Understanding and enforcing robustness in neural networks, modern perspectives and challenges

William Piat

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Understanding and enforcing robustness in neural networks, modern perspectives and challenges

Présentée et soutenue par
WILLIAM PIAT

Thèse soutenue le 06/03/2023 devant le jury composé de		
M. FRANCK IUTZELER	Maître de conférences HDR, Université Grenoble Alpes	Rapporteur du jury
MME BÉATRICE PESQUET-POPESCU	Professeur des universités, Thales Land and Air Systems	Rapporteur du jury
M. SÉBASTIEN DA VEIGA	Maître de conférences, ENSAI	Membre du jury
M. MOHAMED-JALAL FADILI	Professeur des universités, ENSICAEN	Membre du jury
M. LAURENT HEUTTE	Professeur des universités, Université de Rouen Normandie	Membre du jury
MME ELISA FROMONT	Professeur des universités, IRISA/INRIA Rennes	Président du jury
M. FREDERIC JURIE	Professeur des universités, Université de Caen Normandie	Directeur de thèse

Thèse dirigée par FREDERIC JURIE (Groupe de recherche en informatique, image, automatique et instrumentation)

UNIVERSITÉ CAEN NORMANDIE

GREYC
Laboratoire de recherche en sciences du numérique

Acknowledgments

First and foremost I would like to thank my academic supervisors Frédéric Jurie and Jalal Fadili that truly had a broad vision on their academic fields and a rigor that I terribly lacked when I started off this journey. You lent me your vision that guided me through these years.

I would like to thank as well the members of my jury Elisa Fromont, Laurent Heutte and especially Béatrice Pesquet-Popescu and Franck Iutzeler that reviewed the manuscript and suggested improvements.

I am indebted to my (now former) industrial supervisor Sébastien Da Veiga that took on the challenge of putting me back on track after a rather peculiar year of remote working. Your honesty and plain speaking on top of your academic skills were always uplifting and motivating. Where others brought vision, you brought means.

I truly feel grateful towards the CAM team in Safran Tech: Brian, Xavier, Clément, Fabien, Nissrine, Augustin, Julien, Alessandro, Felipe, Ramzi and Etienne. You are able to identify and tackle key challenges in industrial research. Although your research topics are broad you keep the desire to build tools and science for every one to benefit from, this truly is inspiring. Some part of the credit should be given to Christian Rey that managed to gather these great characters in a team that, despite restructuring and reorganising, remains a close-knit group.

I wish to thank my friends that helped me through my study by allowing me to take some welcome breaks: Guillaume L., Margaux, Vincent, Alexandre, Alexis, Damien, Nabil, Haroun, Tahir, Karine, Zakaria, Laurent, Guillaume D., Kun, Rémi and Liam.

I wouldn't fail to thank my family for their support: François, Elisabeth, Quentin, Eline, Tristan, Charlotte and especially my twin brother, Arthur, my teammate for longer than I can remember. From you I constantly find willpower and a helping hand. Where others brought vision or means, you brought drive.

Finally and especially, Marion: where others brought vision, means or drive, you actually bring perspective.

Abstract

The current challenge of Deep Learning is no longer the computational power nor its scope of application that has drastically widened in the couple last decades but the robustness of the produced models. Indeed, now that applications have reached numerous high risk domains such as autonomous cars or power grid management, the concerns for a reliable, explainable and robust artificial intelligence have grown as well. However, as of today, and despite years of research and an ever-growing active community, the question on how to certify neural networks for high risk production environments never came up with a consensus. The question has attracted many different researchers with different backgrounds and thus produced very different guarantees and/or algorithms. In this dissertation we provide various approaches where we detail, when applicable, the extent of the guarantees or algorithms given. This work starts with a survey of robustness techniques and advances that will serve to locate the following chapters in the literature. Then, we empirically put to the test an analogy that ties coverage and robustness together: we disentangle these two concepts via a set of different experiments that resort to a new subdifferentiable metric of coverage. We support our claims with a large range of datasets and state of the art models. Furthermore, we present a novel approach for assessing the quality of algorithms used for computing the Lipschitz constant of neural networks that are closely tied to robustness. Currently, these algorithms are tested against real world use cases where the actual result is unknown. We thus provide use cases with prescribed Lipschitz constants designed as general as possible in order to compare in an objective way the algorithms aiming at upper bounding or approximating the Lipschitz constant. Finally, we present a new optimization objective that acts as a proxy to tackle the distributional robust optimization of a deep neural network. Our procedure provides convergence guarantees under reasonable hypothesis. We present practical examples where the guaranteed smoothed optimization provides equivalent or better results than state of the art robust learning methods.

Keywords: Robustness, Deep Learning, Robust Optimization, Coverage, Lipschitz constants

Résumé

Le plus grand enjeu actuel de l'apprentissage profond n'est plus la puissance de calcul ni son cadre applicatif qui n'a cessé de s'étendre au cours des deux dernières décennies mais la robustesse des modèles induits. En effet, alors que les applications touchent des domaines industriels critiques variés comme la voiture autonome ou la gestion de réseau électrique, la nécessité de fiabilité, d'explicabilité et de robustesse en intelligence machine en est acerbée. Cependant, et malgré des années de recherche et une communauté de plus en plus active sur le sujet, nous en sommes encore aujourd'hui à un manque de consensus concernant la démarche de certification des réseaux de neurones profonds pour les applications à haut risques. Cette question a attiré de nombreux chercheurs de domaines variés qui ont donc produit des algorithmes et des garanties de forme multiple. Dans cette dissertation nous explorons plusieurs approches où nous détaillons, quand cela est applicable, l'étendue des garanties/algorithmes fournis. Ce travail commence par un état de l'art général des dernières avancées et des méthodes pour la robustesse qui va servir à situer les chapitres qui suivent au sein de la littérature. Ensuite nous examinons de manière empirique une analogie qui lie la robustesse et la notion de couverture: nous démêlons ces deux concepts à l'aide d'une série d'expériences impliquant une nouvelle métrique de couverture sous-différentiable. Nous appuyons nos résultats par une grande diversité de modèles et de bases de données. Par la suite nous proposons une approche nouvelle pour tester la performance des algorithmes calculant la constante de Lipschitz de réseaux de neurones (cette dernière étant intimement liée à la robustesse). Actuellement ces algorithmes sont testés sur des cas réels où la vraie constante de Lipschitz est inconnue. Nous fournissons des cas tests avec une constante de Lipschitz donnée qui sont construits de manière la plus générale possible pour pouvoir comparer de manière objective les algorithmes qui visent à majorer ou approcher la constante de Lipschitz. Pour finir nous présentons un nouvel objectif de substitution pour approcher le problème d'optimisation robuste distributionnelle d'un réseau de neurones. Notre méthode bénéficie de garanties de convergence sous le couvert d'hypothèses raisonnables. Nous présentons des exemples pratiques où notre optimisation lissée donne des résultats équivalents si ce n'est meilleurs que les méthodes état de l'art non garanties de converger.

Mots clés: Robustesse, Apprentissage profond, Optimisation Robuste, Couverture, Constantes de Lipschitz.

Table of contents

Chapter 1

Introduction: The numerous problems of robustness

1.1 The problem of robustness, the thorn in the side of modern learning of neural networks

Deep Neural Networks (DNNs) are nonlinear models that, unlike any other model, resort to millions or billions of parameters that are optimized jointly to tackle one given specific task. Applications cover numerous problem types in statistical learning such as regression, classification, unsupervised learning and policy learning, just to name a few. They often provide gains of performance over traditional methods by benefiting from large datasets, that have become more common in the digital era. They are, in practice, very suited for diverse tasks and data structures with dedicated architectures such as convolution networks [115] for vision tasks or transformers networks [182] for sequential data. As of today no model has had this broad scope of applications and this can be mostly explained by the simplicity of use of these technologies, along with the increase of parallel computing on dedicated hardware such as Graphical Processing Units (GPUs). Yet the reasons for this versatility and this unprecedented level of performance are still missing and thus cannot replace existing, well understood and certified technologies. DNNs could be used in addition to more traditional methods relying on simpler decision-making processes in order to support traditional approaches: DNNs are able to consider broader information and identify a pathological example that would have been overlooked otherwise. This can be a crucial advantage as a better prediction model means that safety margins can be reduced due to a higher confidence on a diagnosis. For instance, a more accurate medical diagnosis means being able to give to a patient the correct treatment in the adequate quantity. However, some of these applications have heavy implications in terms of safety and ethics that cannot be overlooked: these are called critical applications.

Critical applications are use cases with costly material or human stakes such as autonomous driving or collision avoidance systems [104], that therefore need a high level of validation and extended testing. The thorough validation and testing of DNNs are relevant to the point that public legislators are currently adapting their policy to account for, what they call, "high risk applications" for artificial intelligence. The most striking example is the AI (Artificial Intelligence) act [47]: a proposal to define and regulate on a European level use cases that fall into the scope of critical applications. The non-exhaustive list of high risk applications concerns biometric and categorization of natural persons, management and operation of critical infrastructure (health, aeronautics, power grids, road traffic), educational and vocational training, access to private services or public services, law enforcement, migration and border control management along with administration of justice and democratic processes. In these fields, ill decision-making can lead to drastic money loss, unfairness and more importantly can have a real negative impact on people's lives. Recently, concerns about fairness were acerbated since statistical algorithms have been shown particularly skilled at reproducing biases and discrimination that might lie in

man-made training sets [51, 40]. A certified system should then be able to provide risk mitigating measures, a validation of the design choices and the data used, an appropriate documentation, measures to ensure robustness and logs that should, when possible, be accessible to users. The AI act explicitly mentions that a compliant AI requires technical solutions to "address AI specific vulnerabilities", namely "data poisoning" and "adversarial examples". Providing a robust and interpretable model that fulfills all these certification requirements is one of the current challenges in statistical learning.

1.1.1 The stakes of robustness

A few notions are thus key to understanding the stakes of these requirements:

a. Interpretability and how it relates to responsibility.

b. Adversarial attacks and how they endanger modern deep learning.

c. Robustness and how it can address vulnerabilities.

d. Validation and how it is essential for certifying DNNs for an industrial application.

Interpretability/explainability and its link to responsibility. A first diagnosis tool in order to understand and explain a given misconduct of a statistical model is interpretability. It seeks to highlight causal relations between inputs and outputs of a model by striving to answer two basic questions: "Why on this example this decision was made over another?" and "On which element(s) of the input the decision relies?". The end goal here is to identify local causes that could enlighten the responsibility in the case of a wrong decision. Unfortunately, the complexity of the models usually comes at the cost of interpretability: it becomes far more complex to understand the decision-making of the algorithm when considering a nonlinear model and high dimensional inputs. Researches on interpretability highlighted that occasionally the prediction is based on naive or short-sighted elements. This is referred to as the "Clever Hans" effect [147, 113], named after a horse Hans that was supposedly able to make arithmetic however, as it turned out, it was simply enumerating with his hoof until he heard the crowd cheering. The "Clever Hans" effect in machine learning encompasses all abuse of contextual information or bias in the dataset to perform the diagnosis or decision. For instance, when a model resorts to an embedded caption or symbol in the image rather than the actual content to make the decision. These findings display a difficulty of knowing whether the cognitive system has actually learned the object of our interest rather than resorting to simpler proxies of it [163]. Such misguided learned correlations can be exploited and having proper diagnosis tools is thus essential for explaining a potential misconception. Interpretability is beyond the scope of this dissertation and we will not elaborate further on the matter, but it is worth noting that it is key to empirically understand the decision-making process and can provide insights that can be leveraged for improvements.

Adversarial samples jeopardize modern deep learning. Adversarial examples are carefully designed light noises that are crafted explicitly to fool the decision-making of neural networks [178]. The adversarial samples are often indistinguishable by a human oracle from the original sample. It questions whether the model really inferred some relevant features in the process of training and if the identified representations are robust. The reason behind the existence of adversarial attacks is not fully understood yet but it appears to be an unavoidable effect of high dimension of the input space [82, 167, 127]. Some attacks can be reproduced through the means of patches or specific occlusions that perturb the prediction of vision algorithms in real-world settings [191, 111]. Adversarial attacks are a key security issue if deep learning based algorithms are, for example, used in autonomous driving, where someone could exploit the vision algorithm of a car to have it select a wrong decision. As of today, there are no procedures that can prevent 100% of adversarial attacks, but there are empirical and theoretical methods to mitigate their effects. A possible safety could be to add filters upstream in the form of anomaly detection mechanisms [122, 117, 143] so that adversarial attacks could be removed before being

processed through the network. However, such techniques are often based on neural networks themselves that are also vulnerable to attacks. The most promising defenses against adversarial samples are based on dedicated training procedures such as adversarial training [126, 32, 204] or regularizations [173, 204, 27] that incorporate robustness directly in the training of the network. These approaches are closely tied to the concept of *robust learning*: an ideal robust training should discard the existence of these litigious samples or at least downplay their effects. Robust learning considers a possible set of perturbations (in which lie adversarial samples) and tries to find the optimal model that remains stable with respect to these alterations. In real-life settings these perturbations can be, for instance, uncertainties in sensor measurements as elaborated in the next paragraph.

Mitigating the effects of uncertainty with robustness. All physical systems are plagued by uncertainty: sensors, cameras, probes or human assertions are subject to perturbations that need to be taken into account when performing any form of statistical learning. Uncertainty can be random, systemic or a combination of both. At a macroscopic scale, it can be categorized into two types [76]: first, epistemic uncertainty comes from the lack of knowledge in the phenomenon we are trying to explain, for instance, if an explanatory variable is missing or if we are fitting a linear model on a second order function. Secondly, aleatoric uncertainty is mostly non-compressible noise on the observed variables such as white noise. In the following we will only consider aleatoric uncertainty since adversaries only modify the observed input space and alterations on unobserved variables have no influence on the prediction. Such uncertainties cannot always be detected and removed before processing the information through the model, but it remains possible to take them into account to some extent in the optimization problem. A naive but efficient idea is to assess what is the overall effect of uncertainty on the prediction: by propagating the uncertainties throughout the system it is possible to apprehend the resulting alterations on the prediction. Of course, this can only be achieved if the uncertainty of the input is known, which is rarely the case in practice. On the other hand, robust optimization [12, 15] can tackle problems where only the support of the uncertainty is known. But this drastically increases the complexity of the learning problem since the optimization procedure of the parameters needs to account for any possible perturbation in the uncertainty support. This support has to be defined the closest possible to the real perturbation otherwise this might provide over-conservative solutions [168]. Models resulting from robust optimization might be penalizing for regular use, but on the other hand they mitigate the worst possible scenario in the range of possibilities. This ultimately ensures a robust behavior of the model.

