lesquelles le modèle a été entrainé, et ce de manière exacte. Pour ce faire, l'adversaire peut réaliser une attaque d'*extraction de données* [41,42]. Tandis que les attaques d'inversion de modèle permettent de retrouver des informations sensibles pour toute donnée d'entrée soumise, avec une grande précision, même si la donnée ne fait pas partie du jeu de données d'entrainement, les attaques d'extraction de données permettent de récupérer de manière exacte des parties du jeu de données d'entrainement. Les attaques d'extraction de données exploitent en fait le phénomène de mémorisation des modèles de machine learning : les modèles ont tendance à mémoriser de manière exacte certaines parties du jeu de données d'entrainement. La mémorisation se produit au début de la phase d'entrainement, et la mémorisation des classes, par exemple, est nécessaire

pour une bonne généralisation [40]. Les attaques d'extraction de données sont donc une menace pour la confidentialité des données d'entrainement du modèle.

Vol de modèle

Quand le but de l'adversaire est de réaliser de la rétro-ingénierie sur un modèle de machine learning, il peut réaliser une attaque dite de *vol de modèle* [45–47]. Ce type de menace est préoccupant pour deux raisons. Premièrement, cela pose des problèmes de confidentialité et de propriété intellectuelle. En effet, le vol de modèle permet de copier un modèle existant sans avoir à supporter les mêmes coûts que le créateur du modèle, à savoir les coûts en recherche et développement et les coûts liés à la phase d'entrainement. Grâce à ce genre d'attaque, un adversaire peut utiliser un modèle sans avoir à payer pour, si le modèle de base n'était disponible que via un accès payant par exemple. Un adversaire peut aussi copier un modèle et le mettre en vente de son côté, ce qui ause des problèmes de propriété intellectuelle et de concurrence déloyale. Deuxièmement, le vol de modèle facilite la réalisation de différents types d'attaques contre un modèle caché. Quelque soit le type d'attaque que l'adversaire veut réaliser, avoir une connaissance plus détaillée du modèle à attaquer facilite la réalisation de cette attaque. L'adversaire peut utiliser le vol de modèle pour obtenir un modèle substitut du modèle ciblé, et s'entrainer à réaliser l'attaque sur le modèle substitut avant de la réaliser sur le modèle ciblé.

Membership inference

Pour déterminer si un exemple particulier fait partie ou non du jeu d'entrainement d'un modèle, un adversaire peut réaliser une attaque de *membership inference* [55, 56]. Ce type d'attaque peut donner lieu à des divulgations graves d'informations personnelles. En effet, savoir qu'un individu fait partie du jeu de données d'entrainement peut permettre de déduire des informations sensibles de sa vie privée. Par exemple, un modèle de machine learning peut être entrainé pour prédire une dose de médication, en se basant sur des données de personnes souffrant d'une certaine maladie. Considérant une donnée particulière, le fait de déterminer que cette donnée faisait partie du jeu d'entrainement implique que la personne associée à cette donnée souffre de cette maladie. Les modèles entrainés avec des données médicales et de santé sont particulièrement concernés par ce type d'attaque. [50]. De plus, les données d'un modèle qui a servi de modèle de base pour du *transfer learning* peuvent aussi être menacées par ce type d'attaques [51].

Exemples adverses

Quand le but de l'adversaire est d'attaquer directement le processus d'inférence d'un modèle, il peut utiliser les *exemples adverses* [11, 64]. Ce type d'attaque consiste à ajouter une perturbation adverse à un exemple normal pour tromper un modèle. La perturbation adverse se doit d'être imperceptible ou d'apparence bénigne, afin de ne pas être détectée par un humain. Le phénomène des exemples adverses tire son origine du fait que les modèles de machine learning utilisent des caractéristiques différentes de celles que les humains utilisent [57]. Notamment, les modèles se reposent sur certains signaux dans une image, auxquels les humains ne sont pas sensibles. Perturber ces signaux permet donc à un adversaire de tromper un modèle de machine learning de manière totalement anodine pour un observateur humain. Puisque ce type d'attaque permet de faire en sorte que le modèle donne des prédictions erronées pour un exemple précis à l'inférence, c'est une menace très sérieuse pour l'intégrité du processus de prédiction des modèles de machine learning. Par exemple, un adversaire peut faire en sore d'inclure des perturbations adverses dans l'apparence de lunettes pour passer outre un système de reconnaissance faciale [58]. Un adversaire peut réaliser des exemples adverses pour des images dans des conditions environnementales changeantes et en tenant compte des différents points de vue possibles [59]. La Figure A.1 propose l'illustration d'un exemple adverse.

Les différentes attaques qui peuvent survenir, que ce soit pendant la phase d'entrainement ou la phase d'apprentissage, sont rappelées dans la Figure A.2.

A.3.4 Confidentialité, Intégrité et Disponibilité

Les attaques présentées ci-dessus peuvent être vues sous l'angle de la triade CIA (Confidentialité, Intégrité et Accessibilité/Disponibilité), qui est un paradigme très connu dans le domaine de la sécurité de l'information. Ces trois concepts permettent de recouvrir de manière exhaustive les différents aspects qui doivent être protégés dans un système.

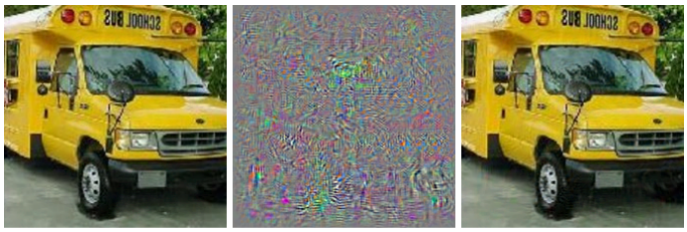

Figure A.1: Illustration d'un exemple adverse, tiré de [11]. (Gauche) L'image originale est prédite comme un bus d'écoliers. (Milieu) La perturbation adverse, amplifiée pour une meileure visualisation. (Droite) L'exemple adverse résultat, prédit par le modèle cible comme une autruche.

