Robustness of neural network image classifiers to meaningful adversarial examples

Lucas Anquetil

Normandie Université

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité **INFORMATIQUE**

Préparée au sein de l'**INSA Rouen Normandie**

Robustesse des réseaux de neurones profonds classifieurs d'images aux exemples adversaires pertinents
Robustness of neural network image classifiers to meaningful adversarial examples

Présentée et soutenue par

LUCAS ANQUETIL

Thèse soutenue le 07/11/2023

devant le jury composé de :

M. STÉPHANE HERBIN	CHERCHEUR HDR - Office National d'Études et de Recherches Aérospatiales de Paris	Rapporteur
MME ILEANA OBER	PROFESSEUR DES UNIVERSITÉS - Université Paul Sabatier, Toulouse	Rapporteur
M. AMAURY HABRARD	PROFESSEUR DES UNIVERSITÉS - Université Jean Monnet, Saint-Etienne	Membre
MME SAMIA AINOUZ	PROFESSEUR DES UNIVERSITÉS - INSA de Rouen Normandie	Président du jury
M. STEPHANE CANU	PROFESSEUR DES UNIVERSITÉS - INSA de Rouen Normandie	Directeur de thèse

Thèse dirigée par **STEPHANE CANU** (LABORATOIRE D'INFORMATIQUE DE TRAITEMENT DE L'INFORMATION ET DES SYSTEMES)

MI/IΣ

litis

Abstract

Machine learning is revolutionizing the world in many ways, enabling the creation of artificial systems capable of performing complex tasks. In medicine, machine learning systems are used to analyze medical data and identify patterns helping doctors diagnose diseases more accurately. In retail, machine learning systems are widely used to personalize customers' shopping experience by recommending products based on their browsing history and behavior. In the transportation sector, machine learning is at the heart of the development of autonomous cars, vehicles that would be capable of driving without human intervention, for which a robustness assessment is of paramount importance.

The problem addressed in this thesis is the robustness evaluation of deep neural networks as image classifiers using adversarial examples, which are original examples to which a perturbation, imperceptible to the human eye, is added changing the classifiers' output. More precisely, we propose a method to compute these adversarial perturbations using a dictionary, a set of images learned from a classifier, and a dataset. We also question the quality of the metrics used to compute distances between two images, leading to reliable estimation of a relevant robustness measure of image classifiers.

In a first contribution, we propose a new way to fool classifiers. While competitive with state-of-the-art methods, this approach learns a dictionary of images for which a certain linear combination produces such an adversarial perturbation. This dictionary makes it possible to construct universal adversarial perturbations, independent of the original examples, and transferable, effective for fooling other classifiers. The learned dictionary also makes it possible to visualize the elementary shapes underlying the creation of adversarial perturbations.

In a second contribution, we propose to learn a similarity measure between images by optimizing the cost matrix of the Wasserstein distance between their representations derived from a neural network. Using image representations enables to compute more relevant and realistic similarities than using their pixels. Pixel-based image distances don't take semantics into account. Our proposal makes it possible to explore the richness of learned representations, to construct a relevant Wasserstein distance. Having such a relevant similarity measure between images may reveal useful when working on life-at-risk computer vision applications.

Résumé

L'apprentissage automatique révolutionne le monde de nombreuses manières en permettant la création de systèmes intelligents capables d'effectuer des tâches complexes. En médecine, les systèmes d'apprentissage automatique sont utilisés pour analyser des données médicales et identifier des motifs qui peuvent aider les médecins à diagnostiquer les maladies de manière plus précise. Dans le domaine du commerce, les systèmes d'apprentissage automatique sont largement utilisés pour personnaliser l'expérience d'achat des clients en recommandant des produits en fonction de leur historique et de leur comportement de navigation. Dans le secteur du transport, l'apprentissage automatique est au cœur du développement des voitures autonomes, des véhicules qui seraient en capacité de circuler sans intervention humaine pour lesquels une évaluation de la robustesse est primordiale.

Le problème abordé dans cette thèse est celui de l'évaluation de la robustesse des réseaux de neurones profonds en tant que classifieur d'images à l'aide d'exemples adversaires, c'est-à-dire des exemples originaux auxquels est ajoutée une perturbation, imperceptible à l'œil nu, changeant la sortie des classifieurs. Plus précisément, nous proposons une méthode de calcul de ces perturbations adversaires à l'aide d'un dictionnaire, un ensemble d'images apprises à partir d'un classifieur et d'un jeu de données. Nous remettons également en question la qualité des métriques utilisées pour calculer les distances entre deux images, afin d'estimer de manière fiable une mesure de robustesse pertinente des classifieurs d'images.

Dans une première contribution, nous proposons une nouvelle manière de tromper les classifieurs. Tout en étant compétitive avec les méthodes de l'état de l'art, cette approche permet d'apprendre un dictionnaire d'images dont une certaine combinaison linéaire produit une telle perturbation adversaire. Ce dictionnaire permet de construire des perturbations adversaires universelles, indépendantes des exemples originaux, et transférables, efficaces pour tromper d'autres classifieurs. Le dictionnaire appris, permet aussi de visualiser les formes élémentaires à la base des perturbations adversaires.

Dans une deuxième contribution, nous proposons d'apprendre une mesure de similarité entre images en optimisant la matrice de coût de la distance de Wasserstein entre leurs représentations issues d'un réseau de neurones. Utiliser les représentations des images permet de calculer des similarités plus pertinentes et réalistes qu'en utilisant leurs pixels. En effet, les distances d'images basées sur les pixels ne considèrent pas la sémantique. Notre proposition permet d'explorer la richesse des représentations apprises, afin de construire une distance de Wasserstein pertinente. Disposer d'une telle mesure de similarité pertinente entre images peut s'avérer utile lorsque l'on travaille sur des applications de vision par ordinateur pour lesquelles des vies sont en jeu.

Contents

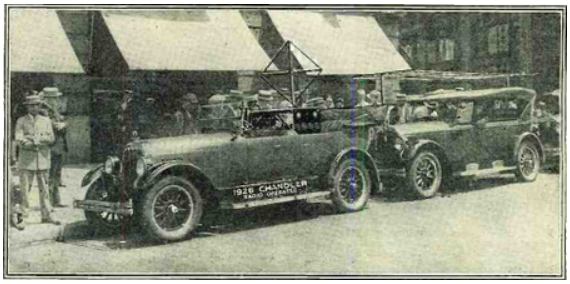
Introduction

Context and motivation

In recent years, the field of artificial intelligence (AI) has experienced unprecedented growth. Problems once considered out of reach, are now easily solved thanks to deep neural networks[1] [146].

In particular, deep networks played a central role in the recent revolution in computer vision. The massive adoption of deep neural networks dates back to 2012, when the Alexnet network [88] largely won the ImageNet image classification challenge, setting a new performance standard in the field.

Since then, numerous neural network based innovations emerged, solving image and video related tasks. This led to the mass adoption of deep neural networks for everyday tasks such as virtual assistants [160], facial recognition payment systems, driving assistants [2], and it is highly likely that in the coming years we will see the advent of fully autonomous driving systems [79].



(a) The radio-operated automobile, American Wonder in 1925. Source: Wikipedia.com

(b) Waymo's robo-taxi. Source: Image courtesy of Waymo.

Figure 1: Autonomous cars in 1925 and 2025, from Chandler (Motor Car Company) to Chandler (Phoenix Suburb).

Autonomous vehicles

Research into autonomous cars has been going on for several decades, with the first experiments and prototypes developed in the 1920s. The first attempt at driverless vehicles dates back to 1925 [3], as illustrated with the Chandler car shown in Figure 1a, and gained momentum in the 1980s when researchers succeeded in developing automated highway systems [57, 77]. This led to semi-autonomous and autonomous vehicles directly connected to the road infrastructure. Improvements in autonomous vehicles were largely achieved in Germany and the USA between 1980 and 2000 [3, 90].

Autonomous vehicles owe much to defense industry research into unmanned equipment, led by the famous US Defense Advanced Research Projects Agency (DARPA) challenge [138]. The

[1] https://chat.openai.com

[2] https://www.tesla.com/autopilot

[3] https://computerhistory.org/blog/where-to-a-history-of-autonomous-vehicles/?key=where-to-a-history-of-autonomous-vehicles

DARPA challenge is a turning point for autonomous vehicles for two reasons. Firstly, it was the first time a car was able to drive autonomously for more than 200 kilometers at an average speed of over 30 km/h, demonstrating that it is indeed possible for cars to drive on their own, without human supervision. Secondly, from a technical point of view, the DARPA challenge is a revolution because Stanford's winning car, shown Figure 2, actually won thanks to the use of machine learning based tools. It was proof that the best way to program an automatic driving system is through machine learning.

Then, private companies became interested in the field and began to carry out in depth research. Google's driverless car drew attention to autonomous vehicles and attracted talent from many disciplines to work on them. Tesla, a competitor of Google's in the autonomous car field, is an electric car company that helped spark interest in autonomous vehicles with its Autopilot product, a semi-autonomous driving system capable of steering, braking and accelerating the car. Two years after the launch of Tesla's Autopilot, Google followed with its Waymo driverless car project, offering a driverless taxi service in Phoenix, Arizona's Chandler neighborhood. Figure 1b shows the inside of such a robo-taxi system. In doing so, it became the first company to offer a commercial service using fully autonomous vehicles. One of the main concerns of autonomous cars is safety. Indeed, autonomous cars incorporate deep neural networks to make driving decisions. However, these models show their limitations in terms of robustness against adversarial examples [22].

Figure 2: DARPA challenge winning car from Stanford, which was the first to integrate machine learning models to drive. Source: herox.com

Adversarial examples

With the recent success of deep learning, several authors [16, 154] highlighted their vulnerability to adversarial attacks. Adversarial attacks are functions of the classifier neural network and an original example producing an adversarial perturbation, i.e. a slight perturbation imperceptible to the human eye, which when added to the original example leads to a misclassification as illustrated in Figure 3. These adversarial perturbations enable neural networks to be attacked by any hacker wishing to corrupt them anonymously. However, thanks to these adversarial perturbations, it is also possible to estimate the vulnerability and robustness of a neural network. In order to estimate the robustness of neural networks, some authors proposed to compute pseudo-random adversarial perturbations which, thanks to efficient optimization, enable the precisely draw the neural network's limits. These perturbations have been criticized for not representing realistic image perturbations, observable in real life, thus leading to an irrelevant estimate of the neural network robustness. In response to these critics, other types of adversarial perturbations have been proposed, such as common corruptions [72], or adversarial patches [20] aiming to compute

a more relevant robustness estimation of neural networks.

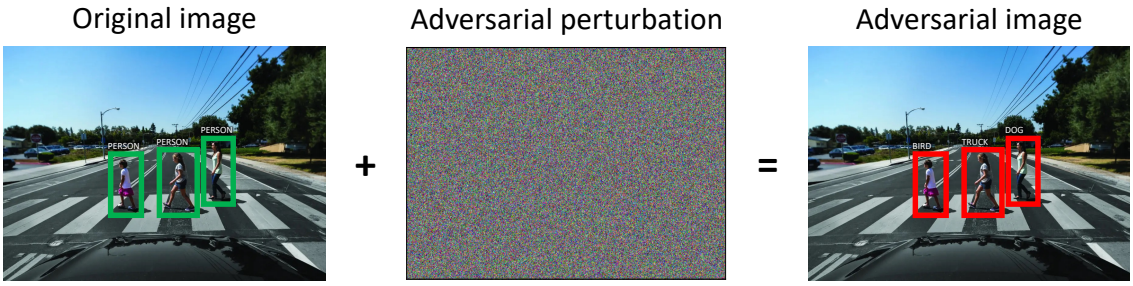


Figure 3: Example of an adversarial image in the context of objects detection by an autonomous car.

Contributions

In this thesis, we proposed two contributions.

First, we proposed to model the adversarial noise space using a dictionary of adversarial atoms, creating an adversarial attack bridging the gap between specific and universal adversarial attacks. Through multiple submissions and revisions, our proposition evolved into an efficient modeling of the adversarial noise space, enabling to derive a robust and transferable adversarial attack. Indeed, the derived attack proves to be competitive with specific state-of-the-art attacks, while being more robust, i.e. more resistant to adversarial example detection mechanisms. Besides, the computed adversarial examples are more transferable, they allow fooling other neural networks. Thanks to the universal model of the adversarial noise space, our experiments show that the computed adversarial perturbations are more universal than state-of-the-art methods, i.e. they can fool a neural network on several original examples. The learned dictionary also makes it possible to visualize the elementary shapes underlying the creation of adversarial perturbations. By relying on a dictionary, our proposition allows us to reveal the global weaknesses and dynamics leading to neural network misclassification. In addition, our proposition also questions the currently used adversarial metrics. Generally speaking, our proposition questions the consideration currently given to pseudo-random adversarial perturbations to effectively assess a relevant robustness estimation of deep neural networks.

In the second contribution, we questioned the currently used similarity measures between images. More precisely, we propose learning a similarity measure between images by optimizing the cost matrix of the Wasserstein distance between the images' representation extracted from a neural network. The similarity measures derived from our proposition are more interesting than the distances currently used, as they use image representations rather than pixels, leading to more relevant and realistic image similarities. Indeed, pixel based image distances do not consider semantics. Our proposition allows to explore the wealth of learned representations, in order to build a relevant Wasserstein distance. Especially, we tested our proposition in different application contexts to make it more robust and efficient, according to the hyper-parameters. Having such a relevant similarity measure between images can be useful when working on life-at-risk computer vision applications. However, due to lack of time, we were unable to address all the problems associated with applying our proposition to more complex and large-scale datasets.

The different contributions have been proposed in the following papers:

1. Jordan Frecon, Lucas Anquetil, Yuan Liu, Gilles Gasso, Stéphane Canu "*Adversarial Dictionary Learning*", published at Conférence sur l'Apprentissage Automatique (CAp), 2021.
2. Lucas Anquetil, Yuan Liu, Jordan Frecon, Stéphane Canu, Gilles Gasso "*LIMANS: Linear Model of the Adversarial Noise Space*", submitted at International Conference on Learning Representations (ICLR), 2023.

Outline

This thesis is divided into four chapters as follows:

- Chapter 1 introduces the background knowledge of deep neural networks and formalizes the different adversarial notations, metrics, attacks, and defenses.
- Chapter 2 presents the proposed dictionary-based adversarial attack.
- Chapter 3 details our proposition to learn a more relevant similarity measure between images.
- Finally, we conclude our work and present future perspectives.

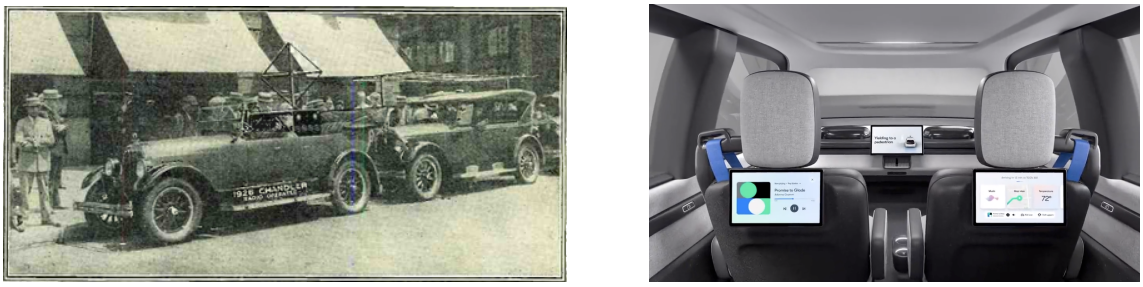
Introduction

Contexte et motivation

Ces dernières années, le domaine de l'intelligence artificielle (IA) a connu une croissance sans précédent. Des problèmes autrefois considérés comme insurmontables, peuvent désormais être résolus facilement grâce aux réseaux de neurones profonds[4] [146].

En particulier, les réseaux de neurones profonds ont joué un rôle central dans la récente révolution de la vision par ordinateur. L'adoption massive de ces réseaux de neurones profond remonte à 2012 lorsque le réseau Alexnet [88] a largement remporté le défi de classification d'images à partir d'ImageNet, établissant ainsi un nouveau standard de performances dans le domaine.

Depuis, de nombreuses innovations reposant sur les réseaux de neurones ont émergé, résolvant des tâches liées à l'image et à la vidéo. Cela a conduit à l'adoption massive de réseaux de neurones profonds pour des tâches quotidiennes telles que les assistants virtuels [160], les systèmes de paiement par reconnaissance faciale, les assistants de conduite [5], et il est fort probable que dans les années à venir nous verrons l'avènement de systèmes de conduite complètement autonomes [79].



(a) L'automobile radiocommandée, American Wonder, en 1925. Source : Wikipedia.com

(b) Le robo-taxi de Waymo. Source: Image fournie par Waymo.

Figure 4: Les voitures autonomes en 1925 et cent ans plus tard, de Chandler (Motor Car Company) à Chandler (banlieue de Phoenix).

Autonomous vehicles

La recherche sur les véhicules autonomes existe depuis plusieurs décennies avec les premières expérimentations et prototypes développés dans les années 1920. Une des première tentative de véhicules sans conducteur remonte à l'année 1925 [6] avec la voiture Chandler présentée Figure 4a. Ces travaux ont pris de l'ampleur dans les années 1980 lorsque des chercheurs ont réussi à développer des systèmes autoroutiers automatisés [57, 77]. Cela a conduit à des véhicules semi-autonomes et autonomes directement connectés à l'infrastructure routière. Les améliorations des

[4] https://chat.openai.com

[5] https://www.tesla.com/autopilot

[6] https://computerhistory.org/blog/where-to-a-history-of-autonomous-vehicles/?key=where-to-a-history-of-autonomous-vehicles

véhicules autonomes ont été largement réalisées en Allemagne et aux États-Unis entre 1980 et 2000 [3, 90].

Les véhicules autonomes doivent beaucoup à la recherche menée par le secteur de la défense sur les équipements sans pilote, dirigée par le célèbre défi de l'Agence pour les projets de recherche avancée de défense (DARPA) des États-Unis [138]. Le défi DARPA est un tournant pour les véhicules autonomes pour deux raisons. Tout d'abord, c'est la première fois qu'une voiture a pu conduire de manière autonome sur plus de 200 kilomètres à une vitesse moyenne supérieure à 30 km/h, ce qui a démontré qu'il est effectivement possible que les voitures circulent seules, sans supervision humaine. Deuxièmement, d'un point de vue technique, le défi DARPA est une révolution car la voiture gagnante de Stanford, présentée Figure 5, a en réalité remporté grâce à l'utilisation d'outils basés sur l'apprentissage automatique. C'était la preuve que la meilleure manière de programmer un système de conduite automatique est l'apprentissage automatique.

Ensuite, des entreprises privées se sont intéressées au domaine et ont commencé à mener des recherches approfondies. La voiture sans conducteur de Google a suscité une attention sur les véhicules autonomes et a attiré de nombreux talents issus de plusieurs disciplines pour y travailler. Tesla, un concurrent de Google dans le domaine de la voiture autonome, est une entreprise de voitures électriques, qui a contribué à susciter l'intérêt pour les véhicules autonomes avec son produit Autopilot, un système de conduite semi-autonome capable de diriger la voiture, de freiner et d'accélérer. Deux ans après le lancement de l'Autopilot de Tesla, Google a suivi avec son projet de voiture sans conducteur Waymo, en proposant un service de taxi sans conducteur à Phoenix, en Arizona, dans le quartier de Chandler. La figure 4b montre l'intérieur d'un tel système de robo-taxi. Elle est ainsi devenue la première entreprise à proposer un service commercial utilisant des véhicules entièrement autonomes. L'une des principales préoccupations des voitures autonomes est la sécurité. En effet, les véhicules autonomes intègrent des réseaux de neurones profonds pour prendre des décisions de conduite. Cependant, ces modèles ont montré leurs limites en termes de robustesse face à des exemples adversaires [22].

Figure 5: La voiture gagnante du défi DARPA de l'Université Stanford, qui a été la première à intégrer des modèles d'apprentissage automatique pour la conduite. Source: herox.com

Examples adversaires

Avec le récent succès de l'apprentissage profond, plusieurs auteurs [16, 154] ont souligné leur vulnérabilité aux attaques adversaires. Les attaques adversaires sont des fonctions du réseau de neurones classifieur et d'un exemple original, produisant une perturbation adversaire, c'est-à-dire une légère perturbation, imperceptible l'œil nu, qui lorsqu'elle est ajouté à l'exemple original conduit à une mauvaise classification, comme illustré Figure 6. Ces perturbations adversaires per-

mettent d'attaquer les réseaux de neurones, pour quiconque voudrait les pirater anonymement. Cependant, grâce à ces perturbations adversaires il est aussi possible d'estimer les vulnérabilités et la robustesse d'un réseau de neurone. Afin d'estimer la robustesse des réseaux de neurones, certains auteurs ont proposé de calculer des perturbations adversaires pseudo-aléatoires qui, grâce à des optimisations efficaces, permettent de délimiter de manière précise le domaine de viabilité des réseaux. Ces perturbations ont été critiquées [140] car elles ne représentent pas des perturbations d'images réalistes, observables dans la vraie vie, menant ainsi à une estimation de la robustesse des réseaux de neurones trop abstraite. Face à ces critiques, d'autres types de perturbations adversaires ont été proposés comme les perturbations communes (où *common corruptions* en anglais)[72], ou bien les patchs adversaires [20] visant à calculer une estimation plus pertinente de la robustesse des réseaux de neurones.

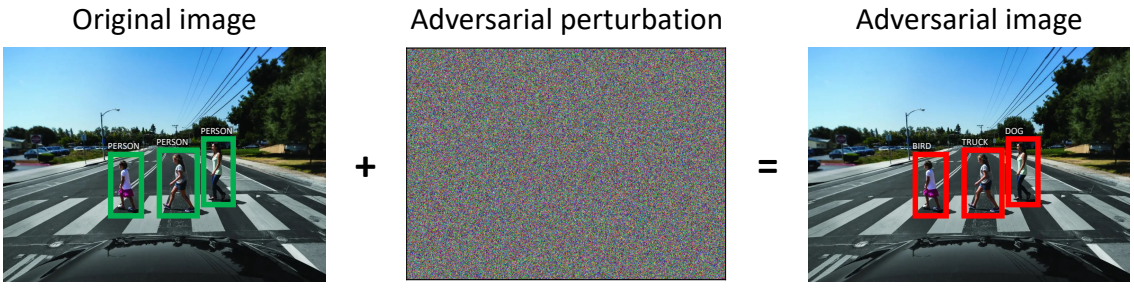


Figure 6: Exemple d'une image adversaire dans le cas d'une detection d'objets par une voiture autonome.

Contributions

Dans cette thèse, nous avons proposé deux contributions permettant de mieux caractériser et retrouver des exemples adversaires.

Dans le deuxième chapitre, nous avons proposé de modéliser l'espace des bruits adversaires à l'aide d'un dictionnaire d'atomes adversaires, créant ainsi une attaque adversaire faisant le lien entre les attaques adversaires spécifiques et les attaques adversaires universelles. À travers plusieurs soumissions et remises en question, notre proposition a évolué vers une modélisation efficace de l'espace des bruits adversaires permettant de dériver une attaque adversaire robuste et transférable. En effet, l'attaque adversaire construite à partir d'un dictionnaire se révèle compétitive par rapport aux attaques spécifiques de l'état de l'art, tout en étant plus robuste, c'est-à-dire qu'elle résiste mieux aux mécanismes de détection d'exemples adversaires. De plus, les exemples adversaires ainsi calculés se révèlent plus transférables, ils permettent de tromper d'autres réseaux de neurones. Du fait d'être calculé à partir d'une modélisation universelle de l'espace des bruits adversaires, nos expériences montrent que les perturbations adversaires calculées sont plus universelles que les méthodes de l'état de l'art, c'est-à-dire qu'elles peuvent tromper un réseau de neurones sur plusieurs exemples originaux. Avoir modélisé l'espace des bruits adversaires par un dictionnaire, nous permet de visualiser les formes élémentaires à la base des perturbations calculées. En s'appuyant sur un dictionnaire, notre proposition nous permet de révéler les faiblesses globales et la dynamique menant à tromper les réseaux neuronaux. En outre, notre proposition remet également en question les métriques adversaires actuellement utilisées. Da manière générale, notre proposition remet en question la considération actuellement donnée aux perturbations adversaires pseudo-aléatoires pour évaluer efficacement et de manière pertinente la robustesse des réseaux de neurones profonds.

Dans la deuxième contribution, nous remettons en question les mesures de similarité entre images actuellement utilisées. Plus précisément, nous proposons d'apprendre une mesure de

similarité entre images en optimisant la matrice de coût de la distance de Wasserstein entre leurs représentations issues d'un réseau de neurones. Les mesures de similarité tirées de notre proposition sont plus intéressantes que les mesures actuellement utilisées car elles utilisent les représentations des images plutôt que leurs pixels, permettant de calculer des similarités entre images plus pertinentes et plus réalistes. En effet, les distances d'images basées sur les pixels, ne considèrent pas la sémantique. Notre proposition permet d'explorer la richesse des représentations apprises, afin de construire une distance de Wasserstein pertinente. Nous avons notamment testé notre proposition dans différents contextes d'application afin de le rendre plus robuste et plus efficace selon le choix des hyperparamètres. Disposer d'une telle mesure de similarité pertinente entre images peut s'avérer utile pour évaluer la robustesse d'un réseau face à des exemples adversaires, notamment lorsque l'on travaille sur des applications de vision par ordinateur dans lesquelles des vies sont en danger. Cependant, par manque de temps, nous n'avons pas pu résoudre tous les problèmes liés à l'application de notre proposition sur des ensembles de données plus complexes et à plus grande échelle.

Les différentes contributions ont été présentées dans les articles suivants :

1. Jordan Frecon, Lucas Anquetil, Yuan Liu, Gilles Gasso, Stéphane Canu "*Adversarial Dictionary Learning*", publié à Conférence sur l'Apprentissage Automatique (CAp), 2021.

2. Lucas Anquetil, Yuan Liu, Jordan Frecon, Stéphane Canu, Gilles Gasso "*LIMANS: Linear Model of the Adversarial Noise Space*", soumis à International Conference on Learning Representations (ICLR), 2023.

Structure du manuscrit

Cette thèse est divisée en quatre chapitres précédés par cette introduction :

- Le premier chapitre introduit les connaissances de base sur les réseaux neuronaux profonds et formalise les différentes notations, métriques, attaques et défenses adversaires.

- Le chapitre deux présente l'attaque adversaire basée sur un dictionnaire que nous avons proposée.

- Le troisème chapitre détaille notre proposition d'apprendre une mesure de similarité entre images plus pertinente.

- Enfin, nous concluons notre travail et présentons les perspectives futures.

Chapter 1

Background and preliminaries

This chapter relies on the notations introduced in [62], making the current work self-sufficient. The work in [62] is considered one of the most comprehensive introductions to deep learning. Readers are invited to refer to it for further details.

1.1 Deep neural network classifiers

Broadly speaking, there have been three waves of deep learning development: at first, deep learning was referred to as cybernetics in the 1940s–1960s [107, 156], then deep learning was known as connectionism in the 1980s–1990s [132], and in 2006 [14, 73, 128] the deep learning and representation learning rise, contrasting from shallow learning which was fancy at the time, to be the current trend we are witnessing today.

The deep learning algorithms we recognize today were first intended to be computational models of biological learning that is, how learning happens in the brain. As a result, one of the names that deep learning has gone by is artificial neural networks (ANNs). While neural networks have sometimes been used to understand brain function [74], they are generally not designed to be realistic models mimicking brain function.

The recent (third wave) rise of deep learning has been made possible thanks to two elements combined, the increase of available large-size datasets and the improvement of computer infrastructure in hardware and software. The age of "Big Data" has made machine learning much easier because the key burden of statistical estimation, generalizing well to new data after observing only a small amount of data, has been considerably lightened [44, 45, 80, 87, 94, 115, 133].

Alongside the data explosion, both the hardware and software enabled a faster training of neural network models and also to build deeper neural networks, that is to increase the complexity of the model, allowing to learn much more complicated patterns within the data and thus reaching a new standard in terms of performances [27, 88, 152].

1.1.1 Neural networks archictures

In this sub-section, we review the different neural network architectures, incrementally according to their complexity.

Feedforward neural networks

In the simplest form, neural networks $f : \mathbb{R}^{\mathrm{P}} \mapsto \mathbb{R}^{\mathrm{C}}$ are called feedforward networks, deep feedforward neural networks, or multilayer perceptrons (MLPs). Considering a feedforward network f_θ, parameterized by θ parameters, and a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{\mathrm{N}}$ of labeled examples with $\mathbf{x}^{(i)} \in \mathcal{X} \subset \mathbb{R}^{\mathrm{P}}$ and its corresponding label $y \in \mathcal{Y} = \{1, ..., \mathrm{C}\}$, the goal of optimizing f is to find the best θ parameters, best modeling the conditional relationship of the data examples $\mathbf{x}^{(i)}$ to their corresponding label $y^{(i)}$ (for ease of reading, f will implicitly refer to f_θ lightening the notation).

A feedforward network defines a mapping from the input space $\mathcal{X}$ to the label space $\mathcal{Y}$, and we define the prediction of the example $\mathbf{x}^{(i)}$ by f with $f(\mathbf{x}^{(i)}) = \hat{y}^{(i)}$.

Feedforward neural networks are called *networks* because they define the composition of different functions (called *hidden layer*) connected only to one after another. These models are prefixed *feedforward* because at example $\mathbf{x} = [x_1, x_2, \ldots, x_P]^T$, its information flows through the intermediate computations of f to finally its output $\hat{y}$,

$$\hat{y} = f(x) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x}))), \tag{1.1}$$

supposing f has three hidden layers. In reality, f can be composed of any number of hidden layers, going up to thousands in some currently used network architecture. The overall number of functions composing f gives the *depth* of the model. The name *deep learning* arose from this terminology. Each hidden layer of the network is typically vector-valued. The dimensionality of these hidden layers determines the *width* of the model.

One way to understand feedforward networks is, to begin with linear functions as hidden layers. Linear functions, such as linear regression, logistic regression, or linear support vector machine, are appealing because they can efficiently and reliably be optimized, either in closed form or with convex optimization. Likewise, we can apply the kernel trick with some nonlinear activation function σ, to obtain a nonlinear learning of the hidden layers producing a nonlinear feedforward network. The previous feedforward network f of three hidden layer shown in Figure 1.1 would be defined as,

$$\begin{aligned} h^{(1)} &= \sigma^{(1)}(\mathrm{W}^{(1)\mathrm{T}}\mathbf{x} + \mathbf{b}^{(1)}), \\ h^{(2)} &= \sigma^{(2)}(\mathrm{W}^{(2)\mathrm{T}}h^{(1)} + \mathbf{b}^{(2)}), \\ h^{(3)} &= \sigma^{(3)}(\mathrm{W}^{(3)\mathrm{T}}h^{(2)} + \mathbf{b}^{(3)}), \\ \hat{y} &= \sigma^{(4)}(h^{(3)}), \end{aligned} \tag{1.2}$$

defining $\theta = [\mathrm{W}^{(1)}, \mathbf{b}^{(1)}, \mathrm{W}^{(2)}, \mathbf{b}^{(2)}, \mathrm{W}^{(3)}, \mathbf{b}^{(3)}]$, while $\sigma^{(1)}, \sigma^{(2)}$ and $\sigma^{(3)}$ are activation functions such as relu, $\sigma(x) = \mathrm{relu}(x) = \max(0, x)$ enabling the hidden layers to be non-linear functions. The activiation function $\sigma^{(4)}$ is quite different as it displays the final prediction of f therefore, for classification $\sigma^{(4)}$ would likely be the softmax function, that is component-wise $\sigma(x)_i^{(4)} = \mathrm{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$ enabling $\hat{y}$ to define a probability distribution over the labels.

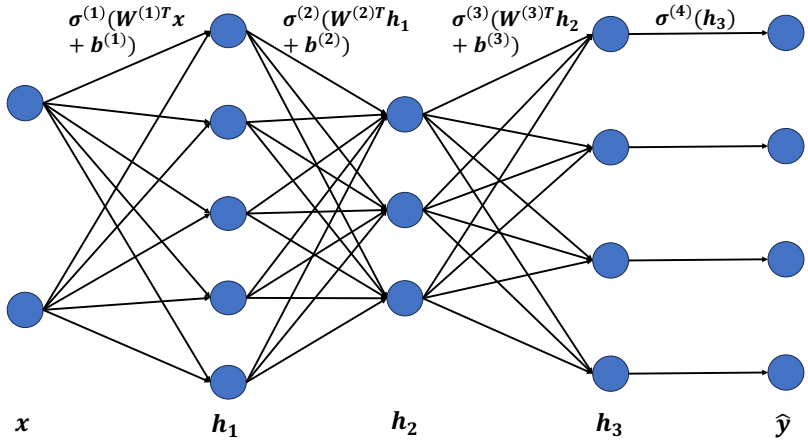


Figure 1.1: Schema of an MLP, composed of three hidden layers. Input and output dimension as well as the dimension of the hidden layers are set arbitrarily for clarity purposes.

Convolutional neural networks

With feedforward neural networks being the simplest case of neural network architecture, the next more complicated architecture type is convolutional neural networks (CNNs), also called convolutional networks [92].

The name "convolutional neural network" indicates the use of a convolution operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their hidden layers. A convolution operation applies a kernel (or filter) to an input signal to extract features or perform spatial transformations.

The output of the convolution $s(\mathbf{x}, \mathrm{K})$ of an input $\mathbf{x} \in \mathbb{R}^{\mathrm{P}}$ and a kernel $\mathrm{K} \in \mathbb{R}^{k}, k \ll \mathrm{P}$, called the feature map, is the sum of the product of each element of K to each element of $\mathbf{x}$ within its range. The output of the convolution $s(\mathbf{x}, \mathrm{K})_i$ that is component-wise,

$$s(\mathbf{x}, \mathrm{K})_i = \sum_{j=1}^{k} \mathbf{x}_{i-j} \mathrm{K}_j. \tag{1.3}$$

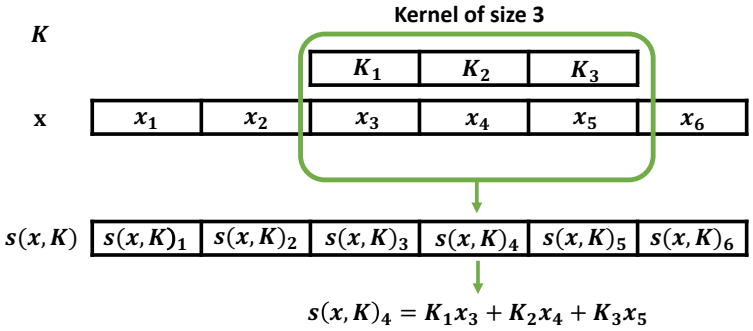


Figure 1.2: Schema of a 1 dimensional convolution operation. The input size and kernel size are set for clarity purpose.

Equation 1.3 and Figure 1.2 define and illustrate a convolution operation over a one-dimensional signal, but the operation can be extended to more dimensions as well. It is important to note that the input of the convolution can be padded with zeros to provide an output feature map of identical shape, preventing edge effects. Including a convolution operation within a neural network architecture, implies that the kernel parameters $\mathrm{K}_j, j \in \{1, \ldots, k\}$, are now part of the θ parameters of f to be found during its optimization.

Adding convolutions improves neural networks in two ways, first by lowering the complexity of the model (size of θ, that is the number of parameters to be found) and second by extracting only the most discriminative features as a pre-process of the next hidden layers. Besides, from a more practical viewpoint, convolution provides a way to deal with different sizes of inputs.

Opposite to feedforward networks, CNNs can set the kernel size much smaller than the input size, resulting in sparse interactions. For example, when working with large-scale data such as high-resolution images, the input size can be 10^5 or 10^6, with convolutions we can detect the small meaningful features in the images, such as corners or edges in the objects, using kernels of size 10^2 or 10^3 only. It means that we need to store fewer parameters, which both reduces the memory requirements of the model and fastens the prediction $\hat{y}$ computation by running fewer operations, improving the runtime complexity. When setting the kernel size to be lower than the input size, the convolution exploits the local connection patterns in the input. These local connections help capture spatial or temporal dependencies within the input, which is particularly useful for tasks such as image and video processing. In practice, convolution is most often used with a pooling operation [185], a downsampling operation that reduces the spatial dimensions of the input by aggregating neighboring values, typically taking the maximum or average value.

Overall CNNs have tremendously outperformed feedforward neural networks thanks to the convolution operation extracting the most discriminative characteristics from the inputs and thus

enabling the overall neural network function to precisely locate and discriminate more complicated patterns.

Residual Networks

CNNs [88, 93] have led to a series of breakthroughs for image classification [88, 141, 181]. At the same time, Simonyan and Zisserman, Szegedy et al. [147, 152] revealed that the network's depth is of crucial importance, as a higher depth leads to better results [70, 78, 147, 152] on the challenging ImageNet dataset [133].

However, up to this point, convolutional neural networks cannot be "too deep" as they become too difficult to optimize. The depth training problem, also known as the vanishing gradient problem, refers to the phenomenon where gradients in deep neural networks diminish exponentially as they backpropagate through multiple hidden layers during the training, making it challenging for earlier hidden layers' weights to be efficiently optimized. Residual connections [71] solve the vanishing gradient problem. Residual connections are links within the deep neural network between the early hidden layers and the final hidden layers as Figure 1.3 shows. As well as proposing the residual connections idea, authors of the original work [71] introduce a series of neural network architectures including residual connections, termed ResNets.

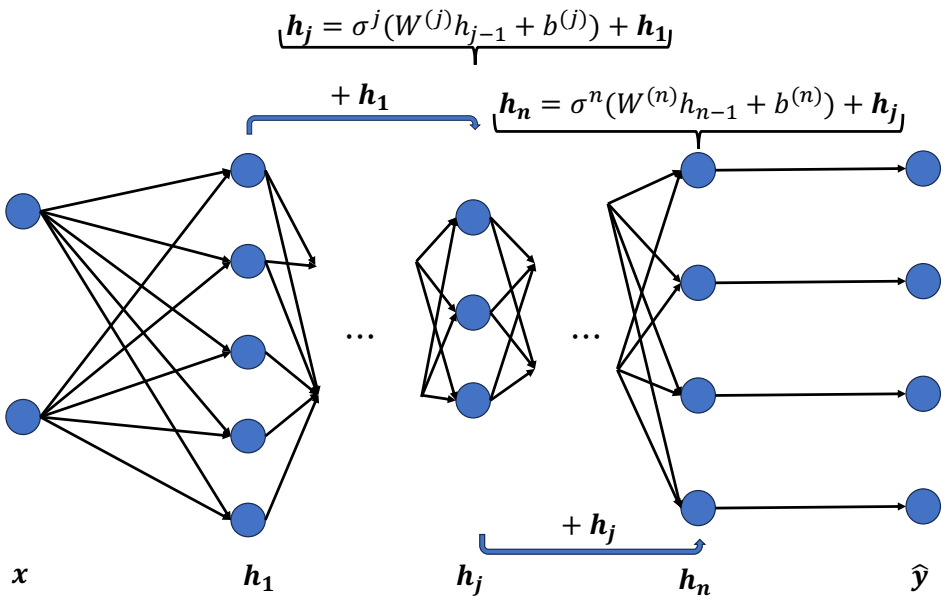


Figure 1.3: Schema of a neural network of n hidden layers with two residual connections. The input size, number of hidden layers, their dimension, and the location of the residual connections are set for clarity purposes.

Figure 1.3 shows a neural network including two residual connections. Even though the residual connections here are only set for display purposes, we can see the hidden layer h_j's parameters will be updated with a gradient of the same magnitude as those of the last layer h_n. By composition, we understand through the other residual connection, that the first hidden layer h_1's parameters will be updated with a gradient only an order of magnitude lower than those of the last layer h_n. This way, early hidden layers can receive insightful gradients allowing us to update them efficiently and therefore solving the vanishing gradient issue.

The neural network architectures used in this thesis involve VGG [147], Inception [153], GoogleNet [152], MobileNet [75], ResNet [71] that are all composed of the presented neural networks architecture tools as well as little specific tricks making them efficient for their purpose at their publication date.

Recent state-of-the-art neural network architectures, such as NAS [186], CoCa[180], ViTs [51] involve new kinds of technology such as variants of attention mechanisms [180] [165] or masking [55] for example. Those technologies are out of the scope of this thesis, however, readers are invited to refer to current benchmark[1] as shown in Figure 1.4, reviewing the current state of the art neural networks models.

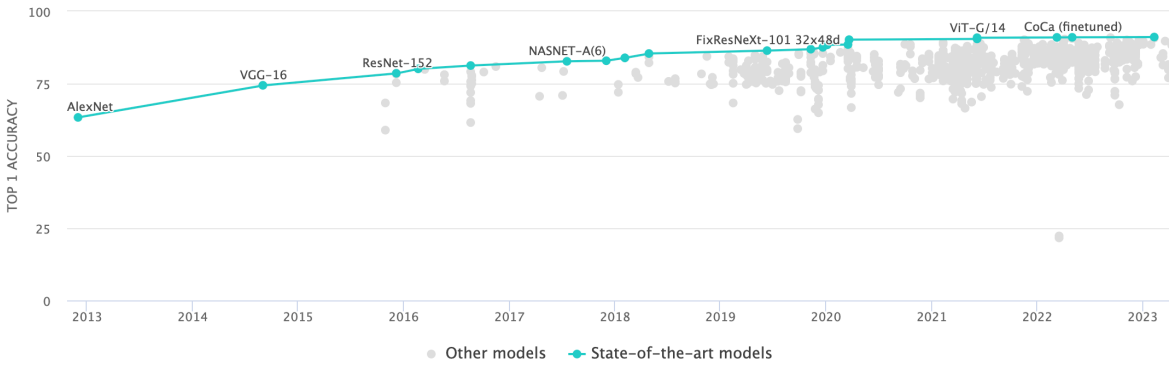


Figure 1.4: Comparison of the state-of-the-art neural network classifiers on the ImageNet dataset as of 2023. Source: paperswithcode.com

1.1.2 Losses

Training a neural network simply means learning (determining) the best values for all the model's parameters θ (weights and bias for example) using empirical examples. In supervised learning, a machine learning algorithm builds a model by examining many labeled examples and attempting to find its best parameters minimizing a loss function (also known as a cost function). Given a labeled dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ with $\mathbf{x}^{(i)} \in \mathcal{X} \subset \mathbb{R}^P$ and $y^{(i)} \in \mathcal{Y} = \{1, \ldots, C\}$ this process is called empirical risk minimization,

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\hat{y}, y), \tag{1.4}$$

with

$$\hat{y} = \text{softmax}([f_1(\mathbf{x}^{(i)}), f_2(\mathbf{x}^{(i)}), \ldots, f_C(\mathbf{x}^{(i)})])^T$$

the predicted label probability vector of $\mathbf{x}^{(i)}$ through f. The predicted label is given by choosing the label receiving the maximum probability, $\text{argmax}_k f_k(\mathbf{x}) = \text{argmax}_k \hat{y}_k$.

In equation 1.4, the loss function is denoted by $\mathcal{L}$ represents a real positive value, a penalty of a bad prediction. The loss is a real value indicating how bad the model's prediction is on a single example. If the model's prediction is perfect then the loss is zero otherwise, the loss is positive. The goal of training a model is to find the best parameters θ, producing the lowest loss, on average, across all examples in $\mathcal{D}$.

The choice of the loss function is of the utmost importance when building machine learning models as it plays a central role in its optimization. Losses are usually associated with the task at hand. It means that some regression loss might not be suited to train a classifier. Here we will review some of the most common classification losses, as other task losses are out of scope for this thesis.

0-1 loss

The simplest classification loss is the 0-1 loss, which simply counts the number of bad predictions,

$$\mathcal{L}(\hat{y}, y) = \begin{cases} 1 & \text{if } \text{argmax}_k \hat{y}_k = y \\ 0 & \text{if } \text{argmax}_k \hat{y}_k \neq y. \end{cases} \tag{1.5}$$

[1] https://paperswithcode.com/sota

Even though the 0-1 loss is quite simple to understand, this loss is not relevant as it can hardly be incorporated into an optimization scheme. Indeed, as we will see in section Section 1.1.3 the 0-1 loss does not allow us to compute gradients with respect to the model's parameters θ, preventing us from updating them.

Hinge loss

Very popular in the early 2000s, Support Vector Machines (SVMs) [34] were at the time, the state-of-the-art classifier models. SVMs are mainly optimized using the hinge loss (also called margin loss). The core idea of hinge loss is to provide a margin-based measure for training classifiers, by maximizing the margin between the decision boundary of the classifier and the training data points. With the hinge loss, the loss is zero if the example is correctly classified, and positive if its prediction lies within a chosen margin from the decision boundary.

$$\mathcal{L}(\hat{y}, y) = \max(0, \rho - \mathbb{1}.\{y = \underset{k}{\operatorname{argmax}}\, \hat{y}_k\}), \tag{1.6}$$

with ρ as the allowed margin hyper-parameter of the hinge loss.

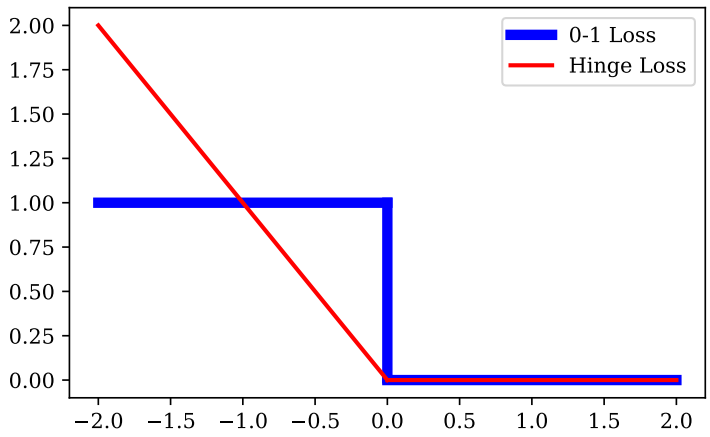


Figure 1.5: Comparison of the Hinge loss and the 0-1 loss. The vertical axis represents the value of the Hinge loss (in red) and zero-one loss (in blue) for fixed ρ = 1, while the horizontal axis represents the value of the prediction y.

Figure 1.5 illustrates the difference between the Hinge and the 0-1 loss. Those two losses are presented because of the impact and fundamental influence they've had in the machine learning community, however, nowadays neural network classifier optimizations do not use them. Instead, the most likely loss involved to optimize neural network classifier is the cross-entropy.

Cross-entropy loss

The cross-entropy loss, as its name suggests, takes roots from the entropy dissimilarity measure. Entropy is a measure of uncertainty or disorder in a probability distribution. The entropy of a discrete probability distribution S is defined as,

$$\text{entropy(S)} = -\sum_{\text{event}} p(\text{event}) log(p(\text{event})), \tag{1.7}$$

with p(event) representing the probability of the realization of an event in the distribution and summing over all possible events.

While entropy measures the disorder in a probability distribution, cross-entropy measures the difference between two probability distributions. In the context of classification, cross-entropy is

used to compare the classifier's probability distribution to the true label distribution represented by a Dirac when considering single-label classification,

$$\mathcal{L}(\hat{y}, y) = -\sum_{k=1}^{\mathrm{C}} o_k log(\hat{y}_k), \tag{1.8}$$

with $\hat{y}_k$ representing the predicted probability of the label k for the example x through f, and o a one-hot vector with 1 at index y representing the true label of x. Cross-entropy is minimized when the probability distribution $\hat{y}$ matches the true distribution o perfectly, which is when $\hat{y}$ puts a probability of 1 on the true label y and 0 elsewhere. Minimizing the cross-entropy effectively maximizes the likelihood of the true labels under the probability distribution $\hat{y}$.

Triplet loss

So far, the presented losses mainly cover the classification task, but in this thesis, we explored the metric learning problem too. The metric learning problem involves learning a distance or similarity metric that captures the relationship or dissimilarity between points. In classification, it is often essential to have an appropriate distance metric $d : \mathbb{R}^{\mathrm{P}} \times \mathbb{R}^{\mathrm{P}} \mapsto \mathbb{R}$ that can effectively measure the similarity or distance between points. In most research and applications, ℓ_p norm distances are set as standard distances however, ℓ_p norm distances may not always be suitable [143], as it treats all dimensions equally and may not reflect the underlying structure or semantics of the data.

Metric learning methods typically leverage labels or pairwise constraints to guide the learning process. A metric learning algorithm optimizes a loss that encourages the desired properties of the learned metric, such as being low for points of the same label, and high for points of different labels. To this end, the triplet loss [137] has been proposed. Given a dataset of labeled examples $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{\mathrm{N}}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^{\mathrm{P}}$ and $y^{(i)} \in \mathcal{Y} = \{1, \ldots, \mathrm{C}\}$, the triplet loss generates triplets $\{(\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n)^{(i)}\}$ from $\mathcal{D}$, with $\mathbf{x}_a$ an anchor point, $\mathbf{x}_p$ a positive point of same label as $\mathbf{x}_a$ and $\mathbf{x}_n$ a negative point of different label. From these triplets, the triplet loss explicitly minimizes the distance between points of the same label $d(\mathbf{x}_a, \mathbf{x}_p)$ and maximizes the distance between points of different label $d(\mathbf{x}_a, \mathbf{x}_n)$ under a margin,

$$\mathcal{L}(d, \mathcal{D}) = \sum_{(\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n) \in \mathcal{D}} max(0, \rho - d(\mathbf{x}_a, \mathbf{x}_n) + d(\mathbf{x}_a, \mathbf{x}_p)), \tag{1.9}$$

with ρ the allowed margin of error.

Optimal transport loss

In recent years, the research community has seen a surge of interest in optimal transport distances [168]. Optimal transport distances, also known as Wasserstein distances (WD) or Earth Mover's distances [131], are mathematical measures that quantify the dissimilarity or distance between two probability distributions. The basic idea behind optimal transport distances is to consider the cost of moving each unit of mass from a source point to a destination point. Originally, the cost function of transporting points is assumed to be known and set as a parameter. In the machine learning context, the WD quantifies the minimum cost required to transform one probability distribution into another, taking into account the distances between the individual points in the distributions.

The WD thus provides a measure of dissimilarity between distributions that considers both their shapes and the geometric structure of their support. With one dimensional point, the WD reads $\mathrm{WD}_{\mathrm{C}} : \mathbb{R}^{\mathrm{N}} \times \mathbb{R}^{\mathrm{M}} \mapsto \mathbb{R}$, and computes the association of N points from one probability distribution a to M points of another probability distribution b, such that the sum of the overall transportation under the cost function C is minimized,

$$\begin{aligned} \mathrm{WD}_{\mathrm{C}}(\mathbf{a},\mathbf{b}) &= \min_{\mathrm{T}\in\mathbb{R}^{\mathrm{N}\times\mathrm{M}}} \sum_{i=1}^{\mathrm{N}}\sum_{j=1}^{\mathrm{M}} \mathrm{T}_{i,j}\mathrm{C}_{i,j}, \\ &\text{such that } \mathrm{T}1 = \mathbf{a} \text{ and } \mathrm{T}^{\mathrm{T}}1 = \mathbf{b}, \end{aligned} \tag{1.10}$$

with the matrix $\mathrm{C} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ being the cost-matrix, that quantifies the effort of moving points, with C_{ij} the effort of moving points from location a_i to location b_j. Figure 1.6 shows a valid transport plan between two points clouds. Its associated WD would be the sum of all the black arrows' costs, representing the cost of moving all the points a_i to all the points b_j.

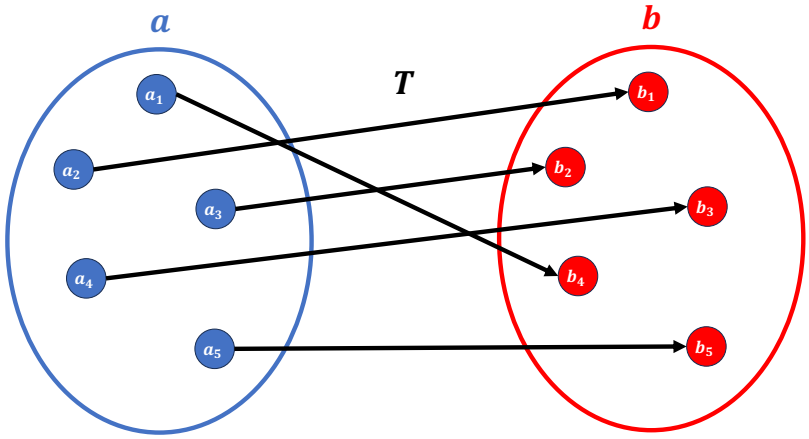


Figure 1.6: Schema of the Wasserstein distance between points from a in red, and points from b in blue. The black arrows represent the association of points from a to points from b, these associations define the transport plan T.

Even though it is very appealing for the geometric understanding and semantic-based definition, computing the exact WD can be computationally expensive, especially for high-dimensional, large-scale problems. To make the problem tractable and computationally efficient, researchers resort to its relaxed version, the Sinkhorn divergence [38]. The Sinkhorn divergence, is an approximation of the WD that leverages the Sinkhorn-Knopp algorithm, a matrix scaling algorithm computing an approximate transport plan efficiently. The Sinkhorn divergence relaxes the strict constraints of the WD by introducing regularization parameters, which control the trade-off between computational efficiency and the accuracy of the approximation. The Sinkhorn divergence $\mathrm{WD}_{\mathrm{C}}^{\alpha}$ simply consists of a regularization term added to the original distance WD_{C},

$$\begin{aligned} \mathrm{WD}_{\mathrm{C}}^{\alpha}(a,b) &= \min_{\mathrm{T}\in\mathbb{R}^{\mathrm{N}\times\mathrm{M}}} \sum_{i=1}^{\mathrm{N}}\sum_{j=1}^{\mathrm{M}} \mathrm{T}_{i,j}\mathrm{C}_{i,j} + \alpha \sum_{k=1}^{\mathrm{N}}\sum_{t=1}^{\mathrm{M}} \mathrm{T}_{i,j}(\log \mathrm{T}_{i,j} - 1), \\ &\text{such that } \mathrm{T}1 = x^s \text{ and } \mathrm{T}^{\mathrm{T}}1 = x^t, \end{aligned} \tag{1.11}$$

with α a regularisation parameter. Usually, the optimization considers instead the dual formulation of the Sinkhorn divergence explicitly including the constraints with the dual variables. Readers are invited to see Section A for details of the dual formulation as well as the Sinkhorn-Knopp algorithm definition.

The design of an appropriate loss according to the objective, the constraints, and the dataset at hand, is complex and requires careful attention. Loss conception and definition is still an ongoing research, see [170] for a survey and more details on the recent improvements in this research field.

1.1.3 Optimization

Once we defined a neural network architecture's parameters θ and the appropriate loss function $\mathcal{L}$, we have the following optimization problem,

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}^{(i)}), y^{(i)}) = \mathcal{L}(\hat{y}, y). \tag{1.12}$$

The simplest and the historically most used strategy to update the parameters θ is to proceed with a stochastic gradient descent (SGD) optimization scheme,

$$\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}(f_\theta(\mathbf{x}^{(i)}), y^{(i)}), \tag{1.13}$$

with $(\mathbf{x}^{(i)}, y^{(i)})$ randomly chosen in the dataset, and γ a fixed step size. Quite basic but efficient, SGD optimization is still used, especially with optimization tricks such as averaged mini-batch gradient computation, or mini-batch sampling strategy. One of the main improvements that have been proposed was that rather than choosing a fixed step size γ, dynamically computing a new step size γ at each gradient update, according to the gradient's magnitude. The Adam [86] optimization strategy has been proposed to dynamically update the step size according to the gradient's magnitude, outperforming the original SGD optimization by far. To calculate the step size, Adam performs an optimization problem on its own by using two moving average variables, namely the first moment of the gradients, the mean with a first hyper-parameter β_1, and the second moment, the variance of the gradients with a second hyper-parameter β_2. These moving averages are updated at each iteration of the optimization process. Readers are invited to refer to Algorithm 1 on page 2 of [86] for a more precise definition of the performed computation of γ.

During the past decade, the Adam optimizer has been the most used and effective, but some variants have been proposed, such as Nadam [53] or Radam [98] for instance. These have been outperformed by other step size computations such as Sophia [97], which enabled us to reach new state-of-the-art performances with transformer models. New state-of-the-art performances are reached using new and better optimizers [31, 110]. Further information and other optimization procedures are given in the fast-ai[2] and Fidle[3] courses.

1.1.4 Generative Adversarial Networks

Another way to train neural networks is to resort to the generative adversarial networks (GAN) frameworks, proposed in [64]. A GAN framework consists of two models competing against each other. On one side of the framework is a generator g trying to generate synthetic data as similar as possible to true data, and on the other side is a discriminator d acting as a classifier, trying to distinguish the true data from the synthetically generated data. More formally a GAN learns to map the simple latent distribution $\mathbf{z} \sim p_z$ to the more complex data distribution, the true data $\mathbf{x} \sim p_{data}$. By having its generator compete against its discriminator, the GAN's objective results in a min/max objective,

$$\min_g \max_d \; \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log d(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log 1 - d(g(\mathbf{z}))]. \tag{1.14}$$

As Figure 1.7 shows, both d and g are optimized from equation 1.14 by backpropagating their loss with respect to their respective model.

[2] https://www.fast.ai/
[3] https://fidle.cnrs.fr

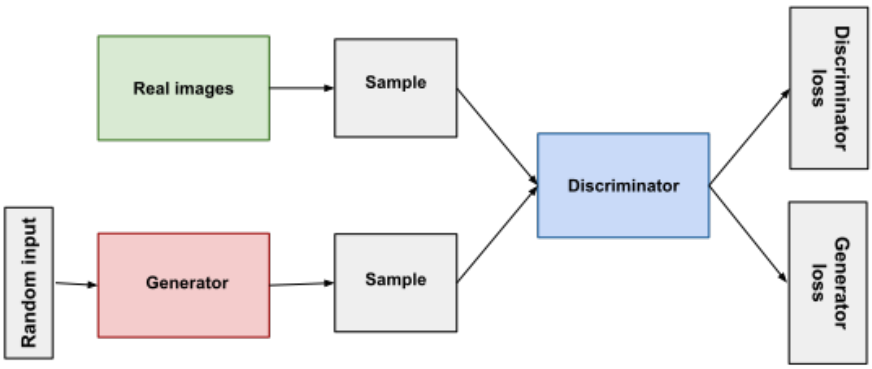


Figure 1.7: Diagram of the GAN optimization scheme.

GAN framework received a lot of attention due to its capacity to optimize a generator producing almost-real synthetic data as Figure 1.8 shows (see [66] for a recent survey on GAN).