Validation and its role in certification. There is an ambiguity in the literature on the terms certification and validation where they are often used to name similar concepts. To avoid any confusion, we first elaborate on these two notions. *Verification* [96] is a method to assess exactly if a formal assertion is met on a system: for instance, in an industrial context, it is essential to verify that a solution abides by the technical specifications composed of design constraints. In the context of statistical learning, this could be asserting if the prediction of a model does not change under some perturbations. *Certification* stands for a list of requirements which are legally needed for a technology to be put into a production state and made available to the public under any kind of retail contract. This list of requirements has to be formulated by a public legislator and ensures that ethics and safety questions have been addressed during the design process and it can include verification of multiple statements. As of today there is little consensus over a general certification process for DNN-based systems. However, some conditions have unanimous backing: on the one hand, a system has to be verified to work even though it is subject to uncertainties, as detailed previously. On the other hand, a statistical decision system has to be proven to work on a production environment that may be slightly different than the training environment. Indeed if the production environment is too far from the training environment then the guarantees proven on the one cannot be applicable to the other (for instance if the inference environment has a different floating precision, or if it is assessed on unseen data). The production environment is usually different from the training one since statistical models are commonly used for interpolating, or if industrial constraints enforce the use of specific software or hardware. But current methods for verification can only provide limited guarantees around training

points [45, 189, 105]. There are a lot of challenges for testing points: the robustness of the prediction is difficult to assess since we theoretically have no way of computing the proximity of a testing sample to a training sample with the knowledge of the trained model only. Certification and validation are two heavily recommended steps to attest of the robustness of a neural network. Unfortunately, verification methods that provide a training procedure are scarce ([189] for instance). In current practice, the usual methodology relies on using any robust training method at training times to increase the safe bounds probed subsequently by verification algorithms. Ultimately a verification method that helps provide information on how to improve the training would be an all-in-one procedure to make neural networks robust.

In the light of the previous considerations, *robustness* appear as the focal point of modern deep learning. We will dedicate the next section to a thorough discussion on this concept.

1.2 Robustness definition

The aim of this section is to clarify the notion of robustness in statistical learning and to explain what to expect from robust models. Ultimately, we will highlight flaws in the current way of dealing with robust learning. In order to give a proper definition of robustness, we first propose to list some key related facts:

(a) **Statistical learning is impossible to achieve without some regularity.** We are performing interpolation from finite datasets to other finite test sets that may differ entirely from the already seen data. If no other assumption is made regarding the model or the data distribution, there is no reason that the learned model would perform any good on the test set. Indeed, it may prove completely wrong the learned model on the training set. A usual implicit and necessary assumption is that the observed phenomenons have some underlying structure and some limited variations along this structure. Statistical learning is nothing but understanding and exploiting this latent structure.

(b) **Uncertainties over explanatory variables need to be accounted for.** Uncertainties regarding the input information have to be dealt with when building a robust model. This may be done by building safety zones around input samples where the prediction should be constant. This is the dominating vision of robustness that assumes separability of the data, yet, it may not hold on more complex datasets.

(c) **Robustness over a dataset is not the robustness over the points in the dataset.** Robustness safety bounds may collide between one another, and dealing with this case naively by building the biggest possible safe zone ends up in overfitting the data. This puts into perspective the previous point where safety zones were an obvious solution to uncertainties. It is essential to consider conflict in the robustness zones in case there are outliers or mislabeled inputs.

(d) **Labels may be subject to perturbations as well.** In some cases (conflicting information in an image, mislabelling because it is man-made), having one single label over a sample is not guaranteed. Building a robust prediction needs to reflect such uncertainty or has to reduce the impact of an obviously wrong label in the dataset. It is essential to take into account possible label misspecifications (or output uncertainty in the case of regression) along with input uncertainty.

(e) **Our models are not flawless.** Many supporters of DL invoke the universal approximation theorem [53, 118] to justify the use of deep neural networks on a given application. This existence theorem is very powerful but it does not provide any practical way of finding the existing approximator nor does it give the architecture that would achieve so. On top of this, it assumes that we possess complete information about the problem regardless of uncertainties related to input/outputs. Neural networks will fail to achieve global optimality since our methods are merely approximating an optimal configuration and optimal parameters: there are still lingering errors coming from our optimization methods.

1.2.1 What is robustness?

A general definition of robustness

In order to give a definition of robustness we first need to assume that the data has some form of structure on which interpolating has meaning. Robustness, then, denotes the principle by which a model or an estimator is stable against light perturbations over the structure of the data or parameters [98]. In the context of building a predictive system, it may be rephrased in a more explicit manner: a robust statistical model provides a prediction that is *reasonably* insensitive to small perturbations either to its parameters, inputs or outputs over their respective structure. Ultimately, a "robust statistical model" is an antithesis since it has to be, on the one hand, robust and thus stable to modifications and on the other hand it has to provide different decision if the inputs were to change (hence the emphasis on the word *reasonably*). This indicates that in our given framework of statistical learning, bringing robustness actually means making compromises between the *generalization* of the model and its *robustness*.

Robust learning, a compromise between generalization and robustness

We call generalization the performance of a model on the real -and often unknown- data distribution. An accuracy on a finite test set is a proxy of the generalization that is valid only if the test set is representative of the real set with small error. There exists a trade-off between robustness and generalization: the more robustness is sought the lower the generalization capacity will be. This has been shown in multiple papers [181, 176, 204, 154, 153, 201] that robustness and generalization are, to some extent, concurrent objectives. In order to illustrate this compromise on a simple example, let us consider a simple binary classification of four points presented in Figure 1.1. A naive yet reasonable way of enforcing robustness could be to have safety zones around the point where the predicted class does not change. This explicitly makes the prediction robust to small uncertainties and if the safety zones are rather small they do not conflict. Yet, if one wants to increase the size of these safety zones as presented in Figure 1.2 there appears conflict and dealing with this case is not trivial. One could seek to build the biggest safety zone close to each point as depicted in Figure 1.3 but this falls into the trap of a nearest neighbor algorithm that has well-known overfitting issues and is not robust. Another possible method could be to assign a label based on the most represented label among safety zones overlapping the point: this would provide the most robust outcome at the cost of some accuracy in the prediction.

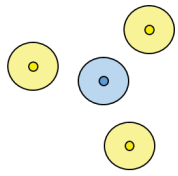

Figure 1.1: Building small robustness zones around samples

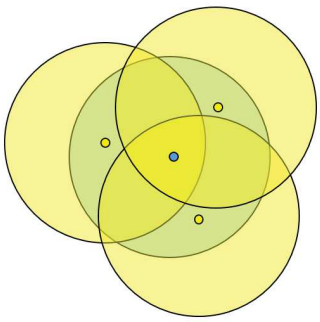

Figure 1.2: Building large robustness zones

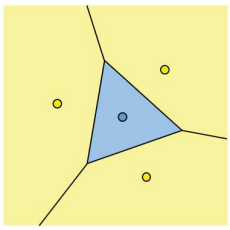

Figure 1.3: Nearest neighbor classification

One cannot expect a model to be able to make swift changes in its decision process and to be stable regarding any modifications of the input as well. The only way to prevent any perturbation of the decision function is to have a prediction that is completely insensitive to its inputs, i.e. a constant model. This does not condemn the search for both robust and accurate models, however, the questions are: what level of robustness do we need to achieve and which toll does it take on generalization? Ideally, one should seek the Pareto front of robustness versus generalization in order to select the best compromise according to the context or application, assuming robustness and generalization can be quantified reliably.

1.2.2 How robustness is addressed in the litterature?

We will distinguish 3 different sources of uncertainty, although they can all be combined altogether. These different sources have had historically separated approaches to deal with each of them. All approaches assume that there exists a regularity in the data (a), yet they tackle different robustness problems:

Robustness regarding the parameters/model (addressing (e))

Bayesian approaches in statistical learning aim at building decision systems where parameters are uncertain: an assumed prior distribution of the parameters, that encompasses our lack of confidence regarding the value of the parameters, is updated considering the input data. Ideally, the more data is observed the more narrow the posterior distribution becomes. Thus, our confidence in our parameters and in the resulting prediction increases. If the posterior distribution does not narrow, this usually means that the model cannot grasp the full extent of the phenomenon at hand and thus displays the lack of robustness of the model. Bayesian inference can be applied to small neural networks [137, 125] but it is difficult to extend them to large neural networks. Some efficient approximations are possible with variational inference [23] or rougher approximations with Monte Carlo dropout [76].

Robustness regarding the prediction (addressing (d))

The existing literature typically considers the case of robustness against label noise that may happen [2, 3, 200] as a consequence of poor mechanisms for attributing the prediction result, for instance man-made labelling. It has become essential for algorithms to take into account possible corruption in the target value. For classification settings the stability assumption specific of robust problems is the fact the correct label is given randomly with a probability greater than $\frac{1}{2}$. For regression settings the true value is assumed to be in a close neighborhood of the given reference value. Output uncertainty might be caused by input uncertainties themselves [162].

Robustness regarding the input space (addressing (b) and (c))

This is, by a wide margin, the robustness that is the most studied in nowadays robust learning: it considers perturbations (either small or large) on the explanatory variables. Such interest of the community around input uncertainty comes from the fact that adversarial attacks are yet an unresolved challenge in deep learning. The literature mostly differs on the nature of the "small perturbations" along which one needs to build resistance, since the notion of closeness varies loosely with the data under consideration. For instance, in $\mathbb{R}^2$ the closeness of two samples is quite straightforward. However, the closeness of two images is more complex than a pixel to pixel distance: we expect slight translations/rotations of images to trigger the same decision by the algorithm even though the example are far apart when considering a euclidean distance. As of today, possible perturbations have been characterized into two macro groups: attacks based on bounded perturbation balls are called *restricted attacks* as opposed to *unrestricted attacks* such as light rotations, translations and deformations. Methods to reduce the impact of input uncertainties are diverse: besides from building models that has an inner regularity [130, 42], the most used method for taking into account these uncertainties remains to sample from these perturbation sets and to find the model parameters that mitigate the most the error with respect to these perturbations [126, 82, 193, 172]. There are attempts at providing networks that are robust to both types of attacks [150], but the approach is overall empirical as the authors used a broad range of man-selected perturbations regardless of any notion of distance. The issue is that the perturbations under consideration are always too specific: the literature lacks of computable, guaranteed methods to mitigate all possible perturbations in a given ball.

1.2.3 The need for robustness in an industrial context

Safran, an aeronautic company designing aircraft engines and onboard systems, has seen DNN use cases emerge in several of its applications as ways to provide better embedded systems with, among others, monocular depth assessment [135], improved predictive maintenance for aging mechanical systems [148] or simulation acceleration [54]. Although gains have been showcased in all these applications, they were rarely put into production for four main reasons:

1. The difficulty of adapting the technology to production environments: the process of transferring a technology from a research department to production has multiple hurdles. For instance, the hardware in production is certified and is not as flexible as the one used in research. On top of that the specifications that need to be respected depend on each application, making this procedure impossible to generalize over a broad range of applications.
2. The fact that an official procedure for certifying the use of DNNs is still missing: the list of requirements is to be elaborated by a collaboration between a public legislator and academic researchers. Industrial actors might have a conflict of interest if they were to design such a document.
3. The presence of vulnerabilities such as adversarial attacks: this is up to the company to provide a system that is the safest possible, this implies building up methodologies that can infuse robustness into DNN solutions.
4. Being able to provide justification for the predictions devised by the algorithms: the research around explainability can be applied to a broad class of models during inference time so as to find the causes of a given prediction. It is up to the company offering a service to provide an explanation mechanism along.

A business that is aiming to sell a technology that resorts to statistical learning can only advance on points (3) and (4). Point (1) is application dependent, while point (2) relies on an independent entity, thus making both of them unfit for our study. Point (3) is the main focus of this thesis since it precedes point (4) to some extent: the conception of the robust model might provide by design an explanation of the decision (for instance, using a linear model provides a direct explanation of the influence of each input). *Robustness* is the aspect that needs the most consideration from an industrial standpoint since its goal is to provide general methodologies to ensure the validity of DNNs over a broad range of problems.

1.2.4 Blind spots in the robustness literature and driving questions of the dissertation

Robustness is central to theoretical deep learning and it is needed from an industrial standpoint. We raise some key questions that need to be addressed in the making of a robust model:

(Q1) Is there a well identified cause or reason for the lack of robustness in DNN models?

(Q2) What are the key metrics that can quantify reliably a level of robustness?

(Q3) What kind of perturbations is it legitimate to consider? Are there more general perturbations than others?

(Q4) Are we able to know, and perhaps model, all kinds of uncertainty that a DNN is exposed to?

(Q5) How one can change the learning phase of neural networks to account for the possible perturbations of the input variables?

1.3 Contributions

After Chapters 2 and 3 respectively dedicated to the mathematical background and a state of the art, we present different contributions in consideration of the questions above. In particular, we push through three angles focused on testing of neural networks and their robust training.

1.3.1 Coverage testing and the potential link between coverage and robustness

In multilayer neural networks there can be parts of the neural network that do not impact the decision made by the model, mostly because there are nonlinearities between each layer. Such blind spots could be exploited by an attacker to change the prediction of specific samples. For these reasons, the notion of *coverage* aims at quantifying which fraction of the network is used on a test input distribution. The current literature has presented successful attempts at building coverage guided attacks and coverage guided testing of DNNs [196, 145]. However, there appears to be little correlations between any of the proposed coverage metrics and robustness metrics [199]. In order to ascertain if adversarial training or adversarial attacks could benefit from coverage related optimization, we devised a range of experiments to highlight the presence of a possible interaction by considering a variety of models and datasets. The experiments failed to showcase a cross effect between an attack and coverage, as if they were completely different objectives. We however show empirically that using coverage for training a neural network brings robustness in the same fashion that regularization does, but that it does not explicitly make models robust to targeted attacks. This research showed that some appealing analogies in deep learning may be erroneous when considering models in high dimension. This contribution questions the cause of vulnerabilities in modern deep learning (Q1) and is the core of Chapter 4.