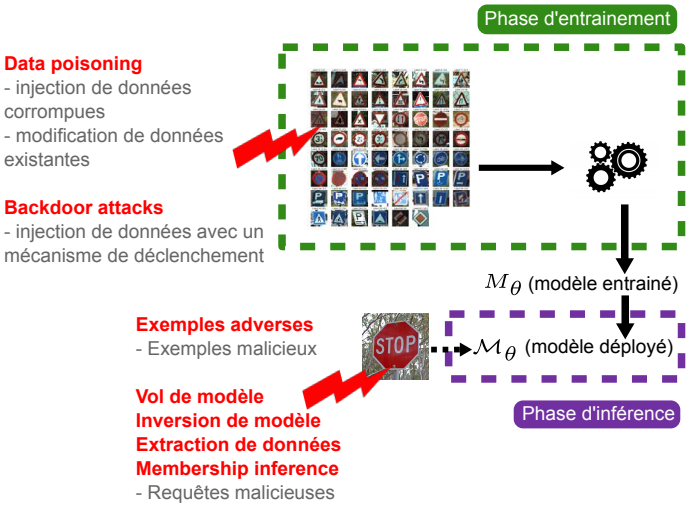

Figure A.2: Differentes manières d'attaquer un modèle de machine learning

Confidentialité. Ce concept fait référence à la protection des données confidentielles, pour éviter une divulgation de celles-ci. Les données confidentielles peuvent être de l'ordre du domaine médical, financier, de la vie privée ou encore relatives à la propriété intellectuelle (brevets, etc.). Dans le cas d'un modèle de machine learning, la confidentalité d'informations sensibles est soit relative aux données, soit au modèle en lui-même. Ainsi, l'*inversion de modèle*, l'*extraction de données*, les attaques *membership inference* ou le *vol de modèle* menace la confidentialité d'un modèle. Pour les deux premières attaques, la confidentialité des données est en jeu. Pour le *vol de modèle*, la propriété intellectuelle est menacée.

Intégrité. Ce concept fait référence à la protection contre toute modification malicieuse qui a pour but de compromettre la complétion d'une tâche. Dans le cas d'un modèle de machine learning, l'intégrité fait référence à l'intégrité du processus d'inférence. Un adversaire qui réalise une attaque contre l'intégrité d'un modèle cible son processus d'inférence car il a pour but de compromettre l'intégrité de la prédiction. Les attaques *backdoor* et les *exemples adverses* menacent l'intégrité d'un modèle de machine learning.

Accessibilité. Le concept d'accessibilité désigne la capacité à garder un système accessible uniquement aux utilisateurs autorisés. Le but d'un adversaire qui cible l'accessibilité d'un système est de diminuer l'accès à ce système, ou de dégrader la qualité de cet accès, relativement à certaines exigences définies pour ce système. De manière importante, les attaques contre l'accessibilité d'un système ne doivent pas être considérées uniquement à l'échelle d'un modèle, mais en termes de système global qui contient un modèle, contrairement aux attaques contre la confidentialité ou l'intégrité d'un modèle. En effet, comme mentionné dans [65] : "L'accessibilité n'est pas bien définie pour un modèle considéré de manière isolé, mais fait sens pour un système basé sur du machine learning dans son ensemble". Par exemple, considérant des exigences de latence, la disponibilité peut être menacée par des attaques par déni de service, qui vont complètement bloquer l'accès au système, ou par d'autres attaques qui vont augmenter le temps de réponse de la part du système. Un autre exemple, considérant des exigences de qualité, est celui d'un adversaire qui va cibler des

exigences d'espace mémoire ou de rapidité de calcul. Les performances relatives à la tâche à réaliser par le système peuvent aussi être menacées, pour compromettre un usage fiable du système. Ainsi, la précision du modèle peut être en jeu. Dans ce cas, les attaques contre l'accessibilité d'un système présentent un lien avec les attaques contre l'intégrité, mais diffèrent fortement dans leur concept. En effet, dans le cas d'une attaque de *data poisoning* ou d'*exemples adverses*, la modification des scores de confiance est un but en soi pour réduire la précision sur des exemples particuliers. Dans le cas d'une attaque contre l'accessibilité concentrée sur des performances relatives à la tâche à réaliser, le but final est d'impacter le comportement du système dans lequel le modèle se trouve. Les attaques de *data poisoning* qui visent une baisse de précision pour tous les exemples sont une forme d'attaque contre l'accessibilité d'un système.

La Figure A.3 propose un résumé des différentes attaques présentées précédemment, et l'intégrité, la confidentialité, ou l'accessibilité d'un réseau de neurones

A.4 Résumé des contributions

A.4.1 Positionnement et préliminaires

Dans cette thèse, nous étudions les exemples adverses pour une tâche de *classification d'images*. Plus précisément, nous concentrons notre travail sur les exemples adverses *digitaux*, et non les exemples adverses physiques. Les exemples adverses digitaux sont relatifs aux perturbations qui interviennent directement au niveau des pixels considérés en entrée du modèle attaque. Pour attaquer une API par exemple, un adversaire va perturber directement les pixels d'une image. Au contraire, pour les exemples adverses physiques, les perturbations adverses sont réalisées directement sur des objets réels. Par exemple, un adversaire va modifier un panneau de signalisation pour tromper le système de reconnaissance d'une voiture autonome.

Les exemples adverses pour une tâche de classification d'image se doivent soit d'apparaître bénins (la perturbation adverse est imperceptible). En pratique, le caractère bénin ou d'imperceptibilité ne peut pas être traduit directement. De ce fait, un proxy mathématique est défini de telle sorte que minimiser ce proxy revient à diminuer le caractère suspect de la perturbation. Ce proxy est aussi utilisé pour définir la *capacité* de l'adversaire, c'est-à-dire à quel point un adversaire peut modifier une image x pour réaliser un exemple adverse $x' = x + \delta$, avec x un image, δ la perturbation adverse, et x' l'image adverse résultante. Dans la grande majorité des travaux sur les exemples adverses, le proxy utilisé est la norme l_p: la capacité de l'adversaire est une borne supérieure pour $\|\delta\|_p$. Plus précisément la norme l_∞ [73], la norme l_2 [105], la norme l_1 [106] ou encore la norme l_0 [107] sont couramment utilisées. Quelques travaux considèrent la distance de Wasserstein [83], les rotations et translations [60], ou encore les changements de couleur [61].