Figure 1.8: Example of fake person images generated by GAN models. Source: thispersondoesnotexist.com

Even though a GAN generator does not create so-called adversarial examples 1.19, the core objective of the generator is still to generate imperceptible to the human eye, fake data.

1.1.5 Frameworks

During the past decade, machine learning research has been extensively supported by various companies such as Google, Meta, and Microsoft for example. Fortunately for academic research these companies released very powerful cutting-edge open-source machine learning frameworks such as Tensorflow [1], Pytorch [120], Apache MXNet [30] or the new in town JAX [60] for example. Each optimizer has its own strengths and weaknesses, depending on either research or production, one can be more appropriate than the other. Figure 1.9 gives a comparison of the usage for the last five years. As of today, the Pytorch framework is on the rise.

The work proposed in this thesis heavily relies on the framework Pytorch, which has many efficient and powerful features enabling us to perform extensive experiments. Pytorch has the main advantage of including the Autograd [119] computation package, which as the name suggests, automatically computes reliable gradients of any differentiable loss with respect to its input. In this thesis, we focused on adversarial examples and resorted to publicly released and peer-reviewed

validated benchmarks of the adversarial attack, namely Torchattacks [85] and RobustBench [35]. All the implementations of this thesis are made available[4] and we thank the Pytorch community for releasing such crucial tools to us.

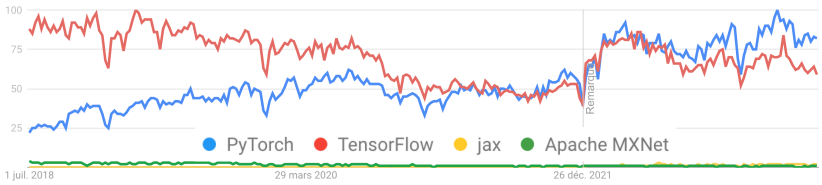


Figure 1.9: Comparison of deep learning framework use, from their Google keyword search. On the x-axis is the time and on the y-axis is displayed a score of the keyword usage in the Google search engine. Source: trends.google.com

1.1.6 Datasets

During this thesis, our experiments were exclusively performed on image datasets, but to some extent, can be extended to other data modalities.

We performed experiments on the widely used benchmark dataset MNIST [94] ("Modified National Institute of Standards and Technology"). This dataset describes a collection of handwritten digit images, specifically a training set with 60,000 examples and a test set with 10,000 examples. Each example in the dataset is a grayscale image of size 28x28 pixels, representing a handwritten digit from 0 to 9. Quickly becoming a limiting dataset, MNIST derived more complex datasets such as Fashion-MNIST [176], EMNIST [33], KMNIST [32] or even QMNIST [178].

The CIFAR-10 dataset [87] is another widely used benchmark dataset in the field of computer vision. CIFAR stands for "Canadian Institute For Advanced Research". The CIFAR-10 dataset includes labeled RGB images of 10 classes. The CIFAR-10 dataset comprises 60,000 RGB images divided into a training set with 50,000 images and a test set with 10,000 images. Each image in the dataset has a size of 32x32 pixels and is labeled with one of ten classes: (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, or truck). The images in CIFAR-10 exhibit more complexity compared to the MNIST dataset. They contain color information and represent a broader range of objects and backgrounds. The dataset is designed to reflect real-world scenarios and challenges in object recognition and classification tasks.

As a step further real-world scenarios, we performed experiments on the ImageNet dataset [45]. It is designed for object recognition and classification tasks and has been crucial in advancing state-of-the-art in computer vision. The ImageNet dataset originally consisted of millions of labeled images from a vast range of object categories. However, the most widely known and used subset of ImageNet is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [133], which focuses on a subset of 1.2 million images across 1000 object classes. Each image in the ImageNet dataset can vary in size and resolution. The dataset covers a wide variety of objects, including animals, vehicles, household items, natural scenes, and more. The dataset also includes a significant number of challenging images, such as those with multiple objects or cluttered backgrounds.

Most of the datasets currently used are available on the Kaggle[5] website. Kaggle is an online platform hosting machine learning competitions, it provides datasets for practice and offers a collaborative environment for machine learning projects. Readers are invited to take a look out of curiosity.

[4] https://github.com/lucasanquetil
[5] https://www.kaggle.com/

1.1.7 The current trend in deep learning

State-of-the-art performances are reached using neural networks to learn complex representations of the data making the final predictions very accurate.

The current trend is to blindly increase the number of parameters of the model (the model's size), allowing it to learn much more complex representations, and achieve outstanding performances. Figure 1.10 shows the rise of the neural network size for different tasks according to their publication date. As we can see, in spite of their consequences, researchers and private companies chose to increase, without any limit, the neural network size.

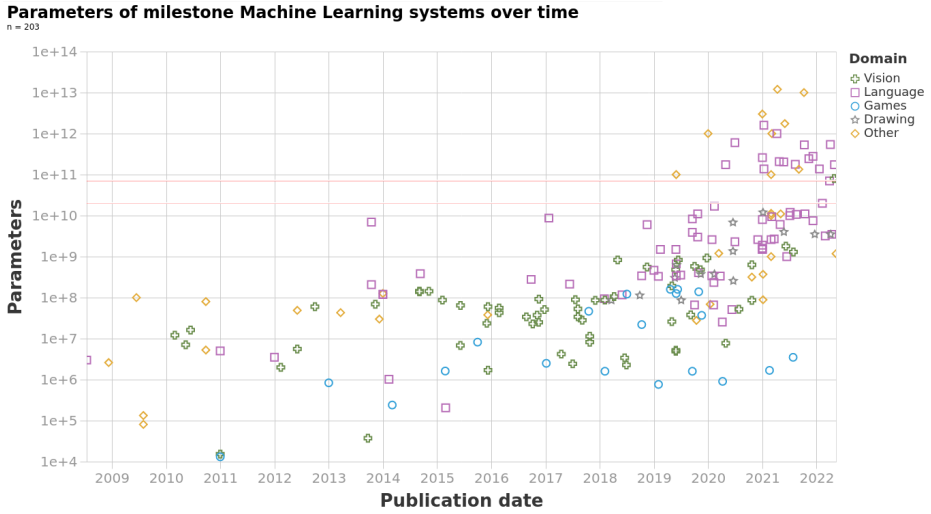


Figure 1.10: Machine learning model's number of parameters (size) according to their publication date and their task. Source: Machine Learning Model Sizes and the Parameter Gap [167].

However, training neural networks is expensive and requires a lot of resources. Training deep neural networks is computationally intensive and requires significant computational resources, including powerful GPUs or specialized hardware. Large models with millions of parameters can be challenging to train and deploy in resource-constrained environments, requiring a significant amount of electrical power and therefore displaying a very poor carbon footprint.

Besides, in this thesis, we empirically show once more, that neural networks are not almighty, and are sensitive to small changes in the distribution of the input data. We show that neural network performances can easily collapse under very small perturbations of the data, showing real harm in releasing them on real-world life at-risk applications.

Overall, this thesis aims to temper the growth of neural network size and to bring the esteemed research institutions, perhaps blinded by the pursuit of gain, back to reason. We believe the focus should be placed on ensuring the security, thorough understanding, and effective control of a bit smaller neural networks but still exhibiting excellent performances.

1.2 Adversarial attacks and examples

In some applications, neural networks are reaching human-level performances when evaluated on an i.i.d. (independent and identically distributed) test set. However, several authors [16, 154] highlighted the vulnerability of neural networks to adversarial attacks. **Adversarial attacks are functions of the deep classifier and original example producing adversarial example, i.e. a slightly perturbed original example misleading the classification.** With images, an adversarial perturbation refers to an "imperceptible" perturbation of an image, i.e. a slight change in the pixels, so that this image remains unchanged for the human eye while causing misclassification as shown in Figure 1.12. These adversarial examples question the trust we put in deep learning models in our everyday tasks, especially in life-at-risk applications such as autonomous cars.

In this thesis, we mainly focus on images, even though adversarial examples could be of any modality (for instance trajectory attack shown in [22]).

Adversarial examples are interesting to anyone who wants to evaluate the robustness of a classifiers and certify their use [166], or to some ill-intentioned hacker who wants to jeopardize the system.

The goal of a hacker is to corrupt the system by any means. In [22], the authors studied the adversarial harm of data-driven trajectory prediction systems within autonomous cars. They devise an adversarial attack framework generating realistic adversarial trajectories. Authors showed their adversarial attack can lead an autonomous car to drive off-road or collide with other vehicles in simulation.

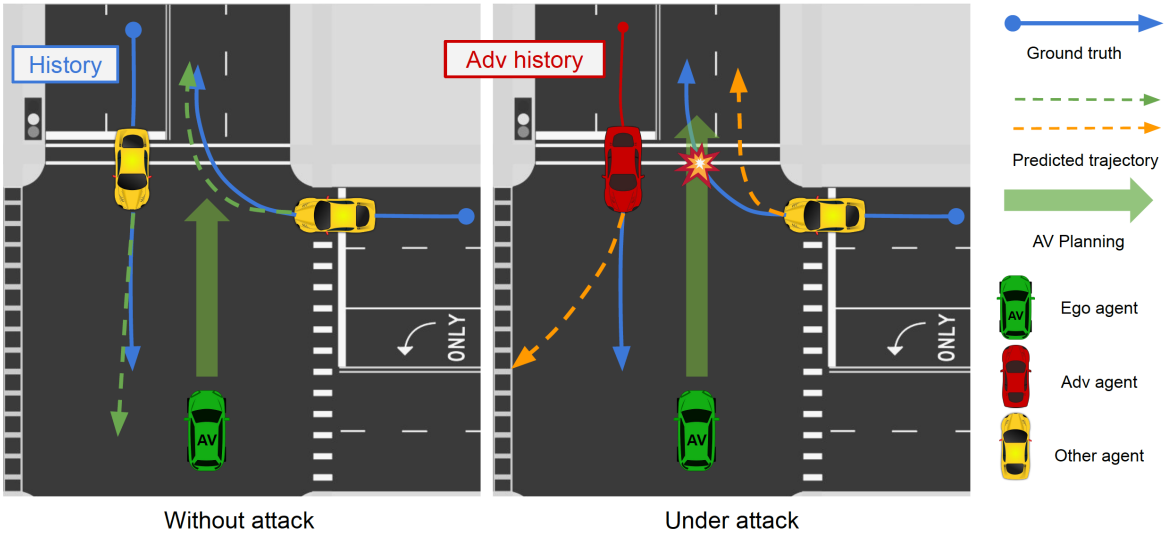


Figure 1.11: Example of adversarial trajectory proposed by the adversarial attack of [22]. By driving along the crafted adversarial history trajectory, the adversarial agent (red car) misleads the prediction of the autonomous car (AV) for both itself and the other agent (yellow car). Consequently, autonomous car planning based on the wrong prediction results in a collision. Source: AdvDO: Realistic Adversarial Attacks for Trajectory Prediction [22].

Indeed, in Figure 1.11, by shifting the left yellow car into an adversarial agent that only goes in the opposite direction, and does not cross the trajectory of the green autonomous car, they make the trajectory prediction fail to predict a safe path leading to a crash with the right yellow car. Opposite to what we study in this thesis, the hacker did not have to perturb the system itself but only its environment, highlighting the fragility of current models used in life-at-risk applications, and our need to assess and improve their robustness.

Adversarial examples are interesting to anyone attempting to evaluate the robustness of a model before releasing it into life-at-risk applications Overall researching the origin and the computation of adversarial examples is of the utmost importance and crucial to the integration of the model into life-at-risk applications.

Evaluating the robustness of a classifier with adversarial examples is tricky because, among all the adversarial attacks, different families co-exist with very different goals regarding the adversarial examples. There are pseudo-random adversarial attacks, common corruptions, adversarial patch attacks, and other types that are out of the scope of this thesis. Among these families, we mainly explored the pseudo-random adversarial attacks, attacks that generate very small pseudo-random adversarial perturbations, uncorrelated to the semantics of the data.

In this section, we first define the adversarial examples optimization problems, and the different visions of it leading to different goals with adversarial examples. In these definitions, we highlight the differences between pseudo-random adversarial examples, common corruptions, and adversarial patches. After defining adversarial examples we formalize their different desired

properties (universality and transferability). Next, we define adversarial metrics that measure the quality of an adversarial example generation. Then, as we deeply explored it, we reviewed some of the most important pseudo-random adversarial attacks (white box, black box, universal attacks, and other types of pseudo-random attacks). Finally, we present the different families of adversarial defenses, mechanisms aiming to protect classifiers from adversarial harm.

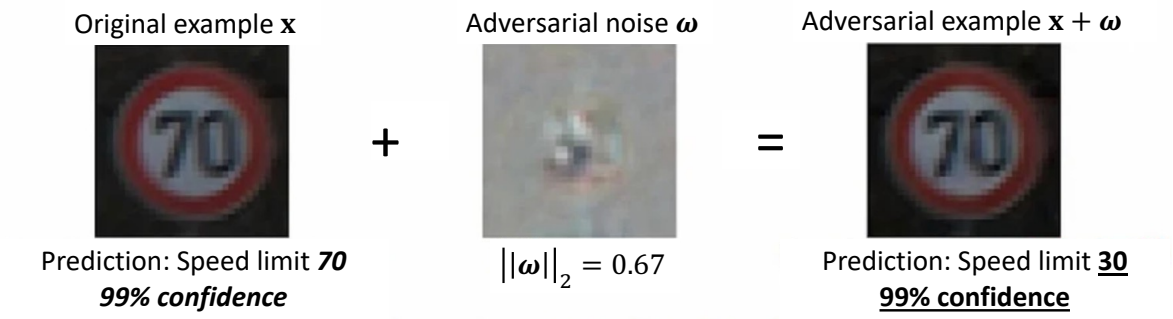


Figure 1.12: Adversarial example produced with an additive pseudo-random adversarial attack. Source: Matthias Hein presentation Adversarial Robustness: Evaluation and Approaches beyond Adversarial Training in "A Blessing in Disguise: The Prospects and Perils of Adversarial Machine Learning" workshop at ICML 2021.

1.2.1 Definitions

Exact adversarial attacks

Adversarial attacks are algorithms generating adversarial examples. They are functions of an original example $\mathbf{x} \in \mathcal{X}$ of which we want to find its adversarial example $\mathbf{x}'$ and a classifier f that misclassifies $\mathbf{x}'$.

Originally, the goal of an adversarial example is to find *the closest point* $\mathbf{x}' \in \mathcal{X}$ to an original example $\mathbf{x} \in \mathcal{X}$ that *maximizes the misclassification gap* (lowering as much as possible the right classification) of a given neural network classifier f. Given an observed (also called clean) example $\mathbf{x}$ labeled as $k' = \text{argmax}_k f_k(\mathbf{x})$ by f, its adversarial example for f reads,

$$\min_{\omega_{\mathbf{x}} \in \Omega} \text{dist}(\mathbf{x}, \mathbf{x} + \omega_{\mathbf{x}}), \tag{1.15a}$$

$$\max_{\omega_{\mathbf{x}} \in \Omega} (\max_{k \neq k'} f_k(\mathbf{x} + \omega_{\mathbf{x}})) - f_{k'}(\mathbf{x} + \omega_{\mathbf{x}}), \tag{1.15b}$$

with dist a similarity metric between data points (most commonly used distances are ℓ_p norms, but we discuss this choice in Chapter 3). In this definition we assumed that the adversarial example $\mathbf{x}' \in \mathcal{X}$ takes the form $\mathbf{x}' = \mathbf{x} + \omega_{\mathbf{x}} \in \mathcal{X}$ as shown in Figure 1.12, which makes the adversarial example generation *additive*. Rather than being additive, other types of adversarial attacks exist, with a different form of the adversarial example such as the Adversarial patch, detailed in Section 1.2.1. However the additive type of adversarial attacks remains the most studied and is the one we focus on in this thesis therefore, unless explicitly rectified, assume $\mathbf{x}' = \mathbf{x} + \omega_{\mathbf{x}}$. Originally adversarial attacks were proposed for binary SVM classifiers by [16], but the latter were extended to more class classifiers.

See that (1.15) defines a bi-criteria optimization in which both criteria are opposing. This makes the problem very difficult to solve thus relaxations have been proposed.

Note that we formalized the adversarial noise space with Ω. This formalism has rarely been done and in this thesis, we explore different constraints we can impose on this space such that interesting adversarial noises can be computed. In an unconstrained setting, pseudo-random adversarial attacks consider $\Omega = \mathbb{R}^P$, while common corruptions [72] define $\Omega = \{\omega = \alpha D \text{ with } \alpha \in$

$\mathbb{R}^{15}$, D = [Gaussian Noise, Shot Noise, ..., JPEG]$\}$, we review them more precisely in Section 1.2.4, and Section 1.2.1 respectively. In equation (1.15), the goal of (1.15b) is to maximize the difference between any other label prediction and the original label prediction. Such an attack is coined untargeted adversarial attack, but its targeted version is just a slight modification and reads,

$$\min_{\omega_{\mathbf{x}} \in \Omega} \text{dist}(\mathbf{x}, \mathbf{x} + \omega_{\mathbf{x}}), \tag{1.16a}$$

$$\max_{\omega_{\mathbf{x}} \in \Omega} f_t(\mathbf{x} + \omega_{\mathbf{x}}) - f_{k'}(\mathbf{x} + \omega_{\mathbf{x}}), \tag{1.16b}$$

with t the targeted label, we want the adversarial example to be classified as.

In practice, the misclassification goal (1.15b) is enforced through the minimization of an adversarial loss function l making the optimization of the adversarial examples easier, more manageable, and more efficient. In a way, equation (1.15b) already is an adversarial loss function. Carlini and Wagner[24] studied multiple adversarial loss functions that we review and detail through the presentation of the adversarial attacks in Section 1.2.4.

In this thesis, unless highlighted otherwise, we mainly explore untargeted adversarial attacks, as some of the most used adversarial attacks simply set the targeted label t as the second best label of the original example $t = \text{argmax}_{k \neq k'} f_k(\mathbf{x}), k' = \text{argmax}_k f_k(\mathbf{x})$. Because solving equation (1.15) is too complicated for state-of-the-art classifiers and high dimensional data, researchers instead proposed to solve relaxation versions.

Adversarial attacks focusing on the quality of adversarial example

The first relaxation of equation (1.15) was proposed by [172], which target, for a classifier f and a given original example $\mathbf{x}$, its best adversarial pair,

$$\min_{\omega_{\mathbf{x}} \in \Omega} \text{dist}(\mathbf{x}, \mathbf{x}') \quad \text{such that} \quad \underset{k}{\text{argmax}}\, f_k(\mathbf{x}') \neq \underset{k}{\text{argmax}}\, f_k(\mathbf{x}), \quad \text{and} \quad \mathbf{x}' \in \mathcal{X}. \tag{1.17}$$

In this definition, the second objective (1.15b) has been replaced with a constraint of misclassification. See that this relaxation eases the original formulation by considering adversarial any example classified with another label than the one of the original example. The gap between the original and adversarial predictions is accepted to be neglected as the purpose of the adversarial examples is to overcome the classifier's decision boundary, by whatever margin. However, in this relaxation, the first objective (1.15a) of the original problem remains thus, these attacks mainly focus on finding the closest adversarial point, that can be seen as **the quality of the crafted adversarial examples (the smallest noise possible, overcoming the classifier's boundaries)**.

When finding the exact smallest adversarial noise of $\mathbf{x}$ overcoming the classifier's boundaries, these adversarial attacks find the exact adversarial example of $\mathbf{x}$. Adversarial attacks producing exact adversarial examples are not that much explored as neural networks are very complex and highly not convex functions, thus making the problem very difficult to solve and intractable for high dimensional data. However, some exact adversarial attacks exist for certain types of neural networks, such as the reluplex [82] relying on satisfiability modulo theories (SMT) solvers [12, 76, 124] to reach exact adversarial examples. The main flaw of the reluplex is that the attack can only be performed on neural networks using only the non-linear relu activation function between the hidden layers. Such restriction of the reluplex attack makes it impractical for state-of-the-art classifiers, preventing it from becoming a standard adversarial attack.

The other envisioned relaxation of original problem 1.15 considers the performance of adversarial attacks to be over multiple original examples while neglecting the precision on each single computed adversarial perturbation. The attacker's objective over multiple original examples is to fool the classifiers on a maximum number of data points, which is measured by the fooling rate (1.24).

Adversarial attacks focusing on the fooling rate

The other proposed relaxation of the original problem is to relax the first objective (1.15a), by allowing all adversarial perturbations to live in a certain ℓ_p norm ball of ϵ radius,

$$\max_{\omega_{\mathbf{x}} \in \Omega} \left(\max_{k \neq k'} f_k(\mathbf{x}') - f_{k'}(\mathbf{x}') \right) \quad \text{such that} \quad \begin{array}{l} \text{dist}(\mathbf{x}, \mathbf{x}') \leq \epsilon, \\ \text{and} \quad \mathbf{x}' \in \mathcal{X}. \end{array} \tag{1.18}$$

This relaxation emphasizes maximizing the probability of finding any adversarial example $\mathbf{x}'$ at maximum ϵ distance from its original example $\mathbf{x}$. In some sense, the first constraint of (1.18) can be seen as setting $\Omega = \{\omega_{\mathbf{x}} \in \mathbb{R}^P, \text{dist}(\mathbf{x}, \mathbf{x}') \leq \epsilon\}$. For ease of reading, because lots of adversarial attacks rely on this relaxation with an ℓ_p norm as dist similarity metric, we set $\Omega_\epsilon = \{\omega \in \mathbb{R}^P, ||\omega_{\mathbf{x}}||_p \leq \epsilon\}$ that is equivalent to the first constraint of (1.18).

This relaxation of the original objective considers the efficiency of the adversarial attack as **the maximum number of points for which the adversarial attack found an adversarial example at ϵ distance from them**.

See that two relaxations (1.17) and (1.18) have a different quality metric, *whereas one evaluates its performance on a single point, the other needs on all of the points to estimate its performance.*

The main advantage of the second relaxation compared to the first one, is that all adversarial attacks are given the same ϵ adversarial budget, and can therefore be fairly compared and benchmarked for comparison. Even though both relaxation problems are very appealing most of the state-of-the-art adversarial attacks have their specific optimization strategy and stop the optimization of ω when all of the following conditions are valid,

$$\text{argmax}_k f_k(\mathbf{x}') \neq \text{argmax}_k f_k(\mathbf{x}), \tag{1.19a}$$

$$\text{dist}(\mathbf{x}, \mathbf{x}') \leq \epsilon, \tag{1.19b}$$

$$\mathbf{x}' \in \mathcal{X}, \tag{1.19c}$$

$$\omega \in \Omega. \tag{1.19d}$$

All these conditions are necessary regarding the purpose of the adversarial example. The first condition (1.19a) enforces to overcome the classifier's decision boundaries, as it is its original main purpose. The second condition (1.19b) stipulates that the adversarial examples should be close to the original example, at an ϵ distance imperceptible to the human eye. Setting an authorized ϵ adversarial budget allows us to fairly compare and benchmark all the different adversarial attacks. Condition (1.19c) is necessary regarding the feasibility of the problem. Indeed, if the device capturing the images displays only positive pixels then negative values in the adversarial examples are prohibited as they are not to be seen in a real-world scenario and therefore, violate the original assumptions. Finally, as a formalism, we define with condition (1.19d) the space Ω in which adversarial noises ω live so that the different attacks' optimization are all made clear and distinct.

Figure 1.13 schematizes an adversarial example along with its optimization direction for 2-dimensional point clouds. The exact adversarial example would lie on the decision boundary in the direction of the arrow from the point $\mathbf{x}$ while an adversarial example $\mathbf{x}'$ computed with the second relaxation is displayed and could be anywhere behind the decision boundary and inside the ϵ ball.

So far, we have seen different optimization problems targeting pseudo-random adversarial perturbations. Pseudo-random adversarial perturbations have the advantage of exploring in depth the original input space $\mathcal{X}$ such that a higher number of adversarial examples can be found. It allows a more precise evaluation of the robustness of the classifier. Besides, from an ill-intentioned person, crafting pseudo-random adversarial perturbations is the easiest and therefore, is the most obvious choice for corrupting the system. Pseudo-random adversarial attacks highlight the attacking vision of adversarial examples.

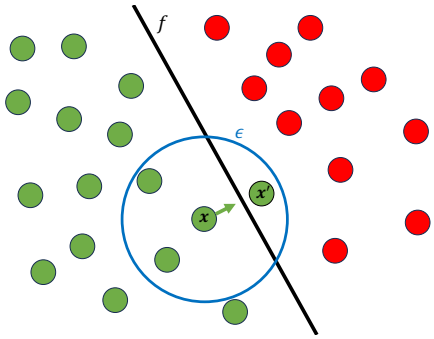


Figure 1.13: Schema of an adversarial example $\mathbf{x}'$ in the ball of radius ϵ centered on the original example $\mathbf{x}$, with the green arrow pointing in the optimization direction of the adversarial example, perpendicular to the classifier f decision boundary.

However, these pseudo-random perturbations are especially criticized because they do not make sense from a real-world viewpoint. Indeed, in real-world scenarios, it is unlikely to find a pink pixel in a garden picture. What's the point of evaluating the robustness of classifiers on data points that will never be seen in real-world applications?

Another kind of adversarial example generation is adversarial patch attacks. These attacks are not of the additive type. They are looking for the solution of a simple human corruption, the stick of a harmful patch on real objects making the classifiers fail at recognizing them. We next detail the patched adversarial examples.

Adversarial patch attacks

To answer the question of a more realistic adversarial example, researchers proposed to compute, instead of pseudo-random adversarial examples, patched adversarial examples (also known as Adversarial Patch) [20].

Adversarial patch attacks are algorithms generating, for a target classifier f, a small patch that, added at a particular position of each image, produces an adversarial example fooling f. Figure 1.14 illustrates such an adversarial patch added to a car image. Once the patch is found and optimized, anyone can print the patch and stick it to any object to make the classifiers fail to recover the right classification of those objects.

Figure 1.14: Example of optimized adversarial patch aiming to fool the detection of light stops. Source: Meta Adversarial Training against Universal Patches[108].

Adversarial patch attacks are not of the *additive type* adversarial attacks but more complicated optimization algorithms. Instead of considering every pixel of the image, the adversarial patch attack creates adversarial examples by completely replacing a part of the image with a patch, which can be seen as a mask of the image. The patch is allowed to take any shape and is optimized using gradient descent over a variety of images, subject to random translation, scaling, and rotation for each image. As quoted in [20], "For a given image $\mathbf{x} \in \mathbb{R}^P$, patch $\mathbf{p}$, patch location l,

and patch transformations t (e.g. rotations or scaling) the attack defines a patch application operator $A(\mathbf{p}, \mathbf{x}, l, t)$ which first applies the transformations t to the patch $\mathbf{p}$, and then applies the transformed patch to the image $\mathbf{x}$ at location l", illustrated in Figure 1.15.

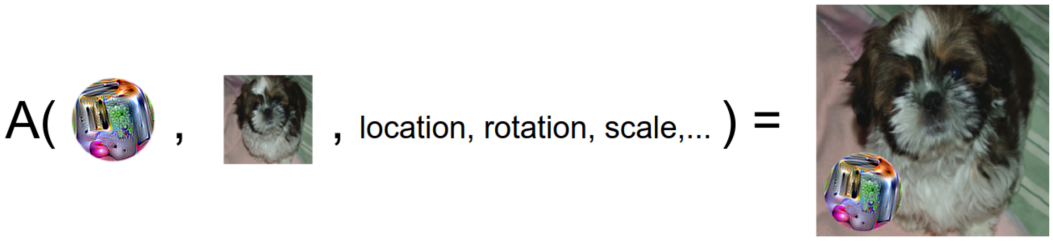


Figure 1.15: An illustration of the patch application operator. Source: Adversarial Patch[20].

The patch $\mathbf{p}$ is optimized using a variant of the Expectation over Transformation (EOT) framework of Athalye, Engstrom, Ilyas and Kwok[7], that is

$$\begin{aligned} \mathbf{p} \quad &= \quad \underset{\mathbf{p}}{\operatorname{argmax}} \mathbb{E}_{\mathbf{x}\in\mathcal{D}, t\in\mathrm{T}, l\in\mathrm{L}}\big[log\, f_{y_t}(\mathrm{A}(\mathbf{p}, \mathbf{x}, l, t))\big], \\ &\text{s.t.} \quad ||\mathbf{x} - \mathbf{x}'|| \leq \epsilon, \end{aligned} \tag{1.20}$$

where y_t is the target label we want the adversarial example to be classified as, $\mathcal{D}$ is a training set of images, T is a set of path transformations, and L is a set of possible locations in the image.

Notice this definition differs from what we saw before as the adversarial attack is not additive. Indeed, the adversarial patch attack considers $\mathbf{x}' = \mathrm{A}(\mathbf{p}, \mathbf{x}, l, t) \neq \mathbf{x} + \omega$. This framework allows much more flexibility in the crafting of the adversarial perturbation but doesn't permit it to be fairly benchmarked to other adversarial attacks as its goal is not the same. However, note that this expectation of equation 1.20 is over the dataset of images, which encourages the trained patch to work regardless of what is in the background. It makes the adversarial patch somewhat universal to every image. Universality is a property we define in Section 1.2.2 and we present universal adversarial attacks in Section 1.2.4. Figure 1.16 shows other types of patched adversarial examples. In Figure 1.16a, an adversary altered a speed limit sign by adding a little patch, to force an autonomous vehicle to fail at recovering the right speed limit, while in Figure 1.16b an adversary added small black and white patches to a stop sign, making autonomous vehicle failing at recognizing it as a stop sign. As we can see these adversarial examples make a lot more sense from our human eye judgment than pseudo-random adversarial examples, highlighted in Figure 1.12.

While [20] is the first work introducing Adversarial patches, newer and fancier patch attacks have been proposed, such as the carpet-bombing patch attack, which requires less knowledge from the targeted model or training data. For newer adversarial patch attacks, readers are invited to refer to [144] for a comprehensive survey on vision-based adversarial patch attacks and defenses.

Opposite to pseudo-random adversarial perturbations and adversarial patches, are common corruptions. The common corruptions are not an attack but a set of real-world perturbations of the data likely to be expected in real-world scenarios. However, as they define a limited number, fixed in stone, of perturbations, shared to all examples and all classifiers, they are mainly limited to modeling the adversarial harm. Therefore, common corruptions are closer to real-world application harm but are too limited to precisely evaluate the robustness of a classifier.

Common corruptions

Alongside adversarial patches, common corruptions are another class of adversarial perturbations that aims to be more realistic than pseudo-random adversarial perturbations. Common corruptions presented in [72] are real-world modifications of the input data, shown in Figure 1.17, that are set in stone and used to measure the robustness of classifiers against real-world perturbations. In their work [72] benchmarked the robustness of classifiers on the dataset ImageNet-C through a

(a) Alteration of a speed limit sign, making autonomous vehicles recognize an 85 mph speed limit, in place of a 35 mph. Source: Macafee.com

(b) Alteration of a stop sign, making autonomous vehicles fail at recognizing it. Source: Robust Physical-World Attacks on Deep Learning Visual Classification [54].

Figure 1.16: Examples of real-world adversarial examples produced by someone altering sign boards.

brand new instance of the dataset, ImageNet-P composed of the original images perturbed by a set of a priori chosen perturbations such as adding Gaussian Noise, adding Zoom Blur or modification of the brightness of original images, detailed in Figure 1.17.

To some extent, common corruptions can be seen as adversarial noise $\omega \in \Omega$, living in a constrained space Ω. Rather than setting $\Omega = \mathbb{R}^P$ and allowing adversarial noises to be pseudo-random, common corruptions constraint the adversarial noise space to be defined as $\Omega = \{\omega = \alpha D \text{ with } \alpha \in \mathbb{R}^{15}, D = [\text{ Gaussian Noise, Shot Noise, ..., JPEG }]\}$, resulting in more realistic adversarial noises.

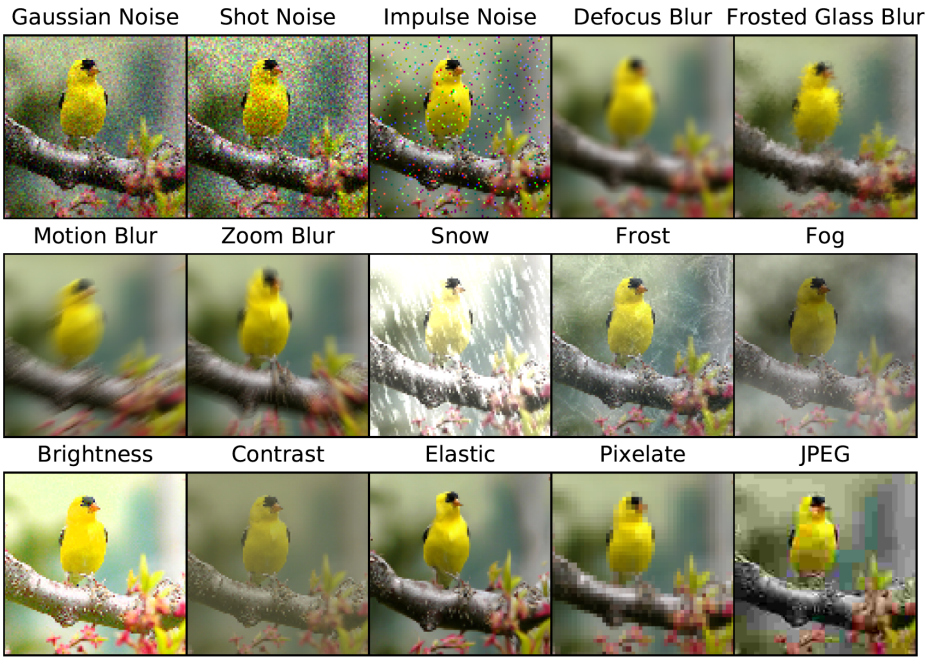


Figure 1.17: Originally proposed common corruptions. Source: Benchmarking neural network robustness to common corruptions and perturbations [72].

As quoted in [109], "Autonomous cars need to be able to cope with wildly varying outdoor conditions such as fog, frost, snow, and night setting, just to name a few (as visualized in Figure 1.18). One of the major weaknesses of autonomous cars is the inability of their recognition models to

function well in adverse weather conditions [40]. Getting data for unusual weather conditions is hard and while many common environmental conditions can (and have been) modeled, including fog [134], rain [169], snow [169] and daytime to nighttime transitions [40], it is impossible to foresee all potential conditions that might occur "in the wild"". Some authors previously proposed a dataset of road scenes under fog captured by polarimetric cameras to model real-world conditions of autonomous cars [17]. But later [109] extended this idea, by proposing several datasets containing common corruptions specifically designed for object recognition tasks (autonomous cars applications included). By setting these object detection common corruptions, Michaelis et al. of [109] allow benchmarking any type of object detection system within autonomous cars, which can be composed solely of one deep learning model or a deep pipeline of small unit model in the shallow learning spirit. Along with the common corruptions, [109] allows the building of more robust classifiers by optimizing them over these common corruptions, which is in the way of improving the classifiers' robustness by modeling more concrete real-world adversarial perturbations.

Figure 1.18: Examples of varying outdoor images conditions acting as adversarial perturbations in the case of autonomous cars object recognition. Source: Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming [109].

In the same spirit [47] proposed to establish a set of 27 common corruptions specifically suited to 3D object detection for both LiDAR (Light Detection and Ranging) and camera inputs considering real-world driving scenarios. These common corruptions are illustrated in Figure 1.19.

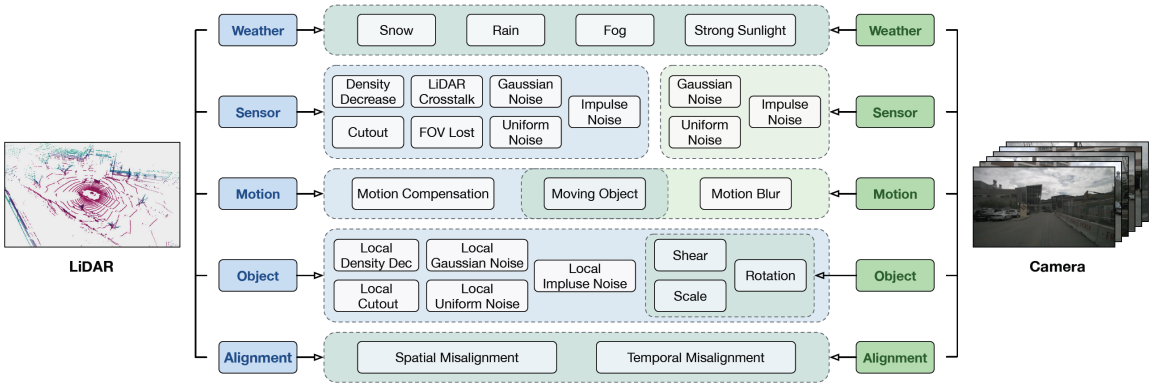


Figure 1.19: Overview of 27 corruptions for 3D object detection proposed by [47]. See that the common corruptions are categorized into weather, sensor, motion, object, and alignment and that some corruptions are effective for one modality, while others are applied to both. Source: Benchmarking Robustness of 3D Object Detection to Common Corruptions in Autonomous Driving [47].

By synthesizing these corruptions on public datasets that are KITTI [61], nuScenes [21], Waymo [150], authors establish three corruption robustness benchmarks namely KITTI-C, nuScenes-C, and Waymo-C. Authors of [47] also conducted extensive experiments of state-of-the-art 3D object recognition. Authors found that camera-only models are more easily affected by common corruptions, demonstrating the indispensability of LiDAR point clouds for reliable 3D detection or the necessity of developing more robust camera-only models. Overall, the different common corruptions propositions share the same goal, evaluating the robustness of deep models (classifiers and object detectors) on a more concrete basis.

These perturbations are interesting for two reasons. First, the classifiers' robustness is evaluated on a more concrete basis as the common corruptions are perturbations that can happen in a real-world application therefore, they are much more interesting from an application standpoint and can be understood by the non-experts.

Secondly, these perturbations are set in stone and are the same as every original example, making them example-agnostic like the pseudo-random universal adversarial perturbations (reviewed in Section 1.2.4). This universality property is desired as it exploits the flaws and weaknesses of classifiers that are always true for every example including the future unseen ones. Exploiting these flaws is of the utmost importance as they will likely remain true when releasing the classifier in production on real-world data examples. Universality is detailed in Section 1.2.2. Common corruptions are a first attempt to compute more realistic adversarial noises ω, by constraining the space Ω they live in.

Some new work has been proposed trying to learn real-world perturbations [65]. In AdvMix[65], authors propose to consider as real-world perturbations, the disentangled latent representations computed by a StyleGAN [81] generator (the GAN framework is detailed in Section 1.1.4). Even though the presented results seem appealing, the problem with these "optimized real-world" perturbations is that they are specific to a s generator, a target classifier, and the dataset used to optimize these perturbations. Compared to the common corruptions, which are real-world perturbations, this proposition is still a step away from computing the real-world robustness of a classifier. Besides these optimized perturbations are not controlled in magnitude which makes impossible any comparison with other adversarial attacks. Indeed, authors do not compare their performances with other adversarial attacks but instead highlight the use of their optimized perturbation within an adversarial defense mechanism (more precisely defined in Section 1.2.5).

As we saw, the definition of the searching space Ω of the adversarial noises is primordial before looking for adversarial examples. Depending on the given constraints and goal, the definition of Ω can be significantly different. In our contributions, we mainly focused on pseudo-random adversarial examples, computed with the second relaxation problem therefore, we used Ω_{ϵ_p}, under which we added some constraints that will be detailed in Chapter 2.

1.2.2 Adversarial properties

So far, we have seen adversarial attacks as functions of a classifier f and an original example $\mathbf{x}$, but other adversarial example generations exist that either are not restricted to a specific original example $\mathbf{x}$ or to a single classifier f.

Universality

Universal adversarial perturbations are adversarial perturbations ω that respect the conditions of the adversarial example (1.15) applied to multiple original examples,

$$\text{argmax}_k f_k(\mathbf{x}^{(i)} + \omega) \neq \text{argmax}_k f_k(\mathbf{x}^{(i)}) \ \ \forall i \in \mathcal{I}, \tag{1.21a}$$

$$\text{dist}(\mathbf{x}^{(i)}, \mathbf{x}^{(i)'}) \leq \epsilon \ \ \forall i \in \mathcal{I}, \tag{1.21b}$$

$$\mathbf{x}^{(i)'} \in \mathcal{X} \ \ \forall i \in \mathcal{I}, \tag{1.21c}$$

$$\omega \in \Omega, \tag{1.21d}$$

with $\mathcal{I}$ an index set.

The more universal the adversarial perturbation is, the more harmful it is. Having a perfectly universal adversarial perturbation signifies that applied to every example, the classifier f cannot ever recover the right prediction for all possible original examples. To some extent, such universal perturbation is the most harmful perturbation possible. Universality is an important property of adversarial examples, that should be included in the performance evaluation of an adversarial attack generation.

However, crafting an adversarial perturbation that is valid for all the original examples is too complicated therefore, the problem is relaxed by lowering the number of points for which the adversarial perturbation is valid. Universal adversarial perturbation generation algorithms are called universal adversarial attacks, we review some of the most important in Section 1.2.4.

Transferability

A transferable adversarial perturbation is an adversarial perturbation ω that respects all the adversarial examples conditions applied to every classifier,

$$\text{argmax}_k f_k^{(i)}(\mathbf{x}') \neq \text{argmax}_k f_k^{(i)}(\mathbf{x}) \;\; \forall i \in \mathcal{I}, \tag{1.22a}$$

$$\text{dist}(\mathbf{x}, \mathbf{x}') \leq \epsilon, \tag{1.22b}$$

$$\mathbf{x}' \in \mathcal{X}, \tag{1.22c}$$

$$\omega \in \Omega, \tag{1.22d}$$

with $\mathcal{I}$ an index set.

Crafting such transferable adversarial perturbation seems very challenging. In practice, when evaluating the performances of an adversarial attack, we craft all the possible adversarial perturbations on a given classifier f and empirically see if the perturbations remain adversarial for other classifiers g, making the adversarial attack more or less transferable.

The transferability property is also not to be neglected when evaluating the performance of an adversarial attack. If one can craft perfectly transferable adversarial perturbations, all life-at-risk applications relying on classifiers are put at risk, as the perturbation remains adversarial to any classifier. Transferability must also be considered when evaluating the performance of an adversarial attack as it reveals some weaknesses/leaks of the classifier remaining true to every original example and therefore, should be one of the first things to consider when evaluating its robustness.

Both the universality and transferability properties of adversarial examples are important to consider and aim for.

1.2.3 Adversarial metrics

We saw that different optimization problems were proposed to craft adversarial examples, implying different visions and different goals with adversarial examples. Depending on the chosen adversarial problem and who solves it, the most relevant performance metric differs and may evaluate only some aspects of the adversarial perturbations. In this section, we explain in detail where the different adversarial metrics come from, what they are measuring, and to whom they are relevant.

From an attacker's point of view

From an attacker's point of view, the adversarial metric that matters the most is the number of data points for which the classifier is fooled, that is maximizing the probability of finding an adversarial example,

$$\max \;\; \Pr_{(\mathbf{x},y)\in\mathcal{D}}\big(\mathbf{x}' \in \mathrm{B}(\mathbf{x},\epsilon),\; \underset{k}{\text{argmax}}\, f(\mathbf{x}') \neq \underset{k}{\text{argmax}}\, f(\mathbf{x})\big) \tag{1.23}$$

with $\mathrm{B}(\mathbf{x},\epsilon)$, the ℓ_p norm ball of radius ϵ centered on $\mathbf{x}$. Empirically this probability measure is estimated with the fooling rate (FR) defined as

$$\text{FR}(f,\mathcal{D}) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathbb{1}\{\underset{k}{\text{argmax}}\, f_k(\mathbf{x}'^{(i)}) \neq \underset{k}{\text{argmax}}\, f_k(\mathbf{x}^{(i)})\}, \tag{1.24}$$

with $\mathbf{x}'^{(i)} = \mathbf{x}^{(i)} + \omega_{\mathbf{x}}, \omega_{\mathbf{x}} \in \Omega_\epsilon$.

This metric is mainly suited to the optimization of the second relaxation problem which allows adversarial perturbations to live in an ϵ ball $\omega \in \Omega_\epsilon$. The FR metric doesn't evaluate the performance of the adversarial attack over a single point $\mathbf{x}$ but over a whole set of points $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ which typically consist of the test dataset, the classifier has not been trained on.

However, this metric doesn't take into account the performances of f on clean data. *Indeed, if the FR only mattered then a constant classifier would be the most robust to adversarial attacks, which is in practice a bad classifier.*

From a defender's point of view

While the FR metric evaluates the performances of the adversarial attack, the robustness of a classifier is measured with another metric. From a defender's point of view when building a classifier f, the goal is to keep a high classification accuracy even though being perturbed by adversarial examples. It leads to a slight modification of the conditions for $\mathbf{x}'$ to be an adversarial example of $\mathbf{x}$. It must originally be well classified, $\text{argmax}_k\, \hat{y} = y$ (see [179] and [172] for more details)

$$\begin{cases} \underset{k}{\text{argmax}}\, f_k(\mathbf{x} + \omega_\mathbf{x}) \neq \underset{k}{\text{argmax}}\, f_k(\mathbf{x}) = y, \\ \omega_\mathbf{x} \in \Omega_\epsilon, \\ \mathbf{x}' \in \mathcal{X}. \end{cases} \tag{1.25}$$

This point of view leads to consider the original performance of the classifier, that is SA the Standard Accuracy

$$\text{SA}(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{\underset{k}{\text{argmax}}\, f_k(\mathbf{x}^{(i)}) = y^{(i)}\}, \tag{1.26}$$

as the empirical counterpart of the expected loss.

Then, the robustness of the classifier (opposite to the efficiency of the adversarial attack) can be measured as the number of well classified clean examples for which the adversarial attack could not find an adversarial example in a ϵ ball around them. This quantity is estimated by the Robust Accuracy RA as the empirical counterpart of the astuteness as defined in [172],

$$\text{RA}(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{\mathcal{A}_{\{(\mathbf{x}^{(i)}, y^{(i)})\}} = \varnothing\}, \tag{1.27}$$

with $\mathcal{A}_{\{(\mathbf{x}^{(i)}, y^{(i)})\}} = \{(\mathbf{x}'^{(i)}, y^{(i)})\}$ the set of adversarial examples associated with $(\mathbf{x}^{(i)}, y^{(i)})$. It is clear that $\text{RA}(f, \mathcal{D}) \leq \text{SA}(f, \mathcal{D})$, the equality being reached when the adversarial attack is completely inefficient. *Therefore, the lower RA is, the more efficient the adversarial attack is.*

Like the FR metric, the RA metric evaluates the classifier's robustness not on a single point but over a whole set of points $\mathcal{D}$, typically the test set that has not been used for training the classifier.

A fair evaluation of the adversarial examples generation

Yet, the latter metric is questioned by Lorenz et al. [100], who emphasizes the importance of taking into account the ability to easily detect the generated adversarial examples. They suggest using an adversarial example detector g plugged before the classifier f when computing the RA. By doing so, Lorenz et al. [100] showed that state-of-the-art adversarial attacks produced easily detectable adversarial examples, making the attacks inefficient when a detector is used. Therefore, they suggest relying on another metric called the Attack Success Rate under Defense (ASRD) (equation 2 of [100]), we rename Robust Accuracy Under Defense (RAUD) for clarity purposes, which stands as a natural performance metric from both the attacker's and defender's points of view. The RAUD is defined as,

$$\text{RAUD}(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{\forall \mathbf{x}' \in \mathcal{A}_{\{(\mathbf{x}^{(i)}, y^{(i)})\}},\ g(\mathbf{x}') = 0\}, \tag{1.28}$$

where $g : \mathbb{R}^P \mapsto \{0, 1\}$ is an adversarial example detector with 0 as the clean example label.

According to the RAUD, an adversarial example is considered when it fools the classifier f and the detection from d. By doing so, the RAUD evaluates the transferability of the produced adversarial example. The RAUD not only is fair for both the attacker and the defender but also evaluates the transferability of the adversarial attack allowing it to more precisely measure the harm of the adversarial attack.

1.2.4 Pseudo-random adversarial attacks

Other than pseudo-random adversarial perturbations, we previously saw that other families of adversarial perturbations exist such as adversarial patch (reviewed in Section 1.2.1 or common corruptions (reviewed in Section 1.2.1) however, in our contributions we mainly studied the pseudo-random adversarial perturbations therefore, here we define them in depth.

Pseudo-random adversarial attacks aim to generate additive adversarial perturbation $\omega \in \Omega = \mathbb{R}^P$ that targets every pixel of the image without any restriction. Some of the pseudo-random adversarial attacks rely on the second relaxation optimization (see Section 1.2.1) to craft adversarial examples, thus setting $\Omega_\epsilon = \{\omega \in \mathbb{R}^P, ||\omega_{\mathbf{x}}||_p \leq \epsilon\}$. It makes it easier to fool classifiers and thus evaluates their robustness more precisely, but computes adversarial perturbations that are less realistic and far from real-world harm.

Table 1.1 summarizes all the attacks we review and highlights their differences.

Table 1.1: Summary of the diverse presented adversarial attacks: The 'ℓ_p-norm' refers to the norm used in $\Omega_\epsilon = \{\omega \in \mathbb{R}^P, ||\omega_{\mathbf{x}}||_p \leq \epsilon\}$. Sources of this table include [145], [9], [182] and [2].

Attacks	Knowledge	Goal	ℓ_p-norm	Optimization
L-BFGS [154]	White-box	Specific	ℓ_2	One-shot
FGSM [63]	White-box	Specific	ℓ_∞	One-shot
BIM [89]	White-box	Specific	ℓ_∞	Iterative
PGD [102]	White-box	Specific	ℓ_2, ℓ_∞	Iterative
DeepFool [113]	White-box	Specific	ℓ_2	Iterative
C & W [24]	White-box	Specific	$\ell_0, \ell_2, \ell_\infty$	Iterative
Autoattack [37]	White-box	Specific	ℓ_2, ℓ_∞	Iterative
Local Substitute Model Attack [116]	Black-box	Specific	ℓ_2, ℓ_∞	One-shot
MIM [48]	Black-box	Specific	ℓ_∞	Iterative
Zoo [29]	Black-box	Specific	ℓ_p	Iterative
Boundary Attack [36]	Black-box	Specific	$\ell_1, \ell_2, \ell_\infty$	Iterative
UAP [111]	White-box	Universal	ℓ_2, ℓ_∞	Iterative
Fast-UAP [41]	White-box	Universal	ℓ_2, ℓ_∞	Iterative
UAP-PGD [142]	White-box	Universal	ℓ_2, ℓ_∞	Iterative
CW-UAP [15]	White-box	Universal	ℓ_2, ℓ_∞	Iterative
NAG Attack [114]	Black-box	Universal	ℓ_∞	One-shot
JSMA [117]	White-box	Specific	ℓ_2	Iterative
Adversarial Patch [20]	White-box	Specific	ℓ_∞	Iterative

The very first work proposing a pseudo-random adversarial attack is [16] which defined the specification of the adversary, its goal, its knowledge, and its capabilities. Every one of these specifics will give birth to a sub-family of pseudo-random adversarial attacks.

An adversary's objective can be a misclassification loss function of one or multiple examples. In (1.15b) we set the misclassification goal as $\max_{\omega_{\mathbf{x}} \in \Omega} (\max_{k \neq k'} f_k(\mathbf{x} + \omega_{\mathbf{x}})) - f_{k'}(\mathbf{x} + \omega_{\mathbf{x}})$ because it is the direct objective of maximizing the FR, but in principle, it could be any loss function enforcing the misclassification of the adversarial example.

The adversary knowledge can include the training set, or part of it, used to build the classifier,

the feature representation of each example, the type of the classifier's learning algorithm and the form of the classifier's decision function, the classifier itself, or the classifier's feedback.

The adversary's capabilities are limited to modifications only of test data. Adversaries are not allowed to modify the training data as it would be another problem to let adversaries penetrate the training of the classifier. Hence, the adversary can only operate modifications to the test examples, modifications of the test examples' features, or independent modifications to specific features (the semantics of the examples may dictate that certain features are interdependent).

Differences in the adversary's knowledge separate adversarial attacks into two categories: White-box and Black-box attacks and differences in the adversary's goal separate adversarial attacks into two categories: Specific attacks and Universal attacks.

All these differences are highlighted in Table 1.1.

White-box adversarial attacks

A white box adversarial attack is a type of attack on a classifier in which the adversary has complete knowledge of the classifier's architecture, parameters, and inputs. In other words, the adversary has full access to the "white box" of the classifier, hence the name. White-box attacks are predominately gradient-based adversarial attacks since they usually rely on the gradient of the adversary's goal with respect to the classifier's parameters to optimize it. This is the first type of proposed adversarial attack. The very first adversarial attacks date back to a few years ago with [154] who first discovered the vulnerabilities of classifiers to adversarial perturbations by solving the following optimization problem

$$\min_{\omega_{\mathbf{x}} \in \Omega} \| \omega_{\mathbf{x}} \|_2 \quad \text{s.t.} \quad \text{argmax}_k f_k(\mathbf{x} + \omega_{\mathbf{x}}) = t, \quad (\mathbf{x} + \omega_{\mathbf{x}}) \in \mathcal{X}, \tag{1.29}$$

with t a target label. With this problem being tough to solve, authors propose to approximate a solution using the Limited Memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm [96], giving the attack its name, the L-BFGS attack. To solve it, authors solve instead a relaxed version, including Lagrange multiplier c,

$$\min_{\omega_{\mathbf{x}} \in \Omega} c \| \omega_{\mathbf{x}} \|_2 + \mathcal{L}(\mathbf{x} + \omega_{\mathbf{x}}, t) \quad \text{s.t.} \quad (\mathbf{x} + \omega_{\mathbf{x}}) \in \mathcal{X}, \tag{1.30}$$

with $\mathcal{L}(\cdot, \cdot)$ as the classifier's loss, and c the dual variable found performing a line-search to get the minimum $c > 0$ for which an adversarial example $\mathbf{x}'$ can be found. The authors showed that their attack transfers well between different classifiers.

The transfer of an adversarial attack refers to the application of an adversarial example $\omega_{\mathbf{x}}$ targeted to fool the original classifier f, on another classifier g (see Section 1.2.2 for more details). This intriguing transfer property of adversarial attacks attracted lots of interest from researchers in the subsequent years and this work.

Solving (1.30) for a large number of images can often be expensive and computationally disastrous therefore, to lighten the complexity of the problem the Fast Gradient Sign Method (FGSM) adversarial attack [63] has been proposed to compute adversarial perturbations more efficiently, as a solution of

$$\omega_{\mathbf{x}} = \gamma \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y)). \tag{1.31}$$

The FGSM adversarial attack is a one-step gradient-based method to compute adversarial examples efficiently. The attack focuses on the "efficiency" of the perturbations rather than achieving high fooling rates. The FGSM attack is among the most influential attacks in the existing literature, it inspired many improved/tweaked versions such as FGVM [130], P-FGVM [26], I-FGSM [49], MI-FGSM [177], DI^2-FGSM [49], M-DI^2-FGSM [49], and many others. The FGSM attack gave rise to one of the first adversarial defense mechanisms, a new way to train classifiers more robustly to adversarial examples, called adversarial training. An overview of adversarial defenses is given in Section 1.2.5.

By being a one-shot adversarial attack, the FGSM is limited in the optimization of the adversarial examples. To enhance adversarial example optimization, the Basic Iterative Method (BIM) [89] has been proposed as an iterative version of FGSM, computing adversarial examples by repeating

$$\omega_{\mathbf{x}}^{i+1} = \text{clip}_{\epsilon}\{\omega_{\mathbf{x}}^{i} + \gamma \text{sign}(\mathcal{L}(\mathbf{x}, y))\}, \tag{1.32}$$

where i denotes the i^{th} iteration, clip_{ϵ} performing clipping at ϵ. Adversarial performances were greatly enhanced and very close to BIM, is the Projected Gradient Descent (PGD) attack [102], which is, as of today, widely considered one of the most powerful attacks in the literature. PGD is a standard optimization technique that projects gradients to the ℓ_p norm ball of radius ϵ,

$$\omega_{\mathbf{x}}^{i+1} = \text{Proj}_{p}^{\epsilon}\{\omega_{\mathbf{x}}^{i} + \gamma \text{sign}(\mathcal{L}(\mathbf{x}, y))\}, \tag{1.33}$$

with the ℓ_p norm being either the ℓ_2 or the ℓ_∞ norm. Even though the PGD attack seems very close to the BIM attack, the proposed Projection operator is a great improvement compared to BIM. It allows us to select another similarity metric than the ℓ_p norm, which can make the adversarial perturbation very different. We present in Section 1.2.6, that there currently is no consensus on which ℓ_p norm should be used and whether even using an ℓ_p norm is a good choice.

Instead of restricting the adversarial perturbations to be in an ϵ norm ball, the DeepFool attack [113] aims at minimizing the norm of the adversarial perturbation by resorting to the first presented adversarial example relaxation problem,

$$\min_{\omega_{\mathbf{x}} \in \Omega} \| \omega \|_2 \quad \text{s.t.} \quad \text{argmax}_k \, f_k(\mathbf{x} + \omega_{\mathbf{x}}) \neq y. \tag{1.34}$$

The main motivation behind resorting to this relaxation was to effectively quantify the adversarial robustness of the target classifier f. The proposed adversarial attack is an iterative algorithm linearizing the class boundaries around the current image to form a convex polyhedron and push the image toward the closest hyperplane to change the class label. Like the FGSM authors, Deepfool authors also proposed a defense mechanism built upon their proposed attack.

Considering the weaknesses of deep classifiers against adversarial attacks [154], researchers proposed defense mechanisms to protect against such harm. One of the first defense mechanisms proposed is Defensive distillation [118] (formalized in Section 1.2.5), however Carlini & Wagner (C&W) [24] proposed a set of adversarial attacks completely breaking the defensive distillation defense. To compute the adversarial perturbations, the (C&W) attack solves the following optimisation problem,

$$\min_{\omega_{\mathbf{x}} \in \Omega} \| \omega_{\mathbf{x}} \|_p + c.l(\mathbf{x} + \omega_{\mathbf{x}}) \quad \text{s.t.} \quad \mathbf{x} + \omega \in \mathcal{X}, \tag{1.35}$$

with c an hyper-parameter and l a function satisfying the misclassification $\text{argmax}_k \, f_k(\mathbf{x}') \neq y$.