1.3.2 Lipschitz prescription for a better robustness assessment

Chapter 5 presents the second part of our research that stems from studying Lipschitz neural networks. Lipschitz continuity is obviously tightly related to robustness as the Lipschitz constant bounds the variations of the model around any given point. The Lipschitz constant of a network is therefore a very good metric showing the robustness of a model against potential perturbations (Q2). Our starting point was to notice that the methods for computing the Lipschitz constant were systematically comparing their performance with concurrent algorithms on either trained models or random networks. However, the discrepancy between the real Lipschitz constant and the one assessed by the algorithm cannot be quantified, since estimating the Lipschitz constant of an unconstrained network is NP-hard [183]. Only computing a guaranteed upper bound is sound for displaying superiority of one algorithm over another (the one lower is therefore the better), but the distance with the real value remains unknown. As a remedy, we offer to the best of our knowledge the first algorithm for designing a neural network with a prescribed Lipschitz constant. Despite having no convergence guarantees yet, our algorithm provides general architectures with a Lipschitz continuity close to the objective. This opens the path for systematic testing of Lipschitz assessing algorithms on some prescribed architectures by computing their error on their estimation of the Lipschitz constant, their dependence to the different dimensions of the problem (input dimension or parameter dimension) and their behavior around their validity domain. Our findings show that algorithms providing approximations of bounds need to be used with caution: many methods often trade approximation errors with computational performance and/or scalability.

1.3.3 Smoothed robust optimization for deep neural networks

In the previous two contributions, we only focused on testing, i.e. checking the correct implementation of a given neural network. The final step is to devise a training procedure that provides a model which has been designed robust from the start. Among approaches for robust training, we select Wasserstein distributional approaches due to the fact that they engulf local robustness approaches. Moreover, the formalism is general enough to derive results for a broader class of models than neural networks. Unfortunately, current approaches that aim at solving the distributional robust optimization often provide intractable algorithms [172, 175]. This is due to the overall complexity of the inner distributional maximization problem that happens on an infinite dimension space. In Chapter 6 we present a novel approach in order to circumvent this difficulty: we propose a proxy method where the distributional robust objective is cast into an equivalent pointwise robust objective. The latter can then be smoothed into an integral computation in infinite dimension and finally be approximated using Monte Carlo

integration. The method enjoys convergence guarantees provided some assumptions on the class of function optimized. We then proceed to show the benefits of using this algorithm on a neural network training where we compare our method with current empirical methods for robust training. The benefits of our algorithm are twofold: while providing at least the same performance in terms of accuracy than concurrent approaches, it also regularizes effectively and implicitly the model. The integral estimation inside our algorithm however adds computational load due to sampling, which may limit its use for high dimensional models. This contribution aims at answering questions (Q3), (Q4) and (Q5).

Chapter 2

Prerequisites and background

This chapter centralizes common notations and the mathematical prerequisites for all chapters. This does not exclude more specific notations to be introduced in the different chapters when judged relevant.

2.1 Euclidean spaces

Often dealing with Euclidean spaces we will define the general notations that will be used on all the chapters of this manuscript (mainly the notions of distance, operator norm and balls).

Distance. We note $\|\cdot\|_q$, $q \in [1, +\infty]$ the ℓ_q norm. $\operatorname{dist}(x, \mathcal{C}) = \inf_{z\in\mathcal{C}} \|x - z\|$ is the distance function to the nonempty set $\mathcal{C}$. The set of nearest points of x in $\mathcal{C}$ is denoted by $\mathrm{P}_{\mathcal{C}}(x)$.

Ball and volume. $\mathbb{B}_r^q(x) = \left[y, \|x - y\|_q \leq r\right]$ is the ℓ_q ball of radius $r \geq 0$ centered at x. $\mu_{\mathcal{L}}(S) = \int_S d\mu$ is the volume of a set S. When the superscript is omitted, this refers to the ℓ_2 norm ball.

Operator norm. Let A be an operator between two normed vector spaces $(V, \|\cdot\|_V)$ and $(W, \|\cdot\|_W)$ then the operator norm is $|||A|||_{V,W} = \sup_{x\in V} \frac{\|Ax\|_W}{\|x\|_V}$. When $\|\cdot\|_W = \|\cdot\|_V = \|\cdot\|_2$ we note the operator norm without the subscript ($|||A|||$) and it is called the spectral norm.

2.2 Deep learning

Deep learning is nowadays a broad field from Machine Learning (ML) and it is a form of machine intelligence. At its core lies the neural network, a parametric statistical model and its variants around which revolve more general fields such as optimization, statistical learning and statistics. Deep learning resorts to advances, both theoretical and practical, in all these fields in order to construct a decision-making process with the most accurate prediction. Most modern applications are to be found where data is available and in quantity to be able to determine correctly the numerous parameters of the model. These models have been increasingly used in the last years in fields such as computer vision [109], language processing [39], games [169], MRI scan analysis [123], molecule shape prediction [165], object detection [208] but also in physics, biology and manufacturing with unprecedented accuracy and performance. Although very experimental and empirical at first the domain is gaining more and more theoretical consolidations in order to explain, improve and certify these technologies. We will succinctly remind the basics of deep learning models:

One hidden layer neural network. For simplicity of notation, we introduce at first the neural network with one hidden layer. This will simplify the generalization to a multilayer neural network. Let $(l, r) \in \mathbb{N}^2$, $\mathcal{Z} \subset \mathbb{R}^l \times \mathbb{R}^r$ be a dataset, $(x, y) \in \mathcal{Z}$ where x is the input (explanatory variables) and y is the variable that we would like to predict by inference from x. We will build a predictor in order to predict y from x (it has meaning if there is causality or correlation between x and y). With $n \in \mathbb{N}$, let us define two linear operators $W_1 \in \mathbb{R}^{l\times n}$ and $W_2 \in \mathbb{R}^{n\times r}$ and two "bias" vectors $b_1 \in \mathbb{R}^n$ and $b_2 \in \mathbb{R}^r$. We also need to equip our network with an activation function [118] $\phi : \mathbb{R} \to \mathbb{R}$ that can be applied element-wise to the elements of a vector. A single hidden layer network writes:

$$f_2(\theta, x) = W_2\phi(W_1x + b_1) + b_2 = y_{\text{pred}}, \tag{2.2.1}$$

with $\theta = (W_1, b_1, W_2, b_2) \in \Theta$. The notation f_2 anticipates a recursive application of these nonlinear operators (see multilayer neural network in the next paragraph). A neuron in this definition is a component of an affine operator, for instance, neuron k in the hidden layer is the affine operator: $(W_1x + b_1)_k = \sum_{j=1}^{l} w_{kj}x_j + b_k$. The linear layer is the collection of all the neurons. A neuron network can then be seen as a collection of small linear units organised in layers on all of which we apply some nonlinear function ϕ.

We select the parameters θ in order to minimize a loss $\ell : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}$ measuring the discrepancy between the prediction $f_2(\theta, x)$ and the real value y.

$$\min_{\theta\in\Theta} \ell(f_2(\theta, x), y).$$

This is not usually made for one point but over all tuples (x, y) drawn from a data probability distribution $\hat{\rho}_k = \frac{1}{k}\sum_{i=1}^{k} \delta_{(x_i,y_i)}$ where $\delta_{(a,b)}$ is the dirac mass in point (a, b):

$$\min_{\theta\in\Theta} \frac{1}{k}\sum_{i=1}^{k} \ell(f_2(\theta, x_i), y_i) = \min_{\theta\in\Theta} \mathbb{E}_{\hat{\rho}_k}[\ell(f_2(\theta, x), y)].$$

For a regression setting ℓ is usually chosen among the euclidean distance, the absolute loss or the Huber smooth loss. For a classification setting ℓ is usually the 0-1 loss or its possible proxies such as a hinge loss or a cross entropy.

Multilayer neural network. For multilayer neural networks one should simply apply recursively the linear product and activation function to the output of the previous layer:

Definition 2.2.1. Given a nonlinear function $\phi : \mathbb{R} \to \mathbb{R}$ that can be applied element-wise on a vector, and a set of $d \in \mathbb{N}$ matrices $(W_i)_{i=1}^d$ with compatible dimensions so that the matrix product $W_1W_2...W_d$ is well defined. A linear neural network with biases is a function $f : \Theta \times \mathcal{X} \to \mathbb{R}^r$, where $\mathcal{X} \subset \mathbb{R}^l$ and $\Theta \subset \mathbb{R}^p$ is the space of parameters (weights and biases $\theta \overset{\text{def}}{=} (W_i, b_i)_{i\in[d]}$)

$$f_1(\theta, x) \overset{\text{def}}{=} W_1x + b_1, f_i(\theta, x) \overset{\text{def}}{=} W_i\phi(f_{i-1}(\theta, x)) + b_i, \forall i \in [d], f(\theta, x) \overset{\text{def}}{=} f_d(\theta, x). \tag{2.2.2}$$

This model has given better experimental results in terms of performance than the one hidden layer one.

In the exact same fashion as the one layer linear network $\theta \in \Theta$ are found by solving an optimization problem:

$$\min_{\theta\in\Theta} \frac{1}{k}\sum_{i=1}^{k} \ell(f(\theta, x_i), y_i) = \min_{\theta\in\Theta} \mathbb{E}_{\hat{\rho}_k}[\ell(f(\theta, x), y)]. \tag{2.2.3}$$

This problem can be approximated with any adequate optimizer, yet the most used algorithms are (sub)gradient descents as they are very efficient in case θ has a high dimension. This is a non-trivial approach since this problem is non-convex and non-differentiable if the activation function chosen is not differentiable, for instance: the Rectified Linear Unit (ReLU) [109]. Nevertheless this approach is gaining serious momentum for its remarkable results.

Although very general and allegedly able to tackle all regression problems just like their shallow counterparts [53, 118], Deep ReLU networks appear to have an extra advantage at representing complex functions as it has

be shown to generalize better [56]. Other architectures dedicated to specific tasks have emerged. For instance, Convolutional Neural Networks (CNNs) [115, 109, 92] that focus on finding local 2D structures rather than pixel to pixel correlations. Theoretically linear networks should be able to perform equivalently well than CNNs but in practice the prior knowledge incorporated in the structure of CNNs is forcing them to consider spatial correlations which gives them a significant advantage on images. That much is attested by the unrivaled performance of CNNs on image classification tasks and object detection. Ever since the achievements made by CNNs in recognition tasks [109] the empirical testing around neural networks has spurred and theoretical justifications lag behind.

2.3 Robust optimization

Vanilla learning. To ease the notations of the next paragraphs and to provide a smoother introduction to the notion of robust learning we present the "traditional" problem met in optimization: let $(\mathcal{Z}, \mathsf{d})$ be a (data) metric space with $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^m$, where $\mathcal{X}$ (resp. $\mathcal{Y}$) is the input (resp. output) space. Let ρ be a probability measure on $\mathcal{Z}$, $\Theta \subset \mathbb{R}^p$ the parameter/action space, $\mathcal{L} : \Theta \times \mathcal{Z} \to \mathbb{R}_+$ a loss function such that $\mathcal{L}(\cdot, \theta)$ is ρ-measurable for all $\theta \in \Theta$. Consider the optimization problem:

$$\min_{\theta \in \Theta} \mathbb{E}_{z \sim \rho} \mathcal{L}(\theta, z). \tag{2.3.1}$$

In the scope of statistical learning, we have, given a distance/divergence function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ adapted to the problem (mean square error for regression, cross entropy or hinge loss for classification etc...), for all $(x, y) \in \mathcal{Z} = (\mathcal{X} \times \mathcal{Y})$ and $\theta \in \Theta$:

$$\mathcal{L}(\theta, (x, y)) = \ell(f(\theta, x), y).$$

This draws the link between general optimization and the procedure for finding the parameters of a deep learning model (2.2.3)

Pointwise robust counterpart. Recalling the notations of equation (2.3.1) , let us define $\mathcal{U}(x, y) \subset \mathbb{R}^m$ a bounded uncertainty set that in all generality depends of x and y, then the robust pointwise counterpart of (2.2.3) writes in the form of the minmax problem:

$$\min_{\theta \in \Theta} \frac{1}{k} \sum_{i=1}^{k} \max_{(\tilde{x}, \tilde{y}) \in \mathcal{U}(x_i, y_i)} \ell(f(\theta, \tilde{x}), \tilde{y}) = \min_{\theta \in \Theta} \mathbb{E}_{(x,y) \sim \hat{\rho}_k} \max_{(\tilde{x}, \tilde{y}) \in \mathcal{U}(x,y)} \ell(f(\theta, \tilde{x}), \tilde{y}). \tag{2.3.2}$$

Equation (2.3.2) means that we are no longer minimizing only on a dataset, but that we minimize the problem with respect to the worst case among the uncertainty set. Uncertainties could theoretically generate unseen configurations or adversarial samples for the statistical model thus one should minimize the impact of uncertainty on the decision model. In case the distribution of uncertainty is well known, it can be included in the learning process and optimized directly with this prior knowledge. In most cases it is not known and it requires a model free approach of learning under uncertainty [12] that is, by essence, much more conservative due to the broad range of possible perturbations. Usually perturbation sets are taken as balls with various distances around the points in the dataset. This is a common model for the perturbation and corresponds to assuming that the prediction of the model should not change when we are close to a known point. Often the uncertainty set is not fully known either and is chosen as a close ball around (x, y) with some ground cost (or distance) $\mathsf{c} : \mathcal{Z}^2 \to \mathbb{R}$, the uncertain set takes the form:

$$\mathcal{U}_\epsilon(x, y) = \{(\tilde{x}, \tilde{y}), \mathsf{c}((x, y), (\tilde{x}, \tilde{y})) \leq \epsilon\}.$$

If the ground cost c is taken to be the ℓ_k norm then $\mathcal{U}_\epsilon(x, y) = \mathbb{B}_\epsilon^k((x, y)$. However, the result of (2.3.2) depends heavily on the metric chosen even if ultimately all vector norms are equivalent.