Considérant un modèle cible, deux grands types de méthodes existent pour construire un exemple adverse. Le premier type de méthode peut être utilisé par un adversaire quand il a un accès complet au modèle (scénario boite blanche). Dans ce scénario, l'adversaire a une connaissance totale du modèle à attaquer: architecture, valeur des paramètres, etc. De manière importante, il peut calculer les gradients (de la fonction de coût, de la fonction de sortie, etc.) du réseau par rapport à son entrée. Cela lui permet d'utiliser des attaques dites *gradient-based* qui se basent sur les calculs des gradients pour construire un exemple adverse. Le second type de méthode d'attaque est utile à un adversaire qui ne peut pas calculer les gradients du modèle cible. Ce type de scénario

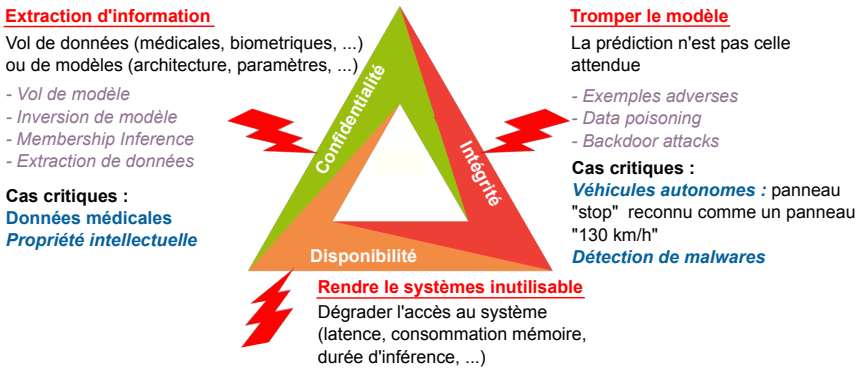

Figure A.3: Triade CIA (Confidentialité, Integrité, Disponibilité) pour un modèle de machine learning, et les attaques correspondantes.

intervient quand l'adversaire n'a pas un accès total au modèle qu'il souhaite attaquer (scénario boite noire). Ce type de scénario présente différentes nuances. L'adversaire peut n'avoir accès qu'à la classe prédite, ou alors de manière plus souple, n'avoir accès qu'aux scores de confiance pour chaque classe prédite. Dans le premier cas, l'adversaire pourra utiliser des attaques dites *decision-based*, et dans le second cas des attaques dites *score-based*.

A.4.2 Objectifs

Pour mieux comprendre en détails la cause de la vulnérabilité d'un réseau de neurones aux exemples adverses, nous cherchons à faire de nouveaux liens entre la robustesse aux exemples adverses et la manière dont le système de vision humain traite les fréquences. Notamment, nous tirons avantage du fait que les humains se reposent principalement sur les composantes basses fréquences d'une image pour prendre une décision.

Dans le but de trouver des méthodes pour réduire la transférabilité des perturbations adverses, nous étudions les raisons de cette transférabilité, et en déduisons des méthodes de défense adaptées au scénario de transférabilité. Pour ce faire, nous commençons par une évaluation de la robustesse des réseaux de neurones déployés avec des méthodes de quantification et l'impact de ces méthodes sur la transférabilité. Ensuite, nous développons une méthode innovante pour restreindre la transférabilité entre deux réseaux de neurones.

De plus, au regard des différentes menaces qui peuvent cibler un modèle de machine learning, nous nous intéressons aux nouvelles menaces potentielles qui pourraient être liées à celles existantes. Plus précisément, nous formalisons une nouvelle attaque qui vise à détériorer l'accès à un système qui se repose sur un modèle de machine learning. De manière intéressante, nous montrons comment ce type d'attaques peut être réalisé en tirant parti des méthodes de construction des exemples adverses.

A.4.3 Analyse fréquentielle des perturbations adverses

Publication proposée pendant la thèse:

- [174] Bernhard, R., Moellic, P. A., Mermillod , M., Bourrier, Y., Cohendet, R., Solinas, M., Reyboz, M. *Impact of Spatial Frequency Based Constraints on Adversarial Robustness.* IEEE International Joint Conference on Neural Networks (2021). 2021.

Dans ce premier travail, nous réalisons une analyse des exemples adverses au niveau fréquentiel. Cela permet notamment de mieux comprendre la différence de sensibilité aux perturbations adverses entre les humains et les réseaux de neurones. En effet, la vulnérabilité des réseaux de neurones standard aux exemples adverses vient du fait que les modèles exploitent sur des caractéristiques différentes de celles exploitées par les humains [57, 113, 114, 175]. De manière intéressante, pour les modèles robustes, il existe un lien fort entre robustesse et interprétabilité : les caractéristiques sur lesquelles se reposent les réseaux de neurones robustes sont interprétables, et vice versa. Plus un modèle est robuste, plus il va se reposer sur des caractéristiques interprétables par des humains [117, 118, 171, 176].

Le lien entre robustesse et interprétabilité pour les réseaux de neurones a été étudié au niveau des fréquences par Zhang *et al.* [114]. Les auteurs montrent via des expériences que pour des modèles robustes, les concepts basse fréquences comme la forme sont plus importants que des concepts moins interprétables liés aux hautes fréquences.

Pour les humains, des résultats expérimentaux en psychologie cognitive montrent l'importance des basses fréquences dans la prise de décision [177–179]. De ce fait, une hypothèse raisonnable serait qu'un modèle entrainé pour exploiter de manière quasi-exclusive les basses fréquences serait alors robustes contre les exemples adverses. Cette hypothèse a déjà été exploitée dans des travaux qui tirent parti de méthodes de pré-traitement [180–182], ou qui forcent un modèle à exploiter l'information contenue dans les bits de poids fort [172]. Cependant, ces méthodes restent agnostiques des propriétés fréquentielles spécifiques des jeux de données en jeu.

Notre contribution dans ce travail a été d'étudier en général le lien entre la robustesse contre les exemples adverses et les propriétés spécifiques des données au niveau fréquentiel.

Dans un premier temps, nous nous sommes intéressés au lien entre les fréquences contenues dans les jeux de donnée et l'information sur lesquels se reposaient les réseaux de neurones entrainés sur ces jeux de donnée. Les résultats des expériences menées ont permis de tirer les conclusions suivantes :

- En étudiant des modèles appris sur des jeux de données filtrés en basse et haute fréquence, nous montrons comment les fréquences sur lesquelles se reposent les modèles sont influencées par la localisation de l'information dans le spectre fréquentiel. Notamment, nous montrons qu'en fonction du jeu de données, un modèle appris sur ces données peut être précis ou pas sur ce même jeu de donnée filtré en basse ou haute fréquence.

- Les modèles appris sur des jeux de donnée filtrés en basse et haute fréquence présentent des sensibilités différentes au bruit localisé dans des fréquences spécifiques. Notamment, les modèles appris sur des jeux de donnée filtrés pour ne garder que les basses fréquences ne présentent que très peu de sensibilité au bruit localisé dans les hautes fréquences.