Multiple l functions were proposed and discussed by the authors. All of the proposed adversarial attacks rely on the second presented relaxation problem therefore, all adversarial perturbations live in the ℓ_p norm ball of radius ϵ. Even though minimizing the norm of the adversarial perturbation is part of the objective, all of their presented algorithms stop the optimization when all the adversarial conditions are met (1.19). It makes the attacks very powerful and allows us to compare all the adversarial attacks as they are given the same ϵ adversarial budget.

As the work wasn't innovative enough, authors propose multiple ℓ_p norms as magnitude metrics of the adversarial perturbations (especially ℓ_2 norm associated with $\epsilon = 0.5$ and ℓ_∞ norm associated with $\epsilon = 8/255$), which impose future adversarial attacks to allow using different similarity metrics to find adversarial examples.

Proposing multiple ℓ_p norm constraints is an improvement in the computer vision community where there was no consensus on which norm should be used to compute distances, highlighted in Table 1.1.

Even though its computation cost is very high, the C&W attack is still considered one of the strongest adversarial attacks.

Finally, as of today, state-of-the-art performances are achieved by [37] who proposed Autoattack an ensemble of diverse parameter-free attacks. An Autoattack instance encapsulates two versions of improved PGD adversarial attack plus two other attacks previously proposed by the same authors (FAB [36] and SquareAttack [4]). An adversarial example computed with Autoattack is an adversarial example computed from one of these four adversarial attack roots, which naturally dope the performances of the overall Autoattack instance.

White-box adversarial attacks make the assumption the adversary has complete knowledge about the target classifier, including its architecture, parameters, and training data which makes the evaluation of the classifier's robustness very precise. While white-box attacks can be very effective in fooling the target classifier, they are not without their flaws. First, the assumption that complete knowledge of the adversary is often not realistic to real-world scenarios. In practice, adversaries usually have limited access to the target classifier and sometimes may not be able to query it, making white box attacks irrealistic. White box attacks produce adversarial examples targeting one classifier f but most likely do not transfer well to other classifiers. It is again a step away from the real-world scenarios. Finally, as white box attacks are designed to exploit specific weaknesses of a target classifier, they are susceptible to countermeasures and defensive mechanisms. It has been shown in [100] that current state-of-the-art white box attacks are made harmless under tiny defense mechanisms.

Black-box adversarial attacks

In contrast to white-box attacks, black-box attacks do not access the classifier's internal parameters. The attack algorithm relies on alternative information like query access to the classifier [29], knowing the training dataset [116], or transferring adversarial examples from one trained classifier to another [184]. Notice that a black-box attacker represents a more practical adversary [28] and one that is closer to real-world scenarios [116].

The very first adversarial black-box attacks were proposed in [116], [29] and [19]. Each attack computes adversarial examples from a different source of information therefore, these three works gave rise to three main families of black-box attacks: transfer-based, score-based, and decision-based black-box attacks.

Transfer based attacks

In transfer-based attacks, the adversary is allowed to query the target classifier and/or access some of the target classifier's training dataset. The adversary then uses this information to create a synthetic classifier from which the attacker generates adversarial examples using a white-box attack. This white-box adversarial example from the synthetic classifier is then transferred to the target classifier, hoping it remains adversarial (respecting conditions of (1.19)).

One of the first black-box adversarial attacks to be proposed is called the local substitute model attack [116]. The idea of the local substitute model attack is to create a copy-cat classifier from which to generate an adversarial example in the hope that it will remain adversarial to the target classifier. The transfer property of adversarial examples arose from this work. In this paper, authors allowed the attacker to access a percentage of the training dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{k}, k \ll \mathrm{N}$, and to query the target classifier f. They proposed to create a synthetic classifier g optimized to mimic the prediction of the target classifier f on the given fragment of the original dataset. Therefore the synthetic classifier g is not optimized to predict the original labels $\{y^{(i)})\}_{i=1}^{k}$ but to predict the target classifier's prediction $\{\underset{c}{\operatorname{argmax}}\, f_c(\mathbf{x})^{(i)})\}_{i=1}^{k}$. The copy-cat classifier g is optimized using,

$$\min_{g} \sum_{i=1}^{k} \mathcal{L}(f(\mathbf{x}^{(i)}), g(\mathbf{x}^{(i)})), \tag{1.36}$$

with $\mathcal{L}$ a classification loss (e.g. cross-entropy as seen in Section 1.1.2). Once the synthetic classifier g is optimized, authors proposed to compute adversarial examples targeting g using the FGSM attack and hope those adversarial examples transfer to the target classifier f. Authors found that by using only 0.3% of $\mathcal{D}$, an adversary can fool any target classifier. However, this last setting is

questioned by researchers who proposed the mixed [104] or adaptive black-box attack [103]. While in the original attack, 0.3% of the training data is used, the adaptive version increases the knowledge of the adversary by using anywhere from 1% to 100% of the original training dataset. Besides, they used newer and fancier white-box attacks than FGSM having better transferability. Authors claimed they found the most effective white-box attack to be the Momentum Iterative Method (MIM) [48] to transfer the adversarial examples to f.

By integrating a momentum term into the iterative process of the adversarial example generation, the MIM attack stabilizes from each iteration update and escapes from poor local maxima during the iterations, resulting in more transferable adversarial examples. To further improve the transfer success rates for black-box attacks, MIM is applied to an ensemble of synthetic classifiers, maximizing the chances that the computed adversarial examples Indeed, transfer to the target classifier f. The MIM iterative optimization for the adversarial examples $\mathbf{x}^{(i)'}$ takes the form of,

$$\begin{aligned} \mathbf{v}_{t+1} &= \mu\mathbf{v_t} + \frac{\nabla_{\mathbf{x}}\mathcal{L}(g(\mathbf{x}),y)}{||\nabla_{\mathbf{x}}\mathcal{L}(g(\mathbf{x}),y)||_1}, \\ \mathbf{x}^{(i)'}_{t+1} &= \mathbf{x}^{(i)'}_{t} + \gamma\text{sign}(\mathbf{v}_{t+1}), \end{aligned} \tag{1.37}$$

with $\mathbf{v}_t$ acting as a velocity vector of the gradient optimization at iteration t, $\mathcal{L}$ a loss function, μ a decay factor, and γ the learning rate.

The two previous black-box attacks require to model of at least one substitute classifier g however, a substitute classifier can be difficult to obtain, particularly if a considerable amount of data labeled by the target classifier is needed.

Score based attacks

Another direction taken by the adversarial black-box community is the score-based attacks. Score-based adversarial attacks allow the attacker to query the defense with input $\mathbf{x}$ and receive the corresponding pre-softmax logits $f_1(\mathbf{x}),\ldots,f_C(\mathbf{x})$ or the post-softmax probability outputs $\hat{y}_1(\mathbf{x}),\ldots,\hat{y}_C(\mathbf{x})$. One of the first score-based adversarial attacks to be proposed is ZOO (Zeroth Order Optimization) [29] which relies on queries to create adversarial examples. Zeroth order methods are derivative-free optimization methods, where only the zeroth order oracle (the objective function value $f(\mathbf{x})$ at any $\mathbf{x}$) is needed during the optimization process. By evaluating the objective function values at two very close points $f(\mathbf{x}+\alpha\mathbf{v})$ and $f(\mathbf{x}-\alpha\mathbf{v})$ with a small α, a proper gradient along the direction of the vector $\mathbf{v}$, can be estimated. For a target classifier f and a target label t, ZOO finds adversarial examples by optimizing the C&W proposed objective, with the minimization of the following adversarial loss function l,

$$l(\mathbf{x}+\omega_{\mathbf{x}}) = \max\Big\{\max_{i\neq t}\log f_i(\mathbf{x}+\omega_{\mathbf{x}}) - \log f_t(\mathbf{x}+\omega_{\mathbf{x}}), -\rho\Big\}, \tag{1.38}$$

with $\rho > 0$ a margin hyper-parameter.

In order to optimize (1.38), ZOO employs a zeroth order optimization and approximate the gradient $\frac{\delta f(\mathbf{x}+\omega_{\mathbf{x}})}{\omega^i_{\mathbf{x}}}$ by using the symmetric difference quotient [91],

$$\frac{\delta f(\mathbf{x}+\omega_{\mathbf{x}})}{\omega^i_{\mathbf{x}}} \approx \frac{f(\mathbf{x}+\omega_{\mathbf{x}}+\alpha\mathbf{e}_i) - f(\mathbf{x}+\omega_{\mathbf{x}}-\alpha\mathbf{e}_i)}{2\alpha}, \tag{1.39}$$

with α a small constant and $\mathbf{e}_i$ a standard basis vector with only the i-th component as 1. Estimating the gradient this way requires $2P$ queries of f which could be inefficient.

Decision based attacks Decision-based adversarial attacks go one step further into the blindness of the adversary. In such an attack algorithm, the adversary is allowed to query the defense with input $\mathbf{x}$ but receives only the defense's final predicted label $\text{argmax}_k\,\hat{y}_k$. In contrast to score-based attacks, the adversary does not receive any probabilistic or logit outputs from the defense, only the predicted label.

The first prominent decision-based attack is the Boundary Attack [19]. The basic intuition behind the Boundary Attack is, for an original example $\mathbf{x}^{(i)}$, to randomly initialize its adversarial example, and then perform a random walk along the classifier's decision boundary between the

adversarial label and the non-adversarial labels region such that the adversarial example stays in the adversarial label region while minimizing the distance from its original counterpart. In other words, Boundary Attack performs rejection sampling with a suitable distribution $\mathcal{P}$ centered on an adversarial example at each step to find progressively smaller adversarial perturbations according to a given adversarial distance dist.

Algorithm 1 Boundary Attack

Require: Original example $\mathbf{x}$; classifier's label prediction F; MAX-STEPS, Adversarial distance $\text{dist}(\cdot,\cdot)$

$k = 0$
$\mathbf{x}'_0 = \mathcal{U}(0,1)$ s.t. $\text{F}(\mathbf{x}'_{k-1}) \neq \text{F}(\mathbf{x})$
for k < MAX-STEPS **do**
  draw random perturbation from distribution centered on $\mathbf{x}'_{k-1}$, $\mu_k \sim \mathcal{P}(\mathbf{x}'_{k-1})$
  if $\text{dist}(\mathbf{x}'_{k-1} + \mu_k) \neq \text{F}(\mathbf{x})$ and $\text{dist}(\mathbf{x}'_{k-1} + \mu_k, \mathbf{x}) < \text{F}(\mathbf{x}'_{k-1}, \mathbf{x})$ **then**
    $\mathbf{x}'_k = \mathbf{x}'_{k-1} + \mu_k$
  else
    $\mathbf{x}'_k = \mathbf{x}'_{k+1}$
  end if
end for
return Adversarial example $\mathbf{x}'_k$ minimizing $\text{dist}(\mathbf{x}', \mathbf{x})$

In their presented work, authors discuss the definition of $\mathcal{P}$. Being the first of its kind, the Boundary Attack is a quite simple optimization and much stronger decision-base black-box attacks have been proposed such as the Geometric decision-based attacks (GeoDA) [126]. GeoDA is a subset of decision-based black-box attacks that can achieve high attack success rates while requiring a smaller number of queries.

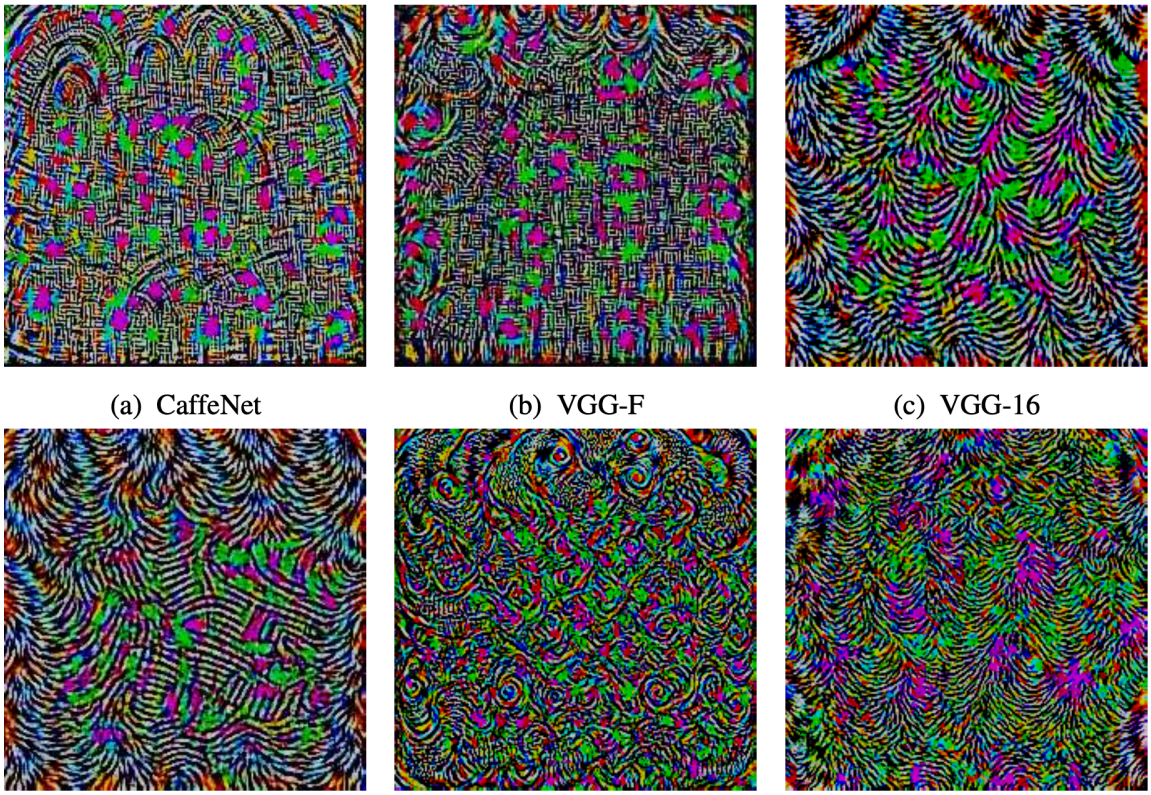
Universal adversarial attacks

So far, every adversarial attack presented consists of finding one adversarial perturbation $\omega_{\mathbf{x}^{(i)}}$ associated with one original example $\mathbf{x}^{(i)}$ producing its paired adversarial example $\mathbf{x}^{(i)'}$. This goal makes these attacks specific as they find specific adversarial perturbations to each original example. Other adversarial attacks have been proposed, aiming to compute one single adversarial perturbation ω capable of producing adversarial examples for a batch of multiple original examples. Such adversarial perturbation is, therefore, universal (as we saw in Section 1.2.2) to the original example, also called example-agnostic. Adversarial attacks producing these universal adversarial perturbations are called universal adversarial attacks.

The existence of universal adversarial perturbations (UAP) to fool deep classifiers for most images has first been demonstrated in [111]. The authors proposed the first universal attack coined UAP. Given a set of labeled images $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{\text{N}}$ with $\mathbf{x}^{(i)} \in \mathcal{X} \subset \mathbb{R}^{\text{P}}$, and a classifier f, the overall UAP objective is to find a single perturbation vector $\omega \in \Omega = \mathbb{R}^{\text{P}}$, such that the classifier f is fooled for most encountered images. More formally, UAP seeks a ω such that,

$$\max_{\omega} \sum_{i=1}^{\text{N}} \mathbb{1}.\Big\{ \underset{k}{\text{argmax}}\, f_k(\mathbf{x}^{(i)}) \neq \underset{k}{\text{argmax}}\, f_k(\mathbf{x}^{(i)} + \omega) \Big\} \quad \text{such that} \quad \mathbf{x}^{(i)} + \omega \in \mathcal{X}, \quad \text{and} \quad ||\omega||_p \leq \epsilon. \tag{1.40}$$

Obviously, from this objective, the main focus of universal adversarial perturbations is not the quality of the adversarial noises, but rather the quantity of examples on which the classifier is fooled. No universality metric has yet been proposed, therefore, the best adversarial metric is the fooling rate. We previously saw that the choice of the norm ℓ_p constraining the adversarial perturbation magnitude was not a consensus in the community. UAP set it to ℓ_∞ as the adversarial perturbations show interesting patterns, displayed in Figure 1.20.



(a) CaffeNet (b) VGG-F (c) VGG-16

(d) VGG-19 (e) GoogLeNet (f) ResNet-152

Figure 1.20: Examples of UAP produced universal adversarial perturbations on different neural network architectures under the ℓ_∞ norm constraint. Source: Universal Adversarial Perturbations [111].

The proposed algorithm computing UAP adversarial perturbations, incrementally accumulates specific DeepFool (1.34) adversarial perturbations over each example separately, hoping that the projection sum of the perturbations in the ϵ radius ℓ_p norm ball will remain adversarial to each specific example. Quite simple, this computation of universal adversarial perturbations has since been improved. Next to the very first universal adversarial attack UAP is, Fast-UAP [41]. This attack follows the UAP procedure but, instead of aggregating all the specific DeepFool perturbations at each iteration, it only adds the perturbation with the closest orientation to the current iterate. Among all the specific DeepFool adversarial perturbations $\omega^{\text{DF}}_{\mathbf{x}^{(i)}}$, Fast-UAP adds to the universal perturbation ω only the one with the maximum cosinus to the universal perturbations,

$$\omega^{t+1} = \omega^t + \max_{i \in [1,\text{N}]} \cos(\omega^t, \omega^{(i)}_{\text{DF}}), \tag{1.41}$$

with t the optimization iteration.

Elaborating on the previous optimizations procedure, UAP-PGD [142] frames the universal perturbation as the solution of the following optimization problem,

$$\max_{\omega \in \Omega} \sum_{i=1}^{\text{N}} l(f(\mathbf{x}^{(i)} + \omega, y^{(i)}) \;\; \text{s.t.} \;\; ||\omega||_p \leq \epsilon, \tag{1.42}$$

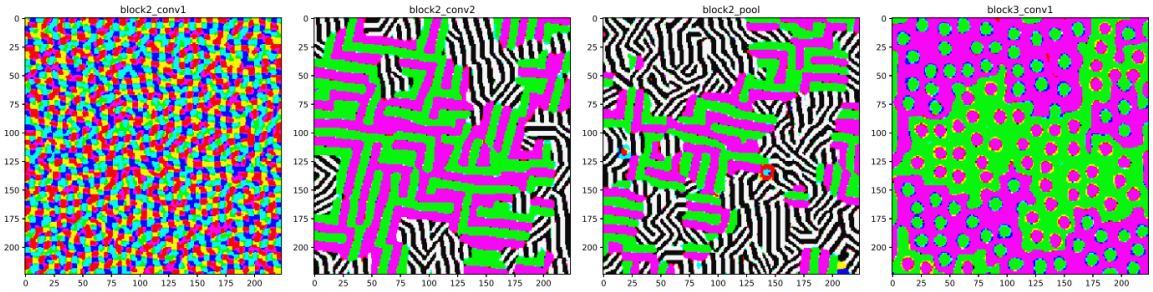
with l an adversarial loss.

Recently, UAP-PGD has been extended to class-wise UAP (CW-UAP) [15] where a single universal perturbation is built for each class.

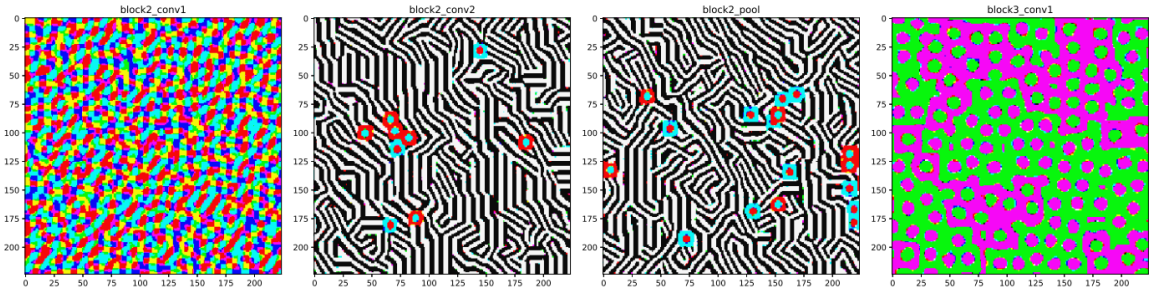
$$\max_{\omega^k, k \in \mathcal{Y}} \sum_{k \in \mathcal{Y}} \sum_{i=1}^{\text{N}^k} l(f(\mathbf{x}^{(i)} + \omega^k, k) \;\; \text{s.t.} \;\; \forall k \in \mathcal{Y}, \; ||\omega^k||_p \leq \epsilon. \tag{1.43}$$

The solution amounts to learning multiple independent UAP-PGD perturbations, one for each class. A different algorithm to craft universal adversarial perturbations has been proposed in [83]. Their method is based on the computation of the singular vectors of the Jacobian matrices of the classifier's feature maps to obtain universal adversarial perturbations. The proposed approach is much more data efficient, with universal perturbations reaching fooling rates of more than 60% on the ImageNet validation set by using only 64 images to optimize the universal adversarial perturbation.

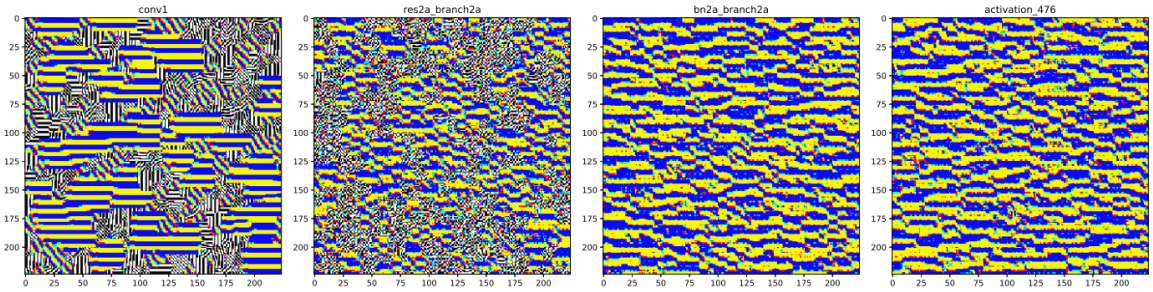
The idea of the semantic captured within the universal adversarial perturbations is again shown by empirical evidence of [83] as Figure 1.21 shows. Indeed, the forms and contours within the perturbations indicate that the universal adversarial perturbations crafted on singular vectors of the hidden layers target the features learned by those layers, such as edges, corners, or rounds. This figure is interesting as it empirically suggest that universal adversarial perturbations might rely on the most meaningful features of the object to fool the classifier. The unrealistic issue of perturbations produced by white-box adversarial attacks would thus be addressed.



(a) VGG-16



(b) VGG-19



(c) ResNet50

Figure 1.21: Examples of singular vector UAP produced universal adversarial perturbations constructed using various layers of various DNNs on different neural network architectures under the ℓ_∞ norm constraint. Source: Art of Singular Vectors and Universal Adversarial Perturbations [83].

The new performances shown in [83] indicate that, by being universal to the examples a universal perturbation has been crafted on, there are good chances that this very same universal perturbation will remain adversarial to other examples. Therefore, the transferability of universal perturbations is not to be neglected and becomes another metric that must be measured by uni-

versal adversarial attacks. The RAUD metric seems to be the most appropriate as it measures both the capacity to fool a targeted classifier and to transfer to other classifiers. The RAUD measures the fooling rate of the adversarial attack on originally well-classified examples, but for adversarial examples also fooling an adversarial example detector, which to some extent measures the transferability of these adversarial examples (see Section 1.2.2 for more details). Unfortunately, the RAUD has only been considered very recently.

Inspired by Generative Adversarial Networks (GAN) framework [64] (presented in Section 1.1.4), Network for Adversary Generation (NAG) was introduced by [114]. NAG aims to model the distribution of universal adversarial perturbations. Therefore, the authors modify the GAN framework by replacing the discriminator with the (frozen) target classifier and introducing a novel loss to train the generator. The novel loss function $\mathcal{L}$ is composed of an adversarial loss l and a diversity loss $\mathcal{L}_d$,

$$\begin{aligned} \mathcal{L} = &\quad l + \lambda \mathcal{L}_d, && \text{with} \\ l = &\quad -log(1 - f_y(\mathbf{x} + \omega)), && \text{and} \\ \mathcal{L}_d = &\quad -\sum_{j=1}^{\mathrm{B}} \mathrm{dist}(f^i(\mathbf{x}^{(j)} + \omega^{(j)}), f^i(\mathbf{x}^{(j)} + \omega^{(\mathrm{K})})), \end{aligned} \tag{1.44}$$

with λ a hyper-parameter, B the batch size, $\mathrm{K} \in [1, \mathrm{B}], \mathrm{K} \neq j$ a random index, $\omega^{(j)}$ and $\omega^{(\mathrm{K})}$ two generated perturbations, $\mathbf{x}^{(j)}$ an original example, i the index of the classifier's hidden layer, and dist a distance metric (e.g. the ℓ_2 norm). As we previously saw, the adversarial loss l is designed such that the generated perturbations produce adversarial examples fooling the target classifier while the diversity objective encourages the diversity of perturbations by increasing the distance of their feature embeddings predicted by the target classifier.

It is worth noticing that NAG is the first and among the few black-box universal adversarial attacks. The combination of both properties was made possible thanks to the modeling of the adversarial noise space by the generator. It indicates that modeling the adversarial noise space is a good idea as it allows us to explicit some adversarial perturbations constraints, enabling more realistic adversarial examples. Another variant of universal adversarial noise generation is generative adversarial perturbations (GAP) using a generator to craft universal adversarial perturbations [122]. Concurrently, the authors of [69] also explored the idea of generating adversarial perturbations with a generator network. Overall, the objective of the three last attacks is to train a (non-linear) generative network that transforms a random pattern to an image-dependant perturbation or universal adversarial perturbation. Unfortunately, all these attacks relied on a non-linear deep neural network to model the adversarial noise space which complicates the analysis of the model thus, preventing us from inspecting and visualizing this adversarial noise space. A more global view of the proposed universal adversarial attacks is given in the survey [182], readers are invited to refer to it for a wider understanding of what researchers proposed.

Overall, universal adversarial perturbations remain not enough explored. In this thesis, we bring in a new adversarial attack in the middle of specific and universal adversarial attacks.

Other types of adversarial attacks

Whereas the majority of adversarial attacks focus on perturbing clean images while enforcing the imperceptibility of the perturbation by restricting their ℓ_2 or ℓ_∞ norms, the Jacobian-based Saliency Map Attack (JSMA) [117] and One-pixel attack [149] deviate from this practice by restricting the perturbations to smaller regions of the image. Opposite to back-propagating the gradient of the adversarial perturbation through the classifier f, JSMA computes the forward gradient of f as,

$$\nabla f(\mathbf{x}) = \frac{\delta f_j(\mathbf{x})}{x_i}, \tag{1.45}$$

with $j = 1, \ldots, \mathrm{C}$ the prediction of f for label j, and x_i the i-th composant of $\mathbf{x}$.

Essentially, (1.45) computes the Jacobian of the function learned by the f. The Jacobian matrix is used as a saliency map to select only a few most influential pixels according to the alteration of the model prediction. [149] demonstrated that an image classifier can even be fooled by restricting the perturbation to a single pixel. The authors also studied the performances of the attack when increasing the number of pixels used for the adversarial example.

1.2.5 Adversarial defenses

Adversarial defenses have been proposed to protect against adversarial attacks harm. There are three main ways to defend against adversarial examples. The first way is to robustly optimize classifiers so they are more robustly trained for adversarial example danger. The second way is, given a classifier, to use an adversarial examples detector protecting the classifier beforehand. Finally, the third option is to use gradient masking [145] techniques, complicating the adversarial example creation, or at least complicating its functioning.

Robust optimization

Robust optimization defenses improve the robustness of the classifier by using regularization, certification bounds, or adversarial examples during the training of the classifier. Robust optimization of classifiers is also called adversarial training of classifiers. The adversarial training has been proposed by [63], as the incorporation of adversarial examples in the training of the classifier. The loss of the classifier does not change however, training examples now consist of valid original training examples along with FGSM computed adversarial examples of the training set.

$$\min_{f} -\sum_{c=1}^{C} o_c \log(f_c(\mathbf{x}')), \tag{1.46}$$

with $\mathbf{x}' \in B(\mathbf{x}, \epsilon)$, a valid or adversarial example in the ϵ ball centered on $\mathbf{x}$.

Overall, the goal of a robust optimization is to expand the decision boundaries of the classifier, to make it more robust, while still maintaining good classification performances. Figure 1.22 highlights such decision boundary expansion, making the classifiers more robust to adversarial examples, in the way that the computed adversarial examples are further away from the original examples, while still keeping the same high classification performances.

New adversarial training methods have been proposed with more sophisticated approaches, see [9] for a survey.

Adversarial examples detection

Another way to defend against adversarial examples is to recognize them before performing classification. This way if adversarial examples are recognized, they are ignored, preventing possible misclassification.

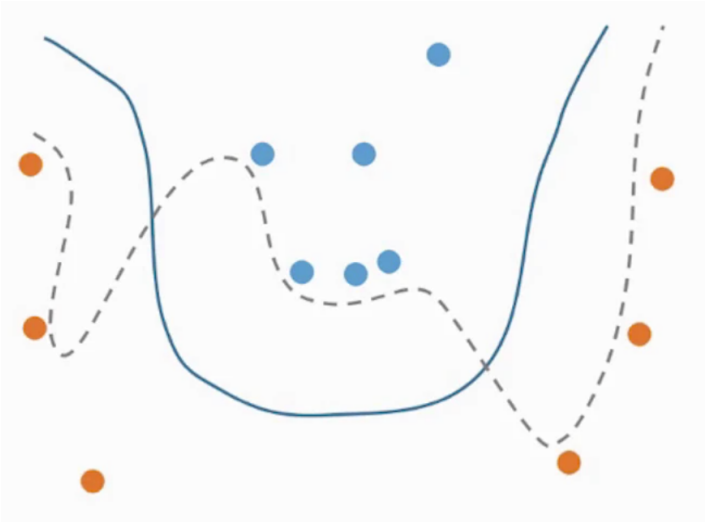


Figure 1.22: Schema of a more robust classifier's decision boundary. In a dashed line, the classifier's decision boundary is close the both cloud points, making the computed adversarial examples very close to the original examples. In a solid line, the shown decision boundary is more robust as the computed adversarial examples are at a higher distance from the original examples. Source: Liwei Wang's talk at "A Blessing in Disguise: The Prospects and Perils of Adversarial Machine Learning" work ICML 2021.

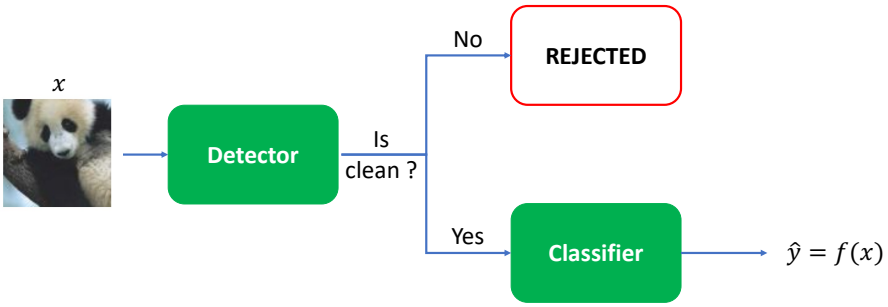


Figure 1.23: Schema of the adversarial examples detector adversarial defense.

The efficiency of adversarial examples detector is highlighted in [100] which proposes to use them not only as a defensive module but as an adversarial attack evaluation metric (see 1.2.3 for more details).

Gradient masking techniques

The last mainly used defense mechanism is gradient masking. The core idea of gradient masking is to build classifiers such that the adversarial examples crafting becomes troublesome. It can be done by using classifiers that are not differentiable, in that case, adversarial example generation via gradient computation is impossible. Another way to perturb the adversarial example generation is to add some tricks inside the classifier's rugs to obfuscate the gradients [6] (e.g. adding noise between each hidden layer). The noise injected in the classifier's internal parameters can, always be injected, or only injected at inference time [121].

Along with the publication of new, more powerful adversarial attacks, researchers also proposed new, fancier adversarial defenses. As Figure 1.24 shows, new adversarial defenses are more efficient in building more robust classifiers, but still working to find better ways to build robust classifiers should be a top priority. Readers are invited to refer to [2] for a survey of adversarial defenses.

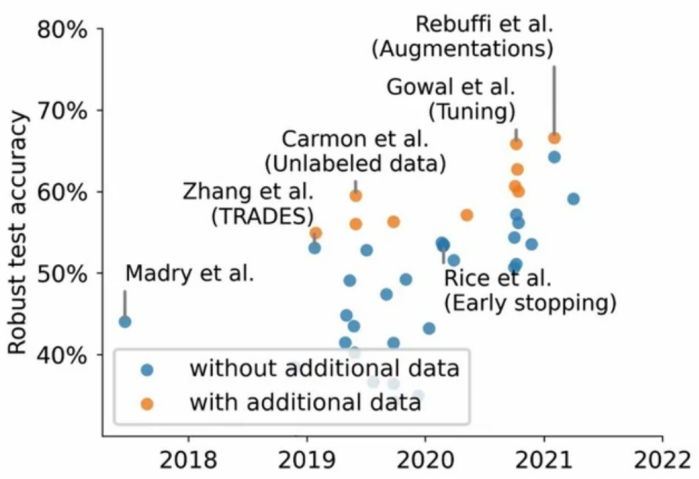


Figure 1.24: Published adversarial defense's robust accuracy according to the year of publication. Source: www.robustbench.io

1.2.6 No community consensus

On the one hand, common corruptions are too limited in the modeling of harmful examples to reliably estimate the robustness of classifiers and on the other hand, adversarial patches are too practical applications of adversarial examples making too many other adversarial perturbations out of reach when performing a robustness evaluation. We thus consider these two adversarial examples visions out of our scope and purposefully ignore them in our research.

As [143] highlights, ℓ_p norm-bounded adversarial noises imply that the similarity between objects, for instance, images, can be efficiently measured by an ℓ_p norm. However, two images don't need to be close to each other as measured by an ℓ_p norm to be perceptually similar. We believe that ℓ_p norm-bounded adversarial examples are neither necessary nor sufficient to measure perceptual similarity, implying realistic adversarial examples, thus a good robustness evaluation.

Within ℓ_p norm-bounded adversarial examples, we think universal adversarial examples are a good trade-off as Figure 1.25 shows. Universal adversarial attacks compute ℓ_p norm-bounded adversarial perturbations therefore, unrealistic perturbations, but are example-agnostic (universal to most examples). These unrealistic noises still evidence the always-open pitfalls of the classifiers. Such universal weaknesses of the classifiers remain true for the unrealistic harmful examples, but also for the realistic ones, that we struggle to compute using common corruptions.

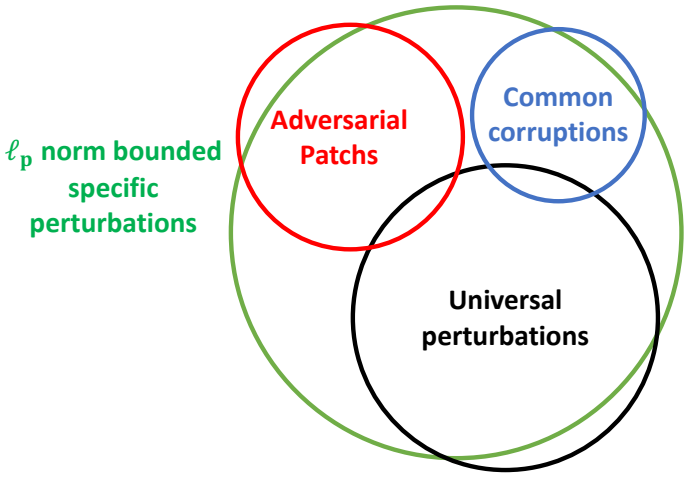


Figure 1.25: Schema of the two adversarial examples vision in the community, with the universal adversarial examples in the middle.

Overall, the universal adversarial perturbations are free to fully explore the ϵ radius ℓ_p norm ball around each example, which is unrealistic, but has the constraint to be universal to most of the examples, which is in the way to compute a realistic robustness evaluation. We believe universal adversarial perturbations act as a good trade-off between both worlds, allowing us to take advantage of the ℓ_p norm-bound computations and to draw conclusions that are useful for real-world applications.

Chapter 2

Adversarial attack through dictionary learning

In this chapter, we present the main contribution of this thesis, which is the modeling of adversarial noise using a dictionary learning framework. First, we present the goal and its motivations. Then, we formalize the dictionary learning problem of the adversarial noise space. Next, we present the three relaxation optimizations we proposed, which are an ADiL (Adversarial Dictionary Learning) proximal gradients solution, an ADiL projected gradient solution, and a LIMANS (Linear Modeling of the Adversarial Noise Space) stochastic gradient solution. For each, we present the proposed optimization problem, give algorithmic solutions, and present its empirical results. The third relaxation offers more results as it is the one we ended up with after reviewing the results of the two previous ones. Finally, we conclude the overall idea and propose its extensions for future promising research.

2.1 Motivations

Among the proposed ℓ_p norm-bounded adversarial attacks, on the one hand, specific adversarial attacks are the most efficient but the produced adversarial examples struggle to transfer to other examples, on the other hand, universal adversarial attacks do produce adversarial noises suited to multiple examples but they suffer from lower attack success. To reconcile universality and efficiency, we proposed to linearly model the adversarial noise space Ω through as set M adversarial directions $D \in \mathbb{R}^{M \times P}$, allowing to frame any specific adversarial perturbation as a linear combination of the M universal adversarial directions. The idea behind an adversarial attack through dictionary learning is to, model the adversarial noise space Ω with a dictionary and then, to look for specific adversarial noises corresponding to the coding vectors of the dictionary within its modeled space.

By modeling the adversarial noise space Ω, we can find adversarial parameters that are example-agnostic and therefore contain all information needed to fool the classifier on any examples. While still modeling the overall space, we still have the possibility, for any original examples $\mathbf{x}$, to navigate within that space to craft an efficient adversarial noise $\omega_{\mathbf{x}}$ suited to this examples, to ensure good adversarial performances. This proposed framework then bridges the gap between universal and specific adversarial attacks, hoping to retain the good adversarial performances of specific attacks, and the transfer property of universal attacks.

2.1.1 Learning the adversarial space

Some studies on the overall structure of deep classifiers tried to clarify and explain the rise and origin of adversarial attacks. Research in [56] illustrated the decision boundaries of a classifier are in the vicinity of examples and flat in most the directions. More recently, [95] claimed that adversarial noise is caused by gradient leakage and the adversarial directions are perpendicular to

the classifier boundaries. Baluja and Fischer [10] trained feed-forward neural networks to generate adversarial examples against other targeted networks or sets of networks. The trained models were termed Adversarial Transformation Networks (ATNs). In the same direction, Hayes and Danezis [68] also used an attacker neural network to learn adversarial examples for black-box attacks. In some sense, generating adversarial examples with a neural network can be seen as modeling the adversarial example space by this neural network.

Tabacof and Valle [155] empirically demonstrated that adversarial examples appear in large regions of the pixel space, which is in line with the similar claim in [63]. Then, [159] discovered that the decision boundaries of different classifiers are closed and proposed to establish a transferable subspace of the space across different classifiers. They proposed a method to estimate the dimensionality of the space of the adversarial examples. However, the space is inferred based on the adversarial noise generated by the FGSM method which impacts the precision of the found space. The hypothesis of the transferability depending only on the dimensionality of this space limited its performance on CNN classifiers.

Most of the previous work tried to model the space of adversarial examples, which is quite different than the space of adversarial noises itself.

Moosavi-Dezfooli, Fawzi, Fawzi and Frossard [111] initially argued that universal adversarial perturbations exploit geometric correlations between the decision boundaries induced by the classifiers. Their existence partly owes to a subspace containing normals to the decision boundaries, such that the normals also surround the natural images. In [112], they built further on their theory and showed the existence of common directions (shared across data points) along which the decision boundary of a classifier can be highly positively curved.

2.1.2 Manifold hypothesis

The manifold hypothesis suggests that high-dimensional data, such as images or text, often lies on or near a lower-dimensional manifold embedded within that high-dimensional space. It implies that data points with similar characteristics or semantic meaning are closer to each other in this lower-dimensional manifold. Autoencoders [8] can help address the manifold hypothesis problem by learning an efficient data representation that captures the underlying structure or manifold of the data. An autoencoder is a type of neural network that is trained to reconstruct its input data. It consists of two main components: an encoder and a decoder as shown in Figure 2.1. During training, the autoencoder learns to encode the input data into a compressed representation in the latent space, while the decoder learns to reconstruct the original input from this representation. By doing so, the autoencoder is forced to capture the essential features of the data and discard the noise or irrelevant information. After training, the autoencoder effectively modeled the underlying structure or manifold of the data.

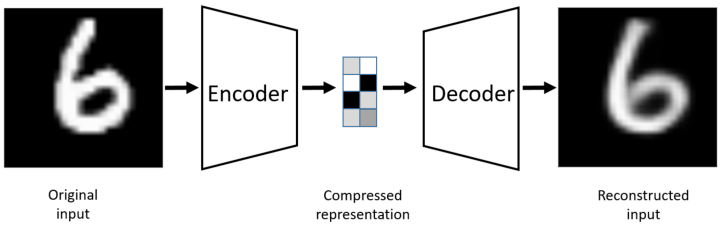


Figure 2.1: Example of an autoencoder performing compression and the decompression of an MNIST image. See that the code outputted by the encoder is of lower dimension than the images but allows for full reconstruction of the image, therefore containing all the information within the image. Source: Autoencoders [11].

Using an autoencoder for adversarial examples has already been proposed by [106] in 2015. Although quite compelling, their approach mainly relied on a GAN generator, which is not exactly the goal we are after.

Here, we assumed that the manifold hypothesis holds for adversarial noise space Ω and can be modeled with a dictionary. Rather than setting $\Omega = \mathbb{R}^P$, we assume this space, if well constrained, can be of lower dimension while maintaining decent adversarial performances. By modeling the underlying structure or manifold of the adversarial noise space, the adversarial dictionary captures the essential features and relationships of the adversarial noises, making the produced adversarial noises both specific to each example, and universal to the modeled adversarial noise space.

As the simplest possible case of modeling, an adversarial dictionary aims to be a linear model of the adversarial noise space, so that both the problem is tractable and the parameters could be inspected.

2.2 Adversarial dictionary learning objective

Let a classifier $f\colon \mathbb{R}^P \mapsto \mathbb{R}^C$ outputting $f(\mathbf{x}) \in \mathbb{R}^C$, the vector of scores for an example $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^P$ to belong to a class of $y \in \mathcal{Y} = \{1, \cdots, C\}$. The predicted class is given by $\operatorname{argmax}_k f_k(\mathbf{x})$. Recall that, given an example $\mathbf{x}$, an adversarial attack seeks an adversarial perturbation $\omega \in \Omega$ such that $\mathbf{x}' = \mathbf{x} + \omega$, the adversarial example, is a valid example *i.e.* $\mathbf{x}' \in \mathcal{X}$, close to $\mathbf{x}$, and induces misclassification $\operatorname{argmax}_k f_k(\mathbf{x}) \neq \operatorname{argmax}_k f_k(\mathbf{x}')$ (respecting the adversarial example constraints 1.19).

The originality of our proposition is to express the adversarial noise paired to $\mathbf{x}$ as $\omega_{\mathbf{x}} = D\mathbf{v}(\mathbf{x})$ where $D \in \mathbb{R}^{P \times M}$ is a dictionary composed of M normalized adversarial noise atoms and $\mathbf{v}(\mathbf{x}) \in \mathbb{R}^M$ a coding vector (further on we write $\mathbf{v}$ for readability). While the dictionary D is universal to every example, the coding vector $\mathbf{v}$ is specifically tailored to any given $\mathbf{x}$. By setting M = 1 atom the learned adversarial perturbation is universal while, by setting M = P (the dimension of the examples) the adversarial perturbation becomes specific.

In a way, an adversarial dictionary attack's novelty can be seen as, constraining the adversarial noise space such that $\omega \in \Omega = \{\omega_k, \omega_k = D\mathbf{v}_k, \mathbf{v}_k \in \mathbb{R}^M\}$, with $D \in \mathbb{R}^{P \times M}$ acting as a base spanning this space.

Given a trained DNN classifier f, the adversarial dictionary attack consists of two stages. First a training stage where the dictionary D is learned using a labeled set $\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$. Then an inference stage where, given D, for any new example $\mathbf{x}^{(k)}$ (yet unseen), the corresponding coding vector $\mathbf{v}^{(k)}$ is crafted to make $\omega = D\mathbf{v}^{(k)}$ an adversarial perturbation of $\mathbf{x}^{(k)}$. Notice that as $M \ll P$, the searching space of the adversarial dictionary attack is a low dimensional space (spanned by the atoms) which is much lower than the original space $\mathcal{X}$. We frame the learning procedure of the adversarial dictionary attack as maximizing the fooling rate under the adversarial example constraints 1.19.

Problem 2.2.1 (adversarial dictionary training objective)**.** *Given the classifier f and the training set $\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$, find $D \in \mathbb{R}^{P \times M}$ and $V \in \mathbb{R}^{M \times N}$ solution of*

$$\begin{aligned} \max_{\substack{D \in \mathbb{R}^{P \times M} \\ V \in \mathbb{R}^{M \times N}}} \quad & \sum_{i=1}^{N} \mathbb{1}\{\operatorname{argmax}_k f_k(\mathbf{x}^{(i)'}) \neq \operatorname{argmax}_k f_k(\mathbf{x}^{(i)})\}, \\ s.t. \quad & \begin{cases} \mathbf{x}^{(i)'} = \mathbf{x}^{(i)} + D\mathbf{v}^{(i)} \in \mathcal{X} & i = 1, \dots, N, \\ ||D\mathbf{v}^{(i)}||_p \leq \epsilon_p & i = 1, \dots, N, \\ \|D_j\|_p = 1 & j = 1, \dots, M, \end{cases} \end{aligned} \tag{2.1}$$

where $\mathbb{1}_{\mathcal{A}}$ denotes the indicator function of the set A. This optimization problem optimizes the dictionary D as well as all the coding vectors of the training set $V \in \mathbb{R}^{M \times N}$, but the latter variable is not interesting as it is useless for future adversarial crafting, it only allows to optimize D.

Once the optimization of D is completed, the adversarial noise space is now modeled and, for any new (yet unseen) adversarial example $\mathbf{x}$, we can navigate in the space spanned by D to find the best specific to $\mathbf{x}^{(i)}$ adversarial noise $\omega^{(i)} = D\mathbf{v}^{(i)}$. Inference problem only optimizes the coding vector $\mathbf{v}^{(i)} \in \mathbb{R}^M$,

Problem 2.2.2 (adversarial dictionary inference objective)**.** *Given the classifier f, an optimized dictionary of adversarial directions $\mathrm{D} \in \mathbb{R}^{\mathrm{P}\times\mathrm{M}}$ and an original labeled example $(\mathbf{x}, y)$, find $\mathbf{v} \in \mathbb{R}^{\mathrm{M}}$ solution of*

$$\begin{aligned} \max_{\mathbf{v}\in\mathbb{R}^{\mathrm{M}}} \quad & \mathbb{1}\{\mathrm{argmax}_k\, f_k(\mathbf{x}') \neq \mathrm{argmax}_k\, f_k(\mathbf{x})\}, \\ s.t. \quad & \begin{cases} \mathbf{x}' = \mathbf{x}^{(i)} + \mathrm{D}\mathbf{v} \in \mathcal{X} \\ ||\mathrm{D}\mathbf{v}||_p \leq \epsilon_p, \end{cases} \end{aligned} \tag{2.2}$$

Both objective 2.1 and 2.2 have the main drawback to be intractable problems. Therefore, they must be relaxed with a differentiable loss function to ensure good optimization.

During our work, we proposed three different relaxations. Namely, we proposed a first solution, Adversarial Dictionary Learning (ADiL) solved using proximal gradients, a second solution Adversarial Dictionary Learning (ADiL) solved using projected gradients, and finally Linear Model of the Adversarial Noise Space (LIMANS), each presented with some extensions. In the following sections, we review the different proposed solutions, in the respective order.

2.3 ADiL proximal gradients solution

At first, we tried to optimize problem 2.2.1 under the scope of a dictionary learning framework, and formulated the optimization as such.

The first ADiL optimization relies on the first relaxation of the adversarial example problem therefore, the adversarial noises are not constrained within an ϵ ball, although the minimization of their norm is at the core of the adversarial example optimization.

Optimization

We proposed a first framework to find a dictionary of shared universal attacks D, named ADiL for *Adversarial Dictionary Learning* [59]. We proposed to address the following optimization problem reminiscent of classical dictionary learning problems (see, e.g., [105, 127]).

$$\underset{\substack{\mathrm{D}\in\mathcal{D} \\ \mathrm{V}\in\mathcal{V}^{\mathrm{N}}}}{\text{minimize}} \sum_{i=1}^{\mathrm{N}} l(f(x^{(i)} + \mathrm{D}\mathbf{v}^{(i)}), y^{(i)}) + \lambda_1 ||\mathbf{v}^{(i)}||_1 + \lambda_2 ||\mathrm{D}\mathbf{v}^{(i)}||_2, \tag{2.3}$$

where l is an adversarial loss enforcing misclassification, whereas $\mathcal{D} \subset \mathbb{R}^{\mathrm{P}\times\mathrm{M}}$ and $\mathcal{V} \subset \mathbb{R}^{\mathrm{M}}$ encode some constraints on D and the specific coding vectors $\mathbf{v}^{(i)}$, respectively.

See that in equation (2.3), the adversarial loss l replaces the non-convex and hard to optimize $\mathbb{1}$ indicator function of equation (2.1). This relaxation shifts the maximization of the number of adversarial examples to the minimization of an adversarial loss function l, which is a typical procedure in machine learning and allows smoother and more manageable optimization procedures. In ADiL we specifically selected the reverse cross-entropy (minus eq. (1.8) as a convenient adversarial loss $\mathcal{L}$ [25, see function f in Section VI.A]).

Even though equation (2.1) explicit three distinct constraints, equation (2.3) does not aim at respecting them. Indeed, at the time, we did not perfectly understood these constraints and thus incorporated them within our proposition. Instead of these three constraints, equation (2.3) does enforce to compute adversarial perturbations $\omega^{(i)} = \mathrm{D}\mathbf{v}^{(i)}$ of low magnitude thanks to the second regularization term. We believed it to be the best option to craft descent and acceptable adversarial perturbations. This optimization problem does not enforce a fixed magnitude budget for the adversarial noises but instead, relies on regularization with the variables λ_2 and λ_1 to ensure respectively, that the magnitude of every adversarial noise stays within the ϵ budget and a good sparsity of the coding vectors $\mathbf{v}^{(i)}$, an essential component of dictionary learning problems. The second regularization term, managed with λ_1, enforces sparsity within the coding vector's entry. This regularization technique is key to well-defined dictionary learning problems, which is in line with our first idea, to frame the original problem under the dictionary learning scope. Therefore

as a good practice, we included these constraints within the optimization problem even though equation (2.1) does not state any particular sparsity constraints.

Note that equation (2.3) is simply the lagrangian formulation of the original problem (2.1), with λ_1 and λ_2 acting as dual variables. Objectively and without context, optimizing the lagrangian formulation of a constrained problem is among the best options. Therefore, before stepping back from the adversarial example definition and criticizing the state-of-the-art propositions, we considered this formulation, the best choice.

Algorithmic solution

Minimizing the objective in (2.3) is a challenge due to the nonconvexity inherent to the dictionary learning formulation and the neural network f. We stress that we are only interested in finding a good stationary point in a limited time. Although classical dictionary learning problems are nonconvex, they are usually solved by alternating the optimization over D and V since each alternating problem is convex. However, here this is no longer the case because of the added term promoting adversarial examples. Hence, we embrace a direct optimization scheme over (D,V) in the spirit of the nonconvex proximal splitting framework of [148] which has also been applied in the context of classical dictionary learning in [127]. To that purpose, we begin by recasting equation (2.3) into,

$$\underset{\substack{\mathrm{D}\in\mathbb{R}^{\mathrm{P}\times\mathrm{M}}\\ \mathrm{V}\in\mathbb{R}^{\mathrm{M}\times\mathrm{N}}}}{\text{minimize}}\ \mathrm{F}(\mathrm{D},\mathrm{V})+\mathrm{R}(\mathrm{D},\mathrm{V}), \tag{2.4}$$

with

$$\begin{cases}\mathrm{F}(\mathrm{D},\mathrm{V}) & =\sum_{i=1}^{\mathrm{N}}\lambda_2\|\mathrm{D}\mathbf{v}^{(i)}\|_2^2+\mathcal{L}\left(f(\mathbf{x}^{(i)}+\mathrm{D}\mathbf{v}^{(i)}),y^{(i)}\right),\\ \mathrm{R}(\mathrm{D},\mathrm{V}) & =\sum_{i=1}^{\mathrm{N}}\lambda_1\|v^{(i)}\|_1,\end{cases}$$

where F is smooth, provided that f is smooth as well.

Here, we promote the linesearch based proximal-gradient method of [18]. For the ease of reading, in what follows we only focus on the proximal-gradient step. Given some sequence of stepsizes $\{\gamma_k\}_{k\in\mathbb{N}}$, each step amounts in finding

$$(\mathrm{D}^{(k+1/2)},\mathrm{V}^{(k+1/2)})=\underset{\mathrm{D}\in\mathbb{R}^{\mathrm{P}\times\mathrm{M}},\mathrm{V}\in\mathbb{R}^{\mathrm{M}\times\mathrm{N}}}{\operatorname{argmin}}\ h^{(k)}(\mathrm{D},\mathrm{V}),$$

where

$$\begin{aligned}h^{(k)}(\mathrm{D},\mathrm{V}) &= \mathrm{R}(\mathrm{D},\mathrm{V})-\mathrm{R}(\mathrm{D}^{(k)},\mathrm{V}^{(k)})\\ &+\nabla_{\mathrm{D}}\mathrm{F}(\mathrm{D}^{(k)},\mathrm{V}^{(k)})^{\top}(\mathrm{D}-\mathrm{D}^{(k)})+\gamma_k^{-1}\|\mathrm{D}-\mathrm{D}^{(k)}\|^2/2\\ &+\nabla_{\mathrm{V}}\mathrm{F}(\mathrm{D}^{(k)},\mathrm{V}^{(k)})^{\top}(\mathrm{V}-\mathrm{V}^{(k)})+\gamma_k^{-1}\|\mathrm{V}-\mathrm{V}^{(k)}\|^2/2.\end{aligned}$$

The overall step can be recast as

$$\begin{pmatrix}\mathrm{D}^{(k+1/2)}\\ \mathrm{V}^{(k+1/2)}\end{pmatrix}=\operatorname{prox}_{\gamma_k\mathrm{R}}\left(\begin{pmatrix}\mathrm{D}^{(k)}\\ \mathrm{V}^{(k)}\end{pmatrix}-\gamma_k\nabla\mathrm{F}(\mathrm{D}^{(k)},\mathrm{V}^{(k)})\right)$$

where the gradient is computed jointly over D and V. Note that since R is separable, it yields that

$$\operatorname{prox}_{\gamma_k\mathrm{R}}(\mathrm{D},\mathrm{V})=\operatorname{Proj}_{\mathcal{D}}(\mathrm{D}),\operatorname{Soft}_{\gamma_k\lambda_1}(\mathrm{V}),$$

where $\operatorname{Soft}_{\gamma_k\lambda_1}(\mathrm{V})$ is the soft-thresholding operator [43]. The full scheme is sketched in Algorithm 2. At inference, for new unseen examples, the same algorithm is used, only the optimization is only done over the coding vectors $v^{(i)}$.

Algorithm 2 ADiL

Require: Parameter $\delta \in]0,1[$
  Set $D^{(0)} \sim \mathcal{N}(0_{P\times M}, 1_{P\times M})$ and $V^{(0)} = 0_{M\times N}$
  for $k = 0$ to $K-1$ **do**
    Provide rough estimate of $\gamma_k > 0$
    Proximal-gradient step
    $D^{(k+1/2)} = \text{Proj}_{\mathcal{D}}(D^{(k)} - \gamma_k \nabla_D F(D^{(k)}, V^{(k)}))$
    $V^{(k+1/2)} = \text{Soft}_{\gamma_k \lambda_1}(V^{(k)} - \gamma_k \nabla_V F(D^{(k)}, V^{(k)}))$
    Compute the difference
    $d_D^{(k)} = D^{(k+1/2)} - D^{(k)}$ & $d_V^{(k)} = V^{(k+1/2)} - V^{(k)}$
    Armijo-like backtracking loop
    $i_k = 0$ and $h_k = h^{(k)}(D^{(k+1/2)}, V^{(k+1/2)})$
    repeat
      $\tilde{D}^{(k)} = D^{(k)} + \delta^{i_k} d_D^{(k)}$ & $\tilde{V}^{(k)} = V^{(k)} + \delta^{i_k} d_V^{(k)}$
      $i_k = i_k + 1$
    until $\mathcal{L}(\tilde{D}^{(k)}, \tilde{V}^{(k)}) \leq \mathcal{L}(D^{(k)}, V^{(k)}) + \delta^{i_k} h_k$
    $D^{(k+1)} = \tilde{D}^{(k)}$ and $V^{(k+1)} = \tilde{V}^{(k)}$
  end for
  return Dictionary $D^{(K)}$, coding vector $V^{(K)}$

Results

It is worth noting that the adversarial noise constraints are respected through both D and V regularization with the regulation variables λ_1 and λ_2. Therefore, very different results can be obtained according to the tuning of each of the two regulation variables. As Figure 2.2 shows, by enforcing a large λ_2 hyper-parameter, the magnitude of the adversarial noises is hugely constrained which results in a decrease in fooling rate performances. At the same time, when lowering the λ_1 hyper-parameter, coding vectors are allowed to be less sparse and thus more freely explore the space spanned by D to compute adversarial noises, which increases the performances of the adversarial attack, but at the cost of producing adversarial noises of higher magnitude.

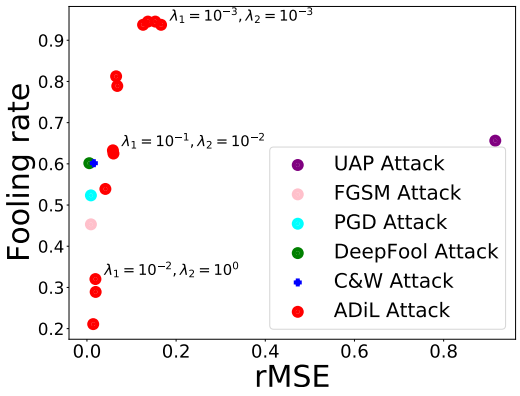


Figure 2.2: Comparison with state-of-the-art adversarial attacks of ADiL with different hyper-parameters λ_1 and λ_2 on a LeNet classifier on the CIFAR-10 dataset. On the y-axis is displayed the test fooling rate performances of adversarial attacks, while the rMSE of the produced adversarial noises is displayed on the x-axis.