Distributional robust counterpart. An alternative to the robust pointwise counterpart is to consider perturbation not on the points of the dataset (x, y) but on their distribution $\hat{\rho}_k$. To that extend we consider a distance/divergence $\mathsf{D} : \mathcal{P}(\mathcal{Z})^2 \to \mathbb{R}$ between two distributions ($\mathcal{P}(\mathcal{Z})$ denotes the positive measures of norm 1 distributed on $\mathcal{Z}$) , most popular choices are Total Variation distance, ϕ-divergences, Wasserstein distance or Maximum Mean Discrepancy. Thus the Distributional Robust Counterpart writes:

$$\min_{\theta\in\Theta} \max_{\mathsf{D}(\rho,\hat{\rho}_k)\leq\epsilon} \mathbb{E}_{(x,y)\sim\rho}[\ell(f(\theta, x), y)]. \tag{2.3.3}$$

Note that in equation (2.3.3) the max and expectancy are reversed compared to (2.3.2) as in the former the perturbation lies on the distribution of the points and on the latter on the points of the nominal distribution. This is particularly suitable when the nominal distributions are discrete samples of a unknown distribution: this formalism allows building a decision system that takes into account the sampling uncertainty of the dataset.

2.4 Adversarial attacks and defenses

2.4.1 Adversarial attacks

2.4.1.1 Perturbing one sample

Adversarial attacks are considered to be one of the greatest menace for certifying neural networks and we discussed them in paragraph 1.1.1. We provide here the mathematical framework of such attacks and the methods for computing them. We need to equip ourselves with a neural network $f : \theta \times \mathcal{X} \to \mathbb{R}^r$ as defined in Section 2.2, a loss function $\ell : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}$ and $\varepsilon > 0$. Generally, adversarial attacks are different methods for approximating the argument of the following problem:

$$\max_{\tilde{x}\in\mathbb{B}_\varepsilon^\infty(x)} \ell(f(\theta, \tilde{x}), y). \tag{2.4.1}$$

Or equivalently with a perturbation variable $\delta \in \mathbb{R}^p$:

$$\max_{\delta\in\mathbb{B}_\varepsilon^\infty(0)} \ell(f(\theta, x + \delta), y). \tag{2.4.2}$$

The most popular methods for approximating this problem are using the very same gradient update used for training the model: a gradient backpropagation and then an update. Instead of computing the gradient of the loss with respect of the parameters we compute it with respect to the inputs this allows to alter slightly the inputs of the model and the update is usually performed with projected gradient (i.e. taking the sign of the gradient before multiplying with the step size). As an example we provide in algorithm 1 of the Projected Gradient Descent (PGD) attack:

Algorithm 1: Projected gradient descent

Input: A loss function L that depends of θ,x,y
Input: A bounded perturbation set C
Input: A training sample $(x, y) \in \mathbb{R}^m$
Input: A number of iterations N
Input: $(\gamma_k)_{k\in\mathbb{N}}$ a decaying learning rate
$\delta := 0$;
for $k = 1, N$ **do**
  $L(\theta, x + \delta, y) := \ell(f(\theta, x + \delta), y)$;
  $u_k \in G_{L(\theta,\cdot,y)}(x + \delta)$;
  $\delta := \mathrm{P}_C\left(\delta + \gamma_k \mathrm{sign}(u_k)\right)$;
return $x + \delta$;

In algorithm 1, P_C denotes the projection on set C, sign denote the function that, applied element-wise, returns the sign of each value and $G_{L(\theta,\cdot,y)}$ denotes generalized gradient of the function L with respect to x.

L has been chosen equal to the learning loss (i.e. $L(\theta, x, y) = \ell(f(\theta, x), y)$ but we can change it to any other appropriated loss that depends of θ, x and/or y.

PGD is thus a procedure that takes a loss function $L : \Theta \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}, L(\theta, x, y) \mapsto \ell(f(\theta, x), y)$, the parameters θ, a bounded perturbation set $C = \mathbb{B}_\varepsilon^\infty(x)$, a tuple (x, y) of input/output and returns a perturbed input $\tilde{x}$. Problem (2.4.1) is then approximated via PGD:

$$\begin{aligned} &\ell(f(\theta, \tilde{x}), y) \\ \text{where} \quad &\tilde{x} = \text{PGD}(L, \theta, \mathbb{B}_\varepsilon^\infty(x), x, y). \end{aligned} \tag{2.4.3}$$

The Fast Gradient Sign Method (FGSM) is the one step variant of PGD where the step size γ_k is taken equal to the radius of C.

Another interesting approach is the Carlini and Wagner attack [32] (C&W): it is only applicable on classification settings and differs from the FGSM and PGD attack by the fact that is does not attempts to reduce the likelihood of the currently predicted label but it tries to increase the likelihood of the second most probable label. This is a proxy of problem 2.4.1 as it resorts to unused information during training coming from negative samples. Formally it seeks to approximate using a multi-step gradient descent algorithm the argument of equation (2.4.4):

$$\min_{\tilde{x} \in \mathbb{B}_\varepsilon^\infty(x), \tilde{y} \neq y, \tilde{y} \in \mathcal{Y}} \ell(f(\theta, \tilde{x}), \tilde{y}), \tag{2.4.4}$$

where $\tilde{y}$ has to lie within the range of possible labels. In practice $\tilde{y}$ is selected as the second most likely output (the first one being usually equal to the real label).

2.4.1.2 Perturbing batches of samples

For computational efficiency the perturbations $\tilde{x}$ are usually computed in a batched manner since the summed contributions to the loss over the dataset $\{(x_i, y_i), i \in [m]\}$ are independent from one another:

$$\frac{\partial}{\partial x_k} \sum_{i=1}^{m} \ell(f(\theta, x_i), y_i) = \frac{\partial}{\partial x_k} \ell(f(\theta, x_k), y_k).$$

Therefore the batched adversarial problem (2.4.1) can be written in a batched manner:

$$\max_{\tilde{x}_i \in \mathbb{B}_\varepsilon^\infty(x_i), \forall i \in [m]} \frac{1}{m} \sum_{i=1}^{m} \ell(f(\theta_i, \tilde{x}_i), y_i). \tag{2.4.5}$$

We can also write a batched version of problem (2.4.4) for which C&W approximates the argument:

$$\min_{\tilde{x}_i \in \mathbb{B}_\varepsilon^\infty(x_i), \tilde{y}_i \neq y_i, \tilde{y}_i \in \mathcal{Y}, \forall i \in [m]} \frac{1}{m} \sum_{i=1}^{m} \ell(f(\theta, \tilde{x}_i), \tilde{y}_i). \tag{2.4.6}$$

These batched expressions are useful if one wants to add a penalty that encourages the diversity of the perturbations $(\tilde{x})_{i \in [m]}$ or any other metric mixing the different $(x_i)_{i \in [m]}$ or $(y_i)_{i \in [m]}$. In Chapter 4 we will be using a coverage metric that will be computed over a batch of data thus only the formulation in equation (2.4.5) allows us to achieve as much.

A direct parallel can be drawn between adversarial attacks and the pointwise robust counterpart inner maximization problem: the procedures PGD, FGSM and C&W attempt to approximate the argument of the inner maximization (2.3.2) with an uncertain set of the form of a infinite ball on the input exclusively and no label perturbation.

2.4.2 Adversarial defenses

The goal of adversarial defenses is to mitigate the impact of adversarial attacks. Defenses did not resort originally to adversarial attacks as they initially used regularizations such a weight decay, dropout [173] or distillation

[142]. However these methods were not giving a satisfying performance against adversarial attacks therefore current methods prefer a two step approach: finding a solution $\tilde{x}$ that approximates equation (2.4.1) and then minimize the loss with respect to this perturbation $\tilde{x}$

Let $L : \Theta \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}, L(\theta, x, y) \mapsto \ell(f(\theta, x), y)$ be a loss function, θ the parameters of the model, (x, y) an input/output tuple and C a bounded perturbation set. We then define an attack (Attack $\in \{\text{FGSM}, \text{PGD}, \text{C\&W}\}$) as a procedure that takes L, θ, C, x and y and returns a perturbed input $\tilde{x}$. The adversarial training writes:

$$\begin{aligned} &\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\hat{\rho}_k} \left[\ell(f(\theta, \tilde{x}), y)\right] \\ &\text{where} \quad \tilde{x} = \text{Attack}(L, \theta, C, x, y). \end{aligned} \tag{2.4.7}$$

If the attack is exactly computing the maximum in a ball $C = \mathbb{B}_\varepsilon^\infty(x)$ (equation 2.4.1) then the problem becomes exactly the pointwise robust counterpart:

$$\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\hat{\rho}_k} \max_{\tilde{x} \in \mathbb{B}_\varepsilon^\infty(x)} \ell(f(\theta, \tilde{x}), y). \tag{2.4.8}$$

Therefore adversarial training and the pointwise robust counterpart are one and the same if the attack manages to find exactly the optimal in the neighborhood. That much cannot be guaranteed when resorting to gradient methods on non convex-concave saddle point problem. Nonetheless, this approach is computationally efficient and it has the best performance results on perturbed datasets.

2.5 Lipschitz continuity

The Lipschitz continuity is a property that bounds the variations of a continuous non-smooth function: the smaller the constant is, the smaller variations of the function are. This is closely intertwined with the notion of robustness as it gives a direct way of upper bounding the changes of a function with respect to perturbations in its arguments. This is overall a suitable indicator of a well-regularized decision process that can be explicitly enforced or conditioned throughout a dedicated training.

Definition 2.5.1. Let $\mathcal{X} \subset \mathbb{R}^l, l \in \mathbb{N}$ and $\mathcal{Y} \subset \mathbb{R}^r, r \in \mathbb{N}$, equipped with the usual norms, a function $f : \mathcal{X} \to \mathcal{Y}$ is called Lipschitz if there exists a constant L such that for all $(x_1, x_2) \in \mathcal{X}^2$

$$\|f(x_1) - f(x_2)\|_2 \leq L\|x_1 - x_2\|_2. \tag{2.5.1}$$

Of course, any $L' > L$ is also a Lipschitz constant of f, for simplicity the minimal Lipschitz constant $L_{f,\mathcal{X}}$ of the function f is called the Lipschitz constant of f.

$L_{f,\mathcal{X}}$ can also be expressed as the upper bound of the differential quotient:

$$L_{f,\mathcal{X}} = \sup_{x_1, x_2 \in \mathcal{X}^2} \frac{\|f(x_1) - f(x_2)\|_2}{\|x_1 - x_2\|_2}. \tag{2.5.2}$$

Let f be a neural network as defined in (2.2.2). In the case of a multilayer perceptron the Lipschitz constant of f over $\mathcal{X}$ in the norm $\|\cdot\|$ is

$$L_{f,\mathcal{X}}(\theta) \overset{\text{def}}{=} \sup_{x \in \mathcal{X}} \left\| \left(\prod_{i=1}^{d-1} W_i^\top \text{diag}(G_\phi(f_i(\theta, x))) W_d^\top \right) \right\|, \tag{2.5.3}$$

and $G_\phi(f_i(\theta, x))$ is an appropriate generalized Jacobian of ϕ evaluated at $f_i(\theta, x)$. Typically G is a conservative field [25] in which case $L_{f,\mathcal{X}}(\theta)$ is indeed a Lipschitz constant whenever $\mathcal{X}$ is convex [37]. When ϕ is differentiable this generalized gradient is the singleton of the Jacobian.

By standard conjugacy results, we also have

$$L_{f,\mathcal{X}}(\theta) \overset{\text{def}}{=} \sup_{x \in \mathcal{X}, U \in \mathbb{B}^*} \langle U, \left(\prod_{i=1}^{d-1} W_i^\top \text{diag}(G_\phi(f_i(\theta, x)))) W_d^\top \right) \rangle, \tag{2.5.4}$$

where $\mathbb{B}^*$ is the unit ball for the dual norm (nuclear norm).

2.6 Kernel density estimation

Kernel density estimation (KDE) [170, 102] is a non-parametric method to build from $k \in \mathbb{N}$ samples $(x_i)_{i\in[k]}, x_i \in \mathbb{R}^p$ an approximation $\tilde{\rho}$ of the unknown underlying density function ρ. The method requires the use of a kernel $K : \mathbb{R}^l \to \mathbb{R}_+$ which is a positive and normalized measure:

$$\forall x \in \mathbb{R}^l, K(x) \geq 0 \text{ and } \int_{\mathbb{R}^p} K(x)dx = 1.$$

The predictor of the distribution ρ is then built by summing contributions from the points $(x_i)_{i\in[k]}$ to the overall estimated distribution:

$$\tilde{\rho}(x) = \frac{1}{hk}\sum_{i=1}^{k} K(\frac{x - x_i}{h}). \tag{2.6.1}$$

The choices of K and h depend on our prior knowledge about the distribution ρ: usually K is taken symmetric, centered around 0. Depending on the physics that are observed when drawing the points other arbitrary choices can be made: using a symmetric kernel might prove to be less efficient than a skewed and long right-tailed kernel when trying to approximate an exponential law. Nonetheless popular choices for K are the Gaussian, uniform, Epanechnikov or triangular kernels. Concerning h, the choice of the value boils down to the regularity class of ρ: for a Gaussian kernel there exists a general criterion [170] that ensures some regularity of the estimated distribution given $h = \hat{\sigma}(\frac{4}{3k^{\frac{1}{5}}})$ where $\hat{\sigma}$ is the standard deviation of the samples $(x_i)_{i\in[k]}$. This h is not suited for all possible approximated distribution and one can add extra specific heuristics in order to select it more suitable.

2.7 Probability measures

Considerations on probability measures are key to distributional robustness. Confusion between a set and the set of probability distributions over it should be avoided since we will switch regularly between the two. We therefore aim here at explicitly distinguish the different metrics in each space.

For a subset $\mathcal{C} \subset \mathbb{R}^m$, let $\mathcal{B}$ be the Borel sigma algebra on $\mathcal{C}$. $\mathcal{M}_+(\mathcal{C})$ denotes the cone of non-negative measures on $(\mathcal{C}, \mathcal{B})$ equipped with the finite total variation norm. We also define $\mathcal{P}(\mathcal{C})$ the space of Borel probability measures supported on $\mathcal{C}$

$$\mathcal{P}(\mathcal{C}) \overset{\text{def}}{=} \left\{ \varphi \in \mathcal{M}_+(\mathcal{C}) : \int_{\mathcal{C}} \mathrm{d}\varphi(x) = 1 \right\}.$$

δ_x is the Dirac measure at x.