- Les perturbations adverses exploitent tout le spectre de fréquence, et ne sont pas localisées uniquement dans les hautes fréquences.

Dans un second temps, nous utilisons les résultats précédents pour mieux comprendre comment apporter de la robustesse aux modèles en utilisant des contraintes fréquentielles lors de la phase d'entrainement. Notamment, nous étudions sous quelles conditions, telle ou telle contrainte, concentrée sur les basses et hautes fréquences, peut être source de robustesse contre les perturbations adverses. Nous montrons notamment que:

- Une contrainte fréquentielle induit des niveaux de robustesse très différents en fonction des caractéristiques fréquentielles du jeu de données. Par exemple, la même contrainte forçant un modèle à se reposer sur les basses fréquences peut donner lieu à une différence en termes de précision sur des exemples adverses de l'ordre de 41%, avec certains modèles ne présentant aucune robustesse.

- Nous montrons qu'une contrainte forçant un modèle à se reposer sur les basses fréquences donne lieu à de la robustesse si le modèle se repose se base de manière prédominante sur les basses fréquences, et ne présente pas de sensibilité au bruit localisé dans les hautes fréquences.

- Pour les modèles se reposant sur de l'information contenu dans tout le spectre fréquentiel, nous identifions des combinaisons de contraintes, mêlant basses et hautes fréquences, qui peuvent apporter de la robustesse.

- On montre que les contraintes utilisées peuvent être utilisées de manière intéressante en combinaison avec d'autres méthodes de défense existantes.

Ce travail a permis de bien appréhender l'importance des propriétés fréquentielles des données en jeu dans la robustesse contre les exemples adverses. Tandis que la littérature existante est concentrée sur des concepts tels que la forme, le contour, ou les textures, nous proposons une étude plus générale directement au niveau des fréquences. Les résultats de ce travail et les conclusions de plusieurs travaux en psychologie cognitive mettent en lumière l'importance de considérer les caractéristiques fréquentielles de l'information dans le développement de futures méthodes de défense.

A.4.4 Impact de la quantification sur la robustesse et la transférabilité

Publication proposée pendant la thèse:

- [188] Bernhard, R., Moellic, P. A., Dutertre, J.-M. *Impact of low-bitwidth quantization on the adversarial robustness for embedded neural networks.* IEEE International Conference on Cyberworlds (CW). 2019.

Au vu des performances qui peuvent être atteintes grâce aux modèles de machine learning, il existe une volonté croissante de déployer ces modèles sur des architectures matérielles. Cependant, ces appareils s'accompagnent de contraintes en termes d'espace mémoire et de rapidité en inférence. Par exemple, un microcontrôleur ARM Cortex-M4 comme le STM32F4 n'a que 384 ko de mémoire vive et 2Mo de mémoire flash [6]. De plus, pour des appareils comme des téléphones mobiles ou des objets connectés comme des capteurs industriels, la consommation d'énergie est un facteur critique. Enfin, la rapidité en inférence est nécessaire pour éviter tout problème de latence.

Trouver des solutions pour embarquer des réseaux de neurones dans ces circuits de type microcontrôleur est donc nécessaire. Certaines APIs comme l' API *Android Neural Network* [7] ont

[6] https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html
[7] https://developer.android.com/ndk/guides/neuralnetworks

déjà été développées pour permettre de déployer efficacement des réseaux de neurones entrainés avec des frameworks répandus comme Tensorflow [8]. L'entreprise ARM, de son côté, propose un kit de développement logiciel pour faire le lien entre un modèle de machine learning et des CPUs ARM (ou GPUs) [9]. D'un côté plus théorique, deux pistes différentes sont étudiées pour répondre aux problématiques d'espace mémoire et de temps en inférence. La première consiste à réduire le nombre de paramètres [189–193]. La seconde piste étudiée concerne les méthodes de quantification [194–203]. La quantification consiste à réduire le nombre de bits utilisé pour stocker les valeurs des paramètres et/ou des sorties de différentes couches d'un réseau de neurones. Cette solution permet donc de diminuer l'espace mémoire nécessaire pour stocker tous les paramètres du modèles, ainsi que d'accélérer leur processus d'inférence.

Notre contribution dans ce travail a été d'étudier la robustesse des réseaux de neurones quantifiés, en considérant divers scénarios. Le premier scénario concerne un réseau de neurones quantifié qu'un adversaire va cibler directement avec différentes méthodes d'attaque. Contrairement à ce qui avait été montré dans la littérature scientifique, nous montrons que la quantification n'apporte aucune vraie robustesse contre les exemples adverses. Plus précisément, la quantification cause l'apparition du phénomène de *gradient masking*, qui induit une fausse impression de sécurité. En effet, le *gradient masking* empêche certaines méthodes d'attaque de trouver un exemple adverse efficace pour tromper le modèle, en rendant les gradients inexploitables. D'une part, nous montrons comment certaines attaques *gradient-based* (c'est-à-dire qui se reposent du le calcul de gradients) souffrent particulièrement du *gradient masking* causé par la quantification. Certaines autres attaques gradient-based, moyennant un travail de recherche sur les paramètres optimaux, arrivent à trouver des exemples adverses qui trompent le modèle. D'autre part, des attaques *decision-based* (c'est-à-dire utilisant les scores de confiance), reposant sur une approximation du gradient, arrivent à trouver des exemples adverses efficaces aussi facilement que pour un modèle standard. La quantification n'apporte alors aucune robustesse, que ce soit pour un adversaire dans un scénario boîte blanche ou boîte noire.

Dans un deuxième temps, nous étudions la robustesse des réseaux de neurones quantifiés dans un scénario d'attaque par transférabilité, qui est un phénomène au coeur de cette thèse. Plus précisément, nous considérons des modèles quantifiés à des niveaux différents (c'est-à-dire différents nombres de bits), et étudions la transférabilité des perturbations adverses entre ces modèles. Ce travail a permis de montrer que les perturbations adverses se transfèrent mal entre modèles avec différents niveaux de quantification. Pour expliquer ce résultat, nous mettons en lumière deux phénomènes. Le premier concerne le fait que les gradients ne sont pas alignés entre les différents modèles. Ainsi, la "direction" qu'il faut prendre pour trouver un exemple adverse sur un modèle n'est pas celle qu'il faudrait prendre pour tromper un autre modèle. Deuxièmement, nous montrons de manière théorique comment un certain niveau de quantification peut ruiner l'effet adverse d'un exemple adverse calculé avec un autre niveau de quantification.