In the proposed framework, the number of atoms M composing the dictionary acts as a hyper-parameter controlling to what extent the adversarial noise information is compressed and therefore universal. On the one hand, when M = 1 each adversarial noise only differs by the intensity of the added perturbation through the quantity $\mathbf{v}^{(i)}$. On the other hand, for M = P, the degree of

freedom is high enough to let adversarial noises be completely specific to each image. In that case, each atom targets one pixel of the adversarial perturbation, making D a basis of $\mathbb{R}^P$. The choice of M has been investigated and indeed impacts the performance of the attacks. Figure 2.3 reports ADiL's test fooling rates for various numbers of M. From Figure 2.3, we observe that ADiL indeed reaches better test fooling rate performances with a higher M but at the cost of becoming less universal.

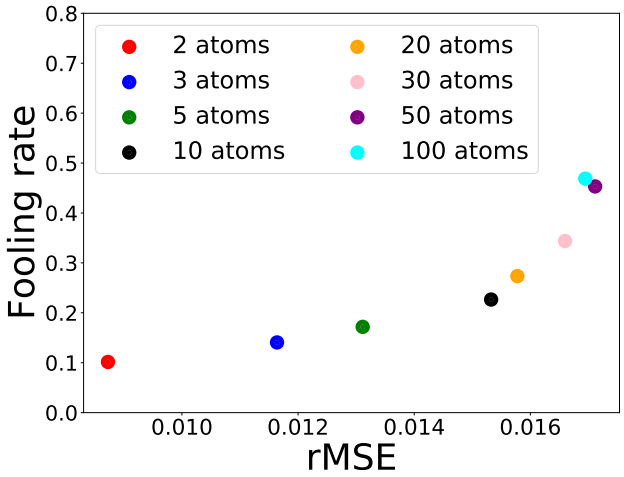


Figure 2.3: ADiL's performances with different numbers of atoms M on a LeNet classifier on the CIFAR-10 dataset. On the y-axis is displayed the test fooling rate performances of adversarial attacks, while on the x-axis is displayed the rMSE of the produced adversarial noises.

Learning ADiL with less data

As a sanity check of the optimization and looking for insights, we tried to optimize problem (2.3) using fewer training data points N. Indeed, the idea of adversarial perturbation modeling was to capture the most common patterns of the data fooling the classifier. Therefore by increasing the number of training points, the inference performances on unseen data points should increase as D captures more generalized fooling patterns. And we empirically confirm this insight. Indeed, in Figure 2.4, we see that by increasing the training set size N the performances D increase for various numbers of atoms M. For $M = 5, M = 10$, and $M = 15$ atoms, the test fooling significantly increases but for only $M = 1$ atom the performance only marginally increases. This artifact is a result of a too-small modeling space. Indeed, with only $M = 1$ atom, D is, to a coefficient factor, universal to every example, which constrains the problem too much to see that big of improvement when varying M. However, for higher values of M, the modeling space is large enough to see significant changes in the test performances according to the training set size N.

Overall and as expected test performances increase according to the number of atoms M, but because of the tuning of the regularization parameters λ_1 and λ_2 and the random seed, we see some discontinuities in the order of performances for the three number of atoms. This variability in performances is confirmed as the higher the number of atoms, the smoother the performances increase. Indeed, for $M = 15$, performances almost increase linearly with the number of training data points. Whereas for $M = 5$, test performances are globally growing with the number of training data points, even though we see some light "abnormal" decrease due to external factors.

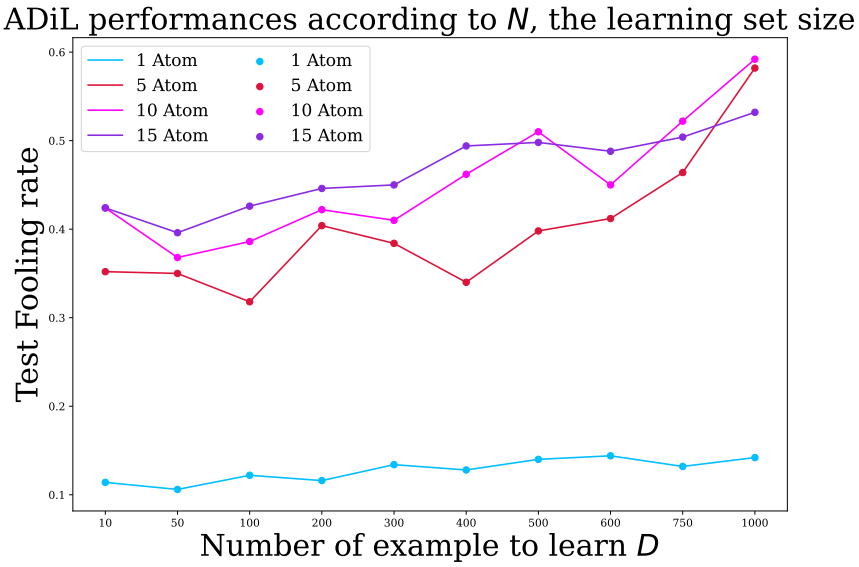


Figure 2.4: ADiL performances with a different number of training data points N for multiple numbers of atoms M = 5, 10, 15. The dictionary D is optimized on a LeNet classifier on the CIFAR-10 dataset. On the y-axis is displayed the test fooling rate performances of ADiL, while on the x-axis is displayed the number of training data points.

2.4 ADiL projected gradients solution

The main drawback of the first proposed dictionary learning optimization is that it does not respect the fixed adversarial budget, which makes the comparison with state-of-the-art attacks biased. It is worth noting that plenty of adversarial attacks do not originally respect this constraint either. In the revisited version of ADiL, we proposed a framework that allows controlling the norm of every adversarial noise to be within an ϵ. This second version of ADiL relies on the second adversarial crafting relaxation, such that $||\omega^{(i)}||_p = ||\mathrm{D}\mathbf{v}^{(i)}||_p \leq \epsilon \forall i$.

Optimization

In order to learn the dictionary of shared attacks, we propose to address the following optimization problem reminiscent of classical dictionary learning problems (see, e.g., [105, 127]).

$$\begin{aligned} \min_{\mathrm{D}\in\mathcal{D},\mathrm{V}\in\mathcal{V}} \quad & \sum_{i=1}^{\mathrm{N}} l\big(f(\mathbf{x}+\mathrm{D}\mathbf{v}^{(i)}), y^{(i)}\big), \\ s.t. \quad & \big\{ \; ||\mathrm{D}\mathbf{v}^{(i)}||_p \leq \epsilon_p \quad i = 1,\dots,\mathrm{N}, \end{aligned} \tag{2.5}$$

Compared to the firstly proposed optimization, we used two adversarial loss $\mathcal{L}$, namely the logit loss and the reverse cross-entropy, both proposed in [25]. In Section 2.4.1, we detail a deeper analysis of these two losses. Depending on the choice of the constrained sets $\mathcal{D}$ and $\mathcal{V}$, one can ensure that the adversarial noises are all ℓ_p-norm constrained. We integrated this constraint within the parameters D, V which ends up in the following proposition,

Proposition 2.4.1 (ℓ_p-Attacks)**.** *Given some budget* $\epsilon > 0$*, we have that* $\|\mathrm{D}v\|_p \leq \epsilon$ *for every* $\mathrm{D} \in \mathcal{D}$ *and* $v \in \mathcal{V}$ *where*

$$\mathcal{D} = \{\mathrm{D} \in \mathbb{R}^{\mathrm{P}\times\mathrm{M}} \mid (\forall m \in \{1,\dots,\mathrm{M}\}), \|d_m\|_p \leq 1, p \in \mathcal{P}\}, \tag{2.6}$$

and

$$\mathcal{V} = \{v \in \mathbb{R}^{\mathrm{M}} \mid \|v\|_1 \leq \epsilon\}. \tag{2.7}$$

Note that, for such a choice, both $\mathcal{D}$ and $\mathcal{V}$ are convex sets. The proof that by framing $\omega = \mathrm{D}\mathbf{v}$, with $\mathrm{D} \in \mathcal{D}, \mathbf{v} \in \mathcal{V}$ results in $||\omega||_p \leq \epsilon$ is given in Section A.1. The integration of this proposition within the optimization problem is solved by Algorithm 4.

2.4.1 Inference optimization

We now consider the definition of the dictionary-based adversarial attack to unseen examples, coined the ADiL attack.

Provided that the dictionary $\mathrm{D} \in \mathcal{D}$ is known, for a new yet unseen original example $\mathbf{x}$ of original label y, we propose two ways to learn its paired adversarial perturbation $\mathbf{x} + \mathrm{D}\mathbf{v}$ such that the dictionary D is considered a parameter, and the optimization is performed only over $\mathbf{v}$,

$$\text{(unsupervised)} \quad v \sim \mathcal{P} \tag{2.8}$$

$$\text{(supervised)} \quad \min_{\mathbf{v} \in \mathcal{V}} l(f(\mathbf{x} + \mathrm{D}\mathbf{v}, y), \tag{2.9}$$

Equation 2.8 derives a black-box ADiL adversarial attack. In the meantime, equation 2.9 gives a white-box ADiL adversarial attack. Equation 2.9 can be solved by resorting to Algorithm 4 and ignoring the optimization steps over D.

However, in practice, we found the optimized adversarial perturbation norm $||\mathrm{D}\mathbf{v}||_p$ might still be far from the ℓ_p norm budget ϵ. This artifact prevents us from fully exploring the space spanned by D, leading to sub-optimal results. When all the adversarial attacks are given an ϵ ℓ_p norm budget, every adversarial perturbation will be of norm ϵ. Therefore by resorting to an algorithm that disables us from reaching this norm budget, the ADiL adversarial attack is unfairly compared to other adversarial attacks and could therefore never reach their performances.

To benefit from the proposed framework, while still being competitive with other adversarial attacks, we proposed to recast the objective of the ADiL white-box adversarial attack (2.9). Given $\mathrm{D} \in \mathbb{R}^{\mathrm{P} \times \mathrm{M}}$ with $\mathrm{M} \ll \mathrm{P}$ and by assuming that D is of full rank, we may write $\mathbf{v} = \mathrm{D}^{\dagger}\mathbf{z}$ with $\mathrm{D}^{\dagger} = (\mathrm{D}^{\top}\mathrm{D})^{-1}\mathrm{D}^{\top}$, therefore, problem (2.9) becomes,

$$\min_{\mathbf{z} \in \mathbb{R}^{\mathrm{P}}} l(f(\mathbf{x} + \mathrm{D}\mathrm{D}^{\dagger}\mathbf{z}, y), \quad \text{s.t.} \quad \|\mathbf{z}\|_p \leq \epsilon. \tag{2.10}$$

Equation (2.10) takes advantage of all the possibilities offered by D, navigating in all of the spanned space, allowed to find a maximum number of adversarial perturbation under the ℓ_p norm ϵ budget constraint. Finally, to ensure that the adversarial example $\mathbf{x} + \mathrm{D}\mathbf{v}$ is a valid image, we additionally perform a projection onto the input manifold $\mathcal{X} \subseteq \mathbb{R}^{\mathrm{P}}$, i.e., $x' = \mathrm{Proj}_{\mathcal{X}}(\mathbf{x} + \mathrm{D}\mathbf{v})$. The biggest flaw of recasting the problem into equation (2.10) is that now the optimization in on $\mathbf{z} \in \mathbb{R}^{\mathrm{P}}$ rather than $\mathbf{v} \in \mathcal{V} \subset \mathbb{R}^{\mathrm{M}}$, which includes as many variables $\mathbf{z}_i, i \in \{1, \ldots, \mathrm{P}\}$ as the number of pixels, which is the problem solved by state-of-the-art adversarial attacks. Indeed, the recast loses the computational gained offered by the storage of a complete dictionary. Algorithm 4 presents the optimization of the white-box ADiL adversarial attack, and details the different performed projections.

Algorithmic solution

Herein we propose a procedure for solving Problem (2.5). Note that minimizing the objective in (2.5) is challenging due to the nonconvexity inherent to the dictionary learning formulation and the neural network f. We highlight that we are only interested in finding a convenient stationary point in a limited time. Since classical dictionary learning problems are bi-convex, they usually are solved by alternating the optimization over D and V since each alternating problem is convex. However, here this is no longer the case because of the adversarial loss involving a neural network. Hence, we embrace a direct optimization scheme over (D, V) in the spirit of the nonconvex proximal splitting framework of [148] which has also been applied in the context of classical dictionary learning in [127]. Algorithm 4 performs a simple projected gradient descent over $\mathbf{z}$ by minimizing an adversarial loss l such that the algorithm outputs a valid adversarial example $\mathbf{x}' = \mathbf{x} + \mathrm{D}\mathbf{v} = \mathbf{x} + \mathrm{D}\mathrm{D}^{\dagger}\mathbf{z}$,

The projection operator $\mathrm{proj}_{\epsilon,p}$ of Algorithm 4 is a simple yet effective rescaling of the variable by its norm, that is $\mathrm{proj}_{\epsilon}(z) = z * \epsilon / ||z||_p$. Given a pre-process of the original image pixels values,

Algorithm 3 ADiL using projected gradients

Require: Step-sizes $\{\gamma_k\}_{k=0}^{K-1}$
  Set $D^{(0)} \sim \mathcal{D}$
  Set $V^{(0)} = 0_{M\times N}$
  for $k = 0$ to $K-1$ **do**
    $\text{loss} = \sum_{i=1}^{N} l(f(\mathbf{x} + D\mathbf{v}^{(i)}), y^{(i)})$
    $D^{(k+1)} = \text{Proj}_{\mathcal{D}}(D^{(k)} - \gamma_k \nabla_D \text{loss})$
    $V^{(k+1)} = \text{Proj}_{\mathcal{V}}(V^{(k)} - \gamma_k \nabla_V \text{loss})$
  end for
  return Adversarial dictionary $D^{(K)}$

Algorithm 4 White-box ADiL adversarial attack

Require: An optimized dictionary $D \in \mathcal{D}$, an unseen original example $\mathbf{x}$, norm budget ϵ, a step-size γ, a max number of iteration T
  $\mathbf{z} = \mathcal{N}(0_P, 1_P)$
  $D^\dagger = (D^\top D)^{-1} D^\top$
  for $t = 1$ to T **do**
    $\mathbf{z}' = \text{proj}_{\epsilon,p}(\mathbf{z})$
    $\mathbf{x}' = \text{proj}_{\mathcal{X}}(\mathbf{x} + DD^\dagger \mathbf{z}')$
    $\text{loss} = l(f(\mathbf{x}'), y^{(i)})$
    $\mathbf{z} = \mathbf{z} - \gamma \nabla_{\mathbf{z}} \text{loss}$
    if $\text{argmax}_k f_k(\mathbf{x} + DD^\dagger \mathbf{z}') \neq \text{argmax}_k f_k(\mathbf{x})$ **then**
      return Adversarial perturbation $\omega = DD^\dagger \mathbf{z}'$
    end if
  end for

the original manifold is expressed $\mathcal{X} = [0:1]^P$. We, therefore, set the projection into manifold operator as $\text{proj}_{\mathcal{X}}(\mathbf{x}) = \min(\max(\mathbf{x}, 1), 0)$, which clamps negative and above 1 adversarial examples pixels values to their bounds. Such projection function is not smooth and could potentially corrupt the optimization, but in practice, we found the ℓ_p norm ϵ budget constraint to be already too constraining. It organically enforces the pixels of the adversarial examples to be within the manifold.

ADiL in between specific and universal adversarial attacks

In this section, we further investigate how the revisited ADiL performs in a more challenging large-scale scenario such as ImageNet.

The revisited ADiL problem allows it to be fairly compared to other state-of-the-art adversarial attacks given the same ϵ adversarial budget. We now compare our approach with the baseline attacks on multiple pre-trained models on ImageNet from the TorchVision repository. Namely, we consider MobileNetV2, DenseNet121, InceptionV3, ResNet18, GoogleNet and VGG11, achieving accuracy of 71.88%, 74.43%, 69.54, 69.76%, 69.78% and 69.02%, respectively. Performance in terms of fooling rate (fr), mean squared error (mse) and time are reported in Table 2.1.

On the one hand, the found results once again shed light on the superiority of specific attacks (e.g. AutoAttack [37]) managing to achieve up to 100% fooling rate. However, such performance comes at a price of high computational cost, up to 10 times more than that required to perform universal attacks. The only two exceptions are FGSM [63] and its variant FFGSM [173], both being one-shot specific attacks. Still, their efficiency and low complexity come at the expense of the highest mse of any attacks.

On the other hand, universal attacks perform relatively well in large-scale experiments, mainly because of the many existing classes allowing them to easily find an adversarial label to target.

Nonetheless, their fooling rates are still significantly far behind those of specific attacks. Regarding ADiL, it displays competitive results at a much lower computational cost than specific attacks while also benefiting from low mse. Indeed, it performs at most 5 times faster than iterative specific attacks and still achieves relatively high fooling rates (e.g., 98.14% for ADiL-ce on MobileNet). In addition, ADiL always drastically improves upon universal attacks in terms of fooling for at most twice or three times their execution times and a comparable mse. Finally, it is unclear whether using the logit loss over the cross-entropy is beneficial since both yield equivalent results on average.

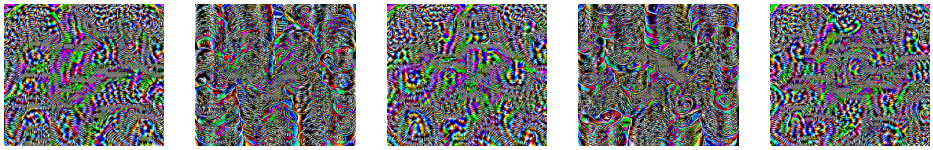
Table 2.1: Performance of ℓ_∞ attacks on ImageNet. Comparisons are drawn in terms of fooling rates (fr), mean squared error (mse), and the average time for evaluating the attack (in ms). Results are divided between specific (*top cell*), ADiL with M = 100 atoms (*middle cell*) and universal attacks (*bottom cell*).

	VGG			DenseNet			GoogleNet		
Attacks	fr	mse	time	fr	mse	time	fr	mse	time
APGD	99.96	95.41	689	99.99	71.67	526	**100**	**69.57**	283
AutoAttack	**100**	95.45	688	**100**	**71.63**	536	**100**	**69.57**	291
FGSM	97.84	144.43	**21**	94.16	144.35	**55**	91.96	144.38	51
FFGSM	98.76	114.83	**21**	96.43	114.67	**55**	94.00	114.71	51
MIFGSM	99.93	94.05	594	99.99	74.46	460	**100**	73.08	233
PGD	99.96	91.74	593	99.99	73.10	458	**100**	71.69	234
ADiL-ce	86.08	90.76	140	88.97	88.12	187	84.12	88.82	101
ADiL-logit	84.52	89.02	140	87.86	86.41	183	86.76	86.05	101
UAP-PGD	68.44	121.70	59	68.14	102.75	67	72.72	120.65	56
UAP	60.37	**84.62**	71	57.52	79.28	67	42.15	74.32	**48**
	Inc.V3			ResNet18			MobileNet		
Attacks	fr	mse	time	fr	mse	time	fr	mse	time
APGD	99.99	**66.47**	910	**100**	**77.23**	180	**100**	**69.31**	364
AutoAttack	**100**	66.49	913	**100**	**77.23**	180	**100**	**69.31**	365
FGSM	83.40	140.50	65	97.89	144.41	**45**	96.16	144.39	51
FFGSM	85.35	112.93	65	99.08	114.70	**45**	98.77	114.74	51
MIFGSM	99.99	68.51	899	**100**	79.03	171	**100**	73.24	354
PGD	99.99	68.53	899	**100**	77.38	170	**100**	71.78	352
ADiL-ce	86.16	75.41	162	92.46	89.38	88	98.14	85.25	86
ADiL-logit	87.75	78.16	139	89.79	87.78	89	96.69	83.43	86
UAP-PGD	52.27	96.10	66	70.85	107.20	50	91.66	106.27	**46**
UAP	18.33	55.44	**61**	65.37	89.84	53	85.44	91.02	50

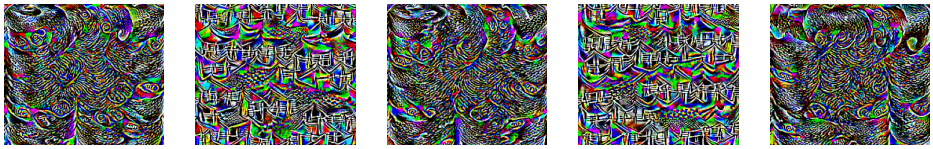
We display in Figure 2.5 a selection of 5 atoms of ADiL-100 learned on ImageNet for fooling either a MobileNet or a GoogleNet network. Observe that they all exhibit strong structured patterns reminiscent of those found in the UAP perturbation [142]. However, contrary to universal attacks which merge all individual perturbations into a single one, the proposed dictionary framework allows to split the individual contributions into multiple diverse atoms.

Inspecting the quality of the adversarial space modeling

In practice, when setting the norm constraint to ϵ, among all the adversarial attack baselines, all of their computed adversarial perturbations are of norm ϵ. In other words, to maximize the performances, adversarial attacks exploit all the possible space Ω_ϵ to find a maximum of adversarial perturbations to maximize the FR. Allowing the adversarial perturbations norm to reach the ϵ adversarial budget is key to being competitive with state-of-the-art adversarial attacks. In ADiL's



(a) MobileNet



(b) GoogleNet

Figure 2.5: Illustration of ADiL's atoms when setting M = 5 on ImageNet dataset for different classifiers, under the ℓ_∞ norm constraint.

case, rewriting the inference problem into equation (2.10) offers an efficient inference problem for which projected gradients are very well suited. However, it designs adversarial perturbations within the ϵ ball, not equal to the ϵ sphere.

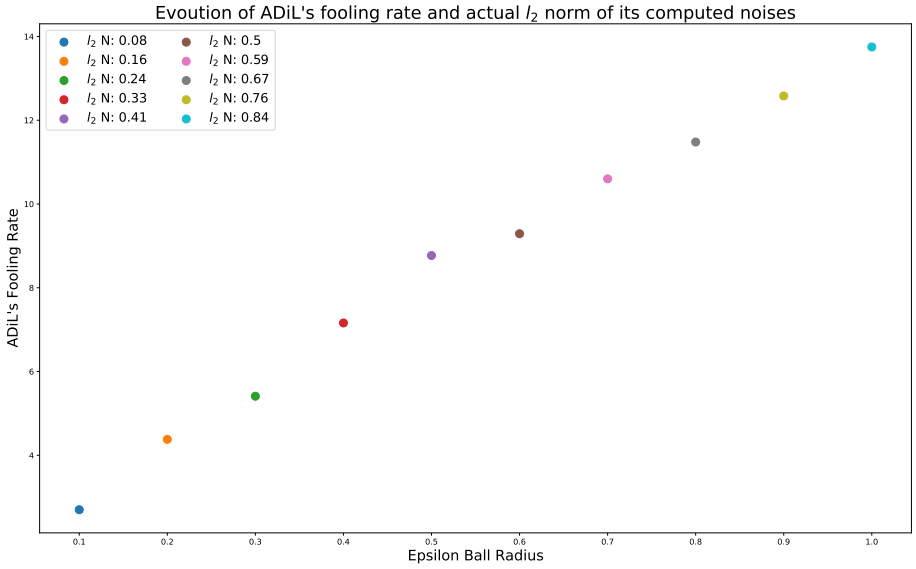


Figure 2.6: ADiL's crafted adversarial perturbation mean ℓ_2 norm with different ϵ norm constraint bound. The dictionary D is optimized on a VGG classifier with M = 50.

We inspected the inferred adversarial noises to better understand the crafted adversarial perturbations beneath a simple FR performance. We found that ADiL's crafted adversarial perturbations are far from the allowed ϵ norm bound. Figure 2.6 displays ADiL's crafted adversarial perturbation average ℓ_2 norm for various ϵ norm bounds.

Through Figure 2.6 we understand that the optimization process is flawed as it does not allow the adversarial perturbations to reach the ϵ norm budget. This artifact is not a flaw as every adversarial constraint is indeed respected. The adversarial perturbation generation is well designed according to the rules set for a fair comparison, but in practice, it prohibits ADiL from reaching state-of-the-art adversarial attack performances. We see that the gap between ADiL's crafted adversarial perturbation mean ℓ_2 norm and ϵ is not negligible, for ϵ = 1, the mean ℓ_2 norm of ADiL's adversarial perturbations is only 0.84 which means that the 16 most promising percentage of the allowed adversarial perturbations norms are unexploited. Indeed, this range is the most promising as obviously the higher the norm of the adversarial noises the more harmful the adversarial perturbation is the the more likely it will fool the classifier while respecting the other adversarial

constraints.

The ℓ_2 norm constraint allowed budget accepted by the community is $\epsilon = 0.5$. Every adversarial attack considered this bound as it allowed every adversarial attack to be fairly compared. In ADiL's case, when setting $\epsilon = 0.5$, we empirically observe that the crafted adversarial perturbations are of magnitude 0.41, which is too far from state-of-the-art adversarial attacks perturbations norm, always equal to $\epsilon = 0.5$, to let ADiL be competitive. From this observation we considered two solutions. First, we could align every adversarial attack to be of the same norm constraint bound as the one we empirically found with ADiL. Therefore in the latter example, we would set $\epsilon = 0.5$ for ADiL, find a mean adversarial norm of 0.41, and then compute the performances of the adversarial attack baselines with $\epsilon = 0.41$. This proposition has not been explored as it would be cheating, unfair to other baselines, and bad scientific behavior.

The other proposition was to compute adversarial attack baseline performances with $\epsilon = 0.5$ just like it is supposed to be, but in ADiL's case, as the inference problem is flawed, we allow a greater ϵ norm bound, making it reach adversarial perturbations norms of 0.5. In Figure 2.6, we empirically observe that when setting $\epsilon = 0.6$ we observe a mean ℓ_2 norm of ADiL's adversarial perturbation equal to 0.5. Therefore, the original adversarial constraints are respected and ADiL would be allowed to be competitive and "fairly" compared with other adversarial attack baselines. In practice, this idea is impossible to consider because it is "cheating" to dig into the inference perturbations and re-adjust the original problem to maximize the inference performances. Second, Figure 2.6 displays the average ℓ_2 norm of the ADiL's adversarial perturbations, meaning that some norms are higher, and some are lower than ϵ. However, higher perturbation norms are forbidden making this idea impossible in practice.

ADiL's projected gradient solution lacks an efficient inference problem allowing it to be competitive with state-of-the-art adversarial attacks while respecting all the adversarial constraints.

Changing the adversarial loss

In [24], multiple adversarial loss l have been proposed and explored for multiple ℓ_p norm budget constraints, but regarding the ADiL optimization framework, we assumed necessary to review different adversarial losses within (2.5) to see if there are significant changes or if the overall optimization is robust to the choice of the adversarial loss. We selected the two most used and most effective adversarial losses. We first ran experiments with the reverse cross-entropy adversarial loss that is, for an original example $\mathbf{x}$ with an original prediction $f(\mathbf{x}) = y$,

$$l\big(f(\mathbf{x}'), y\big) = -\mathcal{L}\big(f(\mathbf{x}'), y\big) = +\sum_{k=1}^{\mathrm{C}} o_k log f(\mathbf{x}')_k, \tag{2.11}$$

with $f(\mathbf{x}')_k$ representing the predicted probability of the label k for the adversarial example $\mathbf{x}'$ through f, and o a one-hot vector with 1 at index y representing the true predicted label of x. The goal of the reverse cross-entropy as the adversarial loss is to decrease as much as possible the original prediction. While the original objective of adversarial examples is to change the classifier's final output, the reverse cross-entropy only intends to lower the prediction of original labels such that other labels receive confidence, up until another label represents the maximum probability value over the possible labels, acting as the prediction shift.

While the reverse cross-entropy is an easy-to-use and straightforward adversarial loss, [24] proposed other losses with interesting properties. The reverse cross-entropy focuses on minimizing the original label prediction but does not intend to maximize any other targeted label prediction. The optimization of the reverse cross-entropy may not be optimal as it does not allow increasing other labels prediction. Carlini & Wagner however, proposed in [24] an adversarial loss capable of both minimizing the original label prediction and maximizing another adversarial label prediction. This loss is called the logit loss and is defined as, for an original example $\mathbf{x}$ with an original prediction $f(\mathbf{x}) = y$,

$$l(f(\mathbf{x}'), y) = \max(f_t(\mathbf{x}') - f_y(\mathbf{x}), -\kappa), \tag{2.12}$$

with κ a margin hyper-parameter and $t = \text{argmax}_{k \neq y} f_k(\mathbf{x})$ the second best label predicted for the original example, becoming the targeted adversarial label to predict.

The logit loss minimizes the original label prediction of the adversarial example and at the same time, maximizes the second-best label prediction of the adversarial example. Carlini & Wagner studied multiple adversarial losses including the reverse cross-entropy suggest relying on the logit loss for ℓ_2 norm bounded adversarial attacks. We assumed it was necessary to compare both adversarial losses, for both the ℓ_2 and ℓ_∞ norm constraints. Indeed, in our optimization framework, we do not have any insights on which is the best therefore, a comparison of both losses is needed.

Table 2.1 compares ADiL's performances to specific and universal adversarial attacks. First, it is very promising to see that even though ADiL does not reach state-of-the-art specific attack performances, ADiL is always a much stronger attack than universal baselines. These results place ADiL where we intended it to be, within both adversarial attack families and acting as a good trade-off between a transferable universal attack and a very effective and harmful specific attack. We also studied the two selected adversarial losses, the logit loss, and the reverse cross-entropy (rce) loss. As Table 2.1 shows, we do not see any differences between both losses. It means that the ADiL learning framework is robust to the chosen adversarial loss function, and promises to be effective for other tasks loss such as an object detection loss, or semantic segmentation loss. Such problems are yet to be studied, but the ADiL framework could easily be applied and promises to be effective with any other adversarial loss.

Time consumption analysis

Table 2.1 also compares the time consumption of the different adversarial attacks. We draw similar conclusions for the position of ADil's time consumption compared to other baselines. On the one hand, universal attacks compute adversarial examples very quickly as only one adversarial perturbation is optimized over all the original examples. This optimization averages to a tiny time consumption per example. On the other hand, specific attacks are very effective at finding adversarial perturbations for most of the original examples, but these optimizations are not free. Indeed, we see the best specific attack Autoattack, sometimes more than 10 times longer than universal attacks to compute all of the adversarial perturbations. It empirically shows a significant gap in the literature to be filled. As expected, ADiL perfectly fits in the middle of both families, computing adversarial examples always faster than specific attacks but longer than universal attacks. ADiL fastens the computation of adversarial perturbations by finding a vector $\mathbf{v} \in \mathbb{R}^{\mathrm{M}}$, composed of only $\mathrm{M} \ll \mathrm{P}$ atoms, drastically lowering the number of variables to optimize.

Black-box ADiL attack

Along with the white-box attack derived from the optimization of equation 2.9, we also tried to derive a black-box attack. Black-box attacks are reviewed and detailed in Section 1.2.4.

Having an ADiL black-box attack is more appealing than an ADiL white-box attack because it reflects real-world scenarios where an attacker has limited or no knowledge about the classifier in use. Besides black-box attacks are more challenging for defenders to detect and mitigate, making the ADiL black-box attack a very desired proposition.

Deriving a black-box attack means that we can optimize D using equation 2.5 in the same way, but at inference time, for new unseen examples $\mathbf{x}^{(i)}$, the corresponding coding vector $\mathbf{v}^{(i)}$ is optimized without the access of the internal parameters of the classifier f. Without any possibility to optimize the coding vectors $\mathbf{v}$ by gradient descent, we opted for a rejection sampling strategy, in which we try T sampling of $\mathbf{v}$ according to a certain probability distribution and stop the trials once a $\mathbf{v}$ produces an acceptable adversarial perturbation $\omega = \mathrm{D}\mathbf{v}$. This black-box attack is detailed in Algorithm 5.

In our experiments, we mainly focused on sampling the coding vector on the ℓ_1 norm sphere. As highlights line 3 and 4 of Algorithm 5, we sampled $\mathbf{v}$ from $\mathcal{P} = \{\mathbf{v} \sim \mathcal{N}(0_{\mathrm{M}}, 1_{\mathrm{M}}), ||\mathbf{v}||_1 = 1\}$. The

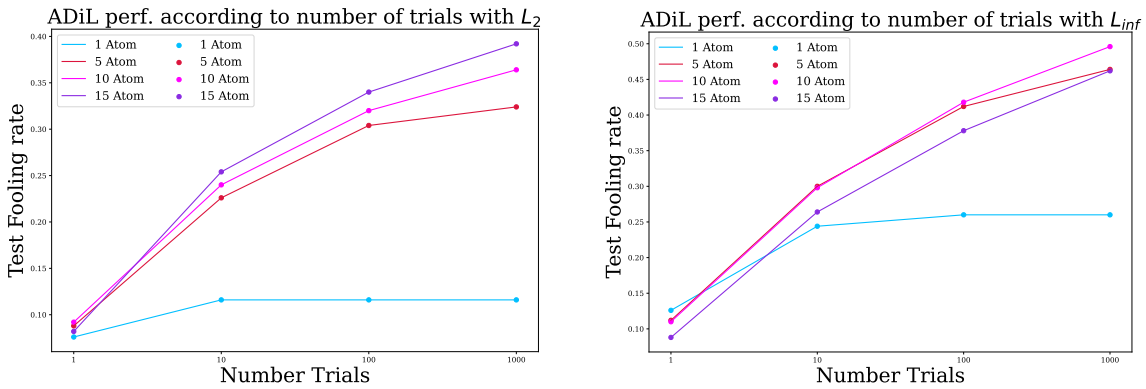
Algorithm 5 Black-box ADiL attack

Require: An optimized dictionary $D \in \mathcal{D}$, an unseen original example $\mathbf{x}$, a max trial number T
  for $t = 1$ to T **do**
    $\mathbf{v} \sim \mathcal{N}(0_M, 1_M)$
    $\mathbf{v} = \mathbf{v}/||\mathbf{v}||_1$
    if $\text{argmax}_k\, f_k(\mathbf{x} + D\mathbf{v}) \neq \text{argmax}_k\, f_k(\mathbf{x})$ **then**
      return Adversarial perturbation $\omega = D\mathbf{v}$
    end if
  end for

choice comes from our choice to respect the adversarial budget constraint of equation (2.5). Indeed, by modeling $D \in \mathcal{D}$, any coding vector vector $\mathbf{v}$ of a 1 ℓ_1 norm produces an adversarial perturbation $D\mathbf{v}^{(i)}$ within the ϵ ball. Therefore the inference sampling would likely be successful if the inference coding vectors do share the same shape.

Figure 2.7 illustrates the results of the black-box adversarial attacks derived from the optimization of equation (2.5). This figure shows the test performances of the black-box ADiL attack for various trial number limits. With M = 1 atom, the same conclusion is drawn as in Section 2.3. When the modeling space is too limited, no real conclusions can be drawn from a possible enhancement of the original version. However, with higher numbers of atoms $M = 5, 10, 15$, we see that the black-box ADiL attack performances increase according to the number of trials, which proves both the attacks potential and the sampling strategy ability to produce effective $\mathbf{v}$. The left figure shows a plateau is not far ahead regarding the ℓ_2 norm of the adversarial perturbations although, according to the ℓ_∞ norm constraint, the test performance increases almost linearly according to the number of trials.

Overall, deriving a black-box attack from the original proposition is a good idea as it derives a more realistic adversarial attack from the ADiL modeling.



(a) Considering the ℓ_2 norm adversarial budget constraint.

(b) Considering the ℓ_∞ norm adversarial budget constraint.

Figure 2.7: Black-box ADiL inference performances according to the number of trails for the sampling of $\mathbf{v}$ for various ℓ_p norm adversarial budget constraints of equation (2.9).

We also tried to rely on the optimized training coding vectors $V \in \mathcal{V}$ (2.7) of equation (2.5). Indeed, we tried some preliminary experiments in which we sampled the inference coding vectors $\mathbf{v}$ from a normal distribution of the optimized training coding vectors, but unfortunately, no better results were found.

ADiL adversarial defense

As the guideline of [158], we proposed, along with the ADiL adversarial attack, an ADiL adversarial defense. Defenses are mechanisms that make classifiers more robust to adversarial attacks. De-

fenses are reviewed and explained in Section 1.2.5. For the ADiL defense, we opted for an ADiL adversarial training defense. During the ADiL adversarial training defense, the classifier currently in training is optimized using original examples as well as ADiL-crafted adversarial examples. Feeding adversarial examples to the optimization of a classifier tends to push its decision boundaries away from the original data points, making the classifier's decision more robust to slight perturbations. However, to access ADiL adversarial attack, ADiL's parameters D need to be optimized as well. Therefore we chose, as a good trade-off to update ADiL's parameters every M epoch of the classifiers' optimization, making the ADiL adversarial attack always up-to-date with the classifier's decision boundaries while building it. ADiL defense is detailed in Algorithm 6.

Algorithm 6 ADiL Defense

Require: A classifier f parameterized by θ, a training set $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}$ a classification loss $\mathcal{L}$, a step-size γ, a maximum number of iteration K, number of epoch when optimizing ADiL's parameters M

for $k = 1$ to K **do**
  Updating D *to the current classifier* f *every* M *epoch*
  if mod(k,M) = 0 **then**
    Optimize D from current f and $\mathcal{D}$ by solving equation (2.5)
  end if
  for $(x^{(i)}, y^{(i)}) \in \mathcal{D}$ **do**
    Optimize on original data
    loss = $\mathcal{L}(f(\mathbf{x}^{(i)}), y^{(i)})$
    $\theta = \theta - \gamma \nabla_\theta loss$
    Optimize on adversarial data
    find $\mathbf{v}^{(i)}$ such that $\mathbf{x}^{(i)'} = \mathbf{x}^{(i)} + \mathrm{D}\mathbf{v}^{(i)}$ is adversarial to f
    loss = $\mathcal{L}(f(\mathbf{x}^{(i)'}), y^{(i)})$
    $\theta = \theta - \gamma \nabla_\theta loss$
  end for
end for
return Robustly trained classifier f

As Table 2.2 and Figure 2.8 show, training a classifier with the proposed defense including a dictionary of M atoms, not only makes it more robust to ADiL attacks of M atoms but also ADiL attacks of any number of atoms. Indeed, the robustness of the classifier is hugely improved against dictionaries that are composed of fewer or as many atoms as the dictionary used to train the classifier, but in a lesser manner, the robustness is also improved against dictionary attacks containing more atoms than the number used to train the classifier. One of the main weaknesses of adversarial training defenses is that when relying on a single adversarial attack to add its produced adversarial examples to the training set of the classifier, the produced classifier is indeed more robust, but only to the adversarial attack used to create the training adversarial examples. By relying on an ADiL attack to augment the training dataset, our empirical proofs, Table 2.2 and Figure 2.8 show that the defense mechanism profits from the modeling of the adversarial noise space, letting users build classifiers that are more robust to not a single bu any adversarial attacks. Indeed, we see classifiers more robust to similar or simpler adversarial attacks (ADiL with less or as many atoms), but also more complex adversarial attacks (ADiL with more atoms).

Optimization configuration

In this new implementation of the adversarial dictionary problem, because we all have the same ϵ adversarial budget, we can be fairly compared and benchmarked to other adversarial attacks. However, almost every adversarial attack includes either hyper-parameters or parameters that need to be fine-tuned, so that every attack reaches its best performance. In our case, because ADiL includes the modeling of the adversarial noise space, we need to fine-tune plenty of critical

$M_{attacker}$	2 atoms	5 atoms	10 atoms	15 atoms	20 atoms
Without Defense	25.78%	56.25%	60.15%	46.09%	57.81%
With Defense	**15.62%**	**30.46%**	**53.90%**	**44.53%**	**56.25%**

Table 2.2: Study of the proposed defense mechanism for LeNet trained on CIFAR-10. By hinging on the proposed defense with $M_{defense}$ = 10 atoms, we report the fooling rate of ADiL attacks for various $M_{attacker}$ atoms.

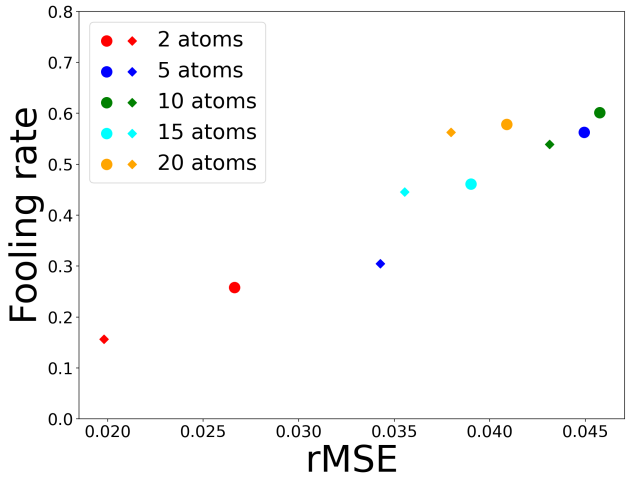


Figure 2.8: Fooling Rate of ADiL with different number of atoms on a LeNet classifier. The dots represent the ADiL attack on a standard LeNet classifier, while the diamonds display fooling rates on a robust LeNet classifier trained with the defense. Fooling rate values are precisely reported in Table 2.2.

parameters that could jeopardize the performance of the attack if not carefully tuned. As well as maximizing the performances of ADiL's attack, finding the best ADiL parameters on a given set of experiments allows us to bring future users a manual, or at least insights, on the right parameters according to their applications. In this section, we review the most critical optimization parameters and evaluate their sensitivity to the correctness of ADiL's modeling and attack performances.

Optimizer

We previously saw in Section 1.1.3 that the optimizer is a critical parameter to the optimization process. We recall that an optimizer is simply the implementation of the strategy to find the right optimization step size at each iteration. Recent works [136] showed that a slight change of optimizer could enhance or decrease the performances by far, therefore we conducted an intensive analysis on which optimizer would be best for the ADiL modeling problem, that is optimizing D and V over the training set.

Regarding the optimization of ADiL, we were looking for an optimizer that is both efficient and robust, as many other parameters are free, such as the number of atoms M, the norm used within the adversarial constraints, the number of training points N and the complexity of the problem at hand. In the case of ADiL, the robustness of the optimizer is as important as the efficiency in reaching the best possible solution Therefore we selected three optimizers, Adam [86], Adam Weighted (AdamW) [101] and SGD [151] that are widely used and enough different to see if ADiL's optimization is agnostic to the choice of optimizer or if there is one optimizer that is prominently better than the others. Because we needed the optimizer to be robust in the task it could handle well, we selected the three mainly used optimizers and were not interested in some lesser-used ones as we do not have any long-term trust and reliability over those.

To make ADiL easy to use for future users we were looking for only one optimizer that meets

our requirements for both the ℓ_2 and ℓ_∞ norm constraints.

Figure 2.9 displays the evolution of the training loss and the relative norm of the parameters D and V that is evaluated as $\frac{\text{norm}(D^{t-1})-\text{norm}(D^t)}{\text{norm}(D^{t-1}-D^t)} + \frac{\text{norm}(V^{t-1})-\text{norm}(V^t)}{\text{norm}(V^{t-1}-V^t)}$, at the iteration t. These two metrics allow us to assess the minimization of the training loss and the convergence towards a minimum. In Figure 2.9, it is clear that for both ℓ_2 and ℓ_∞, the SGD optimizer is a bit too basic compared to other optimizers. Indeed, with the ℓ_2 norm, the SGD optimizer does not reach the best minima and is shown to be too unstable when using the ℓ_∞ norm. Between Adam and Adam Weighted, the choice is open to debate. Indeed, neither exceeds the other in terms of loss minimization or robustness to the initial step size. To achieve a good balance between the optimizer performances and descent robustness to the choice of the learning rate, we chose to carry out our experiments with the Adam optimizer.

Now that we selected the Adam optimizer as the right tool to use under both the ℓ_2 and ℓ_∞ norm, we must select the best initial learning rate, possibly one for each norm. Indeed, Figure 2.9 highlights the need for a different learning rate for each norm. Thus we conducted an initial learning rate experiment comparison under the Adam optimizer for both the ℓ_2 and ℓ_∞ norm.

Learning rate

Even though we purposefully select the Adam optimizer as the most efficient and robust optimizer, the learning rate remains a sensitive parameter. Therefore we conducted experiments with the Adam optimizer paired with different learning rates to select the best setup, hopefully, the same for both the ℓ_2 and ℓ_∞ norm. Figure 2.10 shows us the evolution of the training loss and the relative norm of ADiL's parameters D and V, which is computed the same as in the Optimizer experiments. We understand through Figure 2.10 that a different learning rate must be used according to the ℓ_p norm constraint used. This choice is mainly drawn from the Adam-0.1 curve that could be a decent choice for the ℓ_∞ norm constraint, compared to ℓ_2 norm constraint for which 0.1 is a really bad choice showcasing an under-fitting. According to the different curves, we selected using Figure 2.10a the learning rate for ℓ_∞ norm constraint to be 0.05 and for ℓ_2 norm constraint with Figure using Figure 2.10b to be 0.001. Indeed, with the two values, the training loss is decreasing by far from the starting point and reaches a plateau while displaying a real convergence regarding the parameter's relative norm.

Now we crafted an optimization setup that allows us to exploit ADiL's potential to the fullest for both the ℓ_∞ and ℓ_2 norm constraints. The optimization of Algorithm 3 under this setting produces dictionaries D shown in Figure 2.11.

Figure 2.11 presents the result of the ADiL optimization procedure under the best hyperparameters. The atoms are visually interesting and allow us to unveil the common patterns involved in the crafting of adversarial perturbations. Note that absolutely no constraint on the "readability" of the atoms was included in the optimization process but we still end up with these visual atoms. Possessing such visual atoms allows us to understand the roots of the adversarial perturbations, which are to some extent the universal pitfalls of the neural network classifiers, according to which we can infer new adversarial perturbations. It is worth noting that the atoms differ a lot according to the ℓ_p norm constraint used. Indeed, on the one hand, Figure 2.11a shows a dictionary optimized using the ℓ_∞ norm constraint, and we see that the atoms capture the corners and edges of the objects to allow computing adversarial perturbations based on those. On the other hand, Figure 2.11b presents atoms optimized under the ℓ_2 norm constraint. We see the atoms captured some random noise allowing us to craft adversarial perturbations that are less "understandable" but still universally effective at fooling the classifier.

It is desired for these atoms to be visually available for inspection, as for explaining to a non-expert audience, the origin, and computation of the adversarial perturbations become very clear and easy to understand. These atoms allow us to investigate deeper into the origin of the adversarial noise so we can incorporate those within a defense mechanism producing more robust classifiers. The proposed ADiL adversarial defense is detailed in Section 2.4.1.

Figure 2.11 shows the result of the training procedure, but for new unseen original exam-

ple, the computation of their adversarial perturbations from the learned D needs another hyper-parameter to tune, that is the inference learning rate.

Inference learning rate

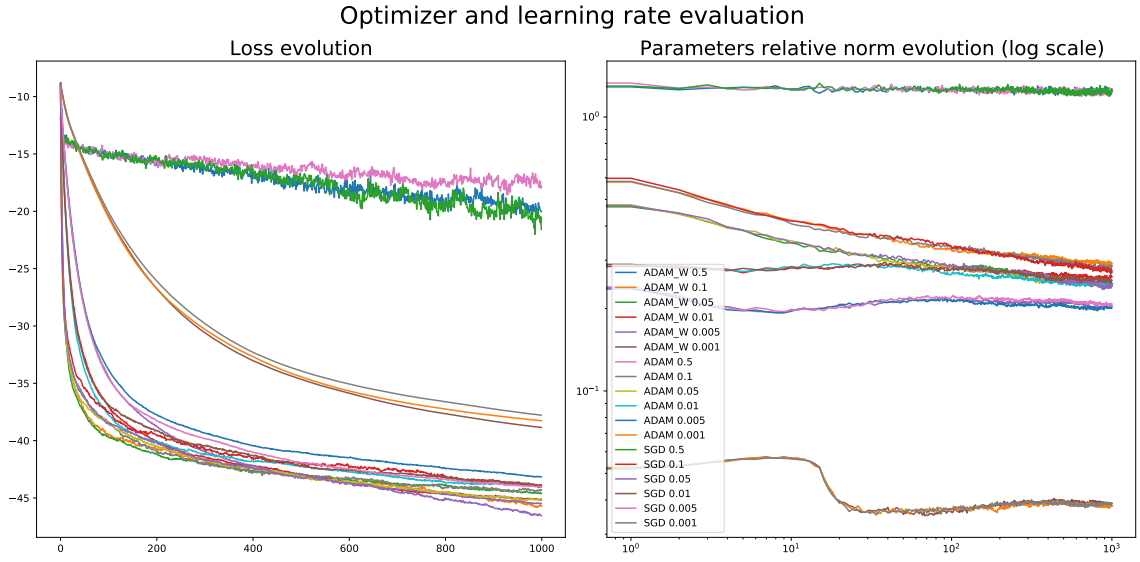
As for the inference problem, we assumed the Adam optimizer to be a decent choice as we previously sanity-checked its efficiency and robustness for the training procedure. However, we could not make the same assumption with the learning rate as the number of variables to optimize drastically changes. Indeed, the inference problem only optimizes the coding vector $\mathbf{v} \in \mathbb{R}^M$ such that $\omega = D\mathbf{v}$ is an effective adversarial perturbation, conditioned to the learned D.

The number of variables to optimize crucially influences the choice of the learning rate therefore, we deep analyzed its impact.

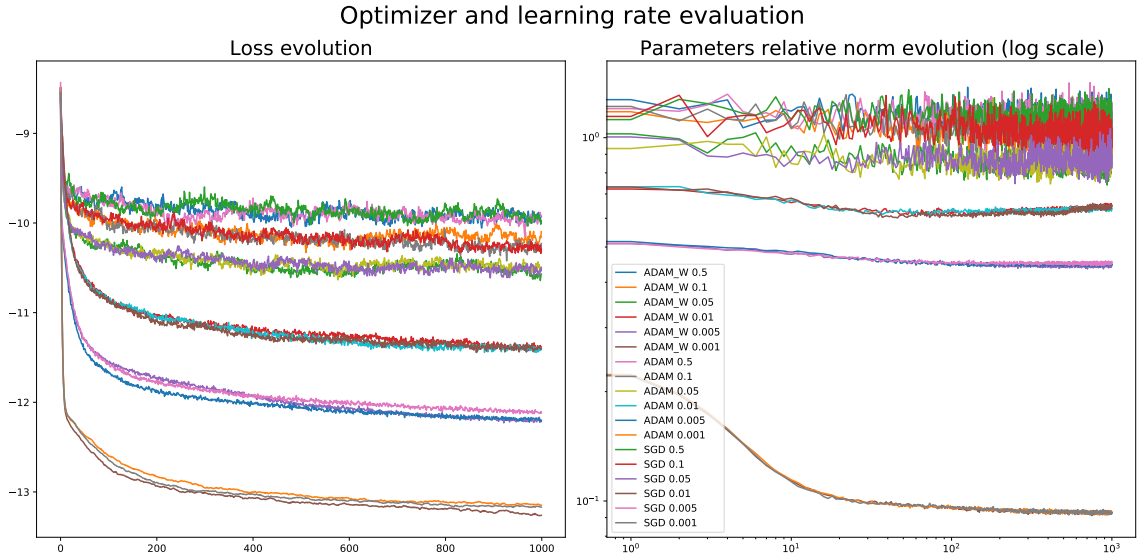
Figure 2.12 displays the evolution of the inference loss along with the final FR reached at the end of the inference optimization. Those two metrics are our only key metrics allowing us to assess a good optimization. The same conclusions can be drawn regarding the inference learning rate, it largely impacts the optimization process and therefore needs to be carefully tuned to ensure good inference performances. Through Figure 2.12b and 2.12a we understand that a different inference learning rate must be used depending the ℓ_p norm constraints used. From Figure 2.12b we selected for the ℓ_2 norm constraints, an inference learning rate of 0.01 which reaches the best FR and visually displays the best optimization behavior, with a plateau for the inference original example for which no adversarial perturbations could be found. As for the ℓ_∞ norm constraints, we select through Figure 2.12a an inference learning rate of 0.005 because of the same reasons, the best FR is reached, while displaying a descent loss minimization.

While the latter ADiL projected gradient solution is interesting and led to good results, the full potential of the idea is not quite reached, as a-posteriori, we observe that the training adversarial noises norm is quite far from the ϵ limit, $||\omega^{(i)}||_p = ||D\mathbf{v}^{(i)}||_p \ll \epsilon$ (see Section 2.4.1 for more details). This artifact limits the modeling of the adversarial noise space by D and prevents the overall attack from reaching its full potential.

Even though it is very promising and theoretically appealing, the empirical results did not match our expectations, and therefore ADiL has only been a stepping stone in the process leading to LIMANS.

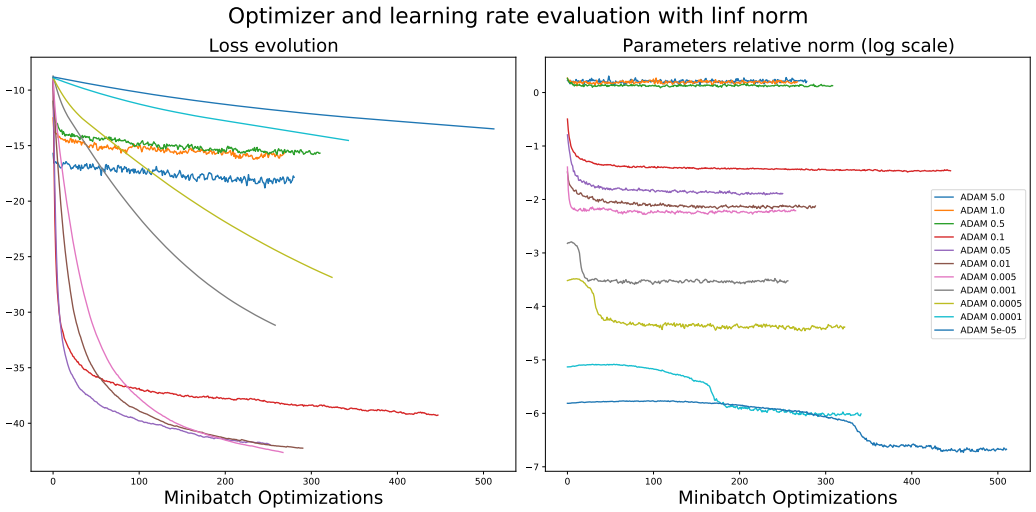


(a) Considering the ℓ_∞ norm adversarial budget constraint.

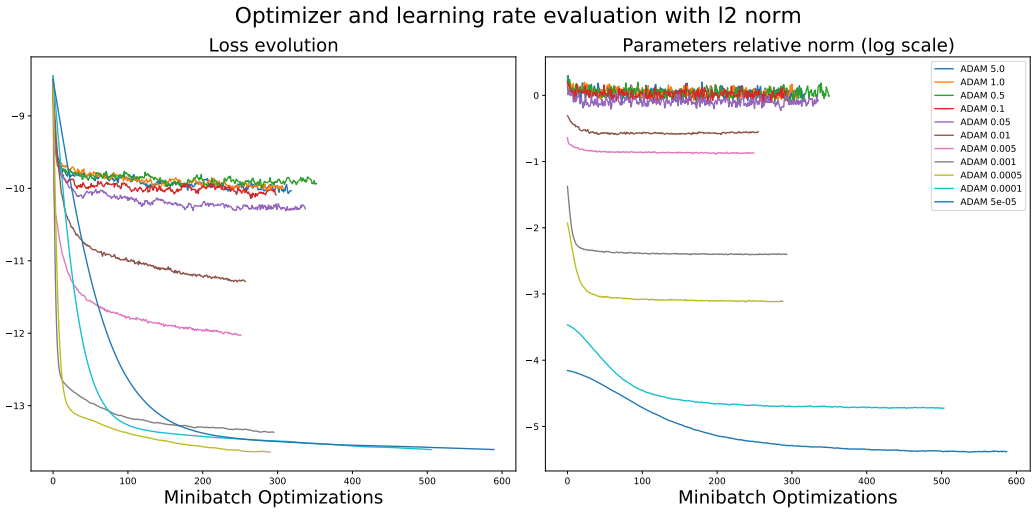


(b) Considering the ℓ_2 norm adversarial budget constraint.

Figure 2.9: Evolution of ADiL's training loss and the relative norm of D and V that is evaluated as $\frac{\text{norm}(\mathrm{D}^{t-1})-\text{norm}(\mathrm{D}^t)}{\text{norm}(\mathrm{D}^{t-1}-\mathrm{D}^t)} + \frac{\text{norm}(\mathrm{V}^{t-1})-\text{norm}(\mathrm{V}^t)}{\text{norm}(\mathrm{V}^{t-1}-\mathrm{V}^t)}$, at the iteration t, with M = 100 during the training procedure, according to different optimizers (ADAM, ADAM-W and SGD) for multiple initial step-sizes and both ℓ_2 and ℓ_∞ norm constraints.

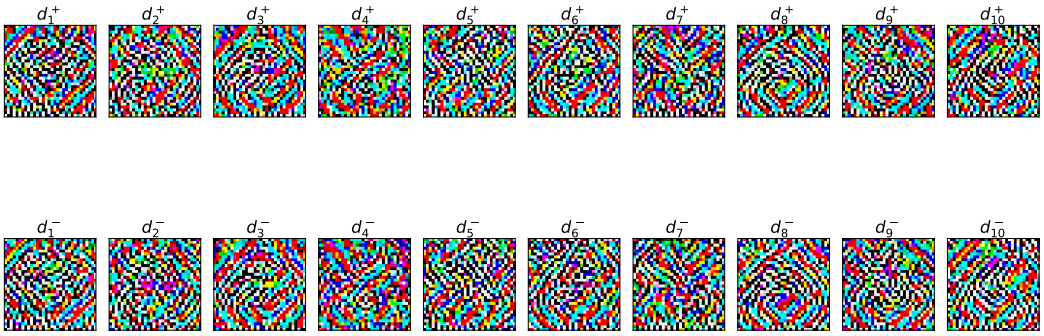


(a) Considering the ℓ_∞ norm adversarial budget constraint.