For any $(\nu, \mu) \in \mathcal{P}(\mathcal{C})$, the Kullback-Leibler divergence between μ and ν is

$$\mathrm{KL}(\mu, \nu) = \begin{cases} \int_{\mathcal{C}} \log(\frac{\mu(x)}{\nu(x)})\mathrm{d}\mu(x) & \text{if } \mu \ll \nu \text{ and } \int_{\mathcal{C}} \left|\log(\frac{\mu(x)}{\nu(x)})\right| \mathrm{d}\mu(x) < \infty \\ +\infty & \text{otherwise,} \end{cases}$$

where $\ll$ stands for absolute continuity of measures.

Let $\mathsf{c} : \mathcal{Z}^2 \to \mathbb{R}_+$ be a continuous and symmetric ground cost function is ($\mathsf{c}(z, z') = \mathsf{c}(z', z)$) and $\mathsf{c}(z, z') = 0 \Leftrightarrow z = z'$. For $(\nu, \mu) \in \mathcal{P}(\mathcal{C})$, denote $\Pi(\mu, \nu)$ their couplings, i.e., joint probability measures π on $\mathcal{Z}^2$ whose marginals are μ and ν. The Wasserstein distance between μ and ν with ground/transportation cost c is

$$W_{\mathsf{c}}(\mu, \nu) = \inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathcal{Z}^2} \mathsf{c}(z, z')\mathrm{d}\pi(z, z').$$

When $\mathsf{c}(z,z') = \|z-z'\|_q^q$, $q \geq 1$, then $W_{\mathsf{c}}^{1/q}$ is indeed a distance, known as the q-Wasserstein distance. We will denote it W_q.

For $(\nu,\mu) \in \mathcal{P}(\mathcal{C})$ let $\mathcal{F}$ be a class of function on $\mathcal{C}$ then the Maximum Mean Discrepancy (MMD) [86, 26] between μ and ν is:

$$\mathrm{MMD}[\mathcal{F},\nu,\mu] \stackrel{\text{def}}{=} \sup_{f\in\mathcal{F}} \left(\mathbb{E}_{x\sim\nu}[f(x)] - \mathbb{E}_{y\sim\mu}[f(y)]\right)$$

If $\mathcal{F}$ is the unit ball of a reproducing kernel Hilbert space with associated positive definite kernel $k : \mathcal{C}^2 \to \mathbb{R}$ then the MMD has a tractable expression:

$$\mathrm{MMD}[\mathcal{F},\nu,\mu] = \mathbb{E}_{\substack{x\sim\nu\\ x'\sim\nu}}\left[k(x,x')\right] - 2\mathbb{E}_{\substack{x\sim\nu\\ y\sim\mu}}\left[k(x,y)\right] + \mathbb{E}_{\substack{y\sim\mu\\ y'\sim\mu}}\left[k(y,y')\right]$$

The MMD distance benefits from a better tractability than the Wassertein distance and less constraints on the support of the distributions than the KL divergence.

2.8 Γ- or epi-convergence

We will invoke the notion of Γ-convergence, which plays a fundamental role in convergence of optimization problems (values and extrema points). In finite dimension, Γ-convergence of a sequence of functions corresponds to convergence of their epigraphs. The interested reader may refer to [128] for a comprehensive treatment.

2.9 Tameness

We will need the notion of tame functions (and sets). A rich family will be provided by semialgebraic functions, i.e., functions whose graph is defined by some Boolean combination of real polynomial equations and inequalities [49]. Definable functions on an o-minimal structure over $\mathbb{R}$ correspond in some sense to an axiomatization of some of the prominent geometrical properties of semialgebraic geometry [48, 63]. O-minimality includes many important structures such as globally subanalytic sets or sets belonging to the log-exp structure hence covering the vast majority of applications in learning, including neural network learning with various activation and loss functions. A slightly more general notion is that of a tame function, which is a function whose graph has a definable intersection with every bounded box. We then use the terminology definable for both. Given the variety of optimization problems that can be formulated within the framework of definable functions and sets, our convergence results will be stated for this class. The reader unfamiliar with these notions can just replace definability by semialgebraicity.

2.10 Clarke subdifferential

We now summarize a few properties of the Clarke subdifferential that will be useful to us in this contribution (see [43]).

Proposition 2.10.1. *Let $f, g : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz continuous functions, then*

(i) $\partial^C(\lambda f)(x) = \lambda \partial^C f(x)$, $\lambda \in \mathbb{R}$.

(ii) $\partial^C(f+g)(x) \subset \partial^C f(x) + \partial^C g(x)$.

(iii) Consider the family of functions $(f_t)_{t\in T}$, where T is a compact space and $t \mapsto f_t(x)$ is upper semicontinuous. Suppose that for each t, $f_t : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous. Let $f(x) = \max_{t\in T} f_t(x)$. Let S be a subset of full Lebesgue measure, then

$$\partial^C f(x) \subset \operatorname{conv}\left\{\lim_{k\to\infty} \nabla f_{t_k}(x_k) : \ x_k \to x, x_k \in S, t_k \in T, f_{t_k}(x) \to f(x)\right\}. \tag{2.10.1}$$

If, moreover, the functions f_t are of class $\mathcal{C}^1$ such that $f_t(x)$ and $\nabla f_t(x)$ depend continuously on (t,x)[1]*, then:*

$$\partial^C f(x) = \left\{\int_T \nabla f_t(x)\mathrm{d}\mu(t) : \ \mu \in \mathcal{P}\Big(\operatorname*{Argmax}_{t\in T} f_t(x)\Big)\right\}. \tag{2.10.2}$$

Remark 2.10.2. We have made no effort to further weaken the assumptions in the calculus rules of Proposition 2.10.1 since they are sufficient for our purpose.

[1] Functions f such that these assumptions are verified are known as lower-$\mathcal{C}^1$ functions; see [158].

Chapter 3

State of the art

In this chapter we summarize the current state of the art concerning robustness of neural networks. The current challenges in deep learning attracted a broad public, since the issue of robustness against adversarial samples remains unsolved and the demand for reliable and trustworthy algorithms is growing. There are lots of promising takes on the problem and we offer here an appreciation of the different shades of this literature. Our starting point is verification methods that can assess the trustworthiness of a given assertion on a neural network, followed by testing methods that aim at detecting possible defects in a implementation. Finally, we present methods made for robust learning that incorporate robustness at training times of the statistical model.

In order to provide a more synthetic vision of the state of the art we categorized the different contributions in three main categories, a color scale is used to present the type of the guarantees offered.

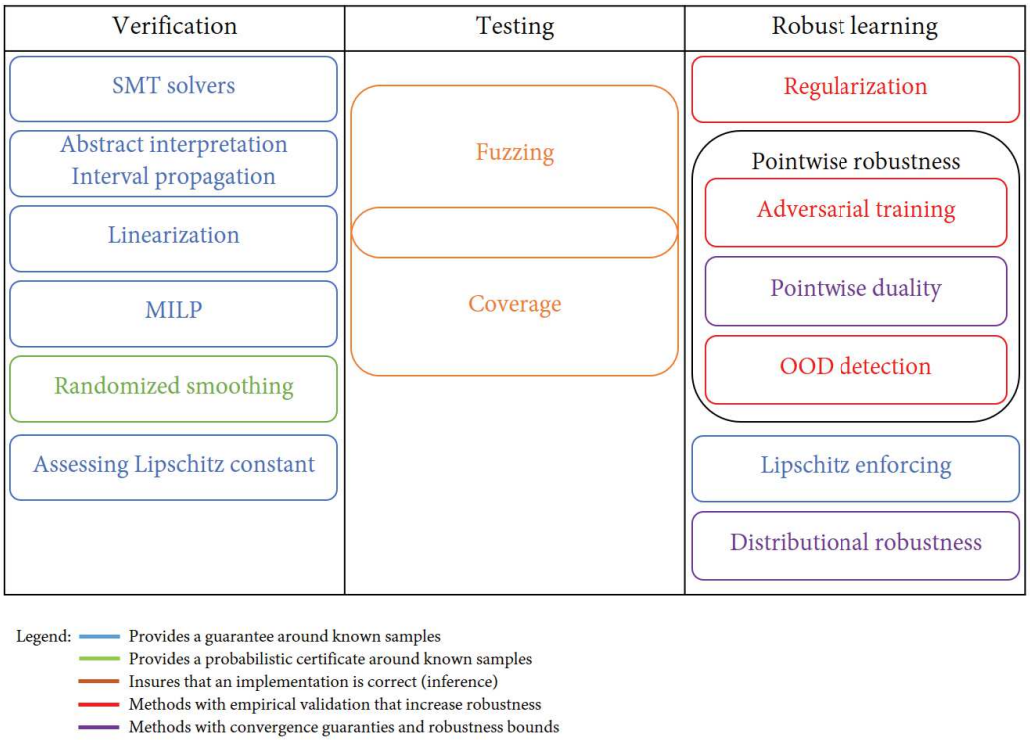

Figure 3.1: Summary of the state of the art on robustness of neural networks.

In Figure 3.1, Chapter 4 is located in the testing bibliography among works on coverage. Chapter 5 is located in the testing literature as well but it focuses on the Lipschitz assessing algorithms. Our third contribution in Chapter 6 deals with robust learning and bridges pointwise robust learning with distributionally robust learning. We detail all these categories hereinafter in their dedicated sections.

3.1 Robustness verification

The first group of methods about robustness is posterior robustness, where the aim is to verify given properties of a network once it is trained. The property consisting of a stable prediction around training points can then be assessed and ideally confirmed. Lots of different techniques have been suggested and they can be qualified of being either deterministic or probabilistic.

3.1.1 Deterministic methods for verification

SMT solvers. Deterministic methods are methods that can guarantee if a model is robust around a given datapoint. We would like to emphasize that, unless explicitly said, a deterministic method can usually validate but not invalidate local robustness (this is a yes/maybe, rather than a yes/no type of answer). Using Satisfiability Modulo Theories (SMT) solvers [105, 9, 149] one can verify explicitly if a property is met, for instance, if the prediction is stable around a datapoint. The method explores the set of constraints to be verified using the piece-wise linear structure of the network and making disjunction of cases at each ReLU activation that encounters 0 on the domain to be verified: combinatorial explosion of cases makes these techniques impractical for large networks.

Abstract interpretation. Abstract interpretation [78, 133, 120, 171] aims at processing a specific convex envelope (such as boxes or polytopes) of the input uncertainty set throughout a modified network that maintains soundness: if the propagated envelope has constant label, then the robustness property is kept on the uncertain set. This method has, however, a major flaw: one needs to be able to test a property continuously on the propagated set which could be multidimensional.

Interval propagation. The principle behind interval propagation [84, 186, 205] is simple: given an input set that is not necessarily finite, the goal is to compute a convex over-approximation of the output set. This approach is fairly close to abstract interpretation that we mentioned in the previous paragraph. This technique allows to certify the neighborhood of a given point even on a test set: if the output interval has constant label then the input interval around the tested point is certified. Such approach has the same flaws as Abstract Interpretation, therefore we grouped them together in our summary in Figure 3.1.

Linearization. Linearization of activations [188] allows to check the property on a finite set of values (vertices of a polytope around the prediction), but for deep neural network the approximation is quite loose and it does not scale very well to large networks.

Mixed Integer Linear Programming (MILP). Mixed integer linear programming [75, 38, 97] is another form of linearization where the approximation of the ReLU function is similar to the one of ReluPlex [105]. These methods do not scale well with the size of the model: they often have to make distinction of cases which number grows exponentially with the size of the network. They are often impractical or give poor results when probed on large networks, since they hardly scale with the dimension of the parameters of the model.

Computing the Lipschitz constant. A measure of the robustness to input perturbations is the Lipschitz constant. It measures how much the output of the function can change when its input is modified: it is therefore a key metric when assessing the variation of the decision-making of a neural network on a small neighborhood. Measuring and controlling the Lipschitz constant has hence become a major challenge to characterize robustness or to build robust neural networks [28]. Indeed, the lack of robustness can be caused by Lipschitz constants not being constrained during training, which in turns, allows small perturbations to cause large changes in the

decision function. But estimating the Lipschitz constant of a neural network is an NP-hard problem [183] resulting in papers often facing an accuracy/complexity dilemma and resorting to different upper bounds [46] that are guaranteed or not. The assessment of a Lipschitz constant answers two specific needs: the first one is the need for verification where the goal is to ensure that around specific datapoints a certain property is met -often stability of the prediction-. The second one is the need to ensure more stability to perturbation on any sample of the data distribution, i.e. robustness. Having access to local Lipschitz constant ensures robustness over local neighborhood [152, 71] as it can provide bounds of possible deformation around data points. The causal link between the Lipschitz property and robustness has fueled research in the last years with loose estimations of the Lipschitz constant first [99] and then tighter ones: for instance Seqlip [183] is reformulating the computation of the Lipschitz constant in a maximization problem on activation variables that, if solved exactly, provides an upper bound for the Lipschitz constant. On the other hand, Semi Definite Programming (SDP) relaxations such as LipSDP [71] or [152] give a certified upper bound. Computing the Lipschitz constant of ReLU networks can also be cast as polynomial optimization, for which the sparse version of the Lasserre's SDP-hierarchy was proposed in [37] to provide upper bounds that are sometimes strict improvements over previously known upper bounds. All these approaches only test their methods on random networks and trained networks that have, by design (or by lack of design), an unknown and varying Lipschitz constant. This results in mixing different sources of variance when approximating it: the variance of the training of the network and the variance of the Lipschitz assessing algorithms. We address this point in Chapter 5.

3.1.2 Probabilistic methods

These methods can scale respectively well on large models contrary to the aforementioned deterministic verification methods but they are still very much dependent of the input dimension. However, their assessment of the robustness relies on a sampling strategy that might require lots of costly computations. Overall, probabilistic methods fail to quantify the presence of null measure sub-spaces that could contains adversarial attacks.

Random smoothing. Random smoothing [45, 119] provides a probabilistic robustness certification bound. It requires the knowledge of the percentage of the perturbation space classified correctly which could be challenging to assess in high dimensions, however the effect of sampling on the probabilistic bound is not assessed as it should depend on the dimension of the problem. The main strength of the method comes from its scalability to large networks, giving a good alternative to other verification methods.

Measuring robustness. [9] suggests statistically relevant metrics such as frequency and severity of adversarial examples. These metrics offer an alternative to the usual "adversarial accuracy" that is used to quantify the robustness of a given network. It provides a better appreciation of the overall neighborhood of points in a dataset. Let us emphasize once again that the sampling method and sampling size are key to providing a proper estimation and that it is heavily dependent on the dimension and its influence on the metrics given is not assessed.