Enfin, en nous basant sur les résultat précédents, nous construisons une défense contre les exemples adverses qui repose sur l'utilisation d'un ensemble de modèles avec des niveaux de quantification différents.

Ce travail a permis de mettre à jour la vulnérabilité des modèles de machine learning déployés avec des méthodes de quantification. Nous avons montré que la transférabilité entre différents modèles quantifiés avec des niveaux de quantification différents est faible, ce que l'on explique par deux phénomènes. En conséquence de ces résultats, nous développons une défense basée sur un ensemble de plusieurs modèles quantifiés avec différents niveaux de quantification.

De manière générale, ce travail souligne le fait que les caractéristiques algorithmiques des modèles qui sont déployés doivent faire l'objet d'une étude particulière vis-à-vis des problématiques de robustesse. Ceci étant d'autant plus appuyé que des modèles de réseaux de neurones vont être de plus en plus déployés sur des architectures matérielles.

A.4.5 Leurrage de perturbations adverses transferées

Publication proposée pendant la thèse:

- [214] Bernhard, R., Moellic, P. A., Dutertre, J.-M. *Luring of transferable adversarial perturbations in the black-box paradigm.* IEEE International Joint Conference on Neural Networks (2021). 2021.

[8] https://www.tensorflow.org/
[9] https://developer.arm.com/products/processors/machine-learning/arm-nn

Considérant un modèle de machine learning pouvant être attaqué, la situation la plus répandue est celle d'un adversaire dans un scénario boîte noire. Notamment, ce type de scénario est relatif à tous les services à bases de machine learning disponibles via des APIs distantes. L'adversaire n'a pas un accès total au modèle, et ne peut pas calculer ses gradients. De ce fait, il doit utiliser des attaques de type *decision-based*, *score-based* ou le phénomène de transférabilité. Concernant les deux premiers types d'attaques, elles sont potentiellement soumises à la limitation venant du nombre de requêtes à effectuer au modèle cible pour obtenir un exemple adverse (qui peut se compter en milliers). Lorsque les requêtes sont tarifées, le coût pour monter ce type d'attaque peut être prohibitif. De plus, ce type de méthode d'attaque se reposant sur des requêtes similaires soumises au modèle, le processus d'attaque peut facilement être détecté [136]. Il peut donc être plus simple pour un adversaire d'utiliser le phénomène de transférabilité : construire un exemple adverse sur un modèle substitut et le transférer au modèle cible. La transférabilité apparaît donc comme une des menaces les plus préoccupantes en termes de sécurité du machine learning. Ce phénomène est d'autant plus aggravé par le fait que le même modèle peut être déployé dans des environnements avec des contraintes de sécurité différentes. Un modèle peu protégé peut alors être utilisé comme modèle substitut pour attaquer via la transférabilité le même modèle dans un système plus sécurisé.

Dans ce travail, nous proposons une nouvelle méthode pour limiter la transférabilité des perturbations adverses entre deux modèles de réseaux de neurones. Etant donné un modèle cible M qu'un défenseur veut protéger contre les exemples adverses, nous proposons une méthode qui permet d'entrainer un modèle T, c'est-à-dire une version augmentée du modèle M, telle que les perturbations adverses ne se transfèrent pas de T à M. De plus, entrainer T requiert uniquement de pouvoir faire des requêtes à M, et ainsi ne nécessite pas de jeu de données labellisé. Notre méthode est donc applicable pour n'importe quel modèle déjà entrainé.

Le modèle T consiste en l'augmentation du modèle M avec un réseau de neurones P placé en aval, pour avoir $T = M \circ P$. Pour que les perturbations adverses ne se transfèrent pas de T à M, nous concevons l'*effet de leurrage*. Cet effet de leurrage tire parti de l'observation que le transfert d'un exemple adverse entre deux modèle survient car les deux modèles utilisent des caractéristiques similaires et vulnérables [57]. En conséquence, on entraine P de sorte que:

- les caractéristiques *vulnérables* de T et M soient différentes, ou alors ne se comportent pas de la même façon face à une attaque. Dans ce dernier cas, les modèles T et M prédisent deux classes différentes pour un exemple adverse x', ce qui permet sa détection.

- les modèles T et M prédisent la même classe pour des exemples normaux. Cela revient à ce que les modèles M et T partagent les mêmes caractéristiques *utiles* à la tâche de classification.

Il apparait donc nécessaire que la manière de prédire de M soit incluse dans P, à la fois pour garder une bonne prédiction pour les exemples normaux, et pour qu'il puisse induire une prédiction différente de M pour les exemples adverses. Pour réaliser ces deux objectifs, nous entrainons le modèle P avec une fonction de coût, qui exploite l'ordre (en termes de grandeur) des scores de confiance du modèle M. L'intuition derrière cette fonction de coût est la suivante. Si le modèle M prédit; : "La classe A est la classe prédite, la classe B est la classe avec le second plus grand score de confiance", cette fonction de coût correspond pour le modèle T à "La classe A est la classe prédite, l'écart entre les scores de confiance pour les classes A et B est le plus grand possible". De ce fait, les modèles M et T se reposent sur des concepts différents, et donc présentent des vulnérabilités différentes. Notre méthode de défense est conceptuellement novatrice, dans le sens où elle ne cherhe pas à rendre le modèle cible M plus robuste contre les exemples adverses, et où elle ne cherche pas à anticiper l'attaque comme c'est le cas avec les méthodes de post-processing par exemple.

Une fois cette effet de luring vérifié expérimentalement, nous proposons trois scénarios dans lesquels on peut en tirer parti:

- Le scénario qui correspond à la manière dont l'effet de leurrage a été conçu. Un défenseur possède un modèle M totalement caché de l'adversaire, qui ne peut pas lui soumettre de requête. Le défenseur souhaite mettre à disposition du public un modèle T tel que les exemples adverses ne se transfèrent pas de T à M. Dans ce cas, il construit T de sorte que $T = M \circ P$ avec P entrainé pour causer l'effet de leurrage. Le modèle T est disponible dans un scénario boîte noire, pour ne pas que son architecture soit disponible, ce qui permettrait à l'adversaire de séparer T de M. Nous montrons de plus que cette méthode est compatible avec des méthodes de défense déjà existantes.