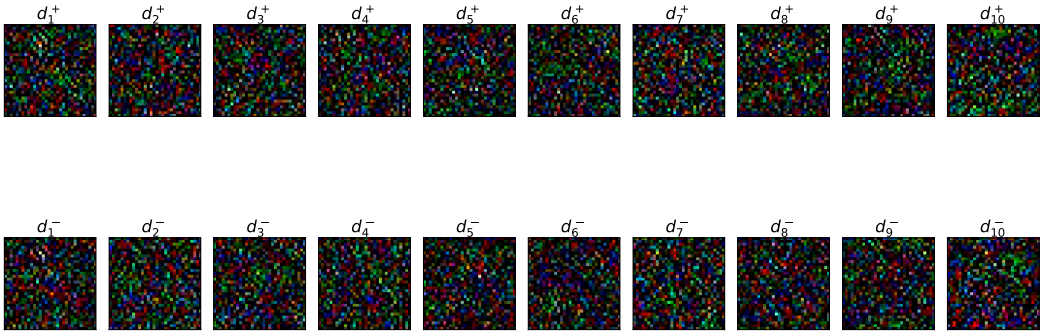


(b) Considering the ℓ_2 norm adversarial budget constraint.

Figure 2.10: Evolution of ADiL's training loss and the relative norm of D and V that is evaluated as $\frac{\text{norm}(\text{D}^{t-1})-\text{norm}(\text{D}^t)}{\text{norm}(\text{D}^{t-1}-\text{D}^t)} + \frac{\text{norm}(\text{V}^{t-1})-\text{norm}(\text{V}^t)}{\text{norm}(\text{V}^{t-1}-\text{V}^t)}$, at the iteration t, with M = 100 during the training procedure, using the Adam optimizer couple with various learning rates for both ℓ_2 and ℓ_∞ norm constraints.

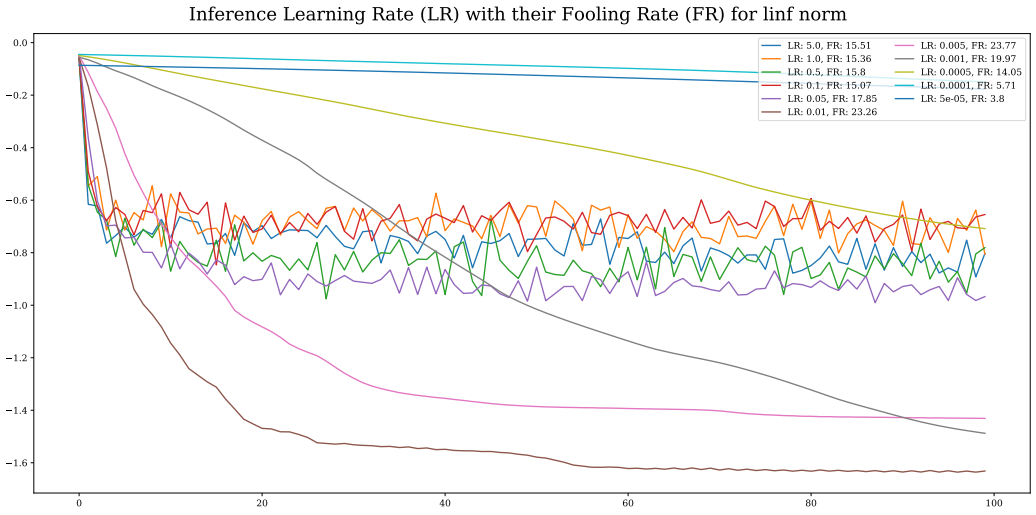


(a) Considering the ℓ_∞ norm adversarial budget constraint.

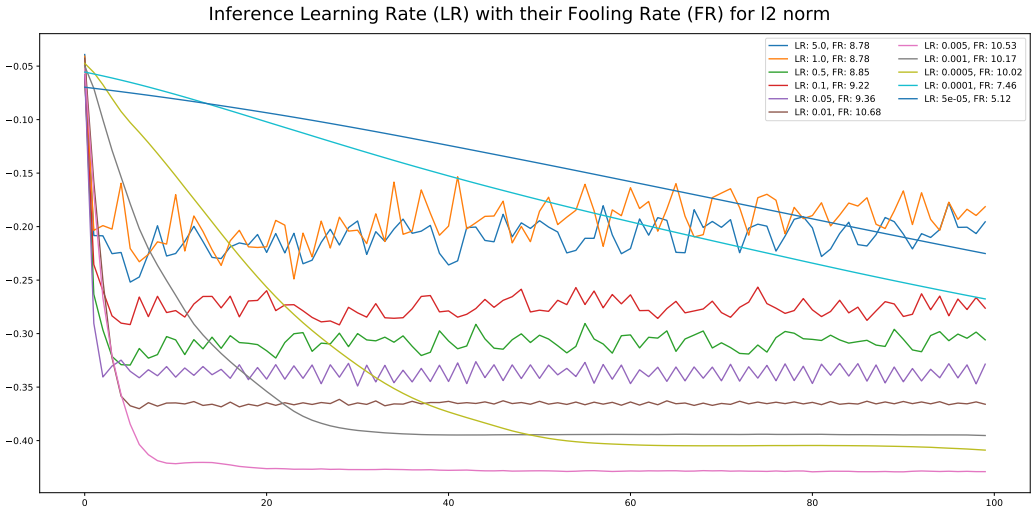


(b) Considering the ℓ_2 norm adversarial budget constraint.

Figure 2.11: Example of D solution with M = 10 found by optimizing Algorithm 3 with the optimization setting proposed in Section Optimizer and Learning rate, for both the ℓ_∞ and ℓ_2 norm constraint.



(a) Considering the ℓ_∞ norm adversarial budget constraint.



(b) Considering the ℓ_2 norm adversarial budget constraint.

Figure 2.12: Evolution of ADiL's inference loss with M = 100, using the Adam optimizer coupled with various inference learning rates for both ℓ_2 and ℓ_∞ norm constraints.

2.5 LIMANS stochastic gradient solution

After a fruitful discussion with prestigious researcher Nicholas Carlini[1], there was a two-sides goal regarding the improvements of ADiL. First, given a budget ϵ, the adversarial noises must exploit this freedom to the fullest, meaning the adversarial noises ℓ_p norm should be close or equal to ϵ, to model the adversarial noise space efficiently. Second, an empirical analysis of the model space should be emphasized as if well modeled, the transfer property of universal attacks should hold using the modeling, allowing an adversarial attack to be both efficient to specific adversarial noise crafting and transferable to fooling other classifiers g than the one through which D has been optimized. Therefore, the original assumption of problem (2.1), that dictionary learning tools are relevant to the modelization of the adversarial noise space, is now questioned. Indeed, the sparsity constraint on the coding vectors is not quite fundamental and should therefore be removed to lower the bias during the optimization. It led to deeply questioning our initial assumptions with ADiL and pivoting toward a new lecture on the problem, engaging other optimization solutions such as stochastic gradients.

To solve the original problem (2.1), we proposed two relaxation optimizations, namely Regularized-LIMANS, a stochastic gradient descent optimization enforcing the constraints through regularization, and Simple-LIMANS, a plain loss function incorporating all the constraints and optimizing both D and V at the same time in a deep learning spirit. A high-level overview of the proposed LIMANS adversarial attack is given in Figure 2.13.

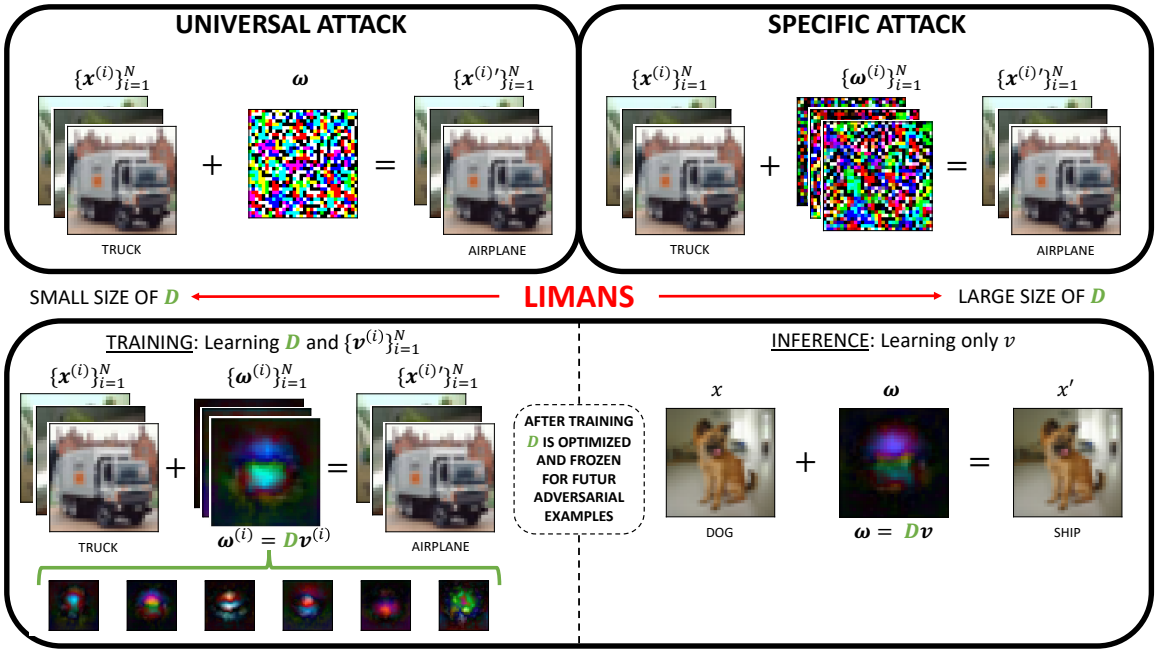


Figure 2.13: A high-level overview of the proposed LIMANS adversarial attack and its optimization. It is highlighted the adversarial model D is universal to every adversarial example, while a specific coding vector v is crafted for each adversarial example.

2.5.1 Regularized-LIMANS solver

In the original problem (2.1), the indicator function $\mathbb{1}$ being non-smooth and highly non-convex makes the optimization untractable. Instead of maximizing the fooling rate, it is usual to consider minimizing a surrogate adversarial loss function $L_\gamma(f(\mathbf{x}'), f(\mathbf{x}))$, parameterized by γ, more amenable to optimization. A typical adversarial loss function of interest is

[1] https://nicholas.carlini.com/

$$l_\gamma(f(\mathbf{x}'), f(\mathbf{x})) = \max\big(-\gamma,\ f_c(\mathbf{x}') - \max_{k\neq c} f_k(\mathbf{x}')\big), \tag{2.13}$$

Still, the problem is hard to solve because of the non-convexity of the constraint $||\mathrm{D}\mathbf{v}^{(i)}||_p \leq \delta_p$ or the account of the constraint $\mathbf{x}^{(i)'} \in \mathcal{X}$. Hence, for computational tractability, we propose to rely instead on the following regularized relaxation problem,

$$\min_{\substack{\mathrm{D}\in\mathcal{D}\\ \mathrm{V}\in\mathbb{R}^{\mathrm{M}\times\mathrm{N}}}} \sum_{i=1}^{\mathrm{N}} l_\gamma(f(\mathbf{x}^{(i)} + \mathrm{D}\mathbf{v}^{(i)}), f(\mathbf{x}^{(i)})) + \lambda h_{(\delta_p,p)}(\mathrm{D}, \mathbf{v}^{(i)}) \quad s.t.\ \mathrm{D}\in\mathcal{D}, \tag{2.14}$$

with $\lambda \in \mathbf{R}_+$ a regularisation parameter, $\mathcal{D} = \{\mathrm{D}\in\mathbb{R}^{\mathrm{P}\times\mathrm{M}}, ||\mathrm{D}_j||_p = 1, \forall\, j\in\{1,\dots,\mathrm{M}\}\}$ and $h_{(\delta_p,p)}$ representing a penalty function. We consider the ℓ_p-norm, with $p=2$ or $p=\infty$, as penalty function leading to $h_{(\delta_2,2)}(\mathrm{D},\mathbf{v}) = \max(||\mathrm{D}\mathbf{v}||_2 - \delta_2, 0)$ and $h_{(\delta_\infty,\infty)}(\mathrm{D},\mathbf{v}) = \sum_k \max(|(\mathrm{D}\mathbf{v})_k| - \delta_\infty, 0)$.

Enforcing a constraint through regularization is one of the most common ways, and the most obvious when considering instead the optimization of a Lagrangian formulation. Therefore, problem (2.14) made total sense as the most natural regularized optimization of the original problem (2.1). Note that the constraint $\mathbf{x}^{(i)'} \in \mathcal{X}$ is not integrated into this formulation and therefore has to be dealt with post-processing. We proposed to optimize (2.1) with a stochastic gradient descent optimization. Algorithm 7 sketches its optimization as well as the post-processing ensuring $\mathbf{x}^{(i)'} \in \mathcal{X}$.

Algorithm 7 Regularized-LIMANS

Require: Classifier f; Learning rate ρ; Training dataset $\mathcal{T}$; ℓ_p budget δ_p; Optimizer Optim; Batch size B; Regularization parameter λ

$\mathrm{D} = \mathcal{N}(0, \mathbb{1}_{\mathrm{M}\times\mathrm{P}})$; $\mathrm{V} = \mathcal{N}(0, \mathbb{1}_{\mathrm{P}\times\mathrm{M}})$
for k = 0 to MAXEPOCH **do**
  loss = 0
  for $(\mathbf{x}^{(i)}, y^{(i)}) \subset \mathcal{T}$ **do**
    // Compute loss_i from Problem (2.14)
    $\mathbf{x}^{(i)'} = \mathbf{x}^{(i)} + \mathrm{D}\mathbf{v}^{(i)}$
    $\hat{y}_{adv} = f(\mathbf{x}^{(i)'})$; $\hat{y} = f(\mathbf{x}^{(i)})$
    $\mathrm{loss}_i = \mathcal{L}_0(\hat{y}_{adv}, \hat{y}) + \lambda h_{(\delta_p,p)}(\mathrm{D}, \mathbf{v}^{(i)})$
    loss = loss + loss_i
    if modulo(i) = B **then**
      $\mathrm{D} \leftarrow \mathrm{Optim}(\nabla_{\mathrm{D}}\mathrm{loss})$ (Update)
      $\mathrm{V} \leftarrow \mathrm{Optim}(\nabla_{\mathrm{V}}\mathrm{loss})$ (Update)
      $\mathrm{D} = \mathrm{Proj}_{\{\mathrm{D}\ |\ \|\mathrm{D}\|_p=1\}}(\mathrm{D})$
      loss = 0
    end if
  end for
end for
$\mathrm{D}\mathbf{v}^{(i)} \leftarrow \mathrm{Proj}_{\{\mathrm{D}\mathbf{v}\ |\ \|\mathrm{D}\mathbf{v}\|_p\leq\delta\}}(\mathrm{D}\mathbf{v}^{(i)})$
$\mathbf{x}^{(i)'} \leftarrow \mathrm{Proj}_{\mathcal{X}}(\mathbf{x}^{(i)} + \mathrm{D}\mathbf{v}^{(i)})$
return $\{\mathbf{x}^{(i)'}\}_{i=1}^{\mathrm{N}}$, (D, V)

Here, we get rid of the constraints $\mathbf{x}^{(i)'} \in \mathcal{X}, \forall\, i$ and enforce small magnitude of $||\mathrm{D}\mathbf{v}^{(i)}||_p$ through the regularizer $h_{(\epsilon_p,p)}$. Empirically this promotes $\mathbf{x}^{(i)} + \mathrm{D}\mathbf{v}^{(i)} \in \mathcal{X}$ to be nearly close to $\mathcal{X}$.

The Regularized-LIMANS optimizes (D,V) in a stochastic fashion, and specifically, D is updated using a projected gradient descent that ensures that the constraints $||\mathrm{D}_j||_p = 1, \forall\, j$ are satisfied. A grid search on λ allows to control of the generalization of the model. In practice, for the selected value of λ, the constraint on $\|\mathrm{D}\mathbf{v}\|_p$ might slightly be violated. If needed a post-processing is performed to ensure the respect of the constraint.

2.5.2 Simple-LIMANS solver

The Regularized-LIMANS requires the cumbersome tuning of the hyper-parameter λ. To alleviate that, we propose a second relaxation of LIMANS that involves an objective function encompassing two of the constraints, the last one being taken care of by post-processing. Specifically, the method termed Simple-LIMANS solves the following problem,

$$\min_{\substack{\mathrm{D}\in\mathbb{R}^{\mathrm{P}\times\mathrm{M}}\\ \mathrm{V}\in\mathbb{R}^{\mathrm{M}\times\mathrm{N}}}} \sum_{i=1}^{\mathrm{N}} l(f(\mathrm{proj}_{\mathcal{X}}(\mathbf{x}^{(i)} + \frac{\epsilon_p \mathrm{D}\mathbf{v}^{(i)} + \mathbf{b}}{\|\mathrm{D}\mathbf{v}^{(i)} + \mathbf{b}\|_p})), f(\mathbf{x}^{(i)})) \tag{2.15}$$

where $\mathrm{proj}_{\mathcal{X}}$ denotes the projection operator that maps its input $\mathbf{x}^{(i)'} = \mathbf{x}^{(i)} + \mathrm{D}\mathbf{v}^{(i)} + \mathbf{b}$ onto $\mathcal{X}$ and $\mathbf{b}$ an offset variable.

Note that the Simple-LIMANS adds a new variables $\mathbf{b}$ to he definition of the adversarial perturbation, that is $\omega^{(i)} = \mathrm{D}\mathbf{v}^{(i)} + \mathbf{b}$. This offset is entirely universal to every adversarial perturbation and, just like in an affine function, it allows to uplift of every adversarial perturbation whatever the value of $\mathbf{v}$. In a way the variable $\mathbf{b}$ can be seen as a rewriting of both D and $\mathbf{v}$ as $[\mathrm{D}, \mathbf{b}]^{\mathrm{T}}$ and $[\mathbf{v}, 1]^{\mathrm{T}}$. Such relaxation of the variables is commonly used within linear classifiers to allow greater flexibility in decision boundary optimization. The role and necessity of the offset $\mathbf{b}$ is questioned and analyzed in Section 2.6.7. To ease the reading and facilitate the comparison between Simple-LIMANS and Regularized-LIMANS we will consider $\mathbf{b}$ to be part D as its relaxation and therefore omit to mention it when unneeded to avoid confusion.

Simple-LIMANS trades off the constraint $\mathrm{D} \in \mathcal{D}$, *i.e.* the unit norm constraint over the atoms of D, for the explicit guarantee that the adversarial example $\mathbf{x}^{(i)'}$ is valid (i.e. belongs to $\mathcal{X}$) and that the adversarial noise is utmost of magnitude ϵ_p by defining $\mathbf{x}_i^{(i)'}$ as: $\mathbf{x}^{(i)'} = \mathrm{proj}_{\mathcal{X}}(\mathbf{x}^{(i)} + \frac{\epsilon_p \mathrm{D}\mathbf{v}^{(i)}}{\|\mathrm{D}\mathbf{v}^{(i)}\|_p})$. Here $\mathrm{proj}_{\mathcal{X}}$ is the projection operator onto $\mathcal{X}$. Simple-LIMANS solves (2.15) by iteratively updating D and V using a gradient descent procedure as shown in Algorithm 8. It proves computationally efficient as it does not require hyperparameter tuning. At termination, a post-processing is used to ensure $\mathrm{D} \in \mathcal{D}$ without changing the adversarial examples by rescaling V accordingly.

Due to their nature, both Regularized-LIMANS and Simple-LIMANS' stochastic optimization apply to large-scale datasets.

2.5.3 Inference optimization

At inference time, provided that D is known, we seek for an unseen example $\mathbf{x}^{(k)}$, its adversarial counterpart $\mathbf{x}'^{(k)} = \mathbf{x} + \mathrm{D}\mathbf{v}^{(k)}$. For both proposed regularization, the original inference problem (2.2) is relaxed in the same way, by optimizing (2.14) and (2.15) only over $\mathbf{v}^{(k)}$. Therefore at inference time, the complexity of the adversarial example crafting is on $\mathrm{M} \leq \mathrm{P}$ which significantly improves the complexity of the adversarial example problem. Once D is known, for a new unseen original example $\mathbf{x}^{(k)}$, both original relaxation optimization problem boil down to,

$$\min_{\mathbf{v}^{(k)}\in\mathbb{R}^{\mathrm{M}}} l(f(\mathrm{proj}_{\mathcal{X}}(\mathbf{x}^{(k)} + \frac{\epsilon_p \mathrm{D}\mathbf{v}^{(k)} + \mathbf{b}}{\|\mathrm{D}\mathbf{v} + \mathbf{b}\|_p})), f(\mathbf{x}^{(k)})), \tag{2.16}$$

for the Simple-LIMANS algorithm and to,

$$\min_{\mathbf{v}^{(k)}\in\mathbb{R}^{\mathrm{M}}} l_{\gamma_2}(f(\mathbf{x}^{(k)} + \mathrm{D}\mathbf{v}^{(k)}), f(\mathbf{x}^{(k)})) + \lambda h_{(\delta_p,p)}(\mathrm{D}, \mathbf{v}^{(k)}). \tag{2.17}$$

for the Regularized-LIMANS algorithm. Note that for the Regularized-LIMANS algorithm, there is still the γ_2 hyper-parameter to tune, which is not the same as γ previously used for the optimization of both D and V during the training stage.

Similar to the training of both D and V, the above problems (2.16) and (2.17) can both be solved directly by gradient descent as in Algorithm 8 and Algorithm 7, but here with the D fixed, considered as a parameter of the optimization.

Algorithm 8 Simple-LIMANS

Require: Classifier f; Learning rate ρ; Training dataset $\mathcal{T}$; ℓ_p budget δ_p; Optimizer Optim; Batch size B

D = $\mathcal{N}(0, \mathbb{1}_{M \times P})$; V = $\mathcal{N}(0, \mathbb{1}_{P \times M})$
for k = 0 to MAXEPOCH **do**
  loss = 0
  for $(\mathbf{x}^{(i)}, y^{(i)}) \subset \mathcal{T}$ **do**
    // Compute loss_i from Problem (2.15)
    $\text{noise}^{(i)} = D\mathbf{v}^{(i)}$
    $\mathbf{x}^{(i)'} = \text{proj}_{\mathcal{X}}(\mathbf{x}^{(i)} + \frac{\delta_p \text{noise}^{(i)}}{\|\text{noise}^{(i)}\|_p})$
    $\hat{y}_{adv} = f(\mathbf{x}^{(i)'})$; $\hat{y} = f(\mathbf{x}^{(i)})$
    $\text{loss}_i = \mathcal{L}_\infty(\hat{y}_{adv}, \hat{y})$
    loss = loss + loss_i
    if modulo(i) = B **then**
      D $\leftarrow$ Optim$(\nabla_D \text{loss})$ (Update)
      V $\leftarrow$ Optim$(\nabla_V \text{loss})$ (Update)
      loss = 0
    end if
  end for
end for
V $\leftarrow$ $[||D_{\bullet j}||_p V_{j\bullet}]$ $\forall j \in \{1, \ldots, M\}$
D $\leftarrow$ $\text{Proj}_{\mathcal{D}}(D)$
return $\{\mathbf{x}^{(i)'}\}_{i=1}^{N}$, (D, V)

All these properties make both relaxations very appealing as solutions to the original problem by theoretically breaking the complexity of the adversarial attack problem.

2.6 LIMANS results

This section presents the experimental evaluations of the adversarial noise space and adversarial perturbations generated with it, providing a comparison with the state-of-the-art attacks on benchmark datasets.

Our experiments are conducted on the three datasets: MNIST, CIFAR-10, and ImageNet all presented in Section 1.1.6. As suggested in [182], we perform the experiments only on the validation set and split it into three parts, the first set for training the model D, the second for the tuning of λ when using Regularized-LIMANS and the last one for testing.

MNIST Experiments. The number of original examples for training, validation, and testing is respectively 8000, 1000, and 1000. The experiments of the proposed model and the robustness estimation were conducted on the trained LeNet classifier [94] achieving a validation accuracy higher than 98.8%. These experiments have been implemented in Pytorch on a MacBook Pro with 2,3 GHz Intel Core i9, 8 cores, and a GPU Nvidia RTX 2080.

CIFAR-10 Experiments. The number of original examples for training, validation, and testing is respectively 8000, 1000, and 1000. The experiments of the proposed model and the robustness estimation were conducted on the pre-trained VGG11 with batch normalization and the robust ResNet-18 [139] classifier. The transferability of the proposed model has been evaluated over four vanilla DNNs, i.e., MobileNet-V2, ResNet50, DenseNet121 and the VGG11 as aforementioned, and two robust DNNs, robust ResNet-18[2] and robust WideResNet-34-10 (R-wr-34-10) [2] [139]. These experiments have been implemented in Pytorch on a MacBook Pro with 2,3 GHz Intel Core i9, 8 cores, and a GPU Nvidia RTX 2080.

ImageNet Experiments. The number of original examples for training, validation, and testing is respectively 10000, 2000, and 5000. We select here the 4 vanilla classifiers, ResNet-18, MobileNet-V2, DenseNet121, and VGG11 and two robust classifiers available on RobustBench[2], robust ResNet-18 and robust WideResNet-50 [135] (that we respectively rename as R-r18 and R-wr50 for simplicity). The experiments on a large-scale dataset were performed on a server equipped with 4 GPU Volta V100-SXM2-32GB.

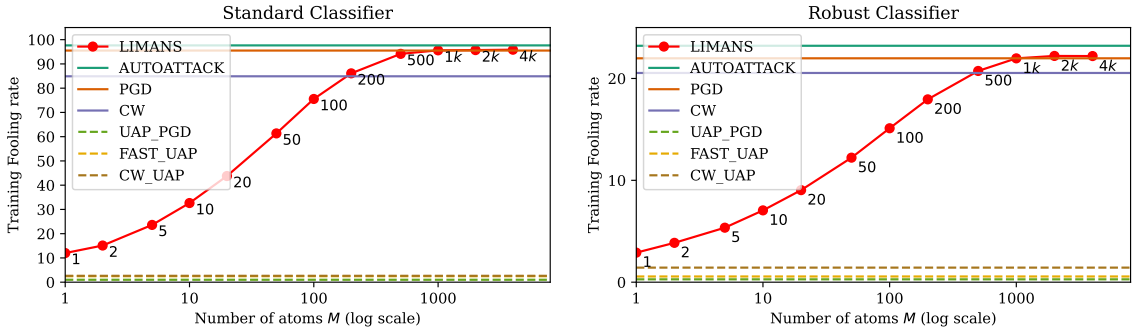
2.6.1 Bridging the gap between specific and universal attacks

With LIMANS, we had better chances to reach a good modeling of the adversarial noise space, than the one that resulted from the ADiL projected gradient solution. Indeed, by relying on projection functions, the adversarial budget is always reached (sometimes overcome with Regularized-LIMANS), allowing D to capture the most relevant information to fool the classifiers.

To check the good modeling of the adversarial noise space, we solved the original problem (2.1) with the fast and reliable Simple-LIMANS solver to check that by increasing the size of the modeling (the number of atoms), its performances increased, sanity checking the overall proposition. Figure 2.14 displays Simple-LIMANS solutions for various numbers of atoms ranging from 1 to 4000.

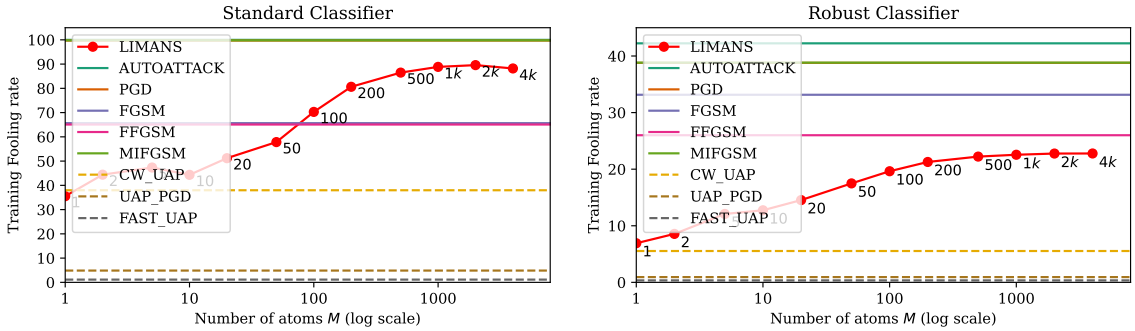
The four figures highlight that when setting M = 1, LIMANS always reach a D solution spanning adversarial noises more effectively than universal attack solutions. This point is essential to sanity-checking the good modeling of the adversarial noise space. Indeed, by having both the $\mathbf{b}$ offset relaxation and the $\mathbf{v}^{(i)}$ coefficient to the universal D, LIMANS is always supposed to be better than the universal attacks solution. This necessary condition is indeed met, by a large margin for a Standard classifier and a smaller margin for a Robust classifier, which is natural because of the more complicated decision boundaries, making universal adversarial perturbations even harder to find. When setting the number of atoms to the input dimension M = P, a solution D exists being the basis of R^{P}, from which all possible adversarial noises can be computed. Therefore, a second necessary condition of the LIMANS good modeling is that when setting M = P, LIMANS performances should get very close to state-of-the-art specific adversarial attacks.

[2] https://robustbench.github.io/



(a) ℓ_2 norm with standard classifier.

(b) ℓ_2 norm with robust classifier.



(c) ℓ_∞ norm with standard classifier.

(d) ℓ_∞ norm with robust classifier.

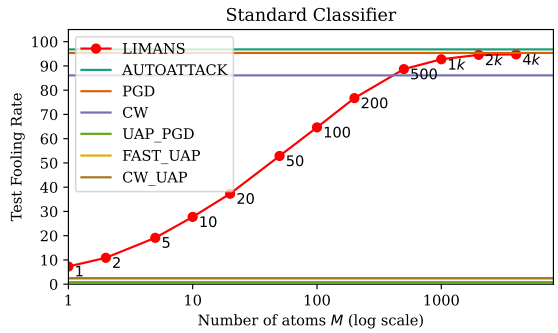
Figure 2.14: Training fooling rate of adversarial attacks under the ℓ_2 and ℓ_∞ norm constraint on CIFAR-10 when fixing several atoms M (x-axis), associated to the classifier (left) VGG11 and (right) robust ResNet-18, with the Simple-LIMANS solver.

As we can see in Figure 2.14a and 2.14b, when applied on both a Standard and a Robustly trained deep classifier VGG11, the LIMANS ℓ_2 performances do reach the state-of-the-art specific adversarial attack baselines performances when M = P $\approx$ 4000. However, for the ℓ_∞ norm, things are a bit more sensitive. Indeed, we see a significant gap with the state-of-the-art specific attacks even when M = P. This bias comes from our Algorithm 8, which does not look for a solution D as a basis of $\mathbb{R}^M$, thus introducing a bias when the atoms are juxtaposing, preventing to possibility to reach all possible specific adversarial perturbation from D. Empirically when considering the ℓ_∞ norm, all atoms found after training are juxtaposing, confirming the introduced bias and explaining the gap found with state-of-the-art specific attacks, whereas, with the ℓ_2 norm, the atoms do not look alike. These atoms are displayed and analyzed in Section 2.6.2.

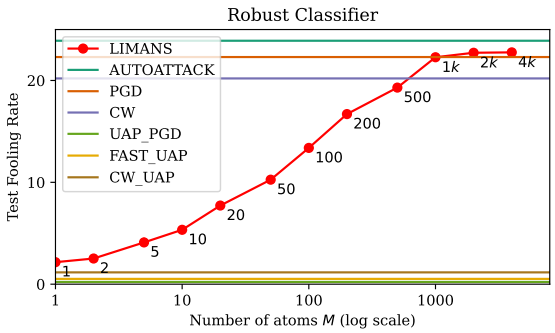
From these conclusions, we understand that the optimization problem is well-defined and empirical modeling results are acceptable. However, training performances do not assess a fair comparison with the state-of-the-art baselines, test performances only do. Therefore we used these D solutions found in Figure 2.14 and solved its paired inference problem (2.16) on the test set to fairly compare LIMANS with the state-of-the-art baselines.

Figure 2.15 tracks the test fooling rate of LIMANS for different numbers of the adversarial atoms under the ℓ_2 and ℓ_∞ norm constraint. It shows that the LIMANS attack is always stronger than universal baselines as even for only a single atom M = 1, LIMANS allows tuning the $\mathbf{v}$ coefficient factor making it more efficient. We see that from M = 500, LIMANS closes the gap with state-of-the-art specific adversarial attacks. This result is interesting as it empirically shows that by tuning the number of atoms M, the proposed **LIMANS does bridge the gap between specific and universal adversarial attacks**.

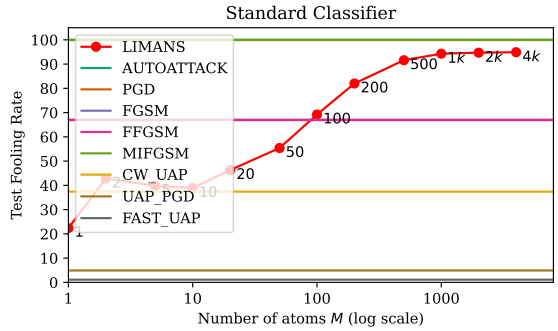
Besides reaching specific attack performances, by setting M = 500, the LIMANS attacks achieve competitive results by only optimizing for a coding vector $\mathbf{v}$ of dimension 500 whereas specific attacks need to optimize 3072 variables. Furthermore, such a result confirms the manifold hy-



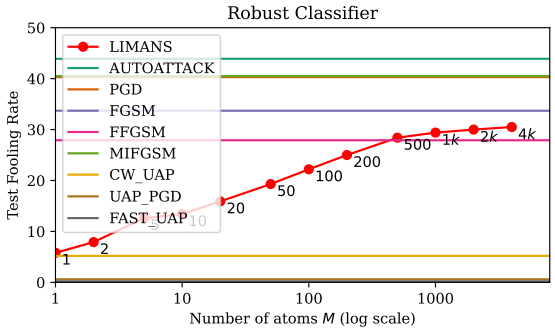
(a) ℓ_2 norm with standard classifier.



(b) ℓ_2 norm with robust classifier.



(c) ℓ_∞ norm with standard classifier.



(d) ℓ_∞ norm with robust classifier.

Figure 2.15: Test fooling rate of adversarial attacks under the ℓ_2 and ℓ_∞ norm constraint on CIFAR-10 when fixing several atoms M (x-axis), associated to the classifier (left) VGG11 and (right) robust ResNet-18, with the Simple-LIMANS solver.

pothesis of the adversarial noise space thus confirming the efficiency of the quick, parameter-free Simple-LIMANS solver. By relying on a linear LIMANS model, users are offered the to visually inspect the adversarial model's parameters. Figure 2.16 displays the optimized model's atoms when M = 5. It shows how the atoms, the roots of LIMANS's adversarial leaks, are structured. This structure differs according to the classifier and the considered ℓ_p norm. In particular, the dictionary of LIMANS-ℓ_2 on the robust classifier is reminiscent of certain Fourier decompositions.

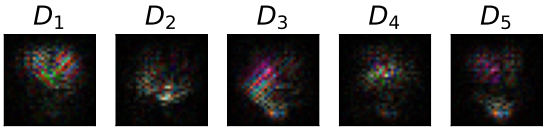
2.6.2 Visually interpretable atoms

In this section, we inspect the LIMANS parameters (atoms of D) found after the training stage and empirically observe interesting patterns that we detail. We give an empirical review of the three mainly used datasets, that is CIFAR-10, ImageNet, and MNIST.

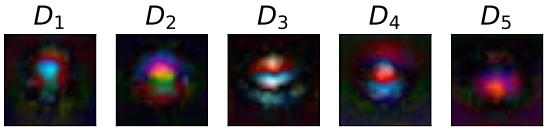
CIFAR-10 dataset

The CIFAR-10 dataset is a good dataset for analyzing the performances of adversarial perturbations as it is a decently complex dataset appropriately representing the real world by offering RGB images of 10 labels each very different from one another. Therefore the claim that the atoms are visually interpretable must be at least proven with this dataset.

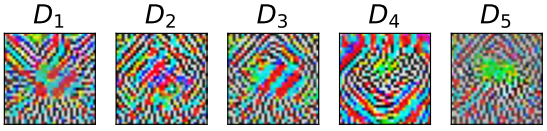
By modeling the adversarial noise space, we empirically observe that LIMANS' parameters capture the most meaningful information fooling the classifier. Figure 2.16 displays the optimized LIMANS' atoms when M = 5, and presents interesting results. Figure 2.16a and Figure 2.16b display LIMANS parameters for a Standard and a Robust classifier under the ℓ_2 norm constraint. As we can see, under the ℓ_2 norm constraint, the atoms do not look alike at all and target locally interesting spots within the images, such that specific adversarial perturbations can be computed,



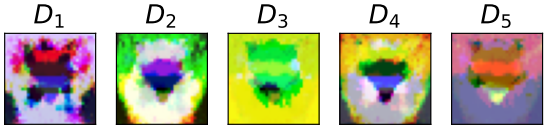
(a) LIMANS-ℓ_2 on Standard classifier.



(b) LIMANS-ℓ_2 on Robust classifier.



(c) LIMANS-ℓ_∞ on Standard classifier.



(d) LIMANS-ℓ_∞ on Robust classifier.

Figure 2.16: Visualization of the learned universal adversarial directions (atoms of the dictionary D) when M = 5, on CIFAR-10 and corresponding to the classifier (left) VGG11 and (right) robust ResNet-18. All atoms are rescaled for display.

taking advantage of these "pitfalls" for which the classifier is the weakest. Figure 2.16a and Figure 2.16d however, consider the ℓ_∞ norm constraint. As previously mentioned, all atoms display the classifier's vulnerabilities for every pixel value, preventing the overall D from being a basis of $\mathbb{R}^P$ and thus limiting the number of possible specific adversarial perturbations that can be computed from. It is interesting, however, to see that even if this modeling of the adversarial noise space is not optimal, the atoms are nevertheless interesting to inspect. Indeed, we see that for the Standard classifier, the atoms capture the corners and edges of the objects, which are the most discriminating features used by the classifier to make its prediction.

In some way, we can read the LIMANS' atoms as a capture of the semantics of the objects within the classification space. Figure 2.16 highlights very particular atoms that indeed spotlight recurring patterns in classification such as edges and corners for the ℓ_∞ atoms and local spots in the images for ℓ_2 atoms.

MNIST dataset

For atom visualization only, we also considered experiments on the MNIST dataset. Indeed, the MNIST dataset offers black-and-white images with pixels in very small manifold values, therefore making the problem too simple for an adversarial perturbation study with deep classifiers. Very few works in the adversarial community target the MNIST dataset. In our case, since we only want to analyze the parameters found after the training, it still is worth looking at the result.

Along with Figure 2.16, we also show that the same conclusions can be reached for simpler datasets such as MNIST with Figure 2.17,

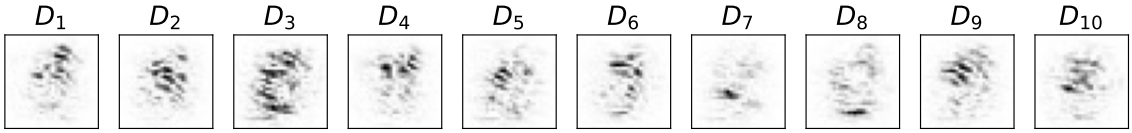
In addition to these visually interesting universal parameters, this empirical observation is further proven when inspecting the LIMANS produced specific adversarial perturbations.

Figure 2.18 displays LIMANS-specific adversarial perturbations on the MNIST dataset (purposely used to ease the observations). LIMANS produces much more interesting adversarial perturbations than state-of-the-art specific adversarial perturbations that are random. LIMANS however, target the most sensitive spots to fool the classifiers, which makes LIMANS adversarial attack a much more realistic attack than state-of-the-art specific adversarial attacks.

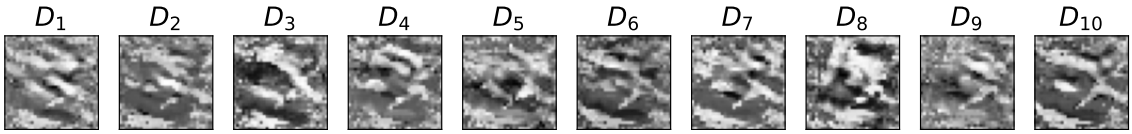
2.6.3 A more robust adversarial attack

In this Section, we observe that LIMANS offers a more robust adversarial attack than state-of-the-art specific attacks. Table 2.3 and 2.4 present the RAUD of LIMANS and state-of-the-art specific adversarial attacks on the CIFAR-10 dataset under respectively the ℓ_∞ and ℓ_2 norm constraint. More details about the RAUD metric are provided in Section 1.2.3.

Table 2.3 and Table 2.4 shows the performances of the robustness of the proposed ℓ_∞ and ℓ_2

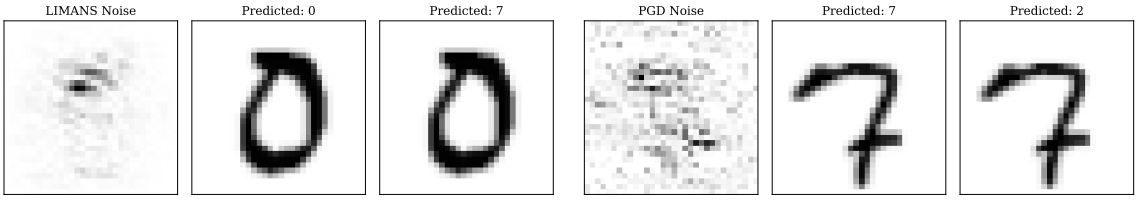


(a) LIMANS-ℓ_2.



(b) LIMANS-ℓ_∞.

Figure 2.17: Visualization of the learned universal adversarial directions (atoms of the dictionary D) when M = 10, on MNIST according to both the ℓ_2 and ℓ_∞ norm targetting a LeNet classifier achieving more than 98.8% of test accuracy. All atoms have been rescaled for display.



(a) LIMANS adversarial perturbation and example.

(b) PGD adversarial perturbation and example.

Figure 2.18: Examples of ℓ_2 adversarial perturbations produced by LIMANS and PGD on the MNIST dataset for a LeNet classifier achieving more than 98.8% of test accuracy.

LIMANS attack and its comparison to the specific adversarial attack baselines targetting a Standard and a Robust classifier. These experiments are conducted with the same settings as in [100].

Lorenz et al. [100] highlights that state-of-the-art specific attacks become nearly or completely harmless when the classifier is protected by an adversarial examples detector. Under the ℓ_∞ norm constraint, we find similar results and, even the proposed LIMANS attack surpasses specific attacks when M is only 10.

With M $\geq$ 500, the proposed LIMANS attack can successfully jeopardize the classical classification system, even equipped with an attack detector. Using a detector before a classification system is one kind of adversarial defense, presented in Section 1.2.5. The robust classifier shows its stability facing adversarial attacks while the proposed LIMANS attack can also ignore the attack detectors and damage the system to some extent. It indicates thus the potential application of the proposed attacks in evaluating the effectiveness of an adversarial example detector plugged prior in a classical or robust classifier.

To a lesser extent, we draw similar conclusions for the ℓ_2 norm constraint. Indeed, when the classifier is protected by a specific attack detector (detector trained to recognize a specific attack's adversarial examples), LIMANS is almost always a more harmful attack. These conclusions are questioned when the detector is trained to recognize LIMANS' adversarial examples. Indeed, when doing so, LIMANS' performances are never better than state-of-the-art specific attacks. However, Table 2.4 tells us that using a LIMANS detector is a poor defense choice. Indeed, before considering any defense detector, Figure 2.15 highlights the natural superiority of specific attacks, which inherently leads to consider a specific attack detector as the most effective defense. Even when using a LIMANS detector, specific attacks RAUD is lowered empirically showing the LIMANS framework is relevant either as an adversarial attack or a defense mechanism.

Under the ℓ_2 norm constraint, using a LIMANS detector will indeed protect from LIMANS adversarial examples, but also from state-of-the-art specific attacks, which indirectly showcases the

Table 2.3: Robustness performance of the LIMANS ℓ_∞ attack ($\epsilon = 8/255$) in terms of RAUD on the CIFAR-10 test data and against the attack detectors plugged in both standard classifier (S.C.) and robust classifier (R.C.). The shown performances are averaged over 5 random seeds. **The smaller the RAUD, the more robust the adversarial attack is**. The best performances are marked in bold.

Detectors d	d_{PGD}		$d_{Autoattack}$		$d_{LIMANS_{10}}$	
Classifiers f	S.C.	R.C	S.C.	R.C	S.C.	R.C
SA	91.1	85.1	91.1	85.1	91.1	85.1
FGSM	*91.1* ± 0.0	*85.1* ± 0.0	*91.1* ± 0	*85.1* ± 0.0	89.3 ± 0.0	79.5 ± 0.0
PGD	*91.0* ± 0.0	84.9 ± 0.1	*91.0* ± 0.0	*85.0* ± 0.0	80.7 ± 0.5	73.3 ± 0.3
Autoattack	91.0 ± 0.0	85.0 ± 0.0	*91.1* ± 0.0	85.0 ± 0.0	78.2 ± 0.3	71.5 ± 0.3
$LIMANS_{10}$	78.3 ± 2.4	79.6 ± 0.0	81.7 ± 2.0	8.0 ± 0.0	86.4 ± 1.6	79.5 ± 0.1
$LIMANS_{500}$	26.3 ± 0.3	71.3 ± 0.1	32.1 ± 0.4	73.2 ± 0.2	36.5 ± 3.1	69.4 ± 0.2
$LIMANS_{1000}$	24.7 ± 0.9	70.4 ± 0.2	31.6 ± 1.1	72.4 ± 0.1	36.9 ± 4.6	68.5 ± 0.2
$LIMANS_{4000}$	**23.7 ± 0.5**	**69.8 ± 0.0**	**30.8 ± 0.9**	**72.9 ± 0.3**	**35.6 ± 2.2**	**68.2 ± 0.1**

robustness of the LIMANS framework.

Table 2.4: Robustness performance of the LIMANS ℓ_2 attack ($\epsilon = 0.5$) in terms of RAUD on the CIFAR-10 test data and against the attack detectors plugged in both standard classifier (S.C.) and robust classifier (R.C.). The shown performances are averaged over 5 random seeds. **The smaller the RAUD, the more robust the adversarial attack is**. The best performance is marked in bold font.

Detectors d	d_{PGD}		$d_{Autoattack}$		$d_{LIMANS_{10}}$	
Classifiers f	S.C.	R.C	S.C.	R.C	S.C.	R.C
SA	91.1	85.1	91.1	85.1	91.1	85.1
PGD	64.0 ± 1.2	82.4 ± 0.0	65.3 ± 0.6	81.8 ± 0.1	42.9 ± 0.6	80.1 ± 0.0
Autoattack	63.1 ± 1.1	82.19 ± 0.1	65.8 ± 0.6	**81.1 ± 0.2**	**42.1 ± 0.3**	**79.4 ± 0.0**
$LIMANS_{10}$	83.5 ± 0.5	87.4 ± 0.1	82.9 ± 0.4	88.1 ± 0.0	86.9 ± 0.9	87.6 ± 0.1
$LIMANS_{500}$	62.9 ± 0.2	84.0 ± 0.4	64.7 ± 1.0	83.8 ± 0.3	51.7 ± 1.3	81.7 ± 0.1
$LIMANS_{1000}$	63.7 ± 0.6	82.7 ± 0.3	63.6 ± 0.9	82.6 ± 0.2	46.8 ± 0.7	80.1 ± 0.1
$LIMANS_{4000}$	**62.6 ± 1.2**	**82.16 ± 0.2**	**62.2 ± 0.6**	82.6 ± 0.3	46.9 ± 0.2	80.0 ± 0.2

Details about the RAUD detector d

Regarding the design of a descent adversarial examples detector, we followed the guidelines proposed by [99] and [67]. In these two works, authors propose to use a random forest binary classifier with at least 100 trees, in the Fourier domain either of the input images or of the Fourier Features of Feature-Maps from the images. Authors showed that by using either one of the inputs, the random forest binary classifier can discriminate with high precision adversarial examples computed from state-of-the-art specific adversarial attacks on several complex datasets such as CIFAR10, CIFAR100, ImageNet, or Celeba. To lower as much as possible the bias introduced by the detector, we chose to use a random forest binary classifier with 300 trees in the Fourier domain of the input images. Our choice has been confirmed by extensive experiments and is highlighted in Table 2.5. Indeed, Table 2.5 displays the confusion matrix of the detectors involved in the results presented in Table 2.3.

The detectors have been trained on the same training dataset as the one used in the training of LIMANS and, the displayed values computed over the validation dataset. Finally, the RAUD metric of the different adversarial attacks using the detector is performed on the test dataset which none has previously seen before, making everything fair.

We empirically observe highly effective detectors discriminating almost perfectly real images rather than adversarial images, without mistaking one or the other by producing low False-Positive and False-Negative values. These performances gave us confidence in the use of these detectors as

Table 2.5: Confusion matrices of the detectors used to compute the RAUD of Table 2.3. All detectors have been trained on the same training dataset as the one used in the training of LIMANS and the displayed values of computed over the validation dataset, such that fair performances are considered. TN: True Negative, FP: False Positive, FN: False Negative, TP: True Positive.

Detectors d			d_{FGSM}		d_{PGD}		$d_{Autoattack}$		$d_{LIMANS_{10}}$	
Confusion Matrix	TN	FP	863	57	814	106	790	130	846	74
	FN	TP	0	920	0	920	0	920	95	825

a tool of the RAUD metric, evaluating both the harmfulness and the transferability of adversarial attacks, which are essential to be measured when assessing the quality of an adversarial attack.

2.6.4 Transfer performances

The results in this part were generated using the algorithm Regularized-LIMANS.

We present a comparison with the specific ℓ_∞ adversarial attacks AutoAttack and three state-of-the-art transferable ℓ_∞ attacks that are, VNI-FGSM [171], NAA [183], and RAP [125]. Besides, we consider also Translation-Invariant Attack (TI-FGSM) [50], VMI-FGSM [171], and universal ℓ_∞ attacks UAP and UAPPGD. However, the well-known DI-FGSM, MI-FGSM, or NI-FGSM are not selected to compare, as all these processes are already integrated into the RAP attack. The LIMANS ℓ_2 attack is compared with the ℓ_2 version of AutoAttack, UAP, UAPPGD, and RAP, as well as the CW adversarial attack.

By adjusting the hyper-parameter λ of the Regularized-LIMANS solver, we managed to find an optimized M = 150 for CIFAR-10, which achieves almost comparable performance with AutoAttack and to analyze the transferability of the learned space without loss of precision as shown in Table 2.6. With the ImageNet dataset (Table 2.8), considering the large size of an image and the memory limitation, this dimension becomes M = 100. It is far from the dimensionality of the full adversarial noise space, and hence, does not lead to comparable performance with specific attacks such as AutoAttack. However, it still offers evidence for the transferability of the learned space on ImageNet. Moreover, for robust classifiers, the decision boundaries are more complicated, leading to failure in closing the performance gap between the LIMANS attack and AutoAttack when only 100 adversarial directions are learned. Nevertheless, it always shows good performance in transferring.

Tramèr et al. [159] claimed that the distance between the decision boundaries of two classifiers trained on the same dataset is small. It means that the LIMANS model of the adversarial noise space is likely to be transferable between classifiers. The results reported in the Table 2.6 confirm this intuition. The LIMANS model builds upon a ResNet50 and a VGG showing better transfer performance over state-of-the-art attacks across other classifiers.

Besides, note that the transferable property also holds between vanilla and robust classifiers. The LIMANS model learned on a vanilla classifier used to fool a robust classifier (and vice versa) produces slightly worse results than the one learned on a classifier of the same category. It might be due to the differences in the dataset split used to train the classifiers, resulting in a larger distance between the decision boundaries of the two classifier types. Yet the performance is still remarkable, e.g., on ImageNet, $FR_{(\text{R-wr50}\rightarrow\text{ResNet-18})} = 78\%$ $FR_{(\text{VGG}\rightarrow\text{ResNet-18})}$ and $FR_{(\text{ResNet-18}\rightarrow\text{R-r18})} = 35\%$ $FR_{(\text{R-r18}\rightarrow\text{R-r18})}$ (with R-r18 standing for Robust ResNet-18 and R-wr50 for Robust WideResNet-50).

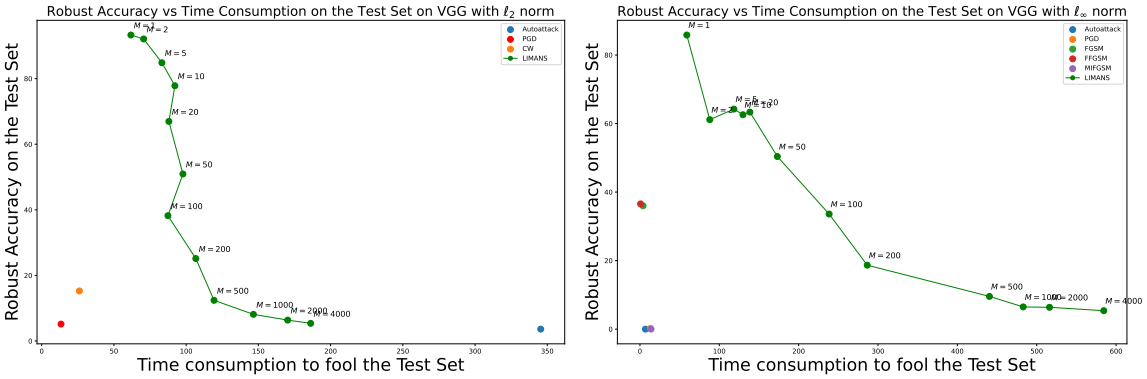
Furthermore, it is worth noting that the performance of LIMANS attack on a target classifier does not depend on its performance on the source classifier, but on the nature of this target classifier. In Table 2.6, the fooling rate of the LIMANS attack on ResNet is 91.3%. However, when the learned LIMANS model generates adversarial perturbation to fool MobileNet, its performance is even better, reaching 96.0%. This is because MobileNet is a simpler and easier classifier to attack. Finally, through comparison and analysis, we conclude that a LIMANS model trained on a robust

classifier is more easily transferable to other classifiers. Generally, the model that is harder to be fooled, shows ascendancy performances in transferring to other models.

2.6.5 Time consumption analysis

By relying on a model of the adversarial noise space $D \in \mathbb{R}^{P \times M}$ with $M \ll P$, the LIMANS adversarial attack problem optimizes a vector $v \in \mathbb{R}^{M}$ rather than an adversarial perturbation $\omega \in \mathbb{R}^{P}$. Intuitively, by optimizing a lower number of variables, the LIMANS attack breaks the time complexity of the adversarial example problem therefore, should take much less time to compute adversarial examples.

In Figure 2.19, we conducted an experiment comparison of the time consumption of LIMANS and state-of-the-art specific adversarial attacks to confirm or not this intuition.



(a) Considering the ℓ_2 norm.

(b) Considering the ℓ_∞ norm.

Figure 2.19: Pareto front of the Robust Accuracy achieved by a standard VGG11 classifier and the time consumption taken by the state-of-the-art specific adversarial attacks and the LIMANS with a different number of atoms M attacks under the ℓ_2 and the ℓ_∞ norm constraint on the CIFAR-10 dataset. On the y-axis is displayed the Robust Accuracy of the classifier, the lower Robust Accuracy is, the more efficient the adversarial attack is. On the x-axis is the time taken (in seconds) to compute adversarial examples over all the test sets of CIFAR-10, the lower the time is, the more efficient the adversarial attack is.

As shown in Figure 2.19, LIMANS does not improve the time consumption of the adversarial example crafting problem. Indeed, LIMANS is always dominated in time consumption by almost every specific attack. The only exception is Autoattack with the ℓ_2 norm constraint, which takes much more time to reach the best Robust Accuracy. We explain this empirical counter-intuitive observation by highlighting that specific adversarial attacks rely on enhanced and efficient optimization procedures to find $\omega \in \mathbb{R}^{P}$, whereas LIMANS optimizes problem (2.17) in a very basic way by relying on an SGD optimization that is not very efficient in time.

Figure 2.19 leads us to consider improving the inference optimization problem, as an important perspective for future work. Indeed, improving the time consumption of LIMANS would likely place it as a strong, more robust, and more transferable attack compared to state-of-the-art specific attacks. Section 2.8 details the envisioned LIMANS perspectives.

2.6.6 Insights on the choice of M

Figure 2.15 tells us that the choice of M is crucial regarding the performances of the derived LIMANS attack.

From our CIFAR-10 experiments, we found it interesting to choose M = 500 as, in terms of RAUD the LIMANS_{500} attack already overcomes state-of-the-art specific adversarial attacks. By setting M = 500, the adversarial examples creation problem boils down to optimizing 500 variables, representing only 16.27% of the original image size. As for ImageNet which is a much more

complex and bigger dataset, our experiments suggest that setting M = 150 is a good trade-off between the LIMANS performances and the space complexity to store such an adversarial noise model in GPU memory.

2.6.7 Optimizations hyper-parameters

This section presents the different implementation choices made for the Regularized-LIMANS and Simple-LIMANS experiments. We conducted several sanity-check experiments to enhance the LIMANS optimization. The following subsections detail each sanity-check experiment.

Regularized-LIMANS hyper-parameters and settings

Figure 2.20 illustrates the impact of the hyper-parameter λ and the number of atoms M on the attack performances. With appropriately increasing M, the performance will be improved, which confirms the conclusions drawn from Figure 2.15 and Figure 2.19. Considering the trade-off between the attack performances and the memory limit of storing the atoms, we choose to use M = 150 for CIFAR-10 and M = 100 for ImageNet when relying on the Regularized-LIMANS solver to estimate the best λ hyper-parameter. From Figure 2.20 we understand that when considering the ℓ_∞ norm constraint, setting λ = 1 provides the best performance, while λ = 0.1 is the suitable setting when dealing with the ℓ_2 norm constraint. This conclusion is valid when extending to other classifiers, as shown in Table 2.9.

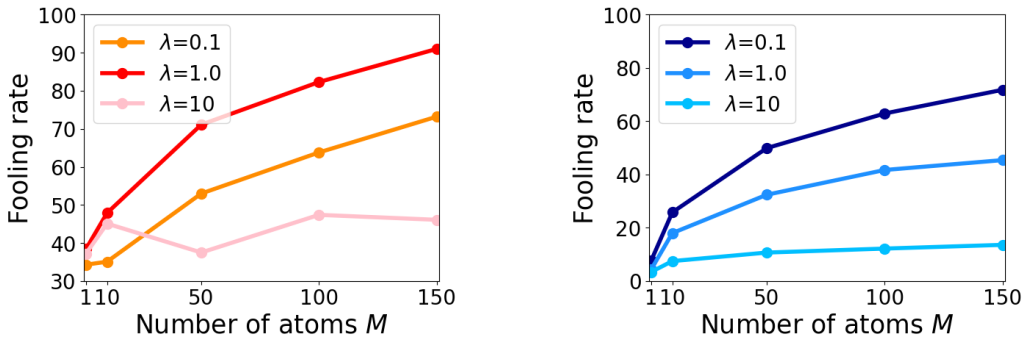


Figure 2.20: Performance of LIMANS (left) ℓ_∞-attacks and (right) ℓ_2-attacks on CIFAR-10 when attacking VGG, under different settings of hyper-parameter in Regularized LIMANS λ, and different number of atoms M.