Verification tools are necessary as they allow to prove formal assertions of the final solution and validate a neural network. However, it must be noted that these methods hardly scale either with the dimension of the model or the dimension of the input. We would like to stress that although a model that passes these tests can be qualified as robust, these methods do not provide ways of improving the robustness of the model. Designing a neural network robust can only be performed prior to verification: these methods are necessary but not sufficient for robustness. They allow to target specific flaws in a trained network and cast a new light on the inner working of these "black-box" models but they do not essentially design robust models.

3.2 Neural network testing

Testing methods answer the need of generating tests suites that probe a statistical model. They differ from verification methods since their goal is to expose vulnerabilities instead of trying to demonstrate resistance to perturbations [206]. In a sense they are mandatory when the production environment is different than the development environment. This mostly covers two scopes of research:

1. Approaches based on code coverage criteria that aim at finding and exploiting blind spots in an activation pattern.
2. Approaches based on fuzzing: an automated test case generation that aims at finding bugs of implementations.

3.2.1 Coverage based testing

Measuring coverage. The goal of coverage based techniques is to generate erroneous behaviors/predictions by studying and exploiting the parts of a statistical model that are not used in its training set. This originated with deepXplore [145], that leveraged the activation pattern to generate erroneous examples. The authors assess, via back-propagation, the gradient direction on the input that increases a specific activation in the same fashion that adversarial attack increases the loss of the network. Following this methodology the authors manage to generate fooling samples, however, the impact of coverage in their method is unclear as it is optimized in conjunction with an adversarial loss to generate erroneous samples. In order to test more plausible transformations, DeepTest [179] optimizes the same coverage criterion with greedy search. DeepGauge [124] extends the notion of coverage with a higher level of granularity by not only considering if a neuron is activated, but by considering also smaller intervals of the activation landscape. This has the advantage of showcasing blind spots even in multimodal patterns. The only drawback from this metric is that there is no explicit way of optimizing it directly: the authors use attacks from the literature and show that they tend to increase coverage. Their experimentation shows only that adversarial attacks imply a greater coverage, but they do not assess if increasing coverage is an efficient way of generating adversarial attacks. [177] introduced structural coverage that aims at quantifying the relations between features, i.e. if neural network activations have a correlated pattern whereas previous contributions considered the coverage of the neurons independently. In a attempt to have a more interpretable coverage metric [195] introduced Neuron Path Coverage as a more elaborated way to highlight paths in neural networks that have an influence on their prediction.

Coverage vs Robustness. In order to investigate the possible correlations between robustness and coverage, [199] and [35] independently computed Pearson's correlations between a wide range of coverage metrics and a wide range of robustness metrics and noticed how little correlation there was between the two. However, Pearson's correlation only displays linear dependencies between variables, thus this does not completely discard a possible link between robustness and coverage. [199] tried as well to include coverage attacks into the training of neural networks just like adversarial attacks could be used at training times to increase the robustness of the learning procedure (the idea was originally suggested in DeepXplore [145] but was not assessed). However, this training on coverage attacks did not prove to be effective in improving the robustness. In order to avoid making the arbitrary choice of the number of sections as done in DeepGauge [124], [90] suggested a metric using a KL divergence between the layer-wise activations values and a uniform distribution. This also has the advantages of providing a subdifferentiable metric that can be optimized concurrently with the loss and to handle activation in a more continuous manner. However, the relevance of using a layer-wise coverage metric instead of a neuron-wise is not discussed in the paper.

3.2.2 Fuzzing

The goal of most papers elaborating on fuzzing is to find implementation mistakes rather than wrong predictions. To that purpose, fuzzing is a method that generates a test suite that covers a maximal percentage of the code and monitors for induced bugs, errors or failures. The test suite needs not be representative of the input distribution, thus enabling a broader search space. The first seminal work on fuzzing adapted to DNN testing, [141], is offering a coverage guided test suite where examples are generated in order to maximize a diversity of activation patterns. DeepHunter [196] tries to achieve the same goal but using mutations that are closer to image perturbations (such as changes in brightness, blur, contrast, translation, rotation or scaling) and the diversity metric for guiding generation of samples is a distance between six different testing criteria. Both approaches work remarkably well for finding defects in implementations. Other methods for fuzzing change the generation method, such as Generative Adversarial Networks (GANs [81]) [207] or changed the coverage criterion used for samples generation [89]. Fuzzing can easily be a compulsory step for certifying a deep learning solution that is to be put to production in order to test the production environment.

Overall, research around coverage based testing is mostly empirical and some of the initial conclusions on the subject were quite optimistic concerning the benefits on robustness, a vision that currently appears flawed [121, 62, 164]: we address this matter in Chapter 4. Concerning Fuzzing, the end goal is different as it aims at finding defect of implementation rather than explaining model misconduct. Coverage is used in accordance with its initial purpose from code coverage: explore all branches of the code in order to ensure that inference can be performed without bugs regardless of possible errors in the prediction.

3.3 Robustness aware learning

As mentioned above, verification and testing focus on trying to validate a deep learning solution. On the other hand, the third group of robustness methods consists in modifications of the learning procedure so as to build a robust model. It can thus be used sequentially with testing methods that can check the extent of the obtained robustness a posteriori. These approaches are a very active field of deep learning: they aim at finding a more suitable set of parameters by incorporating prior knowledge on the possible perturbations. Some methods are explicitly enforcing constraints on the network during training, this ensures the compliance of the resulting network with the constraints. Other methods aim at regularizing the network or making changed training procedures without given constraints, hoping to find an optimal configuration with a suitable robustness.

3.3.1 Preprocessing robustness

Out Of Distribution (OOD) detection. A first approach to prevent spurious behavior caused by an adversarial sample is to detect or preprocess the inputs before performing a prediction on them: the OOD literature tries to achieve as much by adding a preliminary step in the processing of an input. Among the literature, the goals of these preprocessings differ: it can be to spot and remove adversarial samples via a dedicated classification before feeding them to the network [122, 117, 143, 129, 87]. Other methods resort to principal component analysis in order to identify adversarial attacks (linear Principal Component Analysis -PCA- [95], nonlinear PCA [18]), or even analyse the activation pattern with respect to the activation trace (distribution of the activations over the training set) using the MMD distance [87] or KDE estimation [74]. These techniques can be very powerful for monitoring the behavior of the model, but one has to make sure that these methods can not be attacked as well (for instance, methods based on neural networks): [31] showed that these methods can easily be bypassed by powerful adversarial attacks. OOD can help providing a diagnosis, yet it does not make our prediction models less sensitive to perturbation: it simply censors erroneous samples. The guarantees of these methods depend on

the guarantees of the detection/preprocessing methods: if a method is based on neural networks this translates the robustness issue from the model to the diagnosis tool that is supposed to bring some extra robustness.

3.3.2 Prior explicit robustness

Lipschitz enforcing. A first category of methods relies on inducing explicitly boundness of variations in a neural network. [134, 203] for multilayer perceptron, [42, 166, 83] for convolutional networks, and more recently [55] for graph neural networks propose to bound the parameters at training times in order to enforce an upper bound of the Lipschitz constant. But these methods require much more computation than vanilla trainings, even though they allow knowing beforehand the resulting maximal variations of the network. For any classification problem, for example, we can find a 1-Lipschitz network that has 100% accuracy on the problem [28]. Advocating evil, this also means that shrinking the Lipschitz constant for classification tasks never cancels out the risk of overfitting. That much is not true for regression problems, where the phenomenon that one wants to approximate has an intrinsic regularity: any attempt to fit a statistical model that has lower regularity than the phenomenon would be unable to perform the regression perfectly. Bounding the Lipschitz constant has a direct implication in limiting steep variations of the function, and thus overfitting.

Regularization. More traditional techniques already existed in order to provide a sense of sparsity and regularity for a learned function such as weight decay [91]. They apply methods from linear modeling that incorporate ℓ_2 regularization to provide models more robust to input perturbations. Recently, more elaborated methods are emerging, for instance, using RKHS norm as a regularization [19] directly influences the Lipschitz property of the learned model. Alternatively, [204] suggests a regularization that reduces the difference between the prediction and a possible perturbation of this prediction, rather than between the perturbed prediction and the true label. More specific techniques have also been developed for different neural architectures: dropout [173] or batch normalization [101] for instance. These empirical regularizations have proven to be very effective in practice, since they are designed to compensate for some poor conditioning of the neural networks. In addition, they rely on strong practical results and efficient implementations that make them popular. They are based on a sparsity assumption, meaning that between two equivalently performing models, the less complex one should be preferred, since over-parametrization may lead to over-fitting. This abides by the Occam's Razor principle: a simpler theory or explanation is often preferred over more complex or far-fetched ones. Removing unused parameters allows to derive simpler causal links in a model and should lead to simpler models. At this point, it is worth recalling that Occam's Razor principle is an empirical view and that it has no grounding on a nonlinear model that learns on an empirical distribution. On top of that, the assumption of sparsity of a problem might simply not be relevant or adequate, since there might not be a simple solution when considering a complex problem.

3.3.3 Prior implicit robustness

As opposed to the previous group that incorporates robustness in the training formulation by enforcing sparsity, we focus now on the works that attempt to formulate the problem as an optimization under uncertainty (also called robust optimization), where regularization might come as a side effect. It is proven that, for linear predictors or support vector machines, regularization with a proper distance and robust optimization are the one and the same [68, 197], however for models non convex in the parameters and non concave in the input it is relevant to consider the two approaches as different.

Theory on robust optimization [12, 16, 17, 184] is rather advanced when it comes to linear predictors or linear decision-making [11, 12, 36]. It aims at finding the set of parameters that minimizes the worst possible error on a perturbation set. However, the theory does not currently provide a general algorithm and guarantees of convergence for robust nonlinear problems without additional hypothesis. Current empirical approaches

differ on the perturbation they consider: either pointwise perturbations or distributional. These approaches are parameter-free for the perturbation set, in the sense that no hypothesis is made on its distribution. A common way to achieve this is by assuming a finite support for the possible perturbations without knowing its density. This aims at finding models that minimizes the objective for any possible perturbation lying in the uncertain set. These approaches are more compliant with our framework than previous methods due to the fact that industrial applications can have access to detailed information on the possible uncertainties of measures or approximation errors in computer codes. This approach to robust optimization that guarantees mitigation of the loss in the presence of uncertainty can be over-pessimistic if the uncertainty set is overspecified. It is therefore key to include all prior information in the learning process to avoid ending in a suboptimal solution of the optimization process.

3.3.3.1 Pointwise robustness

Adversarial attacks and trainings. A large amount of work on pointwise robustness was achieved through the scope of adversarial attacks [126, 82, 69], which we mentioned in the introduction as a serious menace to modern deep learning. Concerning the creation of these attacks, papers usually resort to local projected gradient ascents [82, 126] in an either single step or multi-step fashion that, despite the lack of convergence guarantees, perform exceptionally well. Other notable approaches resort to GANs [192] or objectives that aim at diverting the prediction towards its second best choice [32]. Early attempts at making specific defenses included distillation [142] that was debunked using strong attacks [32]. An efficient way to make models robust is to use adversarial samples to further train the model to design it less insensitive to gradient based attacks (see the dedicated section 2.4.2 in the prerequisites). This has been called adversarial training [126, 180, 193, 52] as opposed to adversarial attacks. These methods remain as of today the most used methods for making a network robust to local perturbations due to their simplicity of use and their computational efficiency. When considering computer vision applications, adversarial attacks take a new dimension: indeed, it is expected that under light transformations such as light translation, rotation, cropping or brightness changes the prediction should not change (such transformations are called *unrestricted attack* as opposed to *restricted attacks* that are constrained in balls). To that purpose, several papers used the same idea from adversarial training by simply performing data augmentation with a broad range of these unrestricted attacks [150, 88, 4]. These papers manage to build models that are reasonably adversarially robust without the use of adversarial attacks on images. As of today, adversarial training is the best empirical method for making DNN robust against adversarial attacks, despite all these advances. Ultimately, adversarial attacks and defense are broad empirical exploration of pointwise robustness with little theoretical results. This is not to downplay the contributions of the aforementioned authors as they paved the way for a more in depth study of robustness of neural networks but it showcases theoretical shortcomings in robustness.

Pointwise duality. Other pointwise approach took the perspective of robust optimization and offered changed training procedures [189, 190]. They presented theoretical guarantees on the result of their optimization: this has the advantage to provide bounds on the training error. [151, 94] provide similarly certified bound on adversarial attacks on the training set. Both authors do not discuss the generalization on a test set of their approaches.

3.3.3.2 Distributional robustness

This approach has gained momentum as it is more general than the pointwise robustness [174]. Distributional robust learning aims at minimizing a worst case expected loss, where the worst case is chosen among all possible distributions in a bounded perturbation set. This distributional perturbation set is often called an ambiguity set, the name coming from seminal works in robust optimization [12, 58, 30]. For linear regression, SVM [36] or logistic regression [168] the distributional robust problem is tractable and has links with regularization.

But this does not hold true for non convex/non concave settings: generality comes at the cost of tractability. Distributional optimization problems are not always tractable and necessitate extra layers of approximations to solve them. Approaches differ in the definition of their perturbation set such as Wasserstein ambiguity sets [22, 21, 110], Entropy regularized Wasserstein ambiguity sets [185], MMD ambiguity sets [175], ϕ-divergence ones [65, 41]. Changing the distance allows papers to trade generality for guarantees or tractability. For instance [172] deals with Wassertein distributional robustness and provides robustness bounds along with a training procedure. However, the algorithm for training the network resorts to an oracle that is able to find the maximum of a non convex/concave function on a compact set. This can be achieved reasonably accurately in small dimensions but, as dimension increases, it becomes more and more tedious to find the local extrema input since the problem is NP-hard. For a more exhaustive review on distributional robustness, see [155]. We address the problem of approximating this NP-hard problem with smoothing and Monte Carlo sampling that insures convergence in the limit to the robust optimization problem in Chapter 6.