- Un scénario plus souple où le défenseur veut distribuer une version publique de son modèle M, qui soit totalement disponible, c'est-à-dire disponible dans un scénario boîte blanche. Nous décidons d'inclure l'effet de leurrage dans un modèle M', qui a la même architecture que le modèle M. Pour ce faire, nous considérons au départ les mêmes paramètres pour le modèle M' que pour le modèle M, et entrainons les premières couches du modèle M' avec la même fonction de coût que le modèle P du scénario précédent.

- Un scénario de détection pour un système distant composé des deux modèles M et M', auquel on peut faire des requêtes et qui renvoie la réponse du modèle M' (l'effet de leurrage fonctionnant du modèle M' vers le modèle M). En effet, un des objectifs de l'effet de leurrage est de faire en sorte que M' et M prédisent deux classes différentes pour un exemple adverse x'. Un adversaire qui attaquerait ce système avec des attaques *decision-based* ou *score-based* a besoin de faire des milliers de requête au système. Notre défense permet de rapidement détecter ce processus de construction d'un exemple adverse en arrêtant de renvoyer une réponse dès que M et M' prédisent deux classes différentes.

D'un point de vue général, ce travail ouvre une nouvelle voie de recherche pour les défenses dans un scénario boîte noire, qui est la situation la plus rencontrée en termes de menace contre l'intégrite du processus d'inférence d'un réseau de neurones.

A.4.6 Utilisation des perturbations adverses pour attaquer l'accessibilité d'un système

Les menaces contre l'intégrité ou la confidentialité sont vues uniquement au niveau du modèle de machine learning, de sa phase d'entrainement et d'inférence. Au contaire, les menaces contre l'accessibilité ont une définition plus vaste et doivent être vues au niveau d'un système. Ce système utilise un modèle de machine learning et présente des prérequis et contraintes relatifs à son utilisation. Il est nécessaire de considérer les interactions entre le modèle et le système dans son ensemble. Plus particulièrement, il faut voir ce type de menace au niveau du processus qui va traiter l'entrée ou la sortie du modèle, pour prendre des décisions relatives à la raison d'être du système. En conséquence, l'accessibilité d'un système est définie via les performances et prérequis du système. Ces prérequis peuvent concerner une grande diversité de caractéristiques, comme le temps de calcul (c'est-à-dire qu'il ne doit pas y avoir de latence), des limitations mémoire, la bande passante du réseau, etc. Lors d'une attaque contre l'accessibilité d'un système, un adversaire peut cibler l'accès au système, comme c'est par exemple le cas avec les attaques par déni de service, qui exploitent des failles logicielles ou matérielles.

Dans le cas d'un système basé sur un modèle de machine learning, plusieurs méthodes d'attaque ont déjà été proposées. Shumailov *et al.* ont conçu les *sponge examples* [66], qui visent les performances d'un système. Cette attaque modifie l'entrée d'un modèle de réseau de neurones (à l'instar des exemples adverses) pour augmenter la consommation énergétique et la durée en inférence d'une cible matérielle sur laquelle est implémentée ce modèle. C'est à notre connaissance l'unique attaque qui vise l'accessibilité d'un système qui intervient lorsque le modèle sous-jacent est en phase d'inférence. Shumailov *et al.* ont aussi proposé une attaque qui ralentit et détériore le processus d'entrainement d'un modèle en modifiant l'ordre de soumission des données du jeu d'entrainement au modèle [67]. De même, Grosse *et al.* [68] pervertissent le processus d'entrainement en jouant sur l'initialisation des paramètres. Le *data poisoning*, quand il vise à réduire la précision du modèle de machine learning, est aussi une attaque contre l'accessibilité d'un système, car il vise les performances du système se reposant sur ce modèle en termes de précision.

Dans ce travail, nous formalisons une nouvelle attaque qui intervient lorsque le modèle de machine learning est déjà entrainé. Elle vise à dégrader la qualité de la prédiction du modèle, sans toutefois chercher à le tromper (à l'instar des exemples adverses). Cette attaque ne réduit pas la précision du modèle, mais fait en sorte que le modèle renvoie des prédictions incertaines. Ce type d'attaque est donc particulièrement préoccupant pour les systèmes qui, comme dans beaucoup de domaines d'application, ne requièrent pas uniquement une bonne précision de le part du modèle sous-jacent, mais aussi des prédictions *de confiance*.

Nous concevons deux cas possibles où un système considère que le modèle n'est pas assez confiant dans la prédiction de la classe :

- Le score de confiance pour la classe prédite est trop faible.

- La différence de score de confiance entre la classe prédite et d'autres classes est trop faible.

Pour chacun de ces scénarios, nous montrons comment réaliser l'attaque en tirant parti d'une méthode d'attaque existante pour les exemples adverses. De manière intéressante, nous montrons que des modèles protégés contre les exemples adverses avec des méthodes destinées à augmenter leur robustesse, restent vulnérables à ce type d'attaque.

Au-delà de la nouvelle menace contre l'accessibilité d'un système que nous introduisons, ce travail permet de mettre en lumière des liens existants entre la robustesse d'un modèle aux attaques contre l'intégrité et la robustesse d'un système aux attaques contre son accessibilité. De plus, ce travail illustre comment la connaissance d'un mécanisme derrière une menace (intégrité) peut être utilisée pour concevoir un tout autre type d'attaque.

A.5 Conclusion

Le premier objectif de cette thèse était de tirer des nouvelles conclusions à-propos de la robustesse aux exemples adverses et de la nature fréquentielle des données en jeu. Cet objectif faisait partie d'un objectif plus général : mieux comprendre la vulnérabilité des modèles à des perturbations auxquelles les humains ne sont pas sensibles. Nous avons choisi de considérer ce problème sous le prisme fréquentiel, avec comme motivation et base de notre travail des résultats en psychologie cognitive qui montrent la prévalence des caractéristiques en basse fréquence dans le processus de classification humain. Notamment, nous avons identifié des facteurs précis qui conditionnent l'apparition de robustesse lorsqu'un modèle est entrainé à tirer parti de manière prédominante d'information présente dans les basses fréquences. De manière plus générale, ce travail est un encouragement à mieux tenir compte des connaissances en neuro-psychologie dans le développement de futures défenses.

Le second objectif de cette thèse était relatif à l'impact des méthodes de quantification sur la robustesse et le phénomène de transférabilité des perturbations adverses. Cette évaluation de l'influence de la quantification sur les exemples adverses permet une meilleure prise de conscience des problématiques d'intégrité liées à l'usage croissant de méthodes de quantification pour embarquer des modèles sur des architectures matérielles publiques. Notamment, nous avons montré que la quantification n'offrait qu'une robustesse très limitée. Cette conclusion est particulièrement importante considérant le fait que de plus en plus de tâches dites critiques sont réalisées sur des systèmes embarqués.