Table 2.9: Fooling rate performances of the Regularized-LIMANS solver with M = 150 on CIFAR-10, when varying the λ hyper-parameter. The best results are marked in bold style.

	ℓ_∞ norm			ℓ_2 norm		
λ	VGG	MobileNet	R-R18	VGG	MobileNet	R-R18
0.1	73.2	85.3	15.8	**71.7**	**95.4**	**17.6**
1.0	**91.0**	**97.3**	**25.3**	45.4	49.6	12.8
10	46.1	91.7	19.9	13.6	24.4	10.3

Simple-LIMANS hyper-parameters and settings

The main idea behind Simple-LIMANS is to optimize the LIMANS parameters in the simplest way possible, with the least hyperparameter tuning and optimization details possible. However, few remain. We detail here their impact and how we selected those.

Simple-LIMANS ablation study of the offset b

As of good practice, and to see that Simple-LIMANS was indeed capturing the right amount of information without promoting too much the offset b, we performed an ablation study of the LIMANS performances with and without the offset b,

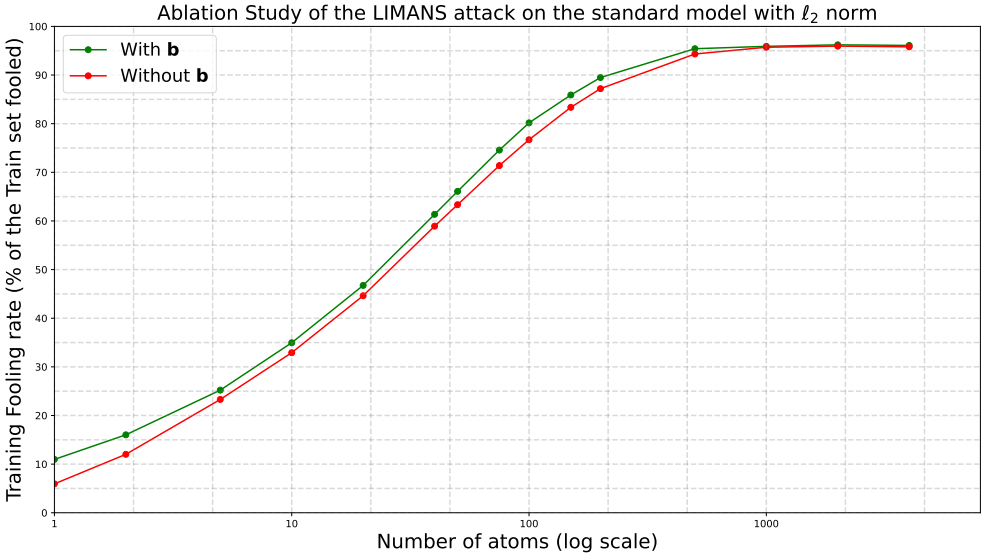


Figure 2.21: Performances of the Simple-LIMANS algorithm under the ℓ_2 norm constraint, with and without the offset $\mathbf{b}$.

Figure 2.21 compares the training performances of a Simple-LIMANS produced solution and the same solution without the offset $\mathbf{b}$. We can see that LIMANS' performances grow accordingly without any unseen artifacts captured by $\mathbf{b}$. Besides, through this figure, we understand its relative importance alongside the atoms of D that constitute the specific adversarial noise of each original example using its corresponding coding vector. The weight of $\mathbf{b}$ relative to the atoms is reasonable, and the balance between both in the implication of the fooling performances is respected. Using an offset $\mathbf{b}$ improves the performances of the LIMANS attack, but is not necessary for the well-being of both the modeling of the adversarial noise space and the crafting of new adversarial perturbations. The offset relaxation of D exactly does what it is intended to do and, sanity checks the optimization procedure.

Learning rate

No regularization parameter tuning is needed and the learning rate is automatically taken care of, by using the scheduler ReduceLROnPlateau. The ReduceLROnPlateau scheduler is a tool proposed by the PyTorch framework, which aims at reducing the learning rate when the loss plateaus. The optimization needs to start with a pretty high learning rate which will avoid bad random initialization local minima, and then will decrease as the optimization proceeds, allowing the parameters to reach a better minimum. The parameters of the used ReduceLROnPlateau are patience=40, factor=0.1, and threshold=1e-1.

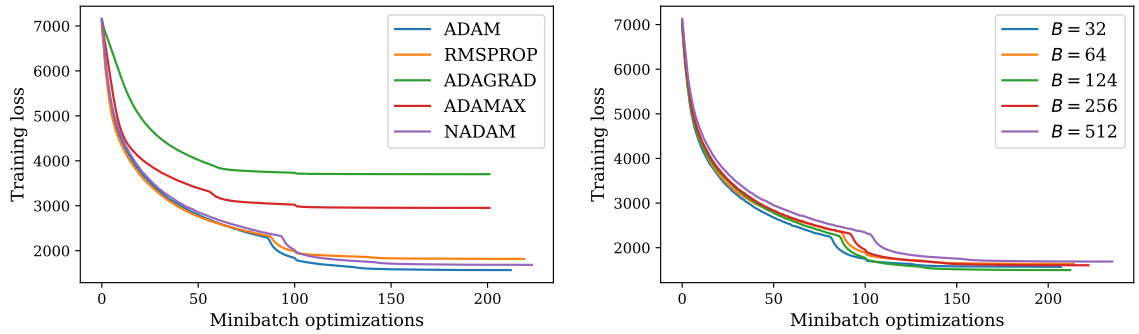
Optimizer

All the Simple-LIMANS optimizations were performed with the Adam optimizer. We found the Adam optimizer to be the best among several ones, as shown in Figure 2.22a. Indeed, similar performances can be reached with a different optimizer such as the RMSProp optimizer, but overall the Adam optimizer or one of its variants seems to be a relevant optimizer choice.

Batch size

As Figure 2.22b shows, when using the Simple-LIMANS solver, the batch size B is not a sensitive hyper-parameter. We found that different batch sizes could end up with similar performances. The only difference resides in the time consumption, higher batch sizes yield faster computations. During our experiments, the batch size B was set to B = 256 during training and B = 64 during

inference.



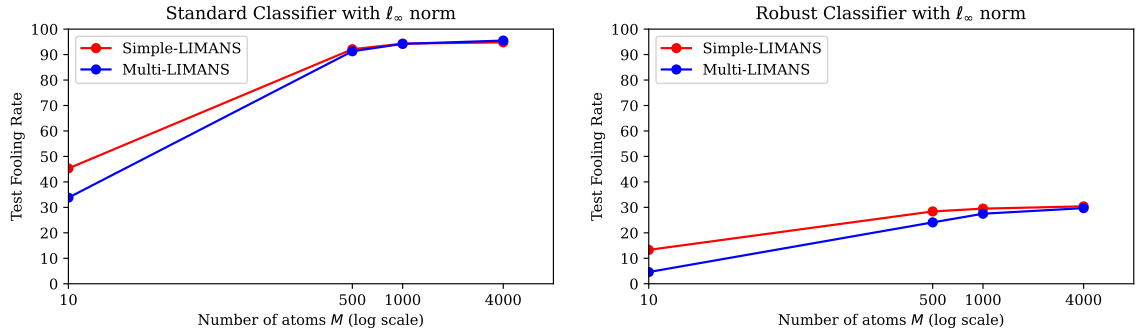
(a) Optimizer comparison with the Simple-LIMANS solver.

(b) Batch size comparison with the Simple-LIMANS solver.

Figure 2.22: Evolution of LIMANS_{10} training loss according to (left) different optimizers and (right) different batch sizes B using the Simple-LIMANS solver on a standard VGG classifier under the ℓ_∞ norm.

2.6.8 Multi-LIMANS

The most obvious extension of LIMANS we thought of, was to make it an ensemble attack coined Multi-LIMANS. In the search for the optimal adversarial noise space model we proposed to learn a shared D on multiple classifiers f_k, each possessing their specific training coding vectors $\{v_k^{(i}\}_{i=1}^{\text{N}}$, and see if, at inference, the specific adversarial noises would be more efficient or more transferable.



(a) Multi-LIMANS applied on a Standard classifier.

(b) Multi-LIMANS applied on a Robust classifier.

Figure 2.23: Test performances of the Simple-LIMANS algorithm under the ℓ_∞ norm constraint, for both the standard classifier (VGG) and the robust classifier (Robust ResNet) with the Multi-LIMANS training on the VGG, DenseNet and robust ResNet classifiers.

Figure 2.23 displays test fooling rate performances between a Multi-LIMANS trained D and, a Simple-LIMANS trained D. We empirically saw that learning D on multiple classifiers rather than only one did not improve the universal adversarial direction but introduced more bias, preventing it from reaching the inference-specific adversarial noise which was originally at hand.

Another envisioned improvement that Multi-LIMANS could bring in, was to improve the transferability. We knew that by learning on multiple classifiers, the dictionary was not better at inference for those classifiers, but did it generalize more and therefore performed better on other classifiers it has not yet been applied to,

Unfortunately, the bias introduced by learning D on multiple classifiers does not balance with the multi-classifiers' learned property. As Figure 2.24 shows, inference performances are never

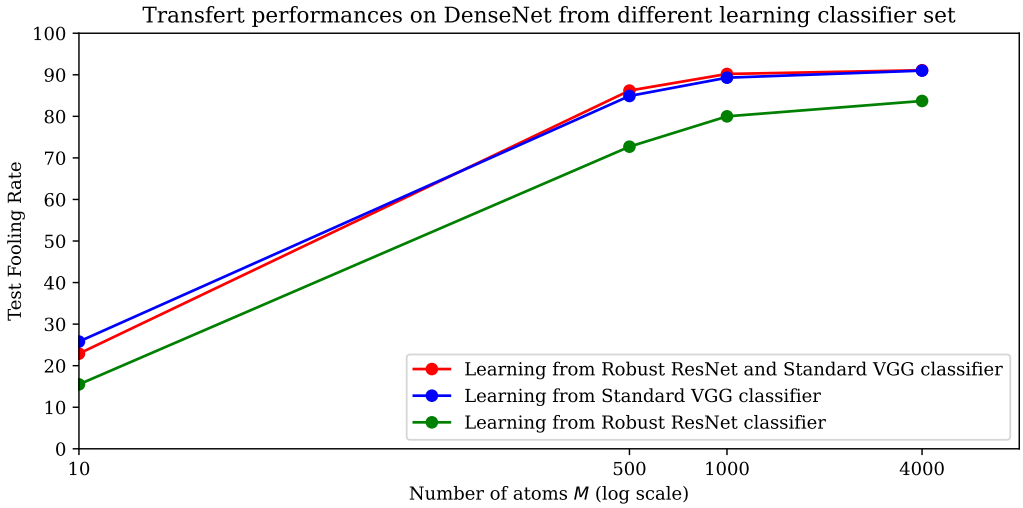


Figure 2.24: Transfer test performances of the Simple-LIMANS algorithm under the ℓ_∞ norm constraint, for both the standard classifier (VGG) and the robust classifier (Robust ResNet) with the Multi-LIMANS training on the VGG, DenseNet, and robust ResNet classifiers, but not trained on the model used during inference.

improved when training the same D to fool multiple classifiers, which means it only adds bias, and, training a D specific to a classifier already captures a generalized misclassification power, that we previously outlined to be transferable to other classifiers.

Table 2.6: Transferability performance of the LIMANS ℓ_∞ attacks on CIFAR-10 ($\epsilon = 8/255$), in terms of fooling rates (FR). The best transferable results are marked in bold font.

Source model \Target model		MobileNet	Inception	ResNet50	DenseNet	VGG	R-r18	R-wr-34-10
MobileNet	AutoAttack	**100**	87.1	37.2	32.8	22.4	1.7	1.5
	UAP	47.3	36.1	9.3	8.3	8.7	1.3	0.8
	UAPPGD	86.2	56.1	20.5	19	21.3	1.6	1.5
	TI-FGSM	87.2	25.2	25.9	29.1	16.1	2.1	2.0
	VMI-FGSM	**100**	87.3	53.1	49.8	38.9	2.2	2.5
	VNI-FGSM	**100**	88.1	54.8	53.1	40.7	2.4	2.9
	NAA	72.2	25.3	6.8	5.9	6.4	1.3	1.0
	RAP	86.7	60.3	38.5	35.7	25.8	3.0	1.6
	LIMANS	97.3	**92.2**	**73.6**	**66.4**	**67.7**	**10.2**	**10.7**
Inception	AutoAttack	54.7	**100**	14.7	12.9	12.0	1.2	1.1
	UAP	39.2	32.9	9.3	9.7	9.4	1.5	1.1
	UAPPGD	73.9	75.5	26.3	23.8	27.3	2.2	1.5
	TI-FGSM	19.7	60.2	19.8	21	11.4	2.2	1.5
	VMI-FGSM	69.8	86.1	40.8	38.3	31.3	2.6	1.6
	VNI-FGSM	75.5	89.5	44.4	42.4	36.3	3.2	2.3
	NAA	38.7	70.5	8.4	8.1	9.2	1.1	1.5
	RAP	61.9	90.2	42.0	41.7	30.3	2.3	2.7
	LIMANS	**98**	95.1	**79.6**	**73.9**	**75.8**	**10.7**	**10.8**
ResNet50	AutoAttack	63.3	52.6	**100**	54.6	25.1	1.2	2.4
	UAP	31.4	23.6	12.1	12.5	11.2	1.3	1.7
	UAPPGD	63.3	49.4	39.4	35.1	26.1	1.1	2.3
	TI-FGSM	18.4	17.1	74.0	38.5	20.4	2.2	3.0
	VMI-FGSM	74.9	75.3	96.0	78.1	53.5	2.1	3.2
	VNI-FGSM	78.3	76.9	95.9	80.3	57.2	2.7	2.1
	NAA	50.7	38.6	64.7	22.9	18.4	1.4	2.1
	RAP	49.0	45.7	75.1	52.5	35.4	1.6	2.8
	LIMANS	**96.0**	**92.9**	91.3	**81.8**	**82.1**	**11.7**	**13.2**
DenseNet	AutoAttack	56.9	51.6	48.8	**100**	21.8	2.1	2.0
	UAP	27.6	20.6	10.6	12.8	11.4	1.6	1.4
	UAPPGD	61.1	49.9	29.3	47.4	27.3	2.7	2.1
	TI-FGSM	17.4	15.8	26.3	65.2	17	2.9	2.3
	VMI-FGSM	73.7	71.8	77.2	93.1	47.9	3.3	3.7
	VNI-FGSM	78.1	76.2	79.5	94.0	53.3	3.5	4.2
	NAA	37.2	31.1	23.7	74.9	12.5	1.2	1.5
	RAP	47.8	43.5	48.7	75.9	35.6	3.2	3.5
	LIMANS	**96.7**	**93.5**	**88.4**	85.5	**82.7**	**12.3**	**13.4**
VGG	AutoAttack	62.5	58.0	43.0	44.0	**100**	2.7	2.7
	UAP	22.0	18.4	10.2	10.2	10.0	1.1	1.3
	UAPPGD	63.6	55.9	27.6	29.4	41.9	3.1	2.1
	TI-FGSM	19.7	16.7	25.6	27.6	74.4	3.7	2.2
	VMI-FGSM	66.2	64.2	57.5	56.9	96.5	3.0	2.6
	VNI-FGSM	69.3	68	62.6	61.4	96.5	3.0	2.6
	NAA	42.3	38.3	14.5	1.8	71.6	1.6	1.2
	RAP	46.5	44.5	39.5	40.9	73.8.	3.3	3.4
	LIMANS	**97.4**	**95.1**	**87.5**	**81.5**	91.0	**11.5**	**12.6**
R-r18	AutoAttack	17.5	15.7	17.2	15.6	17.5	**44.3**	**23.4**
	UAP	14.5	9.5	7.1	6.4	7.6	1.9	2.6
	UAPPGD	18.6	13.3	9.7	8.6	10.5	3.1	3.5
	TI-FGSM	8.4	5.5	8.2	7.8	8.6	26.2	13.1
	VMI-FGSM	24	22.9	24.2	21.9	24.8	38	22.7
	VNI-FGSM	27.1	23.1	25.4	23.8	25.6	38.1	22.9
	NAA	16.2	11.5	11.2	10.4	10.4	18.7	7.2
	RAP	10.9	8.4	7.9	8.9	9.7	23.8	12.2
	LIMANS	**81.3**	**73.2**	**71.7**	**68.3**	**61.7**	25.3	21.6

Table 2.7: Transferability performance of the LIMANS ℓ_2-attacks on CIFAR-10 ($\epsilon = 0.5$), in terms of fooling rates (FR). The best transferable results are marked in bold font.

Source model \Target model		MobileNet	Inception	ResNet50	DenseNet	VGG	R-r18	R-wr-34-10
MobileNet	AutoAttack	**100**	50.20	14.20	13.30	8.20	0.90	0.50
	UAP	7.50	5.20	3.00	2.50	2.40	0.30	0.40
	UAPPGD	37.90	15.20	2.00	1.10	0.90	0.30	0.20
	CW	97.50	11.00	4.20	3.20	2.40	0.30	0.00
	RAP	67.30	11.20	4.20	3.90	2.60	0.50	0.10
	LIMANS	95.40	**91.50**	**61.70**	**59.30**	**51.50**	**4.60**	**5.00**
Inception	AutoAttack	32.80	**100**	6.60	7.90	5.50	0.50	0.50
	UAP	9.80	7.50	2.50	3.50	2.90	0.20	0.10
	UAPPGD	26.90	16.70	1.30	2.30	2.00	0.30	0.10
	CW	16.30	82.80	5.00	5.20	3.70	0.30	0.00
	RAP	13.60	43.50	3.50	3.60	2.70	0.40	0.30
	LIMANS	**94.60**	94.10	**64.30**	**63.90**	**57.20**	**5.10**	**5.20**
ResNet50	AutoAttack	31.00	23.40	**99.70**	26.10	10.00	1.20	0.70
	UAP	5.10	3.80	2.40	1.90	2.80	0.50	0.30
	UAPPGD	4.10	3.20	2.20	2.30	2.20	0.40	0.20
	CW	13.50	9.80	82.40	13.10	6.10	0.50	0.40
	RAP	10.20	8.60	33.00	8.60	4.90	0.40	0.30
	LIMANS	**92.60**	**87.50**	78.10	**71.70**	**61.70**	**7.90**	**7.50**
DenseNet	AutoAttack	32.60	25.20	27.30	**99.50**	10.20	0.50	0.50
	UAP	4.90	3.70	2.60	3.30	1.90	0.20	0.20
	UAPPGD	5.00	4.50	3.20	3.70	2.10	0.20	0.20
	CW	14.60	13.70	14.80	80.00	6.40	0.40	0.30
	RAP	8.00	7.30	7.70	29.00	4.50	0.30	0.30
	LIMANS	**91.10**	**87.60**	**74.00**	74.10	**62.70**	**8.40**	**7.70**
VGG	AutoAttack	32.00	28.20	19.50	21.10	**98.90**	0.80	0.60
	UAP	4.70	3.80	2.20	2.70	2.00	0.50	0.40
	UAPPGD	4.80	5.60	2.00	2.70	2.80	0.40	0.40
	CW	10.00	8.20	5.70	7.40	79.10	0.60	0.30
	RAP	8.80	7.10	5.20	6.50	32.10	0.30	0.50
	LIMANS	**94.20**	**89.00**	**74.80**	**71.00**	71.70	**8.00**	**7.10**
R-r18	AutoAttack	6.70	8.30	8.00	8.50	9.60	**24.60**	11.00
	UAP	3.40	3.10	2.50	2.30	1.80	0.50	0.40
	UAPPGD	2.70	2.10	2.00	1.70	2.60	0.30	0.10
	CW	9.50	11.60	10.60	10.00	11.60	22.90	3.90
	RAP	8.70	7.70	7.60	8.10	9.60	10.70	4.50
	LIMANS	**58.70**	**53.80**	**50.30**	**50.70**	**41.80**	17.60	**14.60**
R-wr-34-10	AutoAttack	7.70	7.80	8.20	7.80	8.90	15.20	**22.50**
	UAP	3.00	3.10	2.40	2.90	2.60	0.90	0.40
	UAPPGD	2.90	2.80	2.20	1.00	1.60	0.70	0.60
	CW	10.30	9.10	13.00	10.60	10.40	8.80	21.20
	RAP	8.10	7.30	8.10	7.60	7.70	7.10	9.90
	LIMANS	**59.10**	**54.80**	**51.80**	**50.00**	**42.50**	**17.00**	14.70

Table 2.8: Transferability performance of the LIMANS ℓ_∞ attack on ImageNet ($\epsilon = 4/255$), in terms of fooling rates. The best transferable results are marked in bold font.

Source model \Target model		MobileNet	ResNet18	DenseNet	VGG	R-r18	R-wr50
MobileNet	AutoAttack	**100**	26.38	20.44	26.94	1.64	1.24
	UAP	48.48	11.5	10.46	17.28	1.8	0.84
	UAPPGD	69.94	18.04	14.34	22.34	2.72	1.56
	TI-FGSM	99.74	36.98	31.66	31.24	3.2	2.56
	VMI-FGSM	**100**	44.84	37.92	42.92	2.92	2.04
	VNI-FGSM	99.98	44.64	36.54	43.62	2.88	2.00
	NAA	84.56	15.1	11.72	16.88	2.1	1.2
	RAP	96.52	**54.58**	**47.24**	**49.16**	3.72	3.16
	LIMANS	75.24	50.06	46.94	44.34	**10.02**	**5.62**
ResNet18	AutoAttack	40.3	**100**	35.76	34.9	1.8	1.34
	UAP	13.34	11.3	9.00	11.72	1.36	0.86
	UAPPGD	25.3	47.22	18.44	23.26	2.5	1.44
	TI-FGSM	32.06	99.84	31.38	31.66	2.98	2.8
	VMI-FGSM	56.5	**100**	51.78	50.2	2.9	2.04
	VNI-FGSM	56.74	99.98	51.4	**51.42**	2.84	2.04
	NAA	22.54	97.94	14.84	19.3	2.12	1.2
	RAP	53.36	96.74	51.30	50.60	3.80	3.14
	LIMANS	**59.16**	59.16	**53.14**	48.28	**10.48**	**6.62**
DenseNet	AutoAttack	37.72	40.4	**100**	30.22	1.8	1.3
	UAP	12.76	9.94	9.8	11.42	1.24	0.92
	UAPPGD	22.72	20.7	40.04	20.18	2.48	1.28
	TI-FGSM	30.1	35.56	99.66	27	3.12	2.32
	VMI-FGSM	52.22	55.44	99.98	44.82	2.9	2.06
	VNI-FGSM	53.88	**56.9**	99.98	46.16	2.64	2.1
	NAA	24.22	25.68	98.34	21.38	1.34	1.42
	RAP	48.16	54.12	96.76	42.00	3.12	3.30
	LIMANS	**58.86**	**56.9**	57.26	**47.74**	**11.3**	**7.32**
VGG	AutoAttack	47.94	40.06	32.62	**100**	2.34	1.42
	UAP	13.34	9.8	8.82	13.6	1.34	0.78
	UAPPGD	24.42	23.16	18.12	46.26	2.54	1.6
	TI-FGSM	33.2	38.26	29.3	99.4	2.96	2.28
	VMI-FGSM	57.52	53.46	43.76	99.86	2.9	2.2
	VNI-FGSM	**57.98**	53.96	42.88	99.84	2.76	2.24
	NAA	19.62	14.92	12.18	79.96	2.18	1.4
	RAP	53.14	53.12	42.68	95.68	3.48	2.84
	LIMANS	57.68	**54.14**	**50.04**	51.62	**10.68**	**6.24**
R-r18	AutoAttack	13.7	15.8	10.82	14.6	**71.74**	10.78
	UAP	11.52	9.32	8.46	10.90	1.44	1.16
	UAPPGD	14.00	12.34	11.20	13.56	3.14	1.66
	TI-FGSM	11.88	13.42	10.08	11.02	54.46	10.14
	VMI-FGSM	17.00	17.80	12.12	16.08	64.98	11.94
	VNI-FGSM	16.14	17.66	12.48	16.08	63.22	11.74
	NAA	11.46	10.86	9.34	11.42	21.48	4.9
	RAP	11.32	10.80	8.16	10.32	45.80	7.94
	LIMANS	**37.14**	**33.2**	**33.76**	**29.90**	29.84	**12.94**
R-wr50	AutoAttack	20.14	22.76	17.36	19.44	15.42	**59.02**
	UAP	9.88	7.60	6.96	8.62	1.84	1.24
	UAPPGD	14.54	12.56	10.92	14.36	2.16	1.38
	TI-FGSM	14.16	16.34	12.68	13.68	17.16	43.66
	VMI-FGSM	24.22	26.66	20.12	23.86	17.82	54.56
	VNI-FGSM	23.88	26.22	19.68	23.28	18.00	52.28
	NAA	14.08	13.12	10.20	14.04	9.82	12.58
	RAP	13.82	14.06	10.52	13.5	15.54	34.1
	LIMANS	**42.18**	**42.5**	**42.46**	**34.22**	**23.7**	18.02

2.7 Conceptual questions raised by our work

In this section, we detail the conceptual questions raised by LIMANS, the originality of the proposed method, and the vision implied behind it. Which gap does it aim to fill and which path does it pave the way to.

2.7.1 Common corruptions or ℓ_p norm bounded adversarial examples?

To evaluate the robustness of classifiers, relying on perturbation to assess its empirical limits is the most straightforward choice. However, defining such perturbation remains hard to do. On the one hand, most research adopted the use of ℓ_p norm bounded perturbation specific to each example to assess the almost exact limit of the classifier's decision boundaries, giving us a precise definition of its robustness (detailed in Section 1.2.4). On the other hand, some research groups criticized this choice by stating that such ℓ_p norm bounded perturbation specific to each example, is not realistic and therefore produces useless and incomplete evaluations of the classifier's robustness. They proposed a set of common corruptions, allowing us to assess a more precise and realistic robustness evaluation of the classifier (detailed in Section 1.2.1). Even though such common corruptions seem appealing, they don't allow to reach the classifier's decision boundaries making the robustness evaluation more realistic but unfortunately more limited than optimized adversarial attacks.

With LIMANS, we propose to use the rely on ℓ_p norm bounded perturbation specific to each example to reach the classifier's decision boundaries, making the robustness evaluation precise, although to compute specific ℓ_p norm bounded perturbations on a more concrete basis than randomly drawn among each feature of the examples. This more concrete basis is modeled by the universal atoms that we reach at the end of the LIMANS training procedure (detailed in Section 2.5), and they reveal interesting universal patterns always involved in the reach of the classifier's decision boundary. In a way, the LIMANS training procedure consists of the capture of the semantics of the objects within the examples and using these extracted features to overcome the classifier's decision boundaries. By relying on the optimized semantic of the object, LIMANS allows computing more realistic adversarial perturbations (see in Section 2.6.2), thus performing a more precise robustness evaluation of classifiers, in a way that the decision boundaries are more precisely reached than with common corruptions, and realistic, in a way that the computed specific ℓ_p norm bounded perturbations are much more object-related and therefore more realistic than other specific ℓ_p norm bounded perturbation.

We strongly believe that neither the specific ℓ_p norm bounded perturbations nor the common corruptions are necessary nor sufficient to efficiently asses a precise and credible robustness evaluation of the classifiers.

LIMANS is just a humble proposition aiming to bridge the gap between these two paradigms, welcoming a reassessment of what has already been proposed, an honest discussion on the usefulness of unrealistic robustness evaluation, and a criticism of the effectiveness of common corruptions to estimate the classifiers' robustness.

2.7.2 Specific or universal adversarial perturbations?

Among the ℓ_p norm-bounded perturbations research community, previous criticisms have already been made. Essentially people denied the specific ℓ_p norm bounded perturbations because they are totally specific to each example, allowing them to reach almost exactly the classifier's decision boundary, but on this very example alone. Evaluating the classifier's robustness on an empirical dataset using specific ℓ_p norm bounded perturbations will only give a robustness measure for this dataset alone and nothing more. This limitation makes the robustness evaluation using specific ℓ_p norm-bounded perturbations, impractical and useless in real-life applications. The original goal is to estimate the overall robustness of the classifiers on every example, even the one we do not have access to, yet.

To resolve this limitation of specific ℓ_p norm-bounded perturbations, several authors proposed to use instead universal ℓ_p norm-bounded perturbations (detailed in Section 1.2.4). These universal ℓ_p norm-bounded perturbations are example-agnostic, making them most likely effective even on future examples we do not know of yet. Universal ℓ_p norm-bounded perturbations are very interesting as they target universal weaknesses and pitfalls of classifiers, to which they are completely broken and are always fooled.

Therefore, universal perturbations must always be included within the robustness evaluation of classifiers as they are, by definition, always included in the set of perturbations we want to model. However, universal perturbations alone are not sufficient to entirely assess the classifiers' robustness. Indeed, by being fully universal, these perturbations do not allow us to get close to the exact classifier's decision boundaries. They are too limited, and don't explore the example space to the fullest allowing to exactly assess the robustness of classifiers.

With LIMANS, we propose to take advantage of both the specific and universal paradigms. Indeed, LIMANS is one of the first propositions bridging the gap between specific and universal ℓ_p norm-bounded perturbations by computing specific perturbations upon a universal perturbation basis. LIMANS almost always capture the universal perturbations necessary to assess the most general robustness of classifiers but also allow us to explore the example space to reach more precisely the classifier's decision boundaries.

LIMANS framework allows drawing robustness evaluations that are more generalized than specific ℓ_p norm bounded perturbations alone, and more efficient than universal ℓ_p norm bounded perturbations alone.

We humbly believe the current interest in universal perturbation is too limited, and that a correction of their value is essential to the sanity of robustness evaluation research. Rather than blindly running after the improvement of some performance metric, a deep questioning of the current work and its importance should be made, engaging in more honest, fair, and unbiased research.

2.7.3 Maximizing the fooling rate or maximizing the perturbation quality?

We previously saw that defining the perturbations involved in the robustness evaluation of classifiers is hard. However, once a definition is chosen, which metric should be considered to define the robustness of the classifier? The choice of the quality metric is crucial to the robustness evaluation. Section 1.2.3 details different quality metrics and explains what vision and goal each envisions. With different considered goals and metrics, a conflict of interest may arise in the robustness evaluation according to who is performing it.

We believe that a framework and constraints (just like the ϵ adversarial budget constraint) should be imposed, allowing the different propositions to be fairly compared and benchmarked. However, such a framework alone is not sufficient. Indeed, the final metric assessing the robustness of the classifier should also but carefully considered, preventing a possible conflict of interest.

To produce the least biased possible robustness evaluation, we believe that future work should explicitly state the used quality metric and to whom it is profitable, informing new readers of the possible limits and advantages of each metric.

We currently are not aware of the best robustness metric. However, we think the perfect robustness metric should integrate every intent of the robustness, whether it is its protection from possible harm or performance under possible hardware bugs.

As the deep community would suggest, one idea could be to parameterize such robustness metric with a neural network framework. The framework would take in the classifier, the original examples, and the perturbed examples outputting a robustness score. Even though it is very straightforward and almost simplistic, this approach would be fair to everyone as the same framework metric would be used, under different classifiers with the same adversarial attack producing the perturbed dataset, or the same classifier under different perturbed datasets or different classifiers. Hence, every new proposed perturbation generation would get a fair robustness score, and every new proposed classifier would receive a fair robustness score.

We believe that fairness and avoiding possible bias should be the center of future research engaging in honest and transparent discussion about new propositions.

2.8 Perspectives

In itself, LIMANS is already a complete proposition on which we spent several months to converge, but it still offers very appealing perspectives that we, unfortunately, did not have enough time to complete. LIMANS framework is one of its kind in the robustness evaluation community, discussing several points that are widely accepted and currently considered sufficient. LIMANS' perspective includes reproducing the ADiL experiments that we did not have enough time to do, such as the training stage with fewer original examples. Do the LIMANS performances and generalization grow according to the number of training examples, or by its linear definition is it immune to the training dataset size?

One original and very interesting ADiL idea was to derive a black-box adversarial attack. We believe that LIMANS could easily be subject to the same extension. A LIMANS black-box attack would step further the realism of the attacker's behavior, estimating the classifier's robustness in an even more realistic way.

Obviously, after proposing the LIMANS adversarial attack, we must propose the LIMANS adversarial defense to contribute to research under the two scopes of attacker and defender.

We also target to impose more constraints on the LIMANS atoms. Indeed, we believe the adversarial patch attacks are very interesting and appealing to estimate a realistic classifier's robustness. We think that, after a careful design of the constraints, we could make LIMANS parameters a dictionary of atoms producing adversarial patches, which could be a real improvement and innovation brought to the research community, going further into the design of realistic yet efficient perturbations.

During our experiments, we mainly focused on images but by definition, LIMANS could be applied to any kind of data modalities as long as we can back-propagate gradients and we do have a similarity metric between the objects allowing us to respect the adversarial budget constraint.

Chapter 3

A meaningful similarity metric

3.1 Observation

Usually, colored images are mathematically encoded as arrays of dimension the number of pixels in which each pixel color is represented by triplets of RGB values. Thus for a colored 24×24 image, its mathematical representation will be defined as $\mathbf{x} \in \mathbb{R}^{24\times24\times3} = \mathbb{R}^{1728}$. Therefore, to estimate the similarity of two images, mathematical tools like ℓ_p norms are well suited. Indeed, ℓ_p norms define distances $d : \mathbb{R}^{\mathrm{P}} \times \mathbb{R}^{\mathrm{P}} \mapsto \mathbb{R}_+$, thus respecting the three following axioms,

$$\begin{array}{rll} \textit{Symmetry:} & \forall \mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \mathbb{R}^{\mathrm{P}}, & d(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = d(\mathbf{x}^{(2)}, \mathbf{x}^{(1)}) \\ \textit{Equality:} & \forall \mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \mathbb{R}^{\mathrm{P}}, & d(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = 0 \Leftrightarrow \mathbf{x}^{(1)} = \mathbf{x}^{(2)} \\ \textit{Triangular inequality:} & \forall \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)} \in \mathbb{R}^{\mathrm{P}}, & d(\mathbf{x}^{(1)}, \mathbf{x}^{(3)}) \leq d(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) + d(\mathbf{x}^{(2)}, \mathbf{x}^{(3)}) \end{array} \tag{3.1}$$

Having a distance is interesting as it offers a certain flexibility in its use, and produces unbiased results. Besides, ℓ_p norm distances scale very well in dimension, which is critical when computing the similarity of two images. The image dimension can quickly increase, easily making the complexity of the problem troublesome, therefore, an appropriate metric is required.

Even though ℓ_p norms are very convenient tools, considered as relevant image distances, they hardly compute the appropriate distances between the images. Indeed, in Figure 3.1, we see two cat images, then the distance between these two images should be low as the same object is displayed, only the cat's color and background change, but in principle, they both represent the same idea. Figure 3.1 displays a third image containing a plane with a vaguely similar background color to the second cat but represents a different idea. Figure 3.1 highlights the meaningless property of ℓ_p norms to compute image distances as the plane is closer to both cats than the cats together.

Even though they are efficient mathematical tools, ℓ_p norms hardly capture the meaning of the images and therefore represent a poor distance function choice.

Up until now, the majority of state-of-the-art pseudo-random adversarial attacks considered an ℓ_p norm to be a relevant distance between images [143]. It led to generating incomprehensible and unrealistic adversarial perturbations still considered state-of-the-art. Opposite to pseudo-random attacks, more realistic attacks were proposed, struggling to efficiently compute meaningful distances between images and therefore, generate realistic and meaningful adversarial perturbations.

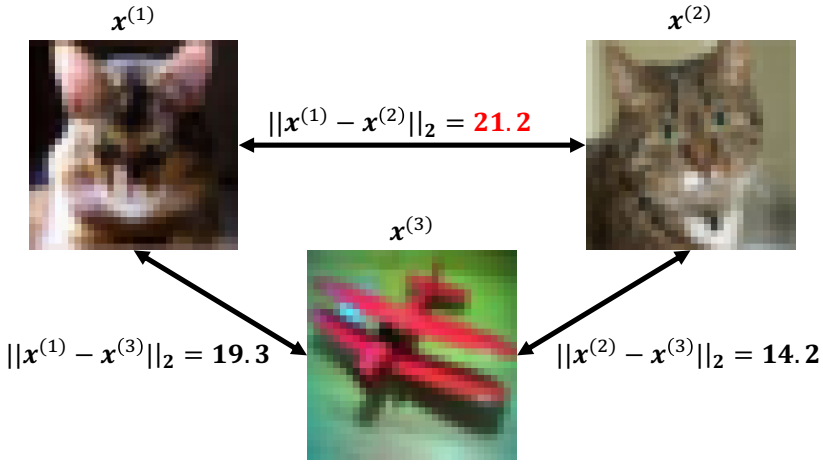


Figure 3.1: Examples of the ℓ_2 norm considered as a distance between images. All of the three images come from the CIFAR-10 dataset.

3.2 Proposition

The goal of this second, more exploratory, project of this thesis is, to come up with a meaningful distance between images leading to generating more realistic adversarial perturbations computed from the very efficient state-of-the-art pseudo-random adversarial attacks. When possessing such a meaningful distance, the original goal, to generate realistic and efficient adversarial perturbations, could be reached, letting us evaluate more precisely and on a realistic basis, the robustness of classifiers.

We mainly relied on [143] to outline the cases for which ℓ_p norms can be used and when they should not be. Readers are invited to refer to it for a more comprehensive and detailed explanation of why ℓ_p norms are neither sufficient nor necessary when used as a similarity metric of images for adversarial perturbation generation.

To ease the problem, our objective is to approximate a meaningful similarity metric, that is not a distance and thus allows us to violate the three axioms 3.1 defining a distance. Given a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{\mathrm{N}}$ of labeled examples with $\mathbf{x}^{(i)} \in \mathcal{X} \subset \mathbb{R}^{\mathrm{P}}$ and its corresponding label $y \in \mathcal{Y} = \{1, ..., \mathrm{C}\}$, we want to approximate a distance $d : \mathbb{R}^{\mathrm{P}} \times \mathbb{R}^{\mathrm{P}} \mapsto \mathbb{R}_+$ such that the points of same labels receive low distances (high similarity), while points of different labels receive high distances (low similarity),

$$\min_{d_{\mathrm{C}}} \sum_{\substack{\mathbf{x}^{(i)} \in \mathcal{Y}_k \\ \mathbf{x}^{(j)} \in \mathcal{Y}_k}} d_{\mathrm{C}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - \sum_{\substack{\mathbf{x}^{(i)} \in \mathcal{Y}_k \\ \mathbf{x}^{(j)} \in \mathcal{D} \setminus \mathcal{Y}_k}} d_{\mathrm{C}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}), \forall \mathcal{Y}_k \tag{3.2}$$

with $\mathcal{Y}_k = \{\mathbf{x}^{(i)}\}_{i=1}^{\mathrm{N}}, y^{(i)} = k$ the of all examples only of label k and C the parameters of the distance d.

The most obvious choice for such a parameterized distance is the Mahalanobis distance,

$$d_{\mathrm{C}}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sqrt{(\mathbf{x}^{(1)} - \mathbf{x}^{(2)})^{\mathrm{T}} \mathrm{C} (\mathbf{x}^{(1)} - \mathbf{x}^{(2)})}, \tag{3.3}$$

with $\mathrm{C} \in \mathbb{R}^{\mathrm{P} \times \mathrm{P}}$ a positive semi-definite matrix, in which the diagonal values C_{ii} serves as a weights of the difference values $[\mathbf{x}^{(1)} - \mathbf{x}^{(2)}]_{ii}$.

A Mahalanobis distance defines a weighted version of the ℓ_p norm. Even though it would still be interesting to explore, such a Manhattan distance only considers the identical relationships between the input entries. Only the diagonal values of the matrix C are considered to compute $d(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$, resulting in a restrained distance and allowing little flexibility in the approximation. Similar to our proposed adversarial dictionary attack proposition, modeling comprehensible and understandable C parameters of the distance is a desired property. Therefore, the distance would allow us to understand why two images are close (a low distance value representing high similarity) or why they are considered different (a high distance representing low similarity).

To extend the Mahalanobis distance proposition, and offer more flexibility we resorted to optimal transport distances.

3.3 Wasserstein Distance as a meaningful distance between images

Optimal transport distances, mostly known through the Wasserstein Distance (WD), are a set of probability distribution distances, that provides a way to compare how different two distributions are, based on the cost of "moving" mass from one distribution to the other according to a cost matrix C.

Given two probability distributions represented by empirical measures $\mathbf{a} \in \mathbb{R}^N$ and $\mathbf{b} \in \mathbb{R}^N$, the WD aims to find the most efficient way to transform the first distribution $\mathbf{a}$ into the second $\mathbf{b}$, where efficiency is defined by a cost function $C \in \mathbb{R}^{N \times N}$ that represents the "distance" or "effort" needed to move mass from one point to another,

$$\begin{gathered} \mathrm{WD}_C(\mathbf{a}, \mathbf{b}) = \min_{T \in \mathbb{R}^{N \times M}} \sum_{i=1}^{N} \sum_{j=1}^{M} T_{i,j} C_{i,j}, \\ \text{such that } T1 = \mathbf{a} \text{ and } T^T 1 = \mathbf{b}. \end{gathered} \tag{3.4}$$

In itself, computing the WD_C solves the optimization problem of finding the best transport plan T, conditioned to the cost matrix C. Figure 1.6 illustrates a valid transport plan between two points clouds.

The cost matrix plays a crucial role in understanding the transportation of mass between two distributions and has several interesting aspects regarding the modeling of a meaningful distance. First, rather than using only the diagonal values like a Mahalanobis distance, all entries of the cost matrix are necessary. Each entry in the matrix C_{ij} represents the cost of transporting a unit of mass from one point $\mathbf{a}_i$ to another $\mathbf{b}_j$. It allows more flexibility and richer modeling possibilities, as well as, an intuitive interpretation of the distance value between the distributions in terms of the transportation required and the associated costs. The WD_C offers a more precise distance value, as well as, an interpretation of its computation. In some cases, the cost matrix can be designed to incorporate application-specific information about the relationships between data points. This allows for a more meaningful measurement of distance that aligns with the problem's context. All these intriguing and favorable properties make the WD_C a strong candidate as a meaningful similarity metric between images.

Even though relying on the WD_C to estimate the similarity between images has already been done, it always was in a limited or flawed context. In [175], Wong et al. proposed the Wasserstein Attack, minimizing the WD_C between image pixels to compute adversarial perturbations. The authors proposed an idea similar to ours, and showed meaningful on-manifold adversarial perturbation results. However, their computed WD_C was revealed to be flawed. Indeed, in [140], authors demonstrate the irrelevancy of any image measure based on their pixels. Distances studied by [140] include the WD, preventing the former Wasserstein Attack from becoming a long-lasting relevant solution to compute meaningful adversarial perturbations of images. However, the displayed empirical results are promising and encourage our pursuit in this direction.

The WD_C between images has especially mainly been considered to estimate the quality of image generations. Indeed, the GAN (defined in Section 1.1.4) community which received lots of attention in the last years, hugely adopted the Fréchet Inception Distance (FID) as the most effective tool to track the quality of the artificially generated images. The FID calculates the distance between two distributions of images f_X and f_Y, most often one is a minibatch of original examples drawn from the dataset, while the second distribution is a minibatch of GAN (or any other image generator) generated examples. Typically, the two distributions of images are represented by summary statistics (mean & covariance matrix) computed over the outputted values of the third pool layer of an inception neural network for all images, acting as a feature extractor. The FID between

the two distributions is defined as,

$$\text{FID}(f_X, f_Y) = ||\mu_X - \mu_Y||_2^2 + \text{Tr}(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{1/2}). \tag{3.5}$$

Specialized variants of FID have been suggested as evaluation metrics for music enhancement algorithms as Fréchet Audio Distance (FAD) [84], for generative models of video as Fréchet Video Distance (FVD) [163], and for AI-generated molecules as Fréchet ChemNet Distance (FCD) [123].

Even though the FID (3.5) may seem a bit different than the WD_C (3.4), they are defining the same thing, only when the cost matrix C is computed using the ℓ_2 norm and under the assumption that both distributions can be represented by their mean and covariance matrix. For the curiosity of readers, the proof is given in Section A.3.

Opposite to the FID, in our case, we want to compute the WD_C not between two minibatch of images but between two images alone. However, as [140] outlines, the WD_C defines a probability distribution similarity metric making it irrelevant to apply it directly over the images' pixels like [175] already did. It led to consider instead, extracting features from the images to represent them. Thus when using extracted features, we operate the WD_C in the meaningful framework making it an appropriate and promising tool applied in the best settings. The extracted features, by definition, contain the most meaningful information about the images, offering us more chances to end up with the definition of a meaningful distance between images. By using extracted features, we now control the complexity of the problem such that, even for high-resolution images, the size of the extracted features is controlled and tuned making it a scalable distance.

In this scenario, however, we are two times conditioned. First, the optimization of the image distance is conditioned to the efficiency of the feature extraction g_θ. Indeed, if an inefficient feature extractor is used, therefore no relevant distance can be derived from it. Secondly and most troublesome, when a meaningful distance is finally modeled, then for the adversarial perturbation generation, we need to get back from the feature space to the original image space using an inverse function of the feature extractor g_θ^{-1}. If one sees the feature extractor as an encoder projecting the images in a feature space of lower dimension, such that labels are well separated, in which we can easily approximate a distance, then its paired decoder g_θ^{-1} is needed to retrieve images from the feature vectors. Figure 3.2 presents a high-level overview of our proposition. Both g_θ and its inverse function g_θ^{-1} are crucial to the efficiency of the proposition.

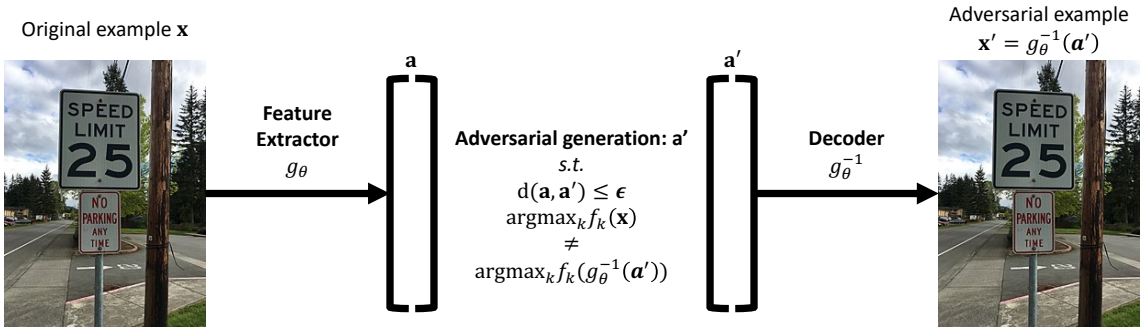


Figure 3.2: High-level overview of our meaningful adversarial attack proposition. The car image is from the CIFAR-10 dataset.

Even though the second limit of representing the images by their extracted features by using a g_θ^{-1} function, is probably the most troublesome, it still is a bit far ahead and most importantly considers a meaningful distance d already optimized, but one step at a time. First, we ignore the g_θ^{-1} decoder issue and focus only on approximating a meaningful distance d between images from their features.

3.4 Optimization

In this section, we detail the similarity metric framework we designed and derive an overall optimization problem

3.4.1 Turning the similarity metric into a distance

In order to represent the images $\mathbf{x}^{(i)} \in \mathbb{R}^{\mathrm{P}}$ we consider their features $\mathbf{a}^{(i)} \in \mathbb{R}^{\mathrm{f}}$ with $\mathrm{f} \ll \mathrm{P}$ extracted from an external model g_θ. For ease of reading we will refer to the extracted features $\mathbf{a}^{(i)}$ as an image even though its associated image is $\mathbf{x}^{(i)}$. As previously outlined, we select the Wasserstein distance ($\mathrm{WD_C}$) as the most appropriate function leading to a meaningful similarity metric between images. The Wasserstein distance ($\mathrm{WD_C}$) between two images $\mathbf{a}^{(s)}$ and $\mathbf{a}^{(t)}$ reads,

$$\begin{aligned} \mathrm{WD_C}(\mathbf{a}^{(s)}, \mathbf{a}^{(t)}) = \min_{\mathrm{T} \in \mathbb{R}^{\mathrm{f}\times\mathrm{f}}} \sum_{i=1}^{\mathrm{f}} \sum_{j=1}^{\mathrm{f}} \mathrm{T}_{i,j}\mathrm{C}_{i,j}, \\ \text{such that } \mathrm{T}1 = \mathbf{a}^{(s)} \text{ and } \mathrm{T}^{\mathrm{T}}1 = \mathbf{a}^{(t)}, \end{aligned} \tag{3.6}$$

with the matrix $\mathrm{C} \in \mathbb{R}^{\mathrm{f}\times\mathrm{f}}$ being the cost matrix containing the cost of pairing each feature entry of $\mathbf{a}^{(s)}$ to each feature entry of $\mathbf{a}^{(t)}$.

Recently, a lot of attention has been put to the regularised version of the Wasserstein Distance, called the Sinkhorn divergence. While the unregularized $\mathrm{WD_C}$ and the Sinkhorn divergence are theoretically equivalent candidates, the Sinkhorn divergence is more suited to similarity metric optimization. Indeed, the Wasserstein distance involves solving a linear programming problem, which can be computationally expensive and challenging to solve, especially for high-dimensional data. In contrast, the Sinkhorn divergence utilizes the Sinkhorn-Knopp algorithm, a more efficient iterative method for approximating the optimal transport plan between distributions. For the readers' curiosity, the development from the original regularized $\mathrm{WD_C}$ expression to the Sinkhorn-Knopp solver is given in Section A.2. Besides, the Sinkhorn divergence includes a regularization parameter helping to stabilize the optimization. This regularization encourages the transport plan to be smoother, improving convergence. By tuning the regularization parameter, Sinkhorn divergence allows for better control over the trade-off between the accuracy of the solution and the stability of the convergence. Overall the Sinkhorn divergence is a good choice as it is better suited to gradient-based optimization techniques commonly used in deep learning, and most likely used here. The Sinkhorn divergence between two images $\mathbf{a}^{(s)}$ and $\mathbf{a}^{(t)}$ reads,

$$\begin{aligned} \mathrm{WD}_{\mathrm{C}}^{\kappa}(\mathbf{a}^{(s)}, \mathbf{a}^{(t)}) = \min_{\mathrm{T} \in \mathbb{R}^{\mathrm{f}\times\mathrm{f}}} \sum_{i=1}^{\mathrm{f}} \sum_{j=1}^{\mathrm{f}} \mathrm{T}_{i,j}\mathrm{C}_{i,j} + \kappa \sum_{i=1}^{\mathrm{f}} \sum_{j=1}^{\mathrm{f}} \mathrm{T}_{i,j}(\log \mathrm{T}_{i,j} - 1), \\ \text{such that } \mathrm{T}1 = \mathbf{a}^{(s)} \text{ and } \mathrm{T}^{\mathrm{T}}1 = \mathbf{a}^{(t)}, \end{aligned} \tag{3.7}$$

with κ the regularization parameter. Rather than optimizing (3.7), usually, people rely on the dual formulation of the problem,

$$\min_{\mathrm{T} \in \mathbb{R}^{\mathrm{f}\times\mathrm{f}}} \max_{\alpha \in \mathbb{R}^{\mathrm{f}}, \beta \in \mathbb{R}^{\mathrm{f}}} \mathcal{L}(\mathbf{a}^{(s)}, \mathbf{a}^{(t)}, \alpha, \beta), \text{ such that } \alpha \geq 0_{\mathrm{f}}, \beta \geq 0_{\mathrm{f}} \text{ with} \tag{3.8}$$

$$\begin{aligned} \mathcal{L}(\mathbf{a}^{(s)}, \mathbf{a}^{(t)}, \alpha, \beta) = \min_{\mathrm{T} \in \mathbb{R}^{\mathrm{f}\times\mathrm{f}}} \sum_{i=1}^{\mathrm{f}} \sum_{j=1}^{\mathrm{f}} \mathrm{T}_{i,j}\mathrm{C}_{i,j} + \kappa \sum_{i=1}^{\mathrm{f}} \sum_{j=1}^{\mathrm{f}} \mathrm{T}_{i,j}(\log \mathrm{T}_{i,j} - 1) \\ + \alpha^{\mathrm{T}}(\mathrm{T}1_{\mathrm{f}} - \mathbf{a}^{(s)}) + \beta^{\mathrm{T}}(\mathrm{T}^{\mathrm{T}}1_{\mathrm{f}} - \mathbf{a}^{(t)}), \end{aligned}$$

with $\alpha \in \mathbb{R}^{\mathrm{f}}$ and $\beta \in \mathbb{R}^{\mathrm{f}}$ the dual variables. This dual problem is efficiently solved using the gradient descent iterations of the Sinkhorn-Knopp solver, detailed in Section A.2.

Up until now, we supposed the cost matrix C between the image features known, however, we don't know such a suitable distance between the features living in $\mathbb{R}^{\mathrm{f}}$ it is precisely the cost matrix

that contains the "meaningful" information defining the meaningful distance. The cost matrix optimization defines our similarity metric optimization.

To make the similarity metric a distance, it must respect the axioms of a distance (3.1). Therefore, the cost matrix is assumed to belong to the cone of distance matrices defined as

$$\mathcal{C} = \left\{ \mathrm{C} \in \mathbb{R}^{\mathrm{f}\times\mathrm{f}} : \forall 1 < i,j,k \leq d, \mathrm{C}_{ii} = 0, \mathrm{C}_{ij} = \mathrm{C}_{ji}, \mathrm{C}_{ij} \leq \mathrm{C}_{ik} + \mathrm{C}_{kj} \right\}. \tag{3.9}$$

3.4.2 Turning the distance into a Linear Program (LP) problem

To ease the reading of the distance optimization problem, we rewrite the Sinkhorn divergence incorporating the cost matrix constraints (3.9) defining the Sinkhorn divergence as a distance, as a linear program (LP) problem that is,

$$\mathrm{WD}_{\mathrm{C}}^{\kappa}(\mathbf{a}^{(s)}, \mathbf{a}^{(t)}) = \begin{cases} \min\limits_{\mathbf{c},\mathbf{t}^{(i)}} & \mathbf{c}^{\top}\mathbf{t}^{(i)} + \kappa \mathbf{t}^{(i)^{\top}}(\log \mathbf{t}^{(i)} - 1) \\ \text{such that} & \mathrm{A}\mathbf{t}^{(i)} = \mathbf{b}^{(i)}, \\ & \mathrm{C}_{ii} = 0, \mathrm{C}_{ij} = \mathrm{C}_{ji}, \\ & \mathrm{C}_{ij} \leq \mathrm{C}_{ik} + \mathrm{C}_{kj}, \forall 1 < i,j,k \leq p\,, \end{cases} \tag{3.10}$$

with $\mathbf{t}$ = flatten(T) the vectorized version of the Optimal Transport Plan T between $\mathbf{a}^{(s)}$ and $\mathbf{a}^{(t)}$, $\mathbf{c}$ = flatten(C) the vectorized version of the cost matrix C, $\mathbf{b} \in \mathbb{R}^{2\mathrm{f}}$ the concatenation of both feature vectors $\mathbf{b} = (\mathbf{a}^{(s)^{\mathrm{T}}}, \mathbf{a}^{(t)^{\mathrm{T}}})^{\mathrm{T}}$ and $\mathrm{A} \in \mathbb{R}^{2\mathrm{f}\times\mathrm{f}^2}$ a fixed binary matrix encoding the right pairing between the transport plan and the feature's entry in $\mathbf{b}$ to respect the original constraints.

Now that the canvas is formally set and we clarified that the optimization of the meaningful distance between images boils down to optimizing $\mathrm{C} \in \mathcal{C}$ we turn its optimization problem into a metric learning problem.

3.4.3 Leveraging a Quadratically Constrained Quadratic Program (QCQP) metric learning problem

Since the images are labeled we could use these labels to solve a metric learning problem from a pseudo-truth distance $w(\cdot,\cdot) : \mathbb{R}^{\mathrm{f}} \times \mathbb{R}^{\mathrm{f}} \mapsto \mathbb{R}$ between images based on their labels. According to such pseudo-truth distance, we then look for a cost matrix C such that its conditioned WD_{C} closer images from the same class while pushing away images from different classes. For example, in [39], Cuturi explored the Wasserstein Distance from a similar metric learning angle but proposed to hand-craft this pseudo-truth distance $w^{(i)} = w(\mathbf{a}^{(s)}, \mathbf{a}^{(t)})$ between two images $\mathbf{a}^{(s)}$ and $\mathbf{a}^{(t)}$ such that $w^{(i)} = 1/\mathrm{N}k$ if both images belong to the same label or $w^{(i)} = -1/\mathrm{N}k$ if they belong to different label, with k being a chosen number of neighbors to consider. Given such pseudo-truth distance $w^{(i)}$ between pairs of images, we can look for the cost matrix encoding the Wasserstein Distance to match these distances,

Given the matrix A, for one couple of points $\mathbf{b}^{(i)} = [\mathbf{a}^{(s)^{\mathrm{T}}}, \mathbf{a}^{(t)^{\mathrm{T}}}]^{\mathrm{T}}$ and their pseudo-truth distance $w^{(i)}$, our ultimate goal is to find C and the $\mathrm{T}^{(i)}$ solution of the following LP problem

$$\mathcal{J}(\mathbf{a}^{(s)}, \mathbf{a}^{(t)}) = \begin{cases} \min\limits_{\mathbf{c},\mathbf{t}^{(i)}} & \mathbf{c}^{\top}\mathbf{t}^{(i)} \left(= \mathrm{Tr}(\mathrm{C}^{\top}\mathrm{T}^{(i)})\right) \\ \text{such that} & \mathrm{A}\mathbf{t}^{(i)} = \mathbf{b}^{(i)} \\ & \mathbf{c}^{\top}\mathbf{t}^{(i)} = w^{(i)} \\ & \mathrm{C}_{ii} = \kappa, \mathrm{C}_{ij} = \mathrm{C}_{ji}, \\ & \mathrm{C}_{ij} \leq \mathrm{C}_{ik} + \mathrm{C}_{kj}, \forall 1 < i,j,k \leq p\,. \end{cases} \tag{3.11}$$

The red term highlights our real implicit goal that is, we are looking for a WD_{C} that matches the pseudo-truth distance for these two images.