3.3.3.3 Parametric robustness

Bayesian approaches in deep learning usually deal with the epistemic uncertainty which, as already elaborated in section 1.1.1, is the uncertainty due to the lack of knowledge of our system (for instance if an explanatory variable is missing). Bayesian neural networks give a confidence interval on the prediction that can be used to appreciate the relevance and reliability of an output. Traditionally a prior distribution is assumed on the weight and this distribution is updated in the light of the incoming data. Pioneers papers date back to [125, 137]. They used costly methods for computing the posterior (Newton method or Hamiltonian Monte Carlo) to tackle their problems, but they are not scalable to larger models. In order to scale to larger models, variational inference was proposed [103, 85], where it assumes a parametric family of distributions for the posterior and tries to find the posterior parameters that fit the most the real posterior distribution. It reduces the amount of computation to perform on larger models but still remains computationally demanding. The latest takes aim at making this approach tractable for highly dimensional neural networks using Dropout [76] where the author presents the dropout mechanism as a form of Bernoulli Bayesian inference. This has the advantage of providing confidence bounds at a much lower cost. Prior assumptions on dense networks have some interpretation when considering one linear layer, however it remains unspecified what prior assumption should be made on convolutional layers: none of the aforementioned papers formally justify the use of priors on more complex structures than linear layers.

Robust learning precedes Verification and Testing as it seeks to improve the training procedures by incorporating uncertainties in the design process of the models. It has been the preferred way of dealing with robustness in the last decade in order to mitigate the effect of all three possible sources of uncertainties: uncertainties from the inputs, the labels or the models. Most of the work focused on input uncertainties i.e. possible changes in the input domain either in a pointwise manner or in a distributional manner. Traditional methods -inherited from classical robust linear learning- resort to regularization or constraint enforcing (such as enforcing Lipschitz constants) but do not always have a proper interpretation or meaning on nonlinear models, despite good empirical results. When it boils down to practical results, adversarial training remains unchallenged: it is computationally the most effective and it scales well to large dimensions. However, it does not provide any theoretical guarantees. Algorithms that are guaranteed to converge towards the optimal model suffer from high dimensions and/or have tractability issues. None of the approaches mentioned in this state of the art meet all the desired requirements of a robust training method: adaptable to all architectures, scalable, tractable, certified and, most importantly, mitigating the effect of uncertainties.

Chapter 4

An experimental approach on robustness following coverage

Main contributions of this chapter

- Provide a (sub)differentiable coverage metric that can be optimized efficiently and concurrently with the training loss.
- Provide empirical assessment of the effect of increasing coverage in conjunction with adversarial training.
- Disentangle the notion of neuron coverage and robustness.

Contents

4.1 Introduction

In the field of deep learning the ReLU activation function [109] triggered a small revolution. It provided significant advantages compared to traditional smooth activation functions (for instance: sigmoid, tanh, softmax): there are no exponentials to compute thus reducing the computation load both of the forward pass (i.e. performing the prediction) and the backward pass (i.e. computation of the gradients for the update of the model). On top of this, the ReLU function is an elegant solution to the vanishing gradient problem: deep neural nets with sigmoid or tanh activation function are shrinking the value of the gradient exponentially in the depth of the network

thus resulting in infinite training times for these deep neural nets. The ReLU function keeps the norm of the gradient constant as it activates and thus allows updating efficiently deep neural structures. The smoothness and boundedness of traditional activation function was traded against computability and now major state of the art architectures in image recognition [93, 6, 157] are resorting to this activation function as part of their building blocks. However, the downside is that, in the processing of a finite number of inputs, some activations (outputs of hidden layers) might never overpass the 0 threshold and thus part of the weights may never have an impact on the prediction of the network. This results in having parameters of the model that remain unaltered during the whole process of training. Since the test distribution is different from the training one, these unused parameters (that have not been designed through the training procedure) may have an influence on the extrapolation result. Having a way to monitor the part of the network that is not used during training time could be an elaborated way of detecting erroneous behaviors at inference times.

Code coverage [132] is a relevant measure of code robustness since it quantifies the amount of source code that was not used while executing a test suite. It provides a good indicator on the possible presence of undetected errors in man-made programs and has become, over the years, a metric for code quality that it is advised to monitor before a code is to be used in a production environment.

In order to find potential erroneous behavior in deep learning models and in analogy to the concept of code coverage, researches sought to exploit the lack of coverage of these statistical models to trigger a wrong prediction. Usually, coverage guided testing aims at creating inputs altering the activation pattern (i.e. the activated units or the distribution of these activations). Thus, coverage testing produces perturbed datasets that maximize the coverage of a given network. Such modified inputs might eventually change the decision provided by the neural net. These perturbations can subsequently be used as a form of data augmentation and serve the purpose of retraining the network to increase the overall usage of the network parameters. Contributions to coverage testing assume that the lack of coverage is responsible for blind spots in the decision-making process and they usually attempt to find ways to reduce such occurrences by increasing coverage at training time.

Although tempting, the analogy does not hold entirely since code coverage seeks defaults of implementations whereas DNN coverage testing seeks ways to twist a perfectly working code. In the light of some recent researches, over-parametrization appears to improve generalization [139, 7, 10] which in turns allows for a more robust model [198]. This result enters in sharp contrast with the coverage reasoning where having more parameters means more unused parameters. By having an extra amount of parameters a neural network should be more vulnerable and yet under some hypothesis (mostly norm constraints on the weights) it has generalization error that is lower than the one of small neural networks that should learn more sparse representations.

Initial works on coverage [145, 124, 196] showcased experiments where adversarial attacks (i.e. fooling samples) tend to increase DNN coverage but the other way around does not hold according to more recent works. Indeed, the recent literature has been more skeptical with results that failed to showcase dependencies between robustness and coverage [199, 90]. Their conclusion is the following: increasing coverage does not provide significant improvements in terms of adversarial robustness. In order to provide additional enlightenment concerning this perspective, we explore in this contribution if an adversarial attack could exploit coverage information in order to improve its capacity at fooling the network. Former approaches on coverage considered the two aspects separately (coverage and adversarial attacks) by investigating correlations between one another whereas we want to assess the effect of mixing the both. In the light of these elements, we formulate the following questions that drive this research: can coverage be used jointly with adversarial attacks (resp. defenses) to efficiently perturb (resp. increase robustness of) neural networks? Can a happy medium where coverage helps shape a better attack (resp. model) be stricken?

This chapter can be classified in the DNN testing literature reviewed in Section 3.2, more specifically the coverage based testing subsection. In order to clarify the contributions made in this chapter detailed hereinafter, we offer subsequently a brief reminder of similar approaches.

4.1.1 Contributions

We propose a novel subdifferentiable coverage metric that does not rely on hyperparameters such as thresholds or binning strategies that may influence the result significantly. This metric, in turns, enables to optimize concurrently coverage and robustness and to find out if they can be exploited altogether to fool a network. The end goal being to highlight a possible joined effect between coverage and adversarial training and/or a joined effect between coverage and adversarial attacks on the accuracy of the model. We will explore the problem following a double perspective: on the one hand, we adopt the point of view of an attacker that seeks to exploit the coverage of a model in order to create the most perturbing attack. On the other hand, we take on the perspective of the defender where we seek to use coverage at training times to make the model more robust. Following the attacker perspective in Section 4.3.1, we present testing of a pure coverage guided attack and explore numerically the effect of coverage used jointly with well known adversarial attacks. In the end, we did not manage to show any gain in using coverage in combination with adversarial attacks and that increasing coverage does not provide a reliable way of perturbing the accuracy of a model. Considering the defense problem, we provide a broad range of experiments in Section 4.3.2 that show the interest of introducing coverage guided metrics in the training and present its effect over a broad range of attackers. We managed to show experimentally that increasing coverage at training times provides some regularization that effectively increases robustness but only significantly compared to an unregularized model. Any attempts to make a training with combinations of adversarial samples made of combinations of coverage and adversarial attacks did not bring any gain compared to the pure adversarial training.

4.1.2 Related work

DeepXplore [145] has been the first paper drawing a parallel between coverage and adversarial attacks: following their logic, adversarial attacks should be exploiting parts of the DNN that were not used during training.

The layer-wise notation is quite unfit for defining neuron coverage, we will prefer the neuron-wise representation (introduced in 2.2): let us consider a neural network $f(\theta, \cdot)$ equipped with the ReLU nonlinearity, with parameters $\theta \in \Theta$ as a collection of neurons $N = \{n_1, n_2, ...\}$. For all $i \in [|N|]$, $n_i : \Theta \times \mathcal{X} \to \mathbb{R}$ is the function that returns the activation of neuron i before applying the nonlinearity for a given input $x \in \mathcal{X}$ of the neural net. Given a test set $X = \{x_1, ..., x_m\} \in \mathcal{X}^m, m \in \mathbb{N}$, Pei et al. [145] defines neuron coverage as the percentage of activated neurons on the test set over the number of neurons:

$$\text{NCov}(X, \theta) = \frac{|\{n_i | \forall x \in X, n_i(\theta, x) > 0\}|}{|N|}. \tag{4.1.1}$$

The authors then devise an algorithm that finds a negative activation $n_j(x), j \in [|N|]$ and they increase its value along with additional penalty terms using a gradient ascent algorithm on x until the activation turns positive. The algorithm then proceeds to increase another negative activation in an attempt to increase indirectly the overall NCov metric for a perturbed version of x.

However, this metric has been experimentally shown [124] to be inefficient in measuring diversity in activation distributions. Indeed, there is no sensitivity with respect to the number of times an activation has passed the 0 threshold over the test set and it does not account for how the activations are distributed over the reals. In order to palliate the latter, DeepGauge [124] introduced k-multi section coverage.

Recalling the notations from equation (4.1.1), for a neuron $n_i, i \in [|N|]$ k-multi section coverage consists of making k subdivisions $(S_j^{n_i})_{j \in [k]}$ of the closed activation interval $[l_i, u_i]$ of a neuron so that $\bigcup_{j \in [k]} S_j^{n_i} = [l_i, u_i]$. Ideally, the subdivisions are taken with equal length. Their coverage metric is a count of the subdivisions that actually contain an activation value when probed on the test set over the total number of subdivisions:

$$\text{KMNCov}(X, \theta, k) = \frac{1}{|N|} \sum_{i \in [|N|]} \frac{|\left\{S_j^{n_i} | \exists x \in X, n_i(\theta, x) \in S_j^{n_i}\right\}|}{k}. \tag{4.1.2}$$

This metric is not differentiable either and thus cannot be optimized directly. On top of that, the influence of the parameter k on which heavily depends the metric is not assessed in the original paper: the authors leave the procedure for finding a suitable k for future work.

There is a growing concern from the software engineering community [121, 62, 164] that current coverage based testing might be unfit for testing neural networks as it seems not to be a good test generation framework for robustness. While undertaking this work, we became aware of parallel and independent researches [199, 90] on the same subject. These two papers are skeptical about a possible correlation between coverage and robustness and present a significant amount of empirical evidence to support their claims. [90] is computing a KL divergence between layer-wise activation distribution and a uniform distribution. Although they display that the increase in layer coverage tend to increase neuron coverage as well, there could be pathological cases where layer-wise coverage is increased and neuron wise coverage is not. [199] provides a large pool of experiments in order to compute Pearson correlations between an extended range of robustness metrics and coverage metrics. They show that the well-used metrics of robustness and the metrics for coverage have little correlation, positive or negative. The literature has been focusing on comparing the correlations of coverage and robustness, however, there has been little consideration over their joined effect: two variables may be independent but can have a cross effect on a third variable. We therefore want to probe if adversarial attacks and coverage can be used jointly to perturb predictions. Our work is in the continuity of DeepGauge as we aim at quantifying the coverage on the whole support of activation distribution but in a rather continuous way. [106] and [74] compute an approximation of the distribution of activations by using KDE (called the activation trace), the former does it in order to increase its entropy and study its effect on the KMNCov criteria, the latter uses it to detect potential out-of-distribution samples. We use the very same activation trace with the purpose of making it uniform since we want to balance the activation pattern. The discrepancy between the activation trace and the uniform distribution constitute our coverage metric on which we elaborate further in the next section.

4.2 Theoretical considerations

4.2.1 Coverage metric

We borrow the notation of part 2.2 for neural networks. The coverage metric from [90] (a KL divergence layer-wise) presents the desirable property of being differentiable [159] or at least subdifferentiable [24] in case activations are non-smooth. Subdifferentiability is key to finding a suitable set of parameters as the gradient descent (or subgradient descent) remains, as of the time of writing of these results, the major procedure for optimizing neural networks. However, this KL divergence is computed layer-wise, thus not accounting completely for the coverage of one specific neuron. A neuron coverage metric should be computed neuron-wise and aggregated throughout all activations in the same fashion that [145, 124]. Subdifferentiability of our metric comes from the fact its definition as a divergence between two measures instead of an enumeration.