La transférabilité des perturbations adverses, qui est un outil très puissant pour un adversaire dans un scénario boîte noire, est fortement liée au déploiement croissant de modèles sur des architectures matérielles. Nous avons observé une faible transférabilité entre des modèles quantifiés avec des niveaux de quantification différents. Ces résultats ont mené au développement d'une défense à base d'un ensemble de modèles quantifiés. Nous avons ensuite étudié spécifiquement les raisons du phénomène de transférabilite. Cela a mené au développement du concept innovant d'*effet de leurrage*. Nous avons montré comment induire l'effet de leurrage, et détaillé comment il pouvait s'appliquer dans de nombreux scénarios. De manière intéressante, puisque ce concept est nouveau, il est compatible avec les méthodes de défense existantes. De plus, l'effet de leurrage peut être utilisé de manière à détecter le processus de construction d'un exemple adverse contre un système distant.

Enfin, nous avons étendu notre travail aux menaces contre l'accessibilité/disponibilité. Notamment, nous avons montré comment des méthodes pour construire des exemples adverses peuvent être utilisées pour attaquer un système qui requiert un certain niveau de confiance pour les prédictions. Dans de nombreux domaines d'applications, la précision d'un modèle n'est pas suffisante en soi. Pour prendre une décision, un système requiert que le modèle sous-jacent renvoie des prédictions avec un certain niveau de certitude. Nous avons conceptualisé une notion d'incertitude basée sur le doute du modèle entre plusieurs classes. Nous avons ensuite montré comment les perturbations adverses peuvent être utilisées pour produire cette notion d'incertitude dans les prédictions d'un modèle. Ce dernier travail de la thèse fait un lien entre les exemples adverses et d'autres types de menace qui ne visent pas l'intégrité d'un modèle. De plus, on souligne le fait que dans de nombreux cas d'application, une protection contre les exemples adverses ne signifie pas que le modèle peut être déployé de manière sécurisée.

A.6 Perspectives

Lors de la phase d'inférence, les menaces algorithmiques (logicielles) ont été très largement couvertes dans la littérature scientifique, qu'elles soient relatives à l'inversion de modèle, l'extraction

de données, le vol de modèle, les attaques *membership inference* ou les exemples adverses. Cependant, une conséquence directe de la volonté croissante de déploiement des modèles est l'extension de la surface d'attaque pour un adversaire. En effet, un adversaire peut tirer avantage de modèles qui deviennent de plus en plus disponibles sur des cibles matérielles, pour monter des attaques physiques (matérielles).

Ce phénomène est particulièrement préoccupant pour l'intégrité de modèles de machine learning, et plus spécifiquement dans le cas des exemples adverses. En effet, un modèle peut être protégé de manière algorithmique par une certaine méthode de défense, qui prévient des modifications malicieuses des entrées données au modèle. Cependant, pour tromper le modèle, un adversaire peut réaliser une attaque par injection de faute, pour faire changer la valeur des bits des paramètres du modèle [221], ce qui revient à une modification malicieuse des poids. Même si le modèle était protégé contre les exemples adverses, son caractère embarqué le rend vulnérable à un autre type d'attaque. De même, considérant un même modèle dans une API distante et embarqué sur un appareil public, un adversaire peut utiliser une attaque par canal auxiliaire pour retrouver l'architecture du modèle [14]. Ainsi, il peut tirer parti de cette information pour réaliser une attaque par transférabilité contre le modèle distant. L'évolution des usages vers une omniprésence des modèles de machine learning dans la vie de tous les jours est donc en train de rendre les modèles vulnérables à un nombre croissant de méthodes d'attaque.

Pour lutter contre les menaces d'ordre physique, certaines méthodes de défense issues du monde de la sécurité matérielle existent, comme l'obfuscation. Cependant, ces méthodes ne sont pas conçues spécifiquement pour les modèles de machine learning. Au vu du déploiement croissant de ces modèles, prendre en compte les possibilités d'attaques à la fois au niveau logiciel et physique, devient une nécessité. En nous basant sur ces conclusions, nous préconisons un développement de méthodes de défense mélangeant des caractéristiques algorithmiques et physiques, c'est-à-dire des méthodes qui tirent parti du meilleur des deux mondes. Plus précisément, nous considérons que les futures schémas de défense ne seront efficaces que si ils combinent des protections contre les menaces à la fois logicielles et matérielles. Notamment, une défense logicielle pourrait tenir en compte d'une possible modification des paramètres pour contrer les attaques qui vont modifier les valeurs des bits des paramètres. Dans la même veine, une protection matérielle pourrait être conçue pour considérer des caractéristiques spécifiques à un modèle de réseau de neurones. Pour résumer, nous croyons fermement qu'un mélange protections physiques et algorithmiques est le meilleur chemin à emprunter vers un machine learning plus sécurisé.

Projets, publications et communications

Projets

Cette thèse est financée par le CEA-Leti. Les travaux menés dans cette thèse s'inscrivent dans plusieurs projets de recherche collaboratifs : le projet Européen InSecTT[10], le programme *Investissements d'avenir* (ANR-10-AIRT-05, irtnanoelec) et le programme ANR PICTURE (AAPG2020)[11]. Cette thèse a pu profiter du supercalculateur français "Jean Zay" grâce à l'*accès AI dynamique*[12].

Publications

Cette thèse, qui a débuté en novembre 2018 et a pris fin en novembre 2021, a donné lieu à la publication d'articles scientifiques:

- **Impact of low-bitwidth quantization on the adversarial robustness for embedded neural networks**
  Rémi Bernhard, Pierre-Alain Moellic, Jean-Max Dutertre
  (2019) IEEE International Conference on Cyberworlds
  Code: `https://gitlab.emse.fr/remi.bernhard/Quantization-and-Adversarial-Robustness`

- **Luring of transferable adversarial perturbations in the black-box paradigm**
  Rémi Bernhard, Pierre-Alain Moellic, Jean-Max Dutertre
  (2021) IEEE International Joint Conference on Neural Networks (IJCNN)
  Code: `https://gitlab.emse.fr/remi.bernhard/Luring_Adversarial/`

[10] `www.insectt.eu`
[11] `https://picture-anr.cea.fr`
[12] `http://www.idris.fr/jean-zay/`