Now we leverage $\mathcal{J}(\mathbf{a}^{(s)}, \mathbf{a}^{(t)})$ to the whole dataset of images' features $\mathcal{D} = \{\mathbf{a}^{(i)}, y^{(i)}\}_{i=1}^{\mathrm{N}}$. We are interested in the following $\mathcal{J}_{\mathcal{D}}$ Quadratically Constrained Quadratic Program (QCQP) problem, given the binary encoding matrix $\mathrm{A} \in \mathbb{R}^{2\mathrm{f}\times\mathrm{f}^2}$ matrix, $\mathrm{B} \in \mathbb{R}^{2\mathrm{f}\times\mathrm{N}}$ a matrix whose columns are the

vectors $\mathbf{b}^{(i)}$, $\mathbf{w}$ a vector in $\mathbb{R}^N$ a concatenation of all the $w^{(i)}$ and $\tau \in \mathbb{R}^{f^2 \times N}$ a matrix whose columns are all the optimal transport plans $\mathbf{t}^{(i)}$.

$$\mathcal{J}_{\mathcal{D}} = \begin{cases} \min\limits_{\mathbf{c},\tau} & \mathbb{1}^\top \tau^\top \mathbf{c} \left(= \sum_{i=1}^{N} \mathbf{c}^\top \mathbf{t}^{(i)}\right) \\ \text{such that} & A\tau = B \\ & \tau^\top \mathbf{c} = \mathbf{w} \\ & C_{ii} = \kappa, C_{ij} = C_{ji}, \\ & C_{ij} \leq C_{ik} + C_{kj}, \forall 1 < i, j, k \leq p\,. \end{cases} \tag{3.12}$$

The red term highlights our real implicit goal that is, we are looking for a WD_C that matches the pseudo-truth distance for all pairs of images.

$\mathcal{J}_{\mathcal{D}}$ defines a bi-level optimization in which every transportation plan $\mathbf{t}$ included in τ are intrinsically defined by the WD_C and thus conditioned to the cost matrix $\mathbf{c}$. However, the cost matrix objective $\mathbf{c}$ is to make the overall WD_C distance closer to the pseudo-truth distances $w(\cdot,\cdot)$ included in $\mathbf{w}$ therefore, $\mathbf{c}$ is only conditioned to the quality of the extracted features and the modeling of the pseudo-truth distance w from the labels.

3.4.4 Algorithm

In this section, we present the proposed relaxation solving the original problem (3.12), and the algorithm solving it.

The original objection $\mathcal{J}_{\mathcal{D}}$ (3.12) is hard to solve because of the non-convex $\mathbb{1}$ indicator function and the strict $\tau^\top \mathbf{c} = \mathbf{w}$ constraint that hard to respect in practice due to the two variables optimized. Therefore we replace them with a more manageable loss function,

$$\begin{cases} \min\limits_{\mathbf{c},\tau} & \frac{1}{m} \sum_{i=1}^{m} ||\mathbf{c}^\top \mathbf{t}^{(i)} - w^{(i)}||_2^2 \\ \text{such that} & A\tau = B \\ & C_{ii} = \kappa, C_{ij} = C_{ji}, \\ & C_{ij} \leq C_{ik} + C_{kj}, \qquad \forall 1 < i, j, k \leq p\,. \end{cases} \tag{3.13}$$

Equation (3.13) optimizes the transport plan $\mathbf{t}^{(i)}$ such that they are valid with respect to the WD_C condition, and optimizes the cost matrix $\mathbf{c}$ to lower the distance of the very WD_C to the pseudo-truth distance $w^{(i)}$ by minimizing their ℓ_2 norm. Equation (3.13) is optimized in two steps, one for each variable. We first will rely on the Linear Solvers available in the library POT (Python for Optimal Transport) [58] to resolve all the transport plans $\mathbf{t}^{(i)}$ forming τ. Then we fix these variables as parameters to optimize the problem over the cost matrix $\mathbf{c}$ by using a sequence of cost matrix solutions to converge towards the optimal one. Finally, a rescaling of $\mathbf{c}$ is performed to ensure the constraint of C and thus produces a well-defined WD_C distance.

At every n optimization, the plans are found using POT's solvers, then frozen to improve the cost matrix $\mathbf{c}$ using mini-batch stochastic gradient descent with gradients from

$$\mathcal{H}_{\mathbf{c}^{(n)}} = \min_{\mathbf{c}^{(n)}} \frac{1}{m} \sum_{i=1}^{m} ||\mathrm{WD}^{\kappa}_{\mathbf{c}^{(n)}}(\mathbf{a}^{(s)^i}, \mathbf{a}^{(t)^i}) - w^{(i)}||_2^2, \tag{3.14}$$

With m being the mini-batch size. We use problem $\mathcal{H}_{\mathbf{c}^{(n)}}$ to improve $\mathbf{c}$ by creating the sequence of solutions

$$\mathbf{c}^{(n+1)} = \mathbf{c}^{(n)} - \gamma \nabla_{\mathbf{c}} \mathcal{H}(\mathbf{c}^{(n)}), \tag{3.15}$$

with γ the learning rate. This sequence of solutions aims at converging toward the optimal solution C^* parameterizing a meaningful Wasserstein distance WD_{C^*} for images by using their extracted features.

The full optimization scheme is sketched in Algorithm 9.

Algorithm 9 Meaningful Wasserstein distance between images optimization.

Require: Labeled dataset of image extracted features $\mathcal{D} = \{\mathbf{a}^{(i)}, y^{(i)}\}_{i=1}^{\mathrm{N}}$; pseudo-truth distance $w(\cdot,\cdot)$; step-size γ, Optimizer Optim; Batch size B; a number of max iteration M

1: $\mathbf{a}^{(i)} = \mathrm{softmax}(\mathbf{a}^{(i)})$ Rebalance every feature vector $\mathbf{a}^{(i)}$ to make it a probability distribution
2: $\mathrm{C} = \mathbb{1}_{\mathrm{f}\times\mathrm{f}}$ Initialize the cost matrix C
3: **for** k = 0 to M **do**
4: loss = 0
5: **for** m = 0 to B **do**
6: $\mathbf{a}^{(i)} \sim \mathcal{D}; \mathbf{a}^{(j)} \sim \mathcal{D}$ Get pair of feature $(\mathbf{a}^{(i)}, \mathbf{a}^{(j)})$ from $\mathcal{D}$
7: $d = \mathrm{WD}_{\mathrm{C}}(\mathbf{a}^{(i)}, \mathbf{a}^{(j)})$ Solve (3.13) w.r.t $\mathbf{t}$ using POT
8: $\mathrm{loss} = \mathrm{loss} + ||d - w(\mathbf{a}^{(i)}, \mathbf{a}^{(j)})||_2^2$
9: **end for**
10: $\mathrm{C} \leftarrow \mathrm{Optim}(\nabla_{\mathrm{C}}\mathrm{loss})$ Update $\mathbf{c}$ from (3.13)
11: Rescaling C to ensure $\mathrm{C} \in \mathcal{C}$ Making the updated WD_{C} a distance
12: **end for**
13: **return** C parameter of WD_{C} the optimized meaningful distance between image

3.5 Experiments

Before tackling more practical applications in our experiments, we first need to study the mechanisms and behavior of cost matrix optimization on a toy problem.

3.5.1 Toy problem

For this purpose, we artificially generate an actual ground-truth cost matrix C^* to recover, and a sample of N pairs of points $\mathcal{D} = \{(\mathbf{a}^{(1)^i}, \mathbf{a}^{(2)^i})\}_{i=1}^{\mathrm{N}}$, where $\mathbf{a}^{(1)}, \mathbf{a}^{(2)} \in \mathbb{R}^{\mathrm{f}}$. By relying on the ground-truth cost matrix C^*, we know the corresponding true distance $\mathrm{WD}_{\mathrm{C}^*}(\mathbf{a}^{(1)^i}, \mathbf{a}^{(2)^i}) = w^{(i)}$. These distances are crucial and allow us to recover C^* by optimizing the problem (3.13). In this toy problem, we indeed bypass the construction of pseudo-distance through Metric Learning. Nevertheless, this provides us with an initial reference point and an overview of our proposed distance optimization framework.

In reality, we may not always access feature vectors $\mathbf{a}$ that are defining probability distribution. Therefore we ignored line 1 of algorithm 9 and we studied the Toy problem under balanced and unbalanced pair of points $(\mathbf{a}^{(1)^i}, \mathbf{a}^{(2)^i})$that is,

$$\begin{aligned} &\text{Balanced Data: } \mathbf{a}^{(1)^i}, \mathbf{a}^{(2)^i} \in \mathbb{R}^{\mathrm{K}} \text{ with } \mathbf{a}^{(1)^i\mathrm{T}}\mathbb{1} = \mathbf{a}^{(2)^i\mathrm{T}}\mathbb{1},\ \forall i \in \{1,\dots,\mathrm{N}\}, \\ &\text{Unbalanced Data: } \mathbf{a}^{(1)^i}, \mathbf{a}^{(2)^i} \in \mathbb{R}^{\mathrm{K}} \text{ with } \mathbf{a}^{(1)^i\mathrm{T}}\mathbb{1} \neq \mathbf{a}^{(2)^i\mathrm{T}}\mathbb{1},\ \forall i \in \{1,\dots,\mathrm{N}\}. \end{aligned} \tag{3.16}$$

To stop the optimization at the point of convergence, we have chosen to use the relative norm of the cost matrix,

$$\text{If } \frac{||\mathrm{C}^{n-1} - \mathrm{C}^n||_2}{||\mathrm{C}^{n-1}||_2} \leq \alpha \text{ Then: STOP.} \tag{3.17}$$

In this notation, and given that we only consider symmetric matrices, by C^{n-1} we refer to the upper triangular part of the cost matrix at the $n-1$st update of the same parameters, and by C^n we refer to the upper triangular part of the cost matrix at the nth update of the same parameters.

This criterion is the least biased and most robust measure we have to detect the convergence of the parameter to be optimized, the cost matrix C (in reality, our parameters correspond only to the $n \times (n-1)/2$ values of the upper triangular part of the cost matrix due to the symmetry condition of C).

Relaxations

For computational reasons, we have simply omitted the inequality constraint on C, the third constraint defining a distance. Furthermore, in [13], Bellet et al. informs us a strict respect of these constraints is still under debate, in the pursuit of constructing a distance metric closer to human perception. Indeed, according to cognitive science research [5], [129], [161], [162], the enforcement of symmetry and the triangle inequality appears to be too rigid for our human perception of similarity between objects. It seems that humans are more flexible in our judgment of similarity than a mathematically perfectly defined distance would suggest.

Is POT a relevant Wasserstein distance solver?

To verify the efficiency of the POT toolbox, we handcrafted a custom mini-batched version of the Sinkhorn-knopp solver to see, which of the two is the most efficient.

Table 3.1: Time Consumption (in seconds) for the optimization of equation (3.13) with the POT solver and our custom Sinkhorn solver under different learning rate with f = 4.

(a) Balanced data.

LR	POT	Custom
1e-3	73.66	152.16
5e-3	24.26	155.78
1e-2	10.91	**140.62**
5e-2	**10.79**	171.27
1e-1	11.44	360.43
5e-1	14.94	138.69
1	64.71	350.12

(b) Unbalanced data.

LR	POT	Custom
1e-3	96.01	**220.23**
5e-3	34.79	445.73
1e-2	17.17	454.97
5e-2	18.04	812.61
1e-1	**14.67**	616.96
5e-1	25.66	395.7
1	25.98	444.54

According to Table 3.1 presenting the results of our optimization on this toy problem for the two solvers under different learning rates, we can completely discard our custom Sinkhorn solver and completely trust the POT toolbox. Even for a small dimension, our custom Sinkhorn solver requires way too much time to achieve an acceptable level of quality, to consider it a relevant solver.

Is the optimization fast enough with small dimensions?

If the optimization framework were to be successful, it would at least converge to a solution very fast(10 seconds max) under the simplest settings (such as a small size of features f). Table 3.1 gives us a first insight into the time required to find the optimal solution with a small f = 4 using the POT solver, but the performances can be greatly enhanced by more finely tuning the hyperparameters. Table 3.2 displays the performances of the cost matrix optimization when using the best (POT) solver of the WD_C problem under different learning rates. Under a small dimension (f = 4) for both balanced and unbalanced data, the optimization converges pretty quickly. Indeed, for balanced data, when tuning $\gamma = 0.01$ the optimization converges to the exact solution in 8 seconds, and for unbalanced data, when tuning $\gamma = 0.05$ the best solution is approximated in 14 seconds.

Note that all these experiments were performed on a MacBook Pro with 2,3 GHz Intel Core i9, 8 cores however, transitioning to GPU computation would drastically enhance our performance as it usually is a great optimization tool when relying on SGD. These first results confirm the minimum acceptable threshold.

Table 3.2: Optimization of the cost matrix with a small dimension (f = 4), using the POT Linear Programming solver for both balanced and unbalanced data, under different learning rates. The optimization error is estimated using the ℓ_2 norm between the found solution and the true cost matrix.

(a) Balanced data.

LR	Error	Time
1e-4	0.061	59.69
5e-4	0.056	53.54
1e-3	0.019	52.8
5e-3	0	18.53
1e-2	**0**	**8.75**
5e-2	0	9.73
0.1	0	10.81
0.5	0	13.36
1	0.004	52.12
5	0.1	58.65
10	0.127	55.52
50	0.171	53.09

(b) Unbalanced data.

LR	Error	Time
1e-4	0.048	78.8
5e-4	0.018	115.3
1e-3	0.045	82.6
5e-3	0.014	82.3
1e-2	0.001	19.5
5e-2	**0**	**14.0**
0.1	0	15.8
0.5	0.0517	15.9
1	0.397	16.7
5	9.082	15.9
10	22.046	7.6
50	111.41	7.7

Is there a trade-off between the quality of the found solution and the time required to find it?

Thanks to Table 3.2 we know that the optimization can reach the exact solution very quickly in small dimensions with either balanced or unbalanced data. However, when the feature size increases, things are a bit different.

Table 3.3: Optimization of the cost matrix with a medium dimension (f = 16), using the POT Linear Programming solver for both balanced and unbalanced data, under different learning rates. The optimization error is estimated using the ℓ_2 norm between the found solution and the true cost matrix.

(a) Balanced data.

LR	Error	Time
1e-4	0.03	0.13
5e-4	0.021	44.18
1e-3	0.015	44.27
5e-3	0.001	44.49
1e-2	0.001	41.47
5e-2	**0**	**13.77**
0.1	0.015	43.94
0.5	0.054	44.01
1	0.091	44.62
5	0.273	44.69
10	0.397	44.15
50	1.309	44.74

(b) Unbalanced data.

LR	Error	Time
1e-4	0.023	44.98
5e-4	0.023	99.97
1e-3	0.012	99.64
5e-3	0.003	95.52
1e-2	0.003	95.63
5e-2	0.008	96.25
0.1	0.018	99.82
0.5	0.156	99.8
1	0.099	100.25
5	**2.348**	**9.17**
10	4.978	9.05
50	24.83	8.79

With balanced data, the optimization process is smoother, and from Table 3.3a and Table 3.4a, we don't see a trade-off between the time required for optimization and the quality of the solution. However, we observe that the learning rate is an extremely sensitive parameter.

However, with unbalanced data, things are more interesting as the optimization becomes more complex. From Table 3.3b, we can observe a trade-off between the time required for optimization and the quality of the solution. Indeed, with medium-sized features, when selecting $\gamma = 0.005$ the

best solution is reached in 95 seconds although, when setting $\gamma = 5$, a descent solution is reached in under 10 seconds. We draw the same conclusion for high-sized features using Table 3.4b. When setting $\gamma = 0.005$, the best solution requires 113 seconds to be found, while an acceptable solution is retrieved at $\gamma = 5$ in only 12 seconds.

Table 3.4: Optimization of the cost matrix with a medium dimension (f = 64), using the POT Linear Programming solver for both balanced and unbalanced data, under different learning rates. The optimization error is estimated using the ℓ_2 norm between the found solution and the true cost matrix.

(a) Balanced data.

LR	Error	Time
1e-4	0.006	0.13
5e-4	0.007	53.65
1e-3	0.0007	53.72
5e-3	0.006	53.5
1e-2	0.006	53.5
5e-2	0.007	53.67
0.1	0.013	53.69
0.5	0.059	56.75
1	0.111	56.85
5	0.574	57.22
10	1.034	57.1
50	4.584	62.69

(b) Unbalanced data.

LR	Error	Time
1e-4	0.007	9.15
5e-4	0.006	110.24
1e-3	0.005	110.1
5e-3	0.003	113.33
1e-2	0.003	115.26
5e-2	0.004	115.33
0.1	0.008	239.51
0.5	0.032	144.19
1	0.018	139.82
5	0.571	12.87
10	1.211	12.56
50	6.168	11.76

We understand through Table 3.3 and Table 3.4 that a trade-off between the quality of the found solution and the time required to find it only with unbalanced data.

Is the optimization framework robust to both balanced and unbalanced data?

We see from Table 3.2, 3.3 and 3.4 that a decent solution can be reached either with balanced or unbalanced data. As the problem is more complex with unbalanced data, the quality of the found solution is lowered, and the time required to find it is increased, but overall the optimal solution can be decently approximated in a reasonable amount of time.

Overall the optimization framework is robust to both balanced and unbalanced data, which is a desirable property when considering more realistic applications in which we are not likely to have balanced image feature vectors.

Is the learning rate a sensitive parameter?

It is very clear from Table 3.2, 3.3 and 3.4 that the learning rate is a very critical parameter. Even for unbalanced data, as a trade-off is observed, it is the learning rate parameter that allows us to navigate within the trade-off between the computational time and the solution quality.

We draw the same conclusion with balanced data. If the learning rate γ is not correctly chosen, then the best solution cannot be retrieved. This conclusion is especially illustrated with the medium-sized features under balanced data presented in Table 3.3a. When setting $\gamma = 0.05$, the best solution is reached in 13 seconds, while with $\gamma = 0.1$, the optimization requires 43 seconds to reach a worse solution.

According to the many conducted optimizations, we observed that the proposed optimization framework is sensitive to the learning rate γ parameter. When it is applied to more realistic applications, a significant time must be allocated to find the most appropriate learning rate γ.

Is the mini-batch size a sensitive parameter?

In stochastic gradient decent optimizations the mini-batch can sometimes be a sensitive parameter. Indeed, when not high enough, the gradients may not always point in the best direction to reach the optimal solution, leading only to a local solution.

Table 3.5: Optimization of the cost matrix for a small dimension (f = 4), medium dimension (f = 16), and high dimension (f = 64), using the POT Linear Programming solver for both balanced and unbalanced data, under different minibatch sizes B. The optimization error is estimated using the ℓ_2 norm between the found solution and the true cost matrix.

(a) f = 4.

B	Error	Time
4	0	0.72
16	0	1.27
32	0	2.15
64	0	3.97
128	0	6.8
256	0	14.9
512	0	29.96
1024	0	42.59

(b) f = 16.

B	Error	Time
4	0.016	73.15
16	0.01	52.55
32	0.01	47.09
64	0	16.75
128	0	26.6
256	0	43.83
512	0	43.46
1024	0	44.3

(c) f = 64.

B	Error	Time
4	0.014	158.29
16	0.011	105.67
32	0.009	60.57
64	0.008	57.06
128	0.007	57.11
256	0.006	56.23
512	0.006	54.91
1024	0.006	55.18

In Table 3.5a we perform the proposed optimization for various sizes of mini-batch. The mini-batch size has absolutely no influence on the quality of the solution found, except for the time consumption. We recall that the displayed results show performances on CPU-performed optimizations, while the mini-batch strategy is especially suited to GPU computations, we do not think it is worth considering the mini-batch size as a time consumption sensitive hyper-parameter.

On the other hand, from Table 3.5c and 3.5b, we can observe that in medium and large dimensions, the mini-batch size becomes a critical parameter. Indeed, it is only when using a relatively large mini-batch size (f = 128) that we achieve rapid convergence. Beyond this required minimum size, increasing the mini-batch size doesn't further improve optimization results.

Therefore, we now know that a sufficiently large mini-batch size is required to ensure the reaching of the optimal solution for realistic feature sizes. However, it won't ultimately produce better results according to its increase.

Does the magnitude of the ground truth cost matrix to recover have any impact on the optimization?

The magnitude of the values within the ground truth cost matrix to recover may severely affect the optimization, as it could easily lead to finding a descent local solution but prevent finding the optimal solution C^*. Thus we performed the same optimization with the cost matrix multiplied by a magnitude coefficient factor which increases the distance between the points values and the cost matrix' entries to recover.

At first, Table 3.6 tells us that the magnitude of the cost matrix to be retrieved has a significant impact on the quality of our optimization.

However, by increasing the learning rate, Table 3.7 demonstrates that it adjusts the optimization to the magnitude of the optimal cost matrix to be found.

Even if in small and medium dimensions, increasing the learning rate led to enhancing the optimization, by increasing the learning rate too much, Table 3.8 reveals that optimization can be damaged in the case of a large dimension.

Once again, the learning rate is a critical factor for successful optimization. We will need to choose it very carefully since it adjusts the optimization for both changes in dimension and changes in the magnitude of the matrix to be retrieved.

Table 3.6: Optimization of the cost matrix for both a small (f = 4) and a medium (f = 16) dimension, using the POT Linear Programming solver for balanced data with different magnitudes of the original cost matrix, employing a small learning rate ($\gamma = 0.05$).

(a) Small dimension (f = 4).

Magnitude	Error	Time
1	0	11.45
2	0	7.13
5	0.002	11.13
10	0.006	18.36
50	3.022	43.81

(b) Medium dimension (f = 16).

Magnitude	Error	Time
1	0	13.77
2	0.001	38.98
5	0.012	47.73
10	0.068	49.75
50	1.126	47.76

Table 3.7: Optimization of the cost matrix for both a small (f = 4) and a medium (f = 16) dimension, using the POT Linear Programming solver for balanced data with different magnitudes of the original cost matrix, employing a medium learning rate ($\gamma = 0.5$).

(a) Small dimension (f = 4).

Magnitude	Error	Time
1	0	11.19
2	0	9.56
5	0	8.4
10	0.002	7.32
50	0.025	12.78

(b) Medium dimension (f = 16).

Magnitude	Error	Time
1	0.049	51.28
2	0.08	44.9
5	0.001	19.71
10	0.002	23.38
50	0.147	50.67

Should we ignore too close neighbors points to ensure a good optimization?

To have a somewhat realistic toy problem, the question of excluding generated points that are too close has been addressed. Would we achieve better optimization of the cost matrix if we only consider points that are sufficiently distant according to the optimal WD_{C^*} using the true cost matrix?

Judging from Table 3.9 displaying experiments results in both small and large dimensions, the optimization does not seem to be impacted at all by the rejection of points that are too close according to the true distance to be retrieved. Therefore, we can consider this parameter as insignificant for our upcoming more realistic applications.

From which dimension can we no longer guarantee a fast and reliable optimization?

It is worth noting that for larger feature dimensions, as we saw, a more detailed study of the learning rate would be more appropriate. However, these initial figures provide us with a preliminary understanding of the limit of the proposed distance optimization.

Even though for a dimension of f = 128, the computational time to reach convergence remains quite reasonable, Table 3.10a and Table 3.10b indicates that starting from a dimension of f = 256, optimizing the cost matrix becomes more challenging. It is important to note that for these larger dimensions, a more detailed study of various optimization parameters (mini-batch size, magnitude, learning rate, or even a learning rate annealing strategy) would be appropriate to achieve the most efficient optimization. Nevertheless, these figures remain illustrative and provide us with practical insights into the optimization process.

After addressing all these questions, we gained insights into the specificity of cost matrix optimization. We understand that the optimization works relatively well for both balanced and unbalanced data, but it takes, on average, 1.5 times longer to converge with unbalanced data. We know that using the POT solver is necessary. We've determined that rejecting points that are too close has no influence. The size of the mini-batch B is a relatively critical parameter to ensure op-

Table 3.8: Optimization of the cost matrix for a large dimension (f = 64), using the POT Linear Programming solver for balanced data with varying magnitudes of the original cost matrix, employing both a small ($\gamma = 0.5$) and a large ($\gamma = 5$) learning rate.

(a) $\gamma = 0.5$.

Magnitude	Error	Time
1	0.058	61.58
2	0.049	66.62
5	0.041	59.88
10	0.045	57.43
50	0.313	56.05

(b) $\gamma = 5$.

Magnitude	Error	Time
1	0.527	62.4
2	0.531	62.48
5	0.537	61.06
10	0.501	61.55
50	0.365	59.74

Table 3.9: Optimization of the cost matrix for both small and large dimensions, using the POT Linear Programming solver for balanced data with varying minimum distance width ρ thresholds accepted between points.

(a) f = 4.

ρ	Error	Time
0	0	49.2
0.001	0	44.6
0.005	0	41.3
0.01	0	40.5

(b) f = 64.

ρ	Error	Time
0	0.007	42.3
0.001	0.007	40.8
0.005	0.007	40.5
0.01	0.007	40.1

timization convergence, along with the magnitude of the solution to be found. However, we can manage these two parameters by choosing an appropriate learning rate, which seems to be the most critical hyperparameter of this optimization framework.

The outcome of this toy problem is positive; it demonstrates that within a certain working framework and under specific conditions (controlled dimension of the features f), we can converge towards the solution C^* in an acceptable time frame. These findings are promising and allow us to tackle a slightly more realistic problem, such as experimenting with image features from the MNIST dataset as a next step.

3.5.2 MNIST images application

Now we gained a preliminary understanding of the optimization problem from the toy problem we can tackle a slightly more concrete application using images from the MNIST dataset [46]. Being one of the easiest image datasets, we considered the MNIST as the first more practical application of our distance optimization framework, which would then allow us to go further into more complex and realistic image datasets.

To reduce the dimensionality of the images and utilize only the most relevant information, we choose to employ discriminative features (referred to as features) extracted from the penultimate layer of a classifier neural network trained on these images. By doing so, we construct a fairly accurate classifier (achieving a minimum of 99% classification accuracy) to ensure that we retain the most relevant features of the images (those responsible for achieving the desired classification accuracy).

Let $f : \mathbb{R}^{\mathrm{P}} \mapsto \mathbb{R}^{\mathrm{C}}$ be a classifier that takes input images of P pixels and assigns them a probability vector over the C possible classes. Since f is a neural network, we can redefine it as $f(x) = g(h(x))$, where $h : \mathbb{R}^{inputDimension} \mapsto \mathbb{R}^{\mathrm{f}}$ is a non-linear feature extraction function, and $g : \mathbb{R}^{\mathrm{f}} \mapsto \mathbb{R}^{\mathrm{C}}$ is the final linear layer of f responsible for the final classification.

This feature extraction process also enables us to increase or lower the feature size f. More precisely, we consider the features from the penultimate layer after its activation function, which is the ReLU (Rectified Linear Unit, $relu(x) = \max(x, 0)$). This activation function has a promising

Table 3.10: Optimization of the cost matrix for various dimensions f, using the POT Linear Programming solver for balanced data with both a small learning rate $\gamma = 0.01$ and a high learning rate $\gamma = 10$.

(a) $\gamma = 0.01$.

f	Error	Time
4	0	15.61
16	0.02	53.52
64	0.006	61.33
128	0.003	95.3
256	0.002	220.39

(b) $\gamma = 10$.

f	Error	Time
4	0.117	64.05
16	0.427	54.75
64	0.924	68.51
128	0.48	107.09
256	0.213	241.18

effect for our purposes, as during the classifier training, it essentially nullifies unused nodes in the penultimate layer to achieve better classification performance. As a result, $h(x) \in \mathbb{R}^f$ will be a very sparse vector. This sparsity is advantageous for us, as it achieves meaningful dimensionality reduction of the features (selecting them based on their relevance to the classification task). It further reduces the complexity of the WD_C inner optimization (the transport plan objective).

3.5.3 Analysis of the MNIST extracted features

As good scientific behavior, data analysis of the features must be performed. Even though we don't have any tools to perform such analysis, we empirically look at the features' entries activation to see if we have some modes emerging according to the labels, which would mean relevant features.

In practice, we observe that for a layer of size f = 16, we end up with non-zero features with an average size of 5. For a layer of size f = 32, we ultimately have non-zero features with an average size of 10. Figure 3.3 displays the distribution of these features when setting f = 16.

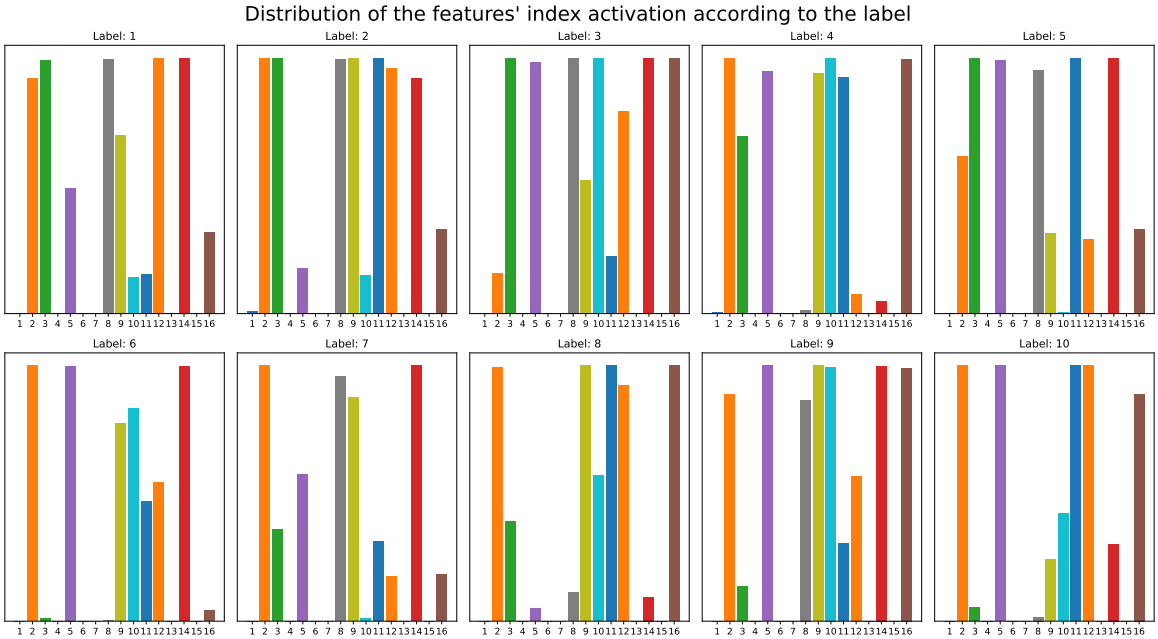


Figure 3.3: Distribution of the features when setting f = 16 according to the 10 possible labels.

From Figure 3.3, we see that only a few components of the features are non-zero. Therefore, we understand that the classifier's training, to effectively separate the classes, has been successful. Indeed, we observe that for each class, only a few dimensions dominate the distribution of features, and we never observe complete overlap between two sets of dominant dimensions among the classes. Along with the label's feature dimensions activation, we considered the mean feature

of each class to assess and confirm the possible modes of each label. Figure 3.4 displays the mean feature vector of each label.

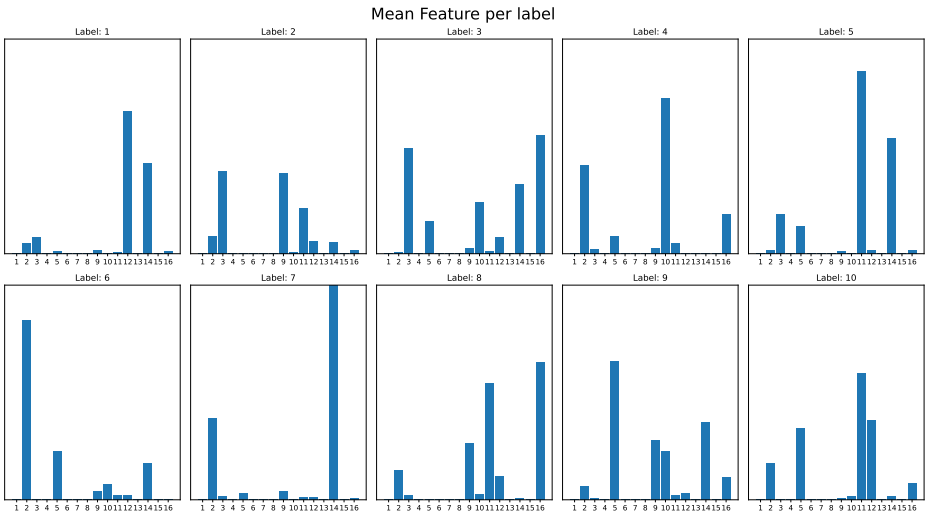


Figure 3.4: Distribution of the features when setting f = 16 according to the 10 possible labels.

The conclusion drawn from Figure 3.3 is strengthened by Figure 3.4. Indeed, there is no mean feature vectors are the same.

Overall, both figures confirm that each class, in the feature space $\mathbb{R}^{\mathrm{f}}$, occupies an isolated local region of the feature space, involving only a few dimensions, and is situated far enough from one another to be accurately distinguished. These features render an almost linear problem and consequently allow us to efficiently optimize our cost matrix.

Before tackling our proposed optimization framework, we attempted to eliminate the step of constructing the pseudo-ground truth distance $w : \mathbb{R}^{\mathrm{f}} \times \mathbb{R}^{\mathrm{f}} \mapsto \mathbb{R}$. Instead, we aimed to treat our Wasserstein Distance WD_{C} directly as the distance to optimize within a metric learning optimization framework.

3.5.4 Optimizing the Wasserstein distance with a metric learning objective

Rather than relying on a pseudo-truth distance $w : \mathbb{R}^{\mathrm{f}} \times \mathbb{R}^{\mathrm{f}} \mapsto \mathbb{R}$ that we don't possess for the MNIST features, we consider our optimization framework of WD_{C} as a metric learning problem in itself.

Thus at each iteration, the cost matrix is optimized according to the triplet loss objective [164],

$$\min_{\mathrm{C}} \mathrm{relu}(\mathrm{WD}_{\mathrm{C}}(\mathbf{a}^{(i)}, \mathbf{p}^{(i)}) - \mathrm{WD}_{\mathrm{C}}(\mathbf{a}^{(i)}, \mathbf{n}^{(i)}) + \rho) \quad \forall (\mathbf{a}^{(i)}, \mathbf{p}^{(i)}, \mathbf{n}^{(i)}) \in \mathrm{Triplets}(\mathcal{D}), \tag{3.18}$$

with ρ an arbitrary margin, and we considered the feature triplets $(\mathbf{a}^{(i)}, \mathbf{p}^{(i)}, \mathbf{n}^{(i)})$, composed of a first feature $ba^{(i)}$ belonging to class $y^{(i)}$, a second feature $\mathbf{p}^{(i)}$ also belonging to class $y^{(i)}$, and finally a third feature $\mathbf{n}^{(i)}$, not belonging to class $y^{(i)}$. The proposed triplet loss replaces the previous cost matrix update (line 8 from algorithm 9).

The goal behind minimizing the triplet loss is to reduce the distance between points belonging to the same class while simultaneously increasing the distance between points belonging to different classes. This objective is widely accepted and used within the Metric Learning community, given the multitude of possible triplets possible from a single dataset. The initial results are both interesting and promising.

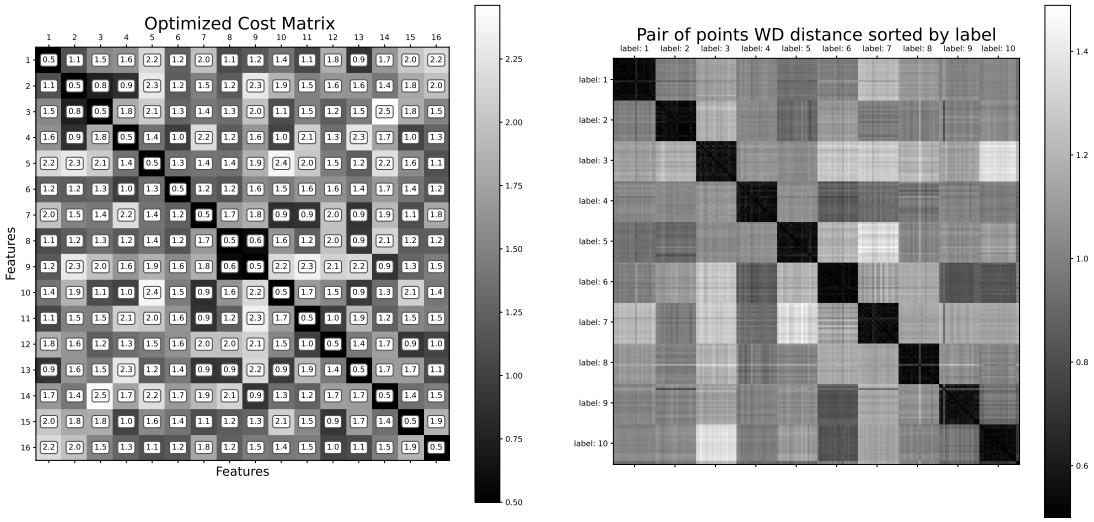
3.5.5 Results

In this section we review the results found during our experiments with the MNIST dataset, and why they reveal an ill-defined optimization problem.

An efficient optimization

During our MNIST experiments, to ensure adherence to a maximum number of distance axioms, we enforce certain conditions on our cost matrix. Specifically, we constrain the matrix to be positive, and symmetric and contain small values on the diagonal (0.5 instead of 0, as 0 is too dominant and inhibits the optimization of other variables). This approach helps achieve symmetry, identity, and positivity for our Wasserstein distance.

Figure 3.5 shows the result of the cost-matrix triplet loss optimization.



(a) Cost Matrix after optimization.

(b) Bootstrap of Test point distances label per label.

Figure 3.5: Triplet loss optimization results.

In addition to achieving outstanding performance, we empirically observe from Figure 3.5b small WD_C distance within classes and large WD_C distance between different classes, we obtain an interesting and explainable cost matrix.

From Figure 3.5a, we see that the cost matrix contains values that can easily be interpreted in conjunction with the feature distribution. If a point C_{ij} in the cost matrix is small, the cost between feature i and feature j is small, indicating a high correlation between these two features. This observation perfectly aligns with our findings from Figure 3.4 and Figure 3.4.

Comparison with the baseline

To some extent, the classifier's probabilities define a somewhat distance between images. Indeed, if two images are considered the same label then according to the classifier, they are somehow similar. Therefore, the optimized WD_C must at least recover the classifier's performance to show any kind of improvement.

Table 3.11 compares the original classifier's performances to a 1-nearest neighbors WD_C classification.

System	Précision Test
Classifier	99.01%
WD_C	98.90%

Table 3.11: Classification performances comparison between the original classifier and a $1-$nearest neighbors according to the WD_C.

Unfortunately, from Table 3.11, we see that the WD_C does not catch up with the classifier's performances. It means that even though the cost matrix optimization is efficient and allows a

retrieval of a decent solution, it does not reach the baseline performances and therefore does not reflect in any kind of improvement.

3.5.6 Intuition on the failure

The bad performances displayed in Table 3.11 might indicate that in reality, we are missing out on some important information contained in the features when optimizing the cost-matrix C.

From Figure 3.4, displaying the mean features, we observe that even though some dimensions of the features are activated (non-zero values), their values are so small that they are insignificant. This raises questions about the relevance of their activation, which could lead to some bias introduced with an overweight of them.

But more importantly, Figure 3.4 tells us that the features seem to already be linearly separable. And we confirm this hypothesis when increasing the feature size f with Figure 3.6.

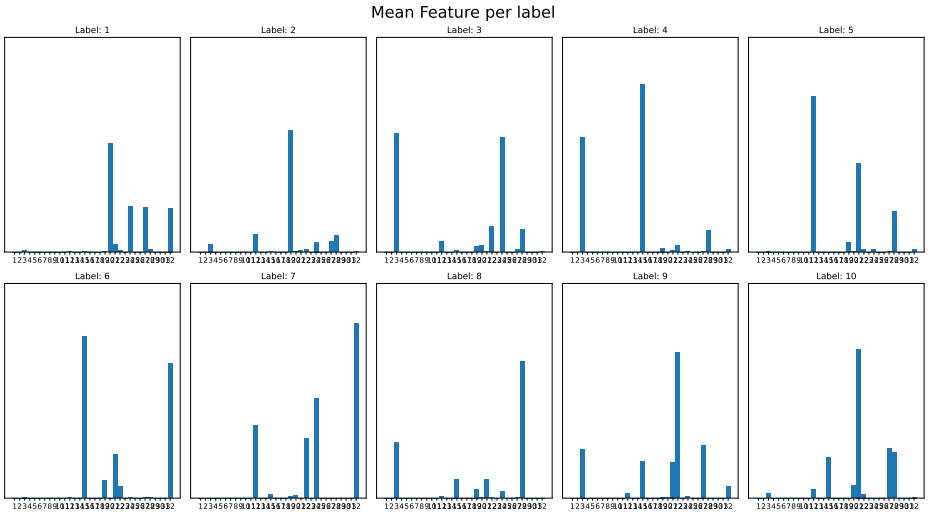


Figure 3.6: Distribution of the features when setting f = 32 according to the 10 possible labels.

Indeed, the average features of each class appear to have very little overlap with others, suggesting a potential existing linear separation.

To explore and confirm or not this intuition, we fixed the cost matrix C with the simplest matrix that doesn't encode any correlations,

$$C^0 = \begin{bmatrix} 0.1 & 1 & \dots & 1 \\ 1 & 0.1 & \dots & 1 \\ 1 & 1 & \dots & 0.1 \end{bmatrix} \tag{3.19}$$

This matrix is indeed symmetric, positive, and has an almost zero diagonal, which enforces symmetry, identity, and triangularity of its parametrized Wasserstein Distance.

When using the Wasserstein distance parameterized with the simplest cost matrix, WD_{C^0}**, we observed that this matrix yielded the best results.** Thus, by initializing the cost matrix in this manner, the optimization didn't manage to improve the test performance even by updating it through triplet loss optimization.

To verify this dual observation of an existing linear separation of features, we performed the same 1-nearest neighbor (1-NN) search using the ℓ_2 norm as the distance from the mean feature of the Train dataset for each class, to the test dataset features. We observed that 99.08% of the test features are closest to the mean training feature of the same class.

3.5.7 MNIST images conclusion

This conclusion indicates that by extracting features from MNIST images using $h(x)$, a function trained jointly with $f(x) = g(h(x))$, where g is a linear function, for a classification task achieving 99% accuracy on the Test dataset as Figure 3.7 highlights, an ℓ_2 norm is sufficient to properly separate the features from each other.

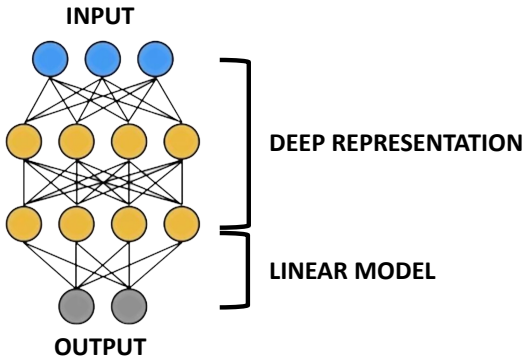


Figure 3.7: Dichotomy of a deep network's architecture. The first layers' job is to discriminate the data points, while the latter are designed to make the prediction. Source: Leveraging sparse linear layers for debuggable deep networks [174].

When building deep neural network classifiers, the first layers' job is to discriminate the data points, while the latter layers are designed to make the prediction. If its training is correctly managed thus, by definition of the linear final layer, the inputs are already well separated before the final layer. Even though we thought we correctly formalized the problem and checked the relevance of all our hypotheses, it revealed that one was not completely accurate leading to a failed intuition.

It leads us to consider other types of features from images and, to apply our framework on more concrete and complex datasets, for which a simple training of a classifier does not reach 99% accuracy, meaning that the features before prediction are not completely linearly separated. In such cases, our optimization framework might be revealed as a relevant tool.

3.6 Conclusion & perspectives

Initially, we aimed to model a meaningful adversarial attack on images, that is, an attack taking into consideration the semantic nature of images. For this purpose, we used extracted features instead of the image pixels. To efficiently distinguish the images in the feature space, we proposed learning a Wasserstein distance.

During this process, we sanity-checked the overall proposed optimization framework on a toy problem and found it very efficient and successful.

However, when applied to a slightly more concrete scenario, we discovered that features derived from before the last layer of a neural network classifier are almost linearly separable, and therefore, an ℓ_2 norm is sufficient to accurately distinguish the features according to their label.

Therefore, we can proceed with the development of a meaningful adversarial attack on images, using an ℓ_2 norm as the distance metric between the features.

During our work on the development of a meaningful adversarial attack, we completely ignored the feature decoder that would map the adversarial features back to the image space as displayed in Figure 3.2. Instead, a relevant choice would suggest considering an autoencoder architecture type to extract the most discriminative features from the images and map them back to the image space.

It would also be worth exploring our optimization framework on more complex datasets, one for which no deep classifier training could linearly separate the images according to their label. In such cases, the modelization of a meaningful image distance with our proposed optimization would be relevant and seem very promising.

The idea we explored is at the core of the current machine learning Large Language Models (LLM) revolution. Indeed, the idea to embed the object in a space where multiple tasks can be performed seems to be a relevant choice and shows outstanding recent results [157] [42]. This current trend confirms the relevance of our idea, and even though we did not reach perfect results, our findings confirm that working to create more meaningful adversarial attacks is the right path for computing more realistic and relevant adversarial perturbations.

Chapter 4

Conclusion & perspectives

Throughout our works, we delved deep into assessing the robustness of image classifiers to certify their suitability for practical use through additive adversarial attacks. We highlighted that properties like transferability and universality are regrettably under-emphasized in current research, although they are fundamental to crafting meaningful adversarial examples leading to a meaningful robustness evaluation. We also took a step back and underscored the significance of the performance evaluation metric in achieving an unbiased and fair assessment of the adversarial example generation.

In the first work, we proposed to model the adversarial noise space using a dictionary such that an attack can be derived, maximizing the desirable properties of adversarial examples, transferability, and universality. Indeed, the derived LIMANS dictionary-based adversarial attack shows to be more robust and more transferable than current state-of-the-art methods. Moreover, our approach allows for visual inspection of the roots of adversarial perturbations, making it easier to communicate with non-expert audiences and gain a better understanding of classifier vulnerabilities. Empirical evidence supports the idea that, according to our approach, the generated adversarial perturbations are more relevant as they exhibit more structure related to objects than pseudo-random adversarial perturbations.

One of the first perspectives on this work is to evaluate the modeling of the adversarial noise space with fewer training examples to see if the LIMANS attack improves in generalization if more training examples are considered. LIMANS' primary goal is to allow for a more relevant robustness evaluation of classifiers using adversarial examples. Therefore, constructing a LIMANS black-box attack would step further the realism of the attacker's behavior, estimating the classifier's robustness in an even more realistic way. The construction of such LIMANS black-box attack is one of the most promising perspectives.

Also, as advocated for instance by Carlini et al. [23], research on adversarial learning should be engaged in developing tools for resistance against these potential attacks. After working and proposing the LIMANS attack, we must design the LIMANS adversarial defense to contribute under the two scopes of attacker and defender, making the work an unbiased contribution. Another interesting perspective is to impose more constraints on the LIMANS atoms. Indeed, the adversarial patch attacks are interesting and appealing to estimate a more realistic classifier's robustness. We believe that, under carefully designed constraints, we could make LIMANS parameters, a dictionary of adversarial patches. It could lead to a real improvement and innovation brought to the research community, going further into crafting more realistic and efficient adversarial perturbations.

During our experiments, we mainly focused on images however, LIMANS could be applied to any data modalities as long as we can back-propagate gradients and have a similarity metric between points allowing us to respect an adversarial budget constraint. The LIMANS attack could be envisioned on any other data modalities, as long as the two previous conditions are met. Overall, this work aims to be a humble first step toward modeling and understanding the origin and nature of adversarial perturbations. This work also steps away from state-of-the-art methods to question

the relevance of the robustness assessment using adversarial examples.

In a second work, we explored the modeling of a meaningful similarity metric between images. While creating adversarial perturbations, we often deal with a biased and irrelevant image distance, which inevitably hampers the creation of relevant adversarial examples, leading to an overly considered robustness estimation.

Despite numerous challenges, we demonstrated that our approach to modeling a relevant similarity metric between images is feasible and promising even though some issues need to be addressed. During our work on the development of a meaningful adversarial attack, we ignored the feature decoder that would map the adversarial features back to the image space.

Instead of considering the image representations of a neural network, an appropriate choice would be to consider an autoencoder architecture type to extract the most discriminative representations of the images while allowing to map them back to the image space.

It would also be worth exploring our optimization framework on more complex datasets, one for which no deep classifier training could linearly separate the images according to their label. In such cases, the proposed modeling of a meaningful image distance would be relevant and promising.

This second work aligns with the first, in the way that it promotes a more meaningful robustness evaluation using adversarial examples. We believe that the considered similarity metric or distance is one of the most critical choices to perform a robustness evaluation of classifiers. Therefore, before performing any meaningful robustness evaluation using adversarial examples, the distance modeling should in itself be an optimization problem to resolve.

To sum up, our work contributes to estimating the image classifier robustness using adversarial examples in a more meaningful way. We have shown that to conduct a fair and realistic robustness evaluation from an application standpoint, adversarial examples must have a certain form and, most importantly, be relevant to the context.

Remerciements

Je commence par remercier Stéphane Canu, mon directeur de thèse, qui m'a accompagné et guidé durant ces trois années de travaux intensifs. Je remercie les membres du jury qui ont accepté de rapporter et d'examiner mon travail de thèse. Vos commentaires m'ont permis de prendre du recul sur mes travaux et de les reconnaitre à leur juste valeur.

Je remercie également les deux membres de mon comité de suivi de thèse, Mike Gartrell et Laurent Heutte, qui m'ont rassuré, conseillé et accompagné sur mes travaux et mes choix personnels. Je remercie avec nostalgie tous les membres du LITIS qui ont fait partie de l'équipe pédagogique qui m'a accompagné, suivi, guidé et qui m'a considéré ces huit dernières années, j'en garde un chaleureux souvenir et une haute estime à leur égard. Merci pour leur investissement, leur professionnalisme et leur sympathie

Je remercie ma psychothérapeute de m'avoir soutenu et de m'avoir accompagné vers un jour plus lumineux.

Enfin je remercie tout particulièrement ma famille, mes amis et ma compagne qui m'ont écouté, aidé et profondement aimé.

Per ardua ad astra

Appendix A

Annexes

A.1 ADiL projected gradient, norm constraint

Proposition A.1.1 (ℓ_p-Attacks)**.** *Given some budget* $\epsilon > 0$*, we have that* $\|\mathrm{D}v\|_p \leq \epsilon$ *for every* $\mathrm{D} \in \mathcal{D}$ *and* $v \in \mathcal{V}$ *where*

$$\mathcal{D} = \{\mathrm{D} \in \mathbb{R}^{\mathrm{P}\times\mathrm{M}} \mid (\forall m \in [\mathrm{M}]), \|d_m\|_p \leq 1\}, \tag{A.1}$$

and

$$\mathcal{V} = \{v \in \mathbb{R}^{\mathrm{M}} \mid \|v\|_1 \leq \epsilon\}. \tag{A.2}$$

Proof. Assuming that $\mathrm{D} \in \mathcal{D}$, then we have that

$$\begin{aligned}\|\mathrm{D}v\|_p = \|\sum_{m=1}^{\mathrm{M}} v_m d_m\|_p &\leq \sum_{m=1}^{\mathrm{M}} \|v_m d_m\|_p, \\ &= \sum_{m=1}^{\mathrm{M}} |v_m| \|d_m\|_p \leq \sum_{m=1}^{\mathrm{M}} |v_m| = \|v\|_1.\end{aligned}$$

Hence $\|v\|_1 \leq \epsilon$ implies $\|\mathrm{D}v\|_p \leq \epsilon$. □

A.2 From a regularised Wasserstein Distance to a Sinkhorn divergence

We will demonstrate how the regularised Wasserstein Distance A.3 boils down to the Sinkhorn divergence which allows to solve it efficiently [39]. The demonstration will be performed in 4 steps:

1. We first write down the Lagrangian formulation of the original problem
2. We find the optimal solution of the plan from this Lagrangian formulation
3. We use a rewriting of the dual variable to rewrite the Lagrangian problem
4. We finish by highlighting the solver to efficiently solve the problem

A.2.1 We first write down the Lagrangian formulation of the original problem

$$\begin{aligned}\mathrm{WD}(a_i, a_j) = \min_{\pi \in \mathbb{R}^{\mathrm{f}\times\mathrm{f}}} \sum_{k=1}^{\mathrm{f}}\sum_{t=1}^{\mathrm{f}} \pi_{k,t}\mathrm{C}_{k,t} + \epsilon \sum_{k=1}^{\mathrm{f}}\sum_{t=1}^{\mathrm{f}} \pi_{k,t}(\log(\pi_{k,t}) - 1)\,, \\ \text{such that } \pi 1 = a_i \text{ and } \pi^{\mathrm{T}} 1 = a_j\,.\end{aligned} \tag{A.3}$$

The Lagrangian formulation including the constraints with, $\alpha, \beta \in \mathbb{R}^{\mathrm{f}}, \alpha_i \geq 0, \beta_j \geq 0 \; \forall i, j$, reads

$$\mathcal{L}(\pi, \alpha, \beta) = \sum_{k=1}^{\mathrm{f}}\sum_{t=1}^{\mathrm{f}} \pi_{k,t}\mathrm{C}_{k,t} + \epsilon \sum_{k=1}^{\mathrm{f}}\sum_{t=1}^{\mathrm{f}} \pi_{k,t}(\log(\pi_{k,t}) - 1) - \alpha^{\mathrm{T}}(\pi 1 - a_i) - \beta^{\mathrm{T}}(\pi^{\mathrm{T}} 1 - a_j)\,.$$

A.2.2 We find the optimal solution of the plan from this Lagrangian formulation

$$\frac{\delta\mathcal{L}(\pi,\alpha,\beta)}{\delta\pi_{kt}} = 0 \Leftrightarrow C_{kt} + \epsilon(\log(\pi_{k,t})) - \alpha_k - \beta_t = 0\,,$$
$$\frac{\delta\mathcal{L}(\pi,\alpha,\beta)}{\delta\pi_{kt}} = 0 \Leftrightarrow \log(\pi_{k,t}) = \frac{-C_{kt} + \alpha_k + \beta_t}{\epsilon}\,,$$
$$\frac{\delta\mathcal{L}(\pi,\alpha,\beta)}{\delta\pi_{kt}} = 0 \Leftrightarrow \pi^*_{k,t} = \exp(\frac{\alpha_k}{\epsilon})\exp(\frac{-C_{kt}}{\epsilon})\exp(\frac{\beta_t}{\epsilon})\,.$$

A.2.3 We use a rewriting of the dual variable to rewrite the Lagrangian problem

Let $u = \exp(\frac{\alpha}{\epsilon})$, $v = \exp(\frac{\beta}{\epsilon})$ and $M = \exp(\frac{-C}{\epsilon})$, then we have

$$\frac{\delta\mathcal{L}(\pi^*,\alpha,\beta)}{\delta\pi_{kt}} = 0 \Leftrightarrow \pi^*_{k,t} = \mathrm{diag}(u)M\,\mathrm{diag}(v)\,.$$

$$\mathcal{L}(\pi^*,u,v) = \sum_{k=1}^{f}\sum_{t=1}^{f}\pi^*_{k,t}C_{k,t} + \epsilon\sum_{k=1}^{f}\sum_{t=1}^{f}\pi^*_{k,t}(\log(\pi^*_{k,t}) - 1) - \alpha^T(\pi^* 1 - a_i) - \beta^T(\pi^{*T}1 - a_j)\,,$$
$$\mathcal{L}(\pi^*,u,v) = \sum_{k=1}^{f}\sum_{t=1}^{f}\pi^*_{k,t}[C_{k,t} + \epsilon\log(\pi^*_{kt}) - \alpha_k - \beta_t] - \epsilon\sum_{k=1}^{f}\sum_{t=1}^{f}\pi^*_{kt} + \alpha^T a_i + \beta a_j\,,$$
$$\mathcal{L}(\pi^*,u,v) = -\epsilon\sum_{k=1}^{f}\sum_{t=1}^{f}\pi^*_{kt} + \alpha^T a_i + \beta a_j\,,$$
$$\mathcal{L}(\pi^*,u,v) = -\epsilon u^T M v + \alpha^T a_i + \beta a_j\,,$$
$$\mathcal{L}(\pi^*,u,v) = -\epsilon u^T M v + \alpha^T\epsilon\log(u) + \beta^T\epsilon\log(v)\,.$$

A.2.4 We finish by highlighting the solver to efficiently solve the problem

$$\begin{cases}\nabla_u\mathcal{L}(u,v) = 0\\ \nabla_v\mathcal{L}(u,v) = 0\end{cases} \Leftrightarrow \begin{cases}-Mv + \frac{\alpha}{u} = 0\\ -M^T u + \frac{\beta}{v} = 0\end{cases} \Leftrightarrow \begin{cases}u^* = \frac{\alpha}{Mv}\\ v^* = \frac{\beta}{M^T u}\end{cases}.$$

The optimization strategy falls back to gradient descent optimization on u and v simultaneously, once convergence is attained for these two variable we compute the optimal plan using $\pi^*_{k,t} = \mathrm{diag}(u)M\,\mathrm{diag}(v)$. We fall back on the same result previously found in the solver proposed in [39].

■

A.3 From the Wasserstein Distance to the Frechet Inception Distance

Let X and Y two multivariate random variables on $\mathbb{R}^N$ following distribution f_X resp. f_Y both belonging to a family of distributions which is closed with respect to changes of location and scale. μ_X, μ_Y and Σ_X, Σ_Y are the respective means and standard deviations of f_X and f_Y. If $WD(f_X, f_Y)$ denotes the Wasserstein distance between the f_X and f_Y, we have

$$WD(f_X, f_Y) = ||\mu_X - \mu_Y||_2^2 + Tr(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{1/2}).$$

According to [52] who is the original author of the proof, the proof of such result follows these three steps:

1. We show that $WD(f_X, f_Y) = \min_{f_W\in\Gamma(f_X,f_Y)}\mathbb{E}||X - Y||_2^2$ where $\Gamma(f_X, f_Y)$ denotes the collection of all measures on $N\times N$ with marginals f_X and f_Y on the first and second factors respectively.

2. We show when $\mu_x = \mu_y = 0$, $\mathrm{WD}(f_X, f_Y) = \mathrm{Tr}(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{1/2})$.
   (a) Reformulation of the problem with $W = \binom{X}{Y}$. Eq. 11
   (b) Expliciting the constraints, $\Sigma_W \succcurlyeq 0$. Eq. 14
   (c) Formulation of the dual problem. Eq. 16
   (d) Solving the dual problem. Eq. 20
   (e) Coming back to the primal problem. Eq. 22
   (f) Reformulation using SVD. Eq. 25
   (g) Concluding. Eq. 28
3. Coming back to the general case where μ_X and μ_Y are not longer assumed to be zero.

A.3.1 We show $\mathrm{WD}(f_X, f_Y) = \min_{f_W \in \Gamma(f_X, f_Y)} \mathbb{E}||X - Y||_2^2$.

The original formulation of the WD between f_X and f_Y is

$$\mathrm{WD}(f_X, f_Y) = \Big(\min_{f_W \in \Gamma(f_X, f_Y)} \int_N \int_N C(x,y)^2 f_W(x,y)\, dx dy \Big)^{1/2}, \tag{A.4}$$

$$\mathrm{WD}(f_X, f_Y) = \Big(\min_{f_W \in \Gamma(f_X, f_Y)} \int_{N\times N} C(x,y)^2 d f_W(x,y) \Big)^{1/2}, \tag{A.5}$$

Where C denotes a cost function between the random variables X and Y and $\Gamma(f_X, f_Y)$ denotes the collection of all measures on the random multivariate variable W = (X, Y) with marginals are $f_X(x)$ and $f_Y(y)$ on the first and second factors respectively.

Since we focus on the l_2 norm as cost function C, we have

$$\mathrm{WD}(f_X, f_Y) = \Big(\min_{f_W \in \Gamma(f_X, f_Y)} \int_{N\times N} ||x-y||_2^2 \, d f_W(x,y) \Big)^{1/2}, \tag{A.6}$$

From the very definition of f_W being a measure on W = (X, Y), the problem simplifies to

$$\mathrm{WD}(f_X, f_Y) = \min_{f_W(X,Y)} \mathbb{E}(||X - Y||_2^2). \tag{A.7}$$

A.3.2 We show when $\mu_x = \mu_y = 0, \mathrm{WD}(f_X, f_Y) = \mathbf{Tr}(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{1/2})$

Let $\mu_X = \mu_Y = 0 \in \mathbb{R}^N$ and W a random multivariate variable on R^{2N} such that $W = \binom{X}{Y}$, then $\mathbb{E}(W) = \binom{\mathbb{E}(X)}{\mathbb{E}(Y)} = \binom{\mu_X}{\mu_Y} = \binom{0}{0} \in \mathbb{R}^{2N}$. W has a covariance matrix $\Sigma_W = \begin{bmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{XY}^T & \Sigma_Y \end{bmatrix}$, $\Sigma_W \in \mathbb{R}^{2N\times 2N}$. We rewrite the problem according to the variable W

$$\begin{aligned}
\mathrm{WD}(f_X, f_Y) &= \min_{\Sigma_{XY}} \mathbb{E}(||X - Y||_2^2), \\
\mathrm{WD}(f_X, f_Y) &= \min_{\Sigma_{XY}} \mathbb{E}(\mathrm{Tr}((X-Y)(X-Y)^T)), \\
\mathrm{WD}(f_X, f_Y) &= \min_{\Sigma_{XY}} \mathrm{Tr}(\mathbb{E}((X-Y)(X-Y)^T)), \\
\mathrm{WD}(f_X, f_Y) &= \min_{\Sigma_{XY}} \mathrm{Tr}(\mathbb{E}(XX^T + YY^T - YX^T - XY^T)), \\
\mathrm{WD}(f_X, f_Y) &= \min_{\Sigma_{XY}} \mathrm{Tr}(\Sigma_X + \Sigma_Y - \Sigma_{XY} - \Sigma_{XY}^T).
\end{aligned} \tag{A.8}$$

There is an implicit assumption, Σ_W being positive semi definite $\Sigma_W \succcurlyeq 0$. A matrix $P \in \mathbb{R}^{2N}$ is only positive semi definite if and only if $\exists B \in \mathbb{R}^{2N\times 2N}$ such that $B^T B = P$. One can prove that

by doing, given a matrix $P \in \mathbb{R}^{2N}$, if $\forall w \in \mathbb{R}^{2N}$ we have $w^T P w \geq 0$, which is true when $w^T P w = w^T Q^T Q w = ||Qw||^2 \geq 0$, then P is positive semi definite. We exhibit such property for Σ_W:

$$\begin{aligned}
\forall w \in \mathbb{R}^{2N}, w^T \Sigma_W w &= \begin{pmatrix} X \\ Y \end{pmatrix}^T \begin{bmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{XY}^T & \Sigma_Y \end{bmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix}^T \begin{pmatrix} \Sigma_X X + \Sigma_{XY} Y \\ \Sigma_{XY}^T X + \Sigma_Y Y \end{pmatrix} \\
\begin{pmatrix} X \\ Y \end{pmatrix}^T \begin{pmatrix} \Sigma_X X + \Sigma_{XY} Y \\ \Sigma_{XY}^T X + \Sigma_Y Y \end{pmatrix} &= X^T \Sigma_X X + Y^T \Sigma_{XY}^T X + X^T \Sigma_{XY} Y + Y^T \Sigma_Y Y,
\end{aligned} \tag{A.9}$$

$$\exists B \in \mathbb{R}^{2N} w^T \Sigma_W w = w^T B^T B w = (Bw)^T (Bw) = ||Bw||^2 = \sum_{i=1}^{2N} (B_i^T w)^2 = \sum_{i=1}^{2N} (B_i^T \begin{pmatrix} X \\ Y \end{pmatrix})^2$$

$$\sum_{i=1}^{2N} (B_i^T \begin{pmatrix} X \\ Y \end{pmatrix})^2 = \sum_{i=1}^{2N} (B_i^T w)^2 = \sum_{i=1}^{2N} (B_i^T \begin{pmatrix} X \\ Y \end{pmatrix})^2 = \sum_{i=1}^{2N} (a_i^T X + b_i^T Y)^2.$$

We decompose the matrix B, the square root matrix of Σ_W, into internal components $a_i \in \mathbb{R}^N, b_i \in \mathbb{R}^N, \forall i = 1, \ldots, 2N$, then $B = \begin{pmatrix} a_1, \ldots, a_{2N} \\ b_1, \ldots, b_{2N} \end{pmatrix} \in \mathbb{R}^{2N \times 2N}$. The two red term should match, which will give us a redifinition of Σ_X, Σ_Y and Σ_{XY}. We develop these terms to see how to redefine the variables (colors are used to highlight equality)

$$\sum_{i=1}^{2N} (a_i^T X + b_i^T Y)^2 = \sum_{i=1}^{2N} (a_i^T X)^2 + (b_i^T Y)^2 + 2((a_i^T X)(b_i^T Y)).$$

Developing each term in the sum:

$$\sum_{i=1}^{2N} (a_i^T X)^2 = \sum_{m=1}^{2N} (\sum_{j=1}^{N} a_{mj} x_j)(\sum_{i=1}^{N} a_{mi} x_i) = \sum_{m=1}^{2N} \sum_{j=1}^{N} \sum_{i=1}^{N} a_{mj} x_j a_{mi} x_i = \sum_{j=1}^{N} \sum_{i=1}^{N} \sum_{m=1}^{2N} a_{mj} a_{mi} x_i x_j,$$

But we know that $X^T \Sigma_{X_{ij}} X = \sum_{i=1}^{N} x_i (\sum_{j=1}^{N} \Sigma_{X_{ij}} x_j) = \sum_{i=1}^{N} \sum_{j=1}^{N} \Sigma_{X_j} x_i x_j$ thus $\Sigma_X = \sum_{m=1}^{2N} a_m a_m^T$,

$$\sum_{i=1}^{2N} (b_i^T Y)^2 = \sum_{m=1}^{2N} (\sum_{j=1}^{N} a_{mj} y_j)(\sum_{i=1}^{N} b_{mi} y_i) = \sum_{m=1}^{2N} \sum_{j=1}^{N} \sum_{i=1}^{N} b_{mj} y_j b_{mi} y_i = \sum_{j=1}^{N} \sum_{i=1}^{N} \sum_{m=1}^{2N} b_{mj} b_{mi} y_i y_j,$$

But we know that $Y^T \Sigma_Y Y = \sum_{i=1}^{N} y_i (\sum_{j=1}^{N} \Sigma_{Y_{ij}} y_j) = \sum_{i=1}^{N} \sum_{j=1}^{N} \Sigma_{Y_{ij}} y_i y_j$ thus $\Sigma_Y = \sum_{m=1}^{2N} b_m b_m^T$,

$$\sum_{i=1}^{2N} 2((a_i^T X)(b_i^T Y)) = 2 \sum_{m=1}^{2N} (\sum_{i=1}^{N} a_{mi} x_i)(\sum_{j=1}^{N} b_{mj} y_j) = 2 \sum_{m=1}^{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} a_{mi} x_i b_{mj} y_j = 2 \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{m=1}^{2N} a_{mi} b_{mj} x_i y_j,$$

$$\begin{aligned}
\text{But we know that } Y^T \Sigma_{XY}^T X + X^T \Sigma_{XY} Y &= \sum_{j}^{N} y_j \sum_{i}^{N} \Sigma_{XY_{ij}} x_i + \sum_{i}^{N} x_i \sum_{j}^{N} \Sigma_{XY_{ij}} y_j, \\
&= \sum_{i}^{N} \sum_{j}^{N} x_i \Sigma_{XY_{ij}} y_j + \sum_{i}^{N} \sum_{j}^{N} x_i \Sigma_{XY_{ij}} y_j, \\
&= 2 \sum_{i}^{N} \sum_{i}^{N} \Sigma_{XY_{ij}} x_i y_j \text{ thus } \Sigma_{XY} = \sum_{m=1}^{2N} a_m b_m^T.
\end{aligned} \tag{A.10}$$

We now rewrite Σ_X, Σ_Y and Σ_{XY} (the red elements) in terms of $a_i \in \mathbb{R}^N, b_i \in \mathbb{R}^N \forall i = 1, \ldots, 2N$, the components of the Σ_W's square root matrix and rewrite Eq. A.8 as well

$$\begin{aligned}
WD(f_X, f_Y) &= \min_{\Sigma_{XY}} Tr(\Sigma_X + \Sigma_Y - \Sigma_{XY} - \Sigma_{XY}^T), \\
WD(f_X, f_Y) &= \min_{a,b} Tr(\sum_{m=1}^{2N} a_m a_m^T + \sum_{m=1}^{2N} b_m b_m^T - \sum_{m=1}^{2N} a_m b_m^T - \sum_{m=1}^{2N} b_m a_m^T), \\
\text{such that } \Sigma_X &= \sum_{m=1}^{2N} a_m a_m^T \text{ and } \Sigma_Y = \sum_{m=1}^{2N} b_m b_m^T.
\end{aligned} \tag{A.11}$$

We integrate the constraints with Lagrangian variables $\Omega \in \mathbb{R}^{N\times N}$ and $M \in \mathbb{R}^{N\times N}$ into the Lagrangian formulation of the problem

$$\begin{aligned}\mathcal{L}(a,b,\Omega,M) = \max_{\Omega,M} Tr(&\sum_{m=1}^{2N} a_m a_m^T + \sum_{m=1}^{2N} b_m b_m^T - \sum_{m=1}^{2N} a_m b_m^T - \sum_{m=1}^{2N} b_m a_m^T) \\ &+ Tr((\Sigma_X - \sum_{m=1}^{2N} a_m a_m^T)\Omega)) + Tr((\Sigma_Y - \sum_{m=1}^{2N} b_m b_m^T)M)), \\ \Leftrightarrow \mathcal{L}(a,b,\Omega,M) = \min_{\Omega,M} Tr(&- \sum_{m=1}^{2N} a_m a_m^T - \sum_{m=1}^{2N} b_m b_m^T + \sum_{m=1}^{2N} a_m b_m^T + b_m a_m^T) \\ &-Tr(\Sigma_X\Omega) + Tr((\sum_{m=1}^{2N} a_m a_m^T)\Omega) - Tr(\Sigma_Y M) + Tr((\sum_{m=1}^{2N} b_m b_m^T)M).\end{aligned} \tag{A.12}$$

The red terms don't interest us since they are either already defined or not related to the dual variables and therefore become constraints. Thus we have the new problem in terms of dual variables

$$\min_{\Omega,M} Tr(\sum_{m=1}^{2N} a_m b_m^T + b_m a_m^T) + Tr((\sum_{m=1}^{2N} a_m a_m^T))\Omega + Tr((\sum_{m=1}^{2N} b_m b_m^T))M. \tag{A.13}$$

The KKT conditions state that :

$$\begin{aligned}\frac{\nabla S}{\nabla a_i} = 0 \Leftrightarrow \frac{\nabla Tr(a_i b_i^T + b_i a_i^T)}{\nabla a_i} + \frac{\nabla Tr(a_i a_i^T)\Omega}{\nabla a_i} = 0 \Leftrightarrow 2b_i^T + 2a_i^T\Omega^T = 0 \Leftrightarrow b_i^T = a_i^T\Omega^T. \\ \frac{\nabla S}{\nabla b_i} = 0 \Leftrightarrow \frac{\nabla Tr(a_i b_i^T + b_i a_i^T)}{\nabla b_i} + \frac{\nabla Tr(b_i b_i^T)M}{\nabla b_i} = 0 \Leftrightarrow 2a_i^T + 2b_i^T M^T = 0 \Leftrightarrow a_i^T = b_i^T M^T.\end{aligned} \tag{A.14}$$

Optimally we have:

$$\begin{aligned}a_i^* = Mb_i,\ \forall i = 1,\dots,2N, \\ b_i^* = \Omega a_i,\ \forall i = 1,\dots,2N.\end{aligned} \tag{A.15}$$

From these two definitions and knowing that $\Sigma_X = \sum_{m=1}^{2N} a_m^* a_m^{*T}$ and $\Sigma_Y = \sum_{m=1}^{2N} b_m^* b_m^{*T}$ we derive a definition of Σ_{XY}

$$\begin{aligned}\Sigma_Y = \sum_{i=1}^{2N} b_i^* b_i^{*T} = \sum_{i=1}^{2N} a_i^*\Omega(\Omega a_i^*)^T = \Omega\sum_{i=1}^{2N} a_i^* a_i^{*T}\Omega = \Omega\Sigma_X\Omega = \Sigma_Y, \\ \Sigma_{XY} = \sum_{i=1}^{2N} a_i^* b_i^{*T} = \sum_{i=1}^{2N} a_i^*(\Omega a_i^*)^T = \sum_{i=1}^{2N} a_i^* a_i^{*T}\Omega = \Sigma_X\Omega = \Sigma_{XY},\end{aligned} \tag{A.16}$$

By combining both we end up with a definition of $\Sigma_{XY}\Sigma_{XY} = \Sigma_{XY}^2$

$$\begin{aligned}\Sigma_{XY}\Sigma_{XY} = \Sigma_X\Omega\Sigma_X\Omega = \Sigma_X\Sigma_Y, \\ \Sigma_{XY}^2 = \Sigma_X\Sigma_Y.\end{aligned} \tag{A.17}$$

We defined Σ_{XY}^2 but for our original problem A.8 we are actually more interested in Σ_{XY}. Assuming Σ_X is non-singular, meaning Σ_X^{-1} is defined, we can establish a definition of Ω,

$$\Omega = \Sigma_X^{-1/2} R \Sigma_X^{-1/2}. \tag{A.18}$$

From the definition of Σ_Y we have a definition of R,

$$\Sigma_Y = \Omega\Sigma_X\Omega \Leftrightarrow \Sigma_Y = \Sigma_X^{-1/2}R\Sigma_X^{-1/2}\Sigma_X\Sigma_X^{-1/2}R\Sigma_X^{-1/2} \Leftrightarrow \Sigma_Y = \Sigma_X^{-1/2}RR\Sigma_X^{-1/2} \Leftrightarrow R^2 = \Sigma_X^{1/2}\Sigma_Y\Sigma_X^{1/2}. \tag{A.19}$$

Let the eigenvalues and eigenvectors of R^2 be $(\lambda_1,\dots,\lambda_{2N})$ resp. $(v_1,\dots,v_{2N})$. The Singular-Value Decomposition of R^2 gives

$$R^2 = VDV^{-1}, \tag{A.20}$$

With $V \in \mathbb{R}^{2N\times 2N}$ a matrix made of the eigenvectors as columns and $D \in \mathbb{R}^{2N\times 2N}$ a diagonal matrix which entries D_{ii} are the eigenvalues. From this form we infer the forms of R

$$R = VD^{1/2}V^{-1} = \sum_{i=1}^{2N} \epsilon_i \lambda_i^{1/2} v_i v_i^T, \tag{A.21}$$

$D^{1/2}$ being a diagonal matrix which entries D_{ii} are the square root of R^2's eigenvalues $\lambda_i \forall i = 1,\dots,2N$. However each λ_i has two possible square root $(-\lambda_i^{1/2}, +\lambda_i^{1/2})$ which confirm 2^N possible $D^{1/2}$ matrices. Thus we encode all of them by timing each eigenvalue' square root with $\epsilon_i \in \{-1,1\}$. With the definition of R we derive a definition of Σ_{XY}

$$\begin{aligned} \Sigma_{XY} = \Sigma_X \Omega \text{ and } \Omega = \Sigma_X^{-1/2} R \Sigma_X^{-1/2}, \\ \Sigma_{XY} = \Sigma_X \Sigma_X^{-1/2} R \Sigma_X^{-1/2}, \\ \Sigma_{XY} = \Sigma_X^{1/2} R \Sigma_X^{-1/2}. \end{aligned} \tag{A.22}$$

Σ_{XY} has the same eigenvalues as R so the maximum and minimum value of $Tr(\Sigma_{XY}) = \sum_{i=1}^{2N} \lambda_i$ are reached when either $\epsilon_i = 1$ or $\epsilon_i = -1, \forall i = 1,\dots,2N$, which we simplify by respectively writing $\pm$. We rewrite Σ_{XY} with this new notation

$$\Sigma_{XY} = \pm \Sigma_X^{1/2} (R^2)^{1/2} \Sigma_X^{-1/2} = \pm \Sigma_X^{1/2} (\Sigma_X^{1/2} \Sigma_Y \Sigma_X^{1/2})^{1/2} \Sigma_X^{-1/2}. \tag{A.23}$$

We already saw that $\Sigma_{XY}^2 = \Sigma_X \Sigma_Y$ but we confirm that $\Sigma_{XY} = (\Sigma_X \Sigma_Y)^{1/2}$. Multiply Σ_{XY} by itself so see it is indeed a square root of $(\Sigma_X \Sigma_Y)$

$$\begin{aligned} \Sigma_{XY}\Sigma_{XY} &= \Sigma_X^{1/2} (\Sigma_X^{1/2} \Sigma_Y \Sigma_X^{1/2})^{1/2} \Sigma_X^{-1/2} \Sigma_X^{1/2} (\Sigma_X^{1/2} \Sigma_Y \Sigma_X^{1/2})^{1/2} \Sigma_X^{-1/2}, \\ \Sigma_{XY}\Sigma_{XY} &= \Sigma_X^{1/2} (\Sigma_X^{1/2} \Sigma_Y \Sigma_X^{1/2}) \Sigma_X^{-1/2}, \\ \Sigma_{XY}\Sigma_{XY} &= \Sigma_X \Sigma_Y. \end{aligned} \tag{A.24}$$

Hence $\Sigma_{XY} = (\Sigma_X \Sigma_Y)^{1/2}$. Which conclude the proof by giving that $Tr(\Sigma_{XY} + \Sigma_{XY}^T)$ is maximised or minimised by taking

$$\Sigma_{XY} = \pm (\Sigma_X \Sigma_Y)^{1/2}. \tag{A.25}$$

A.3.3 Coming back to the general case where μ_X and μ_Y are not longer assumed to be zero

If μ_X and μ_Y are no longer assumed to be zero, we enforce such property by adding the term $||\mu_X - \mu_Y||^2$ to the problem and re-utilise the previous definition we derived. The problem effectively reads

$$\begin{aligned} WD(f_X, f_Y) &= \min_{f_W \in \Gamma(f_X, f_Y)} \mathbb{E}(||X - Y||_2^2), \\ &= \min_{f_W \in \Gamma(f_X, f_Y)} \mathbb{E}(||(X - \mu_X) - (Y - \mu_Y) + \mu_X - \mu_Y||_2^2), \\ &= \min_{f_W \in \Gamma(f_X, f_Y)} \mathbb{E}(||(X - \mu_X) - (Y - \mu_Y)||_2^2) + \mathbb{E}(||\mu_X - \mu_Y||_2^2), \\ &= ||\mu_X - \mu_Y||^2 + Tr(\Sigma_X + \Sigma_Y - 2(\Sigma_X \Sigma_Y)^{1/2}). \end{aligned} \tag{A.26}$$

■

Notations

This section provides a concise reference describing the notation used throughout this work.

Numbers and Arrays

a	A scalar (integral or real)
$\mathbf{a}$	A vector
$\mathbf{A}$	A matrix
$\mathbf{I}_n$	Identity matrix with n rows and n columns
$e^{(i)}$	Standard basis vector $[0,\dots,0,1,0,\dots,0]$ with 1 at position i
$\text{diag}(a)$	A square, diagonal matrix with diagonal entries given by $\mathbf{a}$
$\mathtt{a}$	A scalar random variable
$\boldsymbol{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable

Sets

$\mathbb{A}$	A set
$\mathbb{N}$	The set of natural integers
$\mathbb{R}$	The set of real numbers
$\{0,1\}$	The set containing 0 and 1
$\{0,1,\dots,n\}$	the set of all integers between 0 and n
$[a,b]$	The real interval including a and b
$(a,b]$	The real interval excluding a but including b
$\mathbb{A}\setminus\mathbb{B}$	Set subtraction, i.e., the set containing the elements of $\mathbb{A}$ that are not in $\mathbb{B}$

Indexing

a_i	Element i of vector $\mathbf{a}$ with indexing starting at 1
a_{-i}	All elements of vector $\mathbf{a}$ except for elements i
$\mathbf{A}_{i,j}$	Element i, j of matrix $\mathbf{A}$
$\mathbf{A}_{i,:}$	Row i of matrix $\mathbf{A}$
$\mathbf{A}_{:,i}$	Column i of matrix $\mathbf{A}$

Linear Algebra Operations

$\mathbf{a} \otimes \mathbf{b}$	Outer product of vectors $\mathbf{a}$ and $\mathbf{b}$
$\mathbf{A}^{\mathrm{T}}$	Transpose of matrix $\mathbf{A}$
$\det(\mathbf{A})$	Determinant of $\mathbf{A}$

Calculus

$\frac{df}{dx}$	Derivative with respect to x of $f : \mathbb{R} \mapsto \mathbb{R}$
$\frac{\partial f}{\partial x}$	Partial derivative with respect to x of $f : \mathbb{R} \mapsto \mathbb{R}$
$\nabla_{\mathbf{x}} f$	Gradient of $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ with respect to $\mathbf{x}$
$\int f(\mathbf{x}) d\mathbf{x}$	Definite integral over the entire domain of $\mathbf{x}$

Probability and Information Theory

$\mathtt{a} \perp \mathtt{b}$	The random variables $\mathtt{a}$ and $\mathtt{b}$ are independent
$\mathrm{P}(\mathtt{a})$	A probability distribution over a discrete variable
$p(\mathtt{a})$	A density probability function over a continuous variable,
$\mathtt{a} \sim \mathrm{P}$	Random variable $\mathtt{a}$ has distribution P
$\mathbb{E}_{\mathtt{x} \sim \mathrm{P}}[f(x)]$ or $\mathbb{E} f(x)$	Expectation of $f(x)$ with respect to $\mathrm{P}(\mathtt{x})$
$\mathrm{Var}(f(x))$	Variance of $f(x)$
$\mathrm{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$
$\mathrm{H}(\mathtt{x})$	Shannon entropy of the random variable $\mathtt{x}$
$\mathrm{D}_{\mathrm{KL}}(\mathrm{P} \parallel \mathrm{Q})$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \mapsto \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the function f and g
$f(\mathbf{x}, \boldsymbol{\theta})$	A function of $\mathbf{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\mathbf{x})$ instead and neglect $\boldsymbol{\theta}$ to lighten notation)
$f_i(\mathbf{x})$	Element i of $f(\mathbf{x})$ when f ranges in the vector space
$\log x$	Natural logarithm of x
e^x	Natural exponential of x
$\| \mathbf{x} \|_p$	L^p norm of $\mathbf{x}$
$\| \mathbf{x} \|$	L^2 norm of $\mathbf{x}$
x^+	Positive part of x, i.e., $\max(0, x)$

Datasets and Distributions

p_{data}	The data generating distribution
$\mathcal{D}$	A dataset of examples
$\mathbf{x}^{(i)}$	The i-th example (input) from a dataset
$y^{(i)}$ or $\mathbf{y}^{(i)}$	The label associated with $\mathbf{x}^{(i)}$
$\mathbf{X}$	The $m \times n$ matrix with input example $\mathbf{x}^{(i)}$ in row $\mathbf{X}_{i,:}$

Adversarial

Δ	An adversarial attack,
$\Delta(\mathbf{x}) = \mathbf{x}' = \mathbf{x} + \omega_{\mathbf{x}}$	An adversarial example paired to the original example $\mathbf{x}$
$\omega_{\mathbf{x}}$	An adversarial noise paired to the original example $\mathbf{x}$
Ω	The space in which the adversarial noise $\omega_{\mathbf{x}}$ lives

List of Figures

List of Tables

Bibliography

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. [2016], Tensorflow: a system for large-scale machine learning., *in* 'Osdi', Vol. 16, Savannah, GA, USA, pp. 265–283.

[2] Akhtar, N., Mian, A., Kardan, N. and Shah, M. [2021], 'Advances in adversarial attacks and defenses in computer vision: A survey', *IEEE Access* **9**, 155161–155196.

[3] Anderson, J. M., Nidhi, K., Stanley, K. D., Sorensen, P., Samaras, C. and Oluwatola, O. A. [2014], *Autonomous vehicle technology: A guide for policymakers*, Rand Corporation.

[4] Andriushchenko, M., Croce, F., Flammarion, N. and Hein, M. [2020], Square attack: a query-efficient black-box adversarial attack via random search, *in* 'Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII', Springer, pp. 484–501.

[5] Ashby, F. G. and Perrin, N. A. [1988], 'Toward a unified theory of similarity and recognition.', *Psychological Review* **95**(1), 124–150.
URL: *https://doi.org/10.1037/0033-295x.95.1.124*

[6] Athalye, A., Carlini, N. and Wagner, D. [2018], Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, *in* 'International conference on machine learning', PMLR, pp. 274–283.

[7] Athalye, A., Engstrom, L., Ilyas, A. and Kwok, K. [2018], Synthesizing robust adversarial examples, *in* 'International conference on machine learning', PMLR, pp. 284–293.

[8] Badrinarayanan, V., Kendall, A. and Cipolla, R. [2017], 'Segnet: A deep convolutional encoder-decoder architecture for image segmentation', *IEEE transactions on pattern analysis and machine intelligence* **39**(12), 2481–2495.

[9] Bai, T., Luo, J., Zhao, J., Wen, B. and Wang, Q. [2021], 'Recent advances in adversarial training for adversarial robustness', *arXiv preprint arXiv:2102.01356* .

[10] Baluja, S. and Fischer, I. [2017], 'Adversarial transformation networks: Learning to generate adversarial examples', *arXiv preprint arXiv:1703.09387* .

[11] Bank, D., Koenigstein, N. and Giryes, R. [2020], 'Autoencoders', *arXiv preprint arXiv:2003.05991* .

[12] Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A. and Criminisi, A. [2016], 'Measuring neural net robustness with constraints', *Advances in neural information processing systems* **29**.

[13] Bellet, A., Habrard, A. and Sebben, M. [2022], *Metric Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Springer International Publishing.
URL: *https://books.google.com/books?id=uYRyEAAAQBAJ*

[14] Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H. [2006], 'Greedy layer-wise training of deep networks', *Advances in neural information processing systems* **19**.

[15] Benz, P., Zhang, C., Karjauv, A. and Kweon, I. S. [2021], Universal adversarial training with class-wise perturbations, *in* '2021 IEEE International Conference on Multimedia and Expo (ICME)', IEEE, pp. 1–6.

[16] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G. and Roli, F. [2013], Evasion attacks against machine learning at test time, *in* H. Blockeel, K. Kersting, S. Nijssen and F. Železný, eds, 'Machine Learning and Knowledge Discovery in Databases', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 387–402.

[17] Blin, R., Ainouz, S., Canu, S. and Meriaudeau, F. [2022], 'The polarlitis dataset: Road scenes under fog', *IEEE Transactions on Intelligent Transportation Systems* **23**(8), 10753–10762.

[18] Bonettini, S., Loris, I., Porta, F., Prato, M. and Rebegoldi, S. [2017], 'On the convergence of a linesearch based proximal-gradient method for nonconvex optimization', *Inverse Problems* **33**(5), 055005.

[19] Brendel, W., Rauber, J. and Bethge, M. [2017], 'Decision-based adversarial attacks: Reliable attacks against black-box machine learning models', *arXiv preprint arXiv:1712.04248* .

[20] Brown, T. B., Mané, D., Roy, A., Abadi, M. and Gilmer, J. [2017], 'Adversarial patch', *arXiv preprint arXiv:1712.09665* .

[21] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G. and Beijbom, O. [2020], nuscenes: A multimodal dataset for autonomous driving, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 11621–11631.

[22] Cao, Y., Xiao, C., Anandkumar, A., Xu, D. and Pavone, M. [2022], Advdo: Realistic adversarial attacks for trajectory prediction, *in* 'European Conference on Computer Vision', Springer, pp. 36–52.

[23] Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A. and Kurakin, A. [2019], 'On evaluating adversarial robustness', *arXiv preprint arXiv:1902.06705* .

[24] Carlini, N. and Wagner, D. [2017*a*], Towards evaluating the robustness of neural networks, *in* '2017 IEEE Symposium on Security and Privacy (SP)', pp. 39–57.

[25] Carlini, N. and Wagner, D. [2017*b*], Towards evaluating the robustness of neural networks, *in* '2017 IEEE Symposium on Security and Privacy (SP)', pp. 39–57.

[26] Chatzikyriakidis, E., Papaioannidis, C. and Pitas, I. [2019], Adversarial face de-identification, *in* '2019 IEEE International Conference on Image Processing (ICIP)', pp. 684–688.

[27] Chellapilla, K., Puri, S. and Simard, P. [2006], High performance convolutional neural networks for document processing, *in* 'Tenth international workshop on frontiers in handwriting recognition', Suvisoft.

[28] Chen, J., Jordan, M. I. and Wainwright, M. J. [2020], Hopskipjumpattack: A query-efficient decision-based attack, *in* '2020 ieee symposium on security and privacy (sp)', IEEE, pp. 1277–1294.

[29] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J. and Hsieh, C.-J. [2017], 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.

[30] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. and Zhang, Z. [2015], 'Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems', *arXiv preprint arXiv:1512.01274* .

[31] Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., Pham, H., Dong, X., Luong, T., Hsieh, C.-J. et al. [2023], 'Symbolic discovery of optimization algorithms', *arXiv preprint arXiv:2302.06675* .

[32] Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K. and Ha, D. [2018], 'Deep learning for classical japanese literature', *arXiv preprint arXiv:1812.01718* .

[33] Cohen, G., Afshar, S., Tapson, J. and Van Schaik, A. [2017], Emnist: Extending mnist to handwritten letters, *in* '2017 international joint conference on neural networks (IJCNN)', IEEE, pp. 2921–2926.

[34] Cortes, C. and Vapnik, V. [1995], 'Support-vector networks', *Machine learning* **20**, 273–297.

[35] Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P. and Hein, M. [2020], 'Robustbench: a standardized adversarial robustness benchmark', *arXiv preprint arXiv:2010.09670* .

[36] Croce, F. and Hein, M. [2020*a*], Minimally distorted adversarial examples with a fast adaptive boundary attack, *in* 'International Conference on Machine Learning', PMLR, pp. 2196–2205.

[37] Croce, F. and Hein, M. [2020*b*], Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, *in* 'International conference on machine learning', PMLR, pp. 2206–2216.

[38] Cuturi, M. [2013*a*], 'Sinkhorn distances: Lightspeed computation of optimal transport', *Advances in neural information processing systems* **26**.

[39] Cuturi, M. [2013*b*], Sinkhorn distances: Lightspeed computation of optimal transport, *in* C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Weinberger, eds, 'Advances in Neural Information Processing Systems', Vol. 26, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf*

[40] Dai, D. and Van Gool, L. [2018], Dark model adaptation: Semantic image segmentation from daytime to nighttime, *in* '2018 21st International Conference on Intelligent Transportation Systems (ITSC)', IEEE, pp. 3819–3824.

[41] Dai, J. and Shu, L. [2021], 'Fast-uap: An algorithm for expediting universal adversarial perturbation generation using the orientations of perturbation vectors', *Neurocomputing* **422**, 109–117.

[42] Dao, T., Fu, D., Ermon, S., Rudra, A. and Ré, C. [2022], 'Flashattention: Fast and memory-efficient exact attention with io-awareness', *Advances in Neural Information Processing Systems* **35**, 16344–16359.

[43] Daubechies, I., Defrise, M. and De Mol, C. [2004], 'An iterative thresholding algorithm for linear inverse problems with a sparsity constraint', *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences* **57**(11), 1413–1457.

[44] Deng, J., Berg, A. C., Li, K. and Fei-Fei, L. [2010], What does classifying more than 10,000 image categories tell us?, *in* 'Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part V 11', Springer, pp. 71–84.

[45] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. [2009], Imagenet: A large-scale hierarchical image database, *in* '2009 IEEE conference on computer vision and pattern recognition', Ieee, pp. 248–255.

[46] Deng, L. [2012], 'The mnist database of handwritten digit images for machine learning research', *IEEE Signal Processing Magazine* **29**(6), 141–142.

[47] Dong, Y., Kang, C., Zhang, J., Zhu, Z., Wang, Y., Yang, X., Su, H., Wei, X. and Zhu, J. [2023], Benchmarking robustness of 3d object detection to common corruptions, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 1022–1032.

[48] Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X. and Li, J. [2018*a*], Boosting adversarial attacks with momentum, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 9185–9193.

[49] Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X. and Li, J. [2018*b*], Boosting adversarial attacks with momentum, *in* '2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE Computer Society, Los Alamitos, CA, USA, pp. 9185–9193.
URL: *https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00957*

[50] Dong, Y., Pang, T., Su, H. and Zhu, J. [2019], Evading defenses to transferable adversarial examples by translation-invariant attacks, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 4312–4321.

[51] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. et al. [2020], 'An image is worth 16x16 words: Transformers for image recognition at scale', *arXiv preprint arXiv:2010.11929* .

[52] Dowson, D. C. and Landau, B. V. [1982], 'The fréchet distance between multivariate normal distributions', *Journal of Multivariate Analysis* **12**(3), 450–455.
URL: *https://EconPapers.repec.org/RePEc:eee:jmvana:v:12:y:1982:i:3:p:450-455*

[53] Dozat, T. [n.d.], 'Incorporating nesterov momentum into adam'.

[54] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T. and Song, D. [2018], Robust physical-world attacks on deep learning visual classification, *in* '2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 1625–1634.

[55] Fang, Y., Wang, W., Xie, B., Sun, Q., Wu, L., Wang, X., Huang, T., Wang, X. and Cao, Y. [2022], 'Eva: Exploring the limits of masked visual representation learning at scale', *arXiv preprint arXiv:2211.07636* .

[56] Fawzi, A., Moosavi-Dezfooli, S.-M., Frossard, P. and Soatto, S. [2018], Empirical study of the topology and geometry of deep networks, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 3762–3770.

[57] Fenton, R. E. and Mayhan, R. J. [1991], 'Automated highway studies at the ohio state university-an overview', *IEEE transactions on Vehicular Technology* **40**(1), 100–113.

[58] Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A. and Vayer, T. [2021], 'Pot: Python optimal transport', *Journal of Machine Learning Research* **22**(78), 1–8.
URL: *http://jmlr.org/papers/v22/20-451.html*

[59] Frecon, J., Anquetil, L., Liu, Y., Gasso, G. and Canu, S. [2022], 'Adversarial dictionary learning'.

[60] Frostig, R., Johnson, M. J. and Leary, C. [2018], 'Compiling machine learning programs via high-level tracing', *Systems for Machine Learning* **4**(9).

[61] Geiger, A., Lenz, P. and Urtasun, R. [2012], Are we ready for autonomous driving? the kitti vision benchmark suite, *in* '2012 IEEE conference on computer vision and pattern recognition', IEEE, pp. 3354–3361.

[62] Goodfellow, I., Bengio, Y. and Courville, A. [2016], *Deep Learning*, MIT Press.

[63] Goodfellow, I. J., Shlens, J. and Szegedy, C. [2015], 'Explaining and harnessing adversarial examples'.

[64] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. [2020], 'Generative adversarial networks', *Communications of the ACM* **63**(11), 139–144.

[65] Gowal, S., Qin, C., Huang, P.-S., Cemgil, T., Dvijotham, K., Mann, T. and Kohli, P. [2020], Achieving robustness in the wild via adversarial mixing with disentangled representations, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 1211–1220.

[66] Gui, J., Sun, Z., Wen, Y., Tao, D. and Ye, J. [2021], 'A review on generative adversarial networks: Algorithms, theory, and applications', *IEEE transactions on knowledge and data engineering* .

[67] Harder, P., Pfreundt, F.-J., Keuper, M. and Keuper, J. [2021], Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain, *in* '2021 International Joint Conference on Neural Networks (IJCNN)', IEEE, pp. 1–8.

[68] Hayes, J. and Danezis, G. [2017], 'Machine learning as an adversarial service: Learning black-box adversarial examples', *arXiv preprint arXiv:1708.05207* **2**, 64.

[69] Hayes, J. and Danezis, G. [2018], Learning universal adversarial perturbations with generative models, *in* '2018 IEEE Security and Privacy Workshops (SPW)', IEEE, pp. 43–49.

[70] He, K., Zhang, X., Ren, S. and Sun, J. [2015], Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1026–1034.

[71] He, K., Zhang, X., Ren, S. and Sun, J. [2016], Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

[72] Hendrycks, D. and Dietterich, T. [2019], 'Benchmarking neural network robustness to common corruptions and perturbations', *arXiv preprint arXiv:1903.12261* .

[73] Hinton, G. E., Osindero, S. and Teh, Y.-W. [2006], 'A fast learning algorithm for deep belief nets', *Neural computation* **18**(7), 1527–1554.

[74] Hinton, G. E. and Shallice, T. [1991], 'Lesioning an attractor network: investigations of acquired dyslexia.', *Psychological review* **98**(1), 74.

[75] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. [2017], 'Mobilenets: Efficient convolutional neural networks for mobile vision applications', *arXiv preprint arXiv:1704.04861* .

[76] Huang, X., Kwiatkowska, M., Wang, S. and Wu, M. [2017], Safety verification of deep neural networks, *in* 'Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30', Springer, pp. 3–29.

[77] Ioannou, P. [1997], *Automated highway systems*, Springer Science & Business Media.

[78] Ioffe, S. and Szegedy, C. [2015], Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International conference on machine learning', pmlr, pp. 448–456.

[79] Janai, J., Güney, F., Behl, A., Geiger, A. et al. [2020], 'Computer vision for autonomous vehicles: Problems, datasets and state of the art', *Foundations and Trends® in Computer Graphics and Vision* **12**(1–3), 1–308.

[80] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. and Fei-Fei, L. [2014], Large-scale video classification with convolutional neural networks, *in* 'Proceedings of the IEEE conference on Computer Vision and Pattern Recognition', pp. 1725–1732.

[81] Karras, T., Laine, S. and Aila, T. [2019], A style-based generator architecture for generative adversarial networks, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 4401–4410.

[82] Katz, G., Barrett, C., Dill, D. L., Julian, K. and Kochenderfer, M. J. [2017], Reluplex: An efficient smt solver for verifying deep neural networks, *in* 'Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30', Springer, pp. 97–117.

[83] Khrulkov, V. and Oseledets, I. [2018], Art of singular vectors and universal adversarial perturbations, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 8562–8570.

[84] Kilgour, K., Zuluaga, M., Roblek, D. and Sharifi, M. [2019], Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms., *in* 'INTERSPEECH', pp. 2350–2354.

[85] Kim, H. [2020], 'Torchattacks: A pytorch repository for adversarial attacks', *arXiv preprint arXiv:2010.01950* .

[86] Kingma, D. P. and Ba, J. [2014], 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

[87] Krizhevsky, A., Hinton, G. et al. [2009], 'Learning multiple layers of features from tiny images'.

[88] Krizhevsky, A., Sutskever, I. and Hinton, G. E. [2017], 'Imagenet classification with deep convolutional neural networks', *Communications of the ACM* **60**(6), 84–90.

[89] Kurakin, A., Goodfellow, I. and Bengio, S. [2017], 'Adversarial examples in the physical world', *ICLR Workshop* .
URL: *https://arxiv.org/abs/1607.02533*

[90] Lantos, B. and Márton, L. [2010], *Nonlinear control of vehicles and robots*, Springer Science & Business Media.

[91] Lax, P. D. and Terrell, M. S. [n.d.], *Calculus with applications*, Springer.

[92] Lecun, Y. [1989], *Generalization and network design strategies*, Elsevier.

[93] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. and Jackel, L. D. [1989], 'Backpropagation applied to handwritten zip code recognition', *Neural computation* **1**(4), 541–551.

[94] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. [1998], 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

[95] Li, Y., Cheng, S., Su, H. and Zhu, J. [2020], Defense against adversarial attacks via controlling gradient leaking on embedded manifolds, *in* 'Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16', Springer, pp. 753–769.

[96] Liu, D. C. and Nocedal, J. [1989], 'On the limited memory BFGS method for large scale optimization', *Mathematical Programming* **45**(1-3), 503–528.
URL: *https://doi.org/10.1007/bf01589116*

[97] Liu, H., Li, Z., Hall, D., Liang, P. and Ma, T. [2023], 'Sophia: A scalable stochastic second-order optimizer for language model pre-training', *arXiv preprint arXiv:2305.14342* .

[98] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J. and Han, J. [2019], 'On the variance of the adaptive learning rate and beyond', *arXiv preprint arXiv:1908.03265* .

[99] Lorenz, P., Strassel, D., Keuper, M. and Keuper, J. [2021], 'Is robustbench/autoattack a suitable benchmark for adversarial robustness?'.
URL: *https://arxiv.org/abs/2112.01601*

[100] Lorenz, P., Strassel, D., Keuper, M. and Keuper, J. [2022], Is autoattack/autobench a suitable benchmark for adversarial robustness?, *in* 'The AAAI-22 Workshop on Adversarial Machine Learning and Beyond'.
URL: *https://openreview.net/forum?id=aLB3FaqoMBs*

[101] Loshchilov, I. and Hutter, F. [2019], 'Decoupled weight decay regularization'.

[102] Madry, A., Makelov, A., Schmidt, L., Tsipras, D. and Vladu, A. [2019], 'Towards deep learning models resistant to adversarial attacks'.

[103] Mahmood, K., Gurevin, D., van Dijk, M. and Nguyen, P. H. [2021], 'Beware the black-box: On the robustness of recent defenses to adversarial examples', *Entropy* **23**(10), 1359.

[104] Mahmood, K., Nguyen, P. H., Nguyen, L. M., Nguyen, T. and van Dijk, M. [2019], 'Buzz: Buffer zones for defending adversarial examples in image classification', *arXiv preprint arXiv:1910.02785* .

[105] Mairal, J., Ponce, J., Sapiro, G., Zisserman, A. and Bach, F. [2009], Supervised dictionary learning, *in* D. Koller, D. Schuurmans, Y. Bengio and L. Bottou, eds, 'Advances in Neural Information Processing Systems', Vol. 21, Curran Associates, Inc.

[106] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. and Frey, B. [2015], 'Adversarial autoencoders', *arXiv preprint arXiv:1511.05644* .

[107] McCulloch, W. S. and Pitts, W. [1943], 'A logical calculus of the ideas immanent in nervous activity', *The Bulletin of Mathematical Biophysics* **5**(4), 115–133.
URL: *https://doi.org/10.1007/bf02478259*

[108] Metzen, J. H., Finnie, N. and Hutmacher, R. [2021], 'Meta adversarial training against universal patches', *arXiv preprint arXiv:2101.11453* .

[109] Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M. and Brendel, W. [2019], 'Benchmarking robustness in object detection: Autonomous driving when winter is coming', *arXiv preprint arXiv:1907.07484* .

[110] Mishchenko, K. and Defazio, A. [2023], 'Prodigy: An expeditiously adaptive parameter-free learner', *arXiv preprint arXiv:2306.06101* .

[111] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O. and Frossard, P. [2017], Universal adversarial perturbations, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1765–1773.

[112] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P. and Soatto, S. [2017], 'Robustness of classifiers to universal perturbations: a geometric perspective'.

[113] Moosavi-Dezfooli, S.-M., Fawzi, A. and Frossard, P. [2016], 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.

[114] Mopuri, K. R., Ojha, U., Garg, U. and Babu, R. V. [2018], Nag: Network for adversary generation, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 742–751.

[115] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. and Ng, A. Y. [2011], 'Reading digits in natural images with unsupervised feature learning'.

[116] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B. and Swami, A. [2017], Practical black-box attacks against machine learning, *in* 'Proceedings of the 2017 ACM on Asia conference on computer and communications security', pp. 506–519.

[117] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B. and Swami, A. [2016], The limitations of deep learning in adversarial settings, *in* '2016 IEEE European symposium on security and privacy (EuroS&P)', IEEE, pp. 372–387.

[118] Papernot, N., McDaniel, P., Wu, X., Jha, S. and Swami, A. [2016], Distillation as a defense to adversarial perturbations against deep neural networks, *in* '2016 IEEE Symposium on Security and Privacy (SP)', pp. 582–597.

[119] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A. [2017], 'Automatic differentiation in pytorch'.

[120] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. [2019], 'Pytorch: An imperative style, high-performance deep learning library', *Advances in neural information processing systems* **32**.

[121] Pinot, R., Meunier, L., Araujo, A., Kashima, H., Yger, F., Gouy-Pailler, C. and Atif, J. [2019], 'Theoretical evidence for adversarial robustness through randomization', *Advances in neural information processing systems* **32**.

[122] Poursaeed, O., Katsman, I., Gao, B. and Belongie, S. [2018], Generative adversarial perturbations, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4422–4431.

[123] Preuer, K., Renz, P., Unterthiner, T., Hochreiter, S. and Klambauer, G. [2018], 'Fréchet chemnet distance: a metric for generative models for molecules in drug discovery', *Journal of chemical information and modeling* **58**(9), 1736–1741.

[124] Pulina, L. and Tacchella, A. [2012], 'Challenging smt solvers to verify neural networks', *Ai Communications* **25**(2), 117–135.

[125] Qin, Z., Fan, Y., Liu, Y., Shen, L., Zhang, Y., Wang, J. and Wu, B. [2022], 'Boosting the transferability of adversarial attacks with reverse adversarial perturbation', *Advances in Neural Information Processing Systems* **35**, 29845–29858.

[126] Rahmati, A., Moosavi-Dezfooli, S.-M., Frossard, P. and Dai, H. [2020], Geoda: a geometric framework for black-box adversarial attacks, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 8446–8455.

[127] Rakotomamonjy, A. [2013], 'Direct optimization of the dictionary learning problem', *IEEE Transactions on Signal Processing* **61**(22), 5495–5506.

[128] Ranzato, M., Poultney, C., Chopra, S. and Cun, Y. [2006], 'Efficient learning of sparse representations with an energy-based model', *Advances in neural information processing systems* **19**.

[129] Rosch, E. [1975], 'Cognitive reference points', *Cognitive Psychology* **7**(4), 532–547.
URL: *https://doi.org/10.1016/0010-0285(75)90021-3*

[130] Rozsa, A., Rudd, E. M. and Boult, T. E. [2016], Adversarial diversity and hard positive generation, *in* '2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)', pp. 410–417.

[131] Rubner, Y., Guibas, L. J. and Tomasi, C. [1997], The earth mover's distance, multi-dimensional scaling, and color-based image retrieval, *in* 'Proceedings of the ARPA image understanding workshop', Vol. 661, p. 668.

[132] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. [1986], 'Learning representations by back-propagating errors', *nature* **323**(6088), 533–536.

[133] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al. [2015], 'Imagenet large scale visual recognition challenge', *International journal of computer vision* **115**, 211–252.

[134] Sakaridis, C., Dai, D. and Van Gool, L. [2018], 'Semantic foggy scene understanding with synthetic data', *International Journal of Computer Vision* **126**, 973–992.

[135] Salman, H., Ilyas, A., Engstrom, L., Kapoor, A. and Madry, A. [2020], 'Do adversarially robust imagenet models transfer better?', *Advances in Neural Information Processing Systems* **33**, 3533–3545.

[136] Schmidt, R. M., Schneider, F. and Hennig, P. [2021], Descending through a crowded valley - benchmarking deep learning optimizers, *in* M. Meila and T. Zhang, eds, 'Proceedings of the 38th International Conference on Machine Learning', Vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 9367–9376.
URL: *https://proceedings.mlr.press/v139/schmidt21a.html*

[137] Schroff, F., Kalenichenko, D. and Philbin, J. [2015], Facenet: A unified embedding for face recognition and clustering, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 815–823.

[138] Seetharaman, G., Lakhotia, A. and Blasch, E. P. [2006], 'Unmanned vehicles come of age: The darpa grand challenge', *Computer* **39**(12), 26–29.

[139] Sehwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M. and Mittal, P. [2022], Robust learning meets generative models: Can proxy distributions improve adversarial robustness?, *in* 'ICLR'.
URL: *https://openreview.net/forum?id=WVX0NNVBBkV*

[140] Sen, A., Zhu, X., Marshall, L. and Nowak, R. [2019], 'Should adversarial attacks use pixel p-norm?', *arXiv preprint arXiv:1906.02439* .

[141] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCun, Y. [2013], 'Overfeat: Integrated recognition, localization and detection using convolutional networks', *arXiv preprint arXiv:1312.6229* .

[142] Shafahi, A., Najibi, M., Xu, Z., Dickerson, J., Davis, L. S. and Goldstein, T. [2020], Universal adversarial training, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 34, pp. 5636–5643.

[143] Sharif, M., Bauer, L. and Reiter, M. K. [2018], On the suitability of lp-norms for creating and preventing adversarial examples, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops', pp. 1605–1613.

[144] Sharma, A., Bian, Y., Munz, P. and Narayan, A. [2022], 'Adversarial patch attacks and defences in vision-based tasks: A survey', *arXiv preprint arXiv:2206.08304* .

[145] Silva, S. H. and Najafirad, P. [2020], 'Opportunities and challenges in deep learning adversarial robustness: A survey', *arXiv preprint arXiv:2007.00753* .

[146] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al. [2016], 'Mastering the game of go with deep neural networks and tree search', *nature* **529**(7587), 484–489.

[147] Simonyan, K. and Zisserman, A. [2014], 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

[148] Sra, S. [2012], Scalable nonconvex inexact proximal splitting, *in* F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, eds, 'Advances in Neural Information Processing Systems', Vol. 25, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper/2012/file/7f100b7b36092fb9b06dfb4fac360931-Paper.pdf*

[149] Su, J., Vargas, D. V. and Sakurai, K. [2019], 'One pixel attack for fooling deep neural networks', *IEEE Transactions on Evolutionary Computation* **23**(5), 828–841.

[150] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B. et al. [2020], Scalability in perception for autonomous driving: Waymo open dataset, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 2446–2454.

[151] Sutskever, I., Martens, J., Dahl, G. and Hinton, G. [2013], On the importance of initialization and momentum in deep learning, *in* S. Dasgupta and D. McAllester, eds, 'Proceedings of the 30th International Conference on Machine Learning', Vol. 28 of *Proceedings of Machine Learning Research*, PMLR, Atlanta, Georgia, USA, pp. 1139–1147.
URL: *https://proceedings.mlr.press/v28/sutskever13.html*

[152] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. [2015], Going deeper with convolutions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1–9.

[153] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. [2016], Rethinking the inception architecture for computer vision, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2818–2826.

[154] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R. [2014], 'Intriguing properties of neural networks'.

[155] Tabacof, P. and Valle, E. [2016], Exploring the space of adversarial images, *in* '2016 international joint conference on neural networks (IJCNN)', IEEE, pp. 426–433.

[156] *The organization of behavior: A neuropsychological theory* [2005], Psychology press.

[157] Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y. et al. [2022], 'Lamda: Language models for dialog applications', *arXiv preprint arXiv:2201.08239* .

[158] Tramer, F., Carlini, N., Brendel, W. and Madry, A. [2020], 'On adaptive attacks to adversarial example defenses', *Advances in neural information processing systems* **33**, 1633–1645.

[159] Tramèr, F., Papernot, N., Goodfellow, I. J., Boneh, D. and Mcdaniel, P. [2017], 'The space of transferable adversarial examples', *ArXiv* **abs/1704.03453**.

[160] Tulshan, A. S. and Dhage, S. N. [2019], Survey on virtual assistant: Google assistant, siri, cortana, alexa, *in* 'Advances in Signal Processing and Intelligent Recognition Systems: 4th International Symposium SIRS 2018, Bangalore, India, September 19–22, 2018, Revised Selected Papers 4', Springer, pp. 190–201.

[161] Tversky, A. [1977], 'Features of similarity.', *Psychological Review* **84**(4), 327–352.
URL: *https://doi.org/10.1037/0033-295x.84.4.327*

[162] Tversky, A. and Gati, I. [1982], 'Similarity, separability, and the triangle inequality.', *Psychological Review* **89**(2), 123–154.
URL: *https://doi.org/10.1037/0033-295x.89.2.123*

[163] Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M. and Gelly, S. [2019], 'Fvd: A new metric for video generation'.

[164] Vassileios Balntas, Edgar Riba, D. P. and Mikolajczyk, K. [2016], Learning local feature descriptors with triplets and shallow convolutional neural networks, *in* E. R. H. Richard C. Wilson and W. A. P. Smith, eds, 'Proceedings of the British Machine Vision Conference (BMVC)', BMVA Press, pp. 119.1–119.11.
URL: *https://dx.doi.org/10.5244/C.30.119*

[165] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. [2017], 'Attention is all you need', *Advances in neural information processing systems* **30**.

[166] Vidot, G., Gabreau, C., Ober, I. and Ober, I. [2021], 'Certification of embedded systems based on machine learning: A survey', *CoRR* **abs/2106.07221**.
URL: *https://arxiv.org/abs/2106.07221*

[167] Villalobos, P., Sevilla, J., Besiroglu, T., Heim, L., Ho, A. and Hobbhahn, M. [2022], 'Machine learning model sizes and the parameter gap', *arXiv preprint arXiv:2207.02852* .

[168] Villani, C. et al. [2009], *Optimal transport: old and new*, Vol. 338, Springer.

[169] Von Bernuth, A., Volk, G. and Bringmann, O. [2019], Simulating photo-realistic snow and fog on existing images for enhanced cnn training and evaluation, *in* '2019 IEEE Intelligent Transportation Systems Conference (ITSC)', IEEE, pp. 41–46.

[170] Wang, Q., Ma, Y., Zhao, K. and Tian, Y. [2020], 'A comprehensive survey of loss functions in machine learning', *Annals of Data Science* pp. 1–26.

[171] Wang, X. and He, K. [2021], Enhancing the transferability of adversarial attacks through variance tuning, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 1924–1933.

[172] Wang, Y., Jha, S. and Chaudhuri, K. [2017], Analyzing the robustness of nearest neighbors to adversarial examples, *in* 'International Conference on Machine Learning'.

[173] Wong, E., Rice, L. and Kolter, J. Z. [2020], Fast is better than free: Revisiting adversarial training, *in* 'International Conference on Learning Representations'.

[174] Wong, E., Santurkar, S. and Madry, A. [2021], Leveraging sparse linear layers for debuggable deep networks, *in* 'International Conference on Machine Learning', PMLR, pp. 11205–11216.

[175] Wong, E., Schmidt, F. and Kolter, Z. [2019], Wasserstein adversarial examples via projected sinkhorn iterations, *in* 'International Conference on Machine Learning', PMLR, pp. 6808–6817.

[176] Xiao, H., Rasul, K. and Vollgraf, R. [2017], 'Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms', *arXiv preprint arXiv:1708.07747* .

[177] Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z. and Yuille, A. [2019], 'Improving transferability of adversarial examples with input diversity'.

[178] Yadav, C. and Bottou, L. [2019], 'Cold case: The lost mnist digits', *Advances in neural information processing systems* **32**.

[179] Yang, Y.-Y., Rashtchian, C., Zhang, H., Salakhutdinov, R. R. and Chaudhuri, K. [2020], A closer look at accuracy vs. robustness, *in* 'NeurIPS', Vol. 33, pp. 8588–8601.
URL: *https://proceedings.neurips.cc/paper/2020/file/61d77652c97ef636343742fc3dcf3ba9-Paper.pdf*

[180] Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M. and Wu, Y. [2022], 'Coca: Contrastive captioners are image-text foundation models', *arXiv preprint arXiv:2205.01917* .

[181] Zeiler, M. D. and Fergus, R. [2014], Visualizing and understanding convolutional networks, *in* 'Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13', Springer, pp. 818–833.

[182] Zhang, C., Benz, P., Lin, C., Karjauv, A., Wu, J. and Kweon, I. S. [2021], A survey on universal adversarial attack, *in* 'IJCAI-21', pp. 4687–4694. Survey Track.
URL: *https://doi.org/10.24963/ijcai.2021/635*

[183] Zhang, J., Wu, W., Huang, J.-t., Huang, Y., Wang, W., Su, Y. and Lyu, M. R. [2022], Improving adversarial transferability via neuron attribution-based attacks, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 14993–15002.

[184] Zhou, M., Wu, J., Liu, Y., Liu, S. and Zhu, C. [2020], Dast: Data-free substitute training for adversarial attacks, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 234–243.

[185] Zhou, Y.-T. and Chellappa, R. [1988], Computation of optical flow using a neural network., *in* 'ICNN', pp. 71–78.

[186] Zoph, B. and Le, Q. V. [2016], 'Neural architecture search with reinforcement learning', *arXiv preprint arXiv:1611.01578* .