- **Impact of Spatial Frequency Based Constraints on Adversarial Robustness**
  Rémi Bernhard, Pierre-Alain Moellic, Martial Mermillod, Yannick Bourrier, Romain Cohendet, Miguel Solinas, Marina Reyboz
  (2021) IEEE International Joint Conference on Neural Networks (IJCNN)
  Code: `https://gitlab.emse.fr/remi.bernhard/Frequency_and_Robustness/`

- **A Review of Confidentiality Threats Against Embedded Neural Network Models**
  Raphaël Joud, Pierre-Alain Moellic, Rémi Bernhard, Jean-Baptiste Rigaud
  (2021) Workshop on Wireless Intelligent Secure Trustable Things, IEEE 7th World Forum on Internet of Things

- **An Overview of Laser Injection against Embedded Neural Network Models**
  Mathieu Dumont, Pierre-Alain Moellic, Raphael Viera, Jean-Max Dutertre, Rémi Bernhard
  (2021) Workshop on Wireless Intelligent Secure Trustable Things, IEEE 7th World Forum on Internet of Things

Communications

Durant cette thèse, les avancées et résultats ont été présentés à différentes occasions, pour encourager les avancées scientifiques dans le domaine de la robustesse des réseaux de neurones face aux exemples adverses:

- **Impact of Quantization for Embedded Neural Network Models on the Adversarial Robustness**
  (2019) Journée thématique Sécurité, fiabilité et tesf de SoC 2 : challenges et opportunités dans l'ère de l'Intelligence Artificielle, Paris, France

- **Security of Neural Networks: Attacks, Defenses and Evaluation methods**
  (2019) Journée de la recherche, Centre de Microélectronique de Provence, Gardanne, France

- **Impact of Low-bitwidth Quantization on the Adversarial Robustness for Embedded Neural Networks**
  (2019) Journée de la recherche, Centre de Microélectronique de Provence, Gardanne, France

- **Adversarial Robustness of Quantized Embedded Neural Networks**
  (2019) C&ESAR IA conference, Artificial Intelligence and Defense, Rennes, France

- **Luring Adversarial Perturbations**
  (2020) C&ESAR IA conference, Artificial Intelligence and Defense, Rennes, France

- **Luring of Transferable Adversarial Perturbations in the Black-Box Paradigm**
  (2021) Groupe de Recherche ISIS (Information, Signal, Image and viSion), online

Activités d'enseignement

En parallèle des activités de recherche, j'ai assuré la tenue de cours magistraux et des séances de travaux pratiques pour les étudiants de l'Ecole Nationale Supérieure des Mines de Saint-Etienne:

- **Algorithmique et programmation**. 2019-2020. Travaux pratiques. Etudiants de première année à l'Ecole Nationale Supérieure des Mines de Saint-Etienne.
  Etude et implémentation d'algorithmes de théorie des graphes en C: Dijkstra, Bellman-Ford and Chandy-Misra-Haas.

- **Securité du machine learning**. 2018-2020. Travaux pratiques et cours magistraux. Etudiants de seconde année à l'Ecole Nationale Supérieure des Mines de Saint-Etienne.
  Présentation des principales menaces pouvant viser un modèle de machine learning. Présentation des exemples adverses, de méthodes d'attaque et de défense.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *2021LYSEM033*

Rémi BERNHARD

INTEGRITY OF NEURAL NETWORKS: DEFENDING AGAINST ADVERSARIAL EXAMPLES AND THEIR TRANSFERABILITY

Speciality : Microelectronics

Keywords : Neural networks, adversarial examples, transferability, integrity

Abstract :

Regarding the success of deep learning in various tasks, ranging from image classification to speech recognition, there is a growing will to deploy neural networks models in the everyday life. However, these models have been shown to be vulnerable to multiple threats. Among them, the phenomenon of adversarial examples is of primary interest as a menace to the integrity of a machine learning model. Adversarial examples refer to a type of attack where an adversary maliciously modifies some input, in order to fool a model at inference time.

Whether neural network models are used in distant APIs or embedded on devices, adversarial examples therefore constitute a threat to this deployment, and more generally to the integrity of machine learning models. More particularly, in some use-cases such as autonomous driving, adversarial examples constitute a major threat to the public safety.
Among the different aspects of adversarial attacks, the phenomenon of the transferability of adversarial perturbations is particularly concerning, as it is a powerful tool for an adversar who does not have a direct access to the target model (black-box setting). Indeed, an adversarial example crafted on a substitute model can also fool another model. As the black-box setting is the most common scenario for a threat model, studying and hindering transferability would therefore allow to greatly enhance the security of machine learning models.

In this thesis, we aim at better understanding the vulnerability of neural networks to adversarial examples, and at developing efficient schemes to thwart their transferability.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : *2021LYSEM033*

Rémi BERNHARD

INTEGRITE DES RESEAUX DE NEURONES : DEFENSES CONTRE LES EXEMPLES ADVERSES ET LEUR TRANSFERABILITÉ

Spécialité: Microélectronique

Mots clefs : Réseaux de neurones, exemples adverses, transferabilité, integrité, sécurité, …

Résumé :

Au vu du succès du deep learning dans de nombreuses tâches, de la classification d'images à la reconnaissance de la voix, il y a une volonté croissante de déployer des modèles de réseaux de neurones. Cependant, il a été montré que ces modèles sont vulnérables à de nombreux types d'attaques. Parmi celles-ci, les exemples adverses constituent une menace grandissante pour l'intégrité d'un système de machine learning. Les exemples adverses sont un type d'attaque où un adversaire va modifier de manière malicieuse une entrée pour tromper un modèle à lors de l'inférence.

Que ces modèles soient utilisés via des APIs distantes ou embarqués sur divers appareils, les exemples adverses constituent donc une menace à ce déploiement, et plus généralement à l'intégrité des modèles de machine learning. Plus particulièrement, dans certains cas comme les voitures autonomes, ils constituent une menace sérieuse à la sécurité civile. Parmi les différents aspects des attaques adverses, la transférabilité des perturbations adverses est particulièrement préoccupante, car elle représente un outil puissant pour un adversaire sans accès direct au modèle cible (scénario boîte noire). En effet, un exemple adverse élaboré sur un modèle substitut peut arriver à tromper un autre modèle. Le scénario boîte noire étant le plus commun, étudier la transférabilité est une considération de premier plan.

Le but de cette thèse est de mieux comprendre la vulnérabilité des réseaux de neurones aux exemples adverses, et de développer des schémas de défense efficaces pour contrer leur transférabilité.