Keeping notation from equation (4.1.1), let $a_i \in \mathcal{P}(\mathbb{R})$ be the activation distribution of one neuron i with $i \in [|N|]$ over the input distribution. The activations a_i are taken before applying the nonlinearity, therefore a_i is the distribution of function $n_i(\theta, \cdot) : \mathcal{X} \to \mathbb{R}$ assessed on the input distribution. We dropped the dependency with respect to θ in the notation of the activations a_i for the sake of clarity. a_i is supported on a compact set $[l_i, u_i]$ with $(l_i \leq u_i)$ and $(l_i, u_i) \in \mathbb{R}^2$ since we considered the support of the input space to be bounded. Let $U(l_i, u_i)(\cdot) = \frac{1}{u_i - l_i}\mathbb{I}_{[l_i,u_i]}(\cdot) \in \mathcal{P}(\mathbb{R})$ be the uniform distribution over $[l_i, u_i]$ where $\mathbb{I}_A$ is the indicator function of a set A. Our coverage metric aims at making the activation distribution lean towards the uniform distribution on the activation interval, thus we define it as minus the KL divergence between $U(l_i, u_i)$ and a_i (the coverage is maximal when the distribution is completely uniform, thus the KL is null):

$$\mathrm{Cov}(a_i, \theta) = -\mathrm{KL}(U(l_i, u_i), a_i). \tag{4.2.1}$$

Overall, this metric is appealing for many reasons: it does not resort to any parameters (such as the threshold

in [145] or the number of sections in [124]), apart from the bounds l_i and u_i that can be determined when the network is initialized by processing the input distribution. It provides a better sense of coverage than previous metrics: NCov and KMNCov only check if there is one occurrence of an activation among their respective binning strategy, whereas optimizing our coverage discrepancy aims at flattening evenly the activation pattern. The major caveat of our metric comes from the fact that the a_i are not necessarily known since all practical use cases have finite datasets. Nevertheless, the distribution a_i can be approximated using m samples $(n_i(\theta, x_1), ..., n_i(\theta, x_m)) = (a_i^1, ..., a_i^m)$. We choose to perform a kernel density estimation presented in Section 2.6 for its simplicity and its plain dependency to activation values $(a_i^1, ..., a_i^m)$. We select the Gaussian kernel for approximating the density, we discuss the choice and relevance of the parameters at the end of this section. Let $h > 0$:

$$a_i(t) \approx \frac{1}{mh} \sum_{j=1}^{m} K\left(\frac{t - a_i^j}{h}\right),$$
$$\text{where} \quad K(x) = \frac{\exp(\frac{-x^2}{2})}{\sqrt{2\pi}}.$$

This expression is the same as [74, 106] where they use it as a way to quantify surprise in an input. However, here we use it for the purpose of estimating our coverage metric. We approximate equation (4.2.1) using this estimation of the a_i:

$$\text{Cov}(a_i, \theta) \approx \widehat{\text{Cov}}(a_i, \theta) \overset{\text{def}}{=} -\text{KL}\left(U(l_i, u_i), \frac{1}{mh} \sum_{j=1}^{m} K\left(\frac{\cdot - a_i^j}{h}\right)\right). \tag{4.2.2}$$

We define the coverage of a network as the mean of the coverage of all the neurons in the network:

$$\text{Cov}(f, \theta) = \frac{1}{|N|} \sum_{i \in [|N|]} \text{Cov}(a_i, \theta). \tag{4.2.3}$$

The dependency in θ is key since we will further aim at designing the parameters of the model with respect to this coverage metric. It obviously depends on the unknown input distribution $\rho \in \mathcal{P}(\mathcal{X})$ and can be approximated on a batch of data $X = (x_1, ..., x_m)$, here $a_i^j = n_i(x_j)$, i.e. the local sampled activation i for input sample j:

$$\text{Cov}(f, \theta) \approx \widehat{\text{Cov}}(f, \theta)(X) = \frac{1}{|N|} \sum_{i \in [|N|]} -\text{KL}\left(U(l_i, u_i), \frac{1}{mh} \sum_{j=1}^{m} K\left(\frac{\cdot - a_i^j)}{h}\right)\right). \tag{4.2.4}$$

Selection of various hyperparameters and general remarks:
Some details on the different errors of approximation and the selection of some of the constants need to be discussed:

- **Selection of the kernel function:** for the optimization process, that resorts to subgradients, the kernel needs to be $\mathcal{C}^1$ almost everywhere thus discarding the use of window kernels. In order to approximate a distribution on a closed using a kernel with finite support would have been more advised. However, the KL divergence requires the two measures to share the support of the first argument in order to be computed correctly: using a finite support kernel on an empirical distribution does not ensure that the KL divergence is well defined for any samples. For computational simplicity, the Gaussian kernel was much more suitable: it ensures the KL divergence to have a finite real value for any sampled activations.
- **The selection of l_i and u_i:** for each activation an l_i and u_i are needed for the computation of the KL divergence. One can select them arbitrarily but this would mean more control over the initialization of the parameters. Throughout our experiments we compute them after the initialization of the weights by processing the training set, l_i and u_i are then kept the same throughout the whole training procedure.

- **Selection of the bandwidth h:** h defines how smooth the approximated distribution should be. An h too large would make an estimated distribution that has consequent mass outside the $[l_i, u_i]$ interval that would introduce bias. On the other hand, a small bandwidth would not ensure a proper estimation of the activation distribution as it would be plagued by sampling errors from the input distribution. We followed [170] and selected $h = \frac{4\hat{\sigma}}{3k^{\frac{1}{5}}}$ where $\hat{\sigma}$ is the standard deviation of the data points from which we want to estimate the distribution. This choice is to ensure some regularity on the estimated distribution. Another possible choice would have been to make an optimization with respect to h in order to fit the target uniform distribution at each density estimation. This would have been a more expensive fitting step but more accurate. For an in-depth review on methods for bandwidth selection see [102].
- **The sampling error:** in order to compute the activation distribution, the input sampling has to be representative enough of the input data, it is the case if the whole training set is used in order to compute the coverage metric. However, as deep learning usually resorts to stochastic subgradient descents, equation (4.2.4) is bound to be further approximated using mini-batches. Each mini-batch therefore needs to be representative enough to have small sampling error on the coverage metric.
- **The approximation error of the KDE method:** although we discussed the choice of the kernel and the bandwidth in previous remarks, it remains that any amount of Gaussian kernel cannot approximate exactly a bounded uniform distribution. Therefore, there will always be an error of approximation in the activation distribution as long as the KL is used jointly with a Gaussian kernel. Possible alternatives for a lower error could be using a distributional distance well defined in the case of disjointed supports such as the Maximum Mean Discrepancy (MMD) or the Wasserstein distance.
- **Approximation of the KL divergence:** in practice, we evaluate equation (4.2.2) using a rectangle rule for numerical integration. Of course, $\mathrm{Cov}(a_i, \theta)$ depends on the parameters θ of the network since the activation samples do and therefore so does the approximated activation distribution.

The expression of the coverage depends on both the parameters θ and the data batch X meaning that we can either seek to increase the coverage by perturbing slightly the input values or we can seek the adequate parameters that would ensure a well covered network. This give us two possible ways of exploiting this coverage metric: from the perspective of an attacker the coverage can be maximized by changing slightly the input in order to find erroneous behavior. From the perspective of a defender, it is possible to find the parameters that increase coverage during training times to mitigate the exploitation of blind spots by an attack. In order to put these hypotheses to the test in section 4.3 we have to define the attacks and trainings that we will be comparing.

4.2.2 Coverage as an attack on DNNs

In this section, we present different ways to perturb the prediction of a neural network by altering the input values within a given perturbation set. We first present an approach that resorts to coverage only and then we proceed to combine this coverage optimization with more traditional methods from adversarial attacks in an attempt to make the most of both world. Throughout this section we will be noting $p \in \mathbb{N}$ the input dimension and $q \in \mathbb{N}$ the output dimension.

Coverage attack. This "attack" is simply a search of a bounded local perturbation $\delta = \{\delta_1, ..., \delta_m\} \in (\mathbb{R}^p)^m$ of the input values of the test set $X = \{x_1, ..., x_m\} \in (\mathbb{R}^p)^m$ in order to increase the coverage of the model. It corresponds to tackling the following problem:

$$\max_{\|\delta_i\|_\infty \leq \epsilon, i \in [m]} \widehat{\mathrm{Cov}}(f, \theta)(\{x_i + \delta_i, i \in [m]\}). \tag{4.2.5}$$

In practice, the $(\delta_i)_{i\in[m]}$ are found by performing a PGD-like algorithm (see algorithm 1) on the coverage metric from equation (4.2.5). Due to our definition of our coverage metric, we can only compute it on a batch of data. Hence the necessity of having m inputs and m perturbations.

FGSM and PGD attack with added coverage. We introduced the FGSM and PGD method in Chapter 2 Section 2.4 as heuristics for approximating the argument of equation (2.4.1). The batched version in equation (2.4.5) is key to adding a coverage penalty: in order to provide some sense of coverage to the adversarial attack, we concurrently increase the loss with the coverage objective of equation (4.2.5). We resort to a constant ν to balance the contribution of each of the objectives:

$$\max_{\|\delta_i\|_\infty \leq \epsilon, \forall i \in [m]} \left[\frac{1}{m} \sum_{i=1}^{m} \ell(f(\theta, x_i + \delta_i), y_i) + \nu . \widehat{\text{Cov}}(f, \theta)(\{x_i + \delta_i, i \in [m]\}) \right]. \tag{4.2.6}$$

ν allows exploring between the pure adversarial attack for $\nu = 0$ and the pure coverage attack when $\nu \to \infty$. In practice, we seek the $(\delta_i)_{i \in [m]}$ with a PGD-like algorithm as well by increasing this composed objective.

Carlini and Wagner attack with added coverage. The C&W attack can be as well processed in a batched manner thus allowing us to define a coverage version of this attack in the same fashion that we did in equation (4.2.6):

$$\min_{\|\delta_i\|_\infty \leq \epsilon, y_i' \neq y_i, y_i' \in \mathcal{Y}, \forall i \in [m]} \left[\frac{1}{m} \sum_{i=1}^{m} \ell(f(\theta, x_i + \delta_i), y_i') - \nu . \widehat{\text{Cov}}(f, \theta)(\{x_i + \delta_i, i \in [m]\} \right]. \tag{4.2.7}$$

The sign before the coverage has changed since the C&W attack is solution to a minimization problem, not a maximization like FGSM or PGD.

Random attack. This "attack" serves the purpose of giving a baseline: it is adding a random noise to the input, thus perturbing the decision-making. Any perturbing $\delta \in \mathbb{R}^m$ is thus sampled uniformly on $\mathbb{B}_\varepsilon^\infty(0)$.

4.2.3 Coverage as a defense for DNNs

We present two main strategies for improving a training procedure using coverage: the first one is similar to a regularized training where we add coverage in the form of a penalty function that helps to reduce the variance by introducing some bias in a vanilla training. The second approach is basically a data augmentation method where the noise introduced to perturb the dataset comes from the attacks that we defined in the previous section

Vanilla training with added coverage. A first logical approach when considering a coverage metric, is basically to increase coverage at training times in order to ensure that activations are well covered. More precisely, we act on the parameters $\theta \in \Theta$ of the neural network in order to concurrently minimize an empirical loss and maximize the coverage. In the same manner than we did for adversarial attacks, we will here balance the objectives by resorting to a constant ν:

$$\min_{\theta \in \Theta} \sum_{i=1}^{m} \ell(f(\theta, x_i), y_i) - \nu . \widehat{\text{Cov}}(f, \theta)(\{x_1, ..., x_m\}). \tag{4.2.8}$$

This problem can be approximated using the classical tools from deep learning (i.e. diverse subgradient based optimizers).

Robust trainings against attacks. This approach resorts to the attacks elaborated in section 4.2.2 and uses them to perform adversarial training. For the sake of clarity we shall remind the readers of the method described in the preliminaries (Section 2.4.2): An attack (Attack $\in \{\text{Coverage}, \text{PGD}, \text{FGSM}, ...\}$) is a procedure that resorts to a loss function L (defined at the same time than the attack in section 4.2.2 or 2.4), parameters θ of the neural network, an input/output tuple (x, y) and a bounded perturbation $C = \mathbb{B}_\varepsilon^\infty(x)$ set. It returns a perturbed

input $\tilde{x} = x + \delta$, the adversarial training writes:

$$\begin{aligned} &\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\hat{\rho}_k} \left[\ell(f(\theta, x+\delta), y)\right] \\ &\text{where} \quad x + \delta = \text{Attack}(L, \theta, \mathbb{B}_\varepsilon^\infty(x), x, y). \end{aligned} \tag{4.2.9}$$

In other words, it exploits adversarial attacks to create fooling samples that are later used as a part of the training set in order to design the parameters θ robust to perturbations. Ultimately, this approach allows building as many defenses as there are of attacks, namely: defense against the coverage attack, adversarial defense against FGSM, PGD and C&W and their versions with added coverage or even defense against random attacks. Since we are using subgradient based methods for computing the attacks, our applications do not come with guarantees of optimality.

The following section will elaborate empirically on the advantages/disadvantages of using coverage conjointly with adversarial attacks or adversarial training.

4.3 Experiments

Throughout these experiments we will perform tests on classification settings on public datasets such as MNIST [115], FMNIST [194], CIFAR10 [108] and SVHN [138]. We used the Cross Entropy loss between the prediction from the neural network and the real labels: $\ell(y_{\text{pred}}, y_{\text{real}}) \overset{\text{def}}{=} \sum_{i=1}^{q} y_{\text{real},i} \log y_{\text{pred},i}$. The classification setting has been chosen since it is a classical framework for comparing empirically the quality of different attacks and on top of that it is the only framework on which one can apply the C&W attack. We will first consider the point of view of an attacker trying to leverage coverage to fool the model in section 4.3.1. It has been showcased [124, 196] that adversarial attacks tend to increase coverage, we intend to assess if the opposite holds true as well: whether increasing coverage makes good adversarial perturbations. Secondly we will take on the role of a defender that aims at building a reliable decision system with a limited number of blind spots in the activation pattern: our metric allows to perform trainings that maximize coverage conjointly with minimizing the objective. We will assess the benefits of using such a mixed approach in Section 4.3.2.

4.3.1 Assessing coverage attacks

4.3.1.1 Pure coverage against usual attacks

The first experiment aims at highlighting if the coverage metric is sufficient for generating adversarial samples, i.e. if solving problem (4.2.5) provides perturbations δ altering significantly the prediction of the model. We compare the performance of a vanilla trained model on this "Coverage attack" against performance on a pool of common adversarial attacks: FGSM, PGD (equation 2.4.1), C&W (equation 2.4.4). We add two performance metrics that will serve as our baselines: the accuracy (i.e. percentage of correctly classified samples) on an unaltered test set and the accuracy on a test set perturbed with bounded white noise. Currently, our interest lies in the capacity of an attack to perturb the model thus the lower is the performance of the model, the better is the attack.

We tested all the previously named attacks on a Resnet18 network (non-adversarially) trained with the ADAM optimizer [107] on different classification tasks: the MNIST [115], FMNIST [194], CIFAR10 [108] and SVHN [138] datasets. On all databases the network was trained for 20 epochs and a batch size of 50. The coverage attack uses the activations of the penultimate linear layer: this reduces the total computational load. The comparison between the coverage attack and the other approaches is slightly biased as the computation of the coverage can only be performed over a batch of data, whereas the computation of an adversarial attack can be performed input-wise.

On Table 4.1 the coverage on its own does not appear to perturb significantly the prediction since it is systematically the attack with the lowest success rate (i.e. the percentage of successful perturbations) for all the datasets

Attack	None	Random	Coverage	FGSM	PGD	C&W
MNIST, $\epsilon = 0.1$	99.5%	99.2%	99.3%	80%	63%	65%
FMNIST, $\epsilon = 0.1$	92%	86%	88%	12%	6%	6%
CIFAR10, $\epsilon = 0.03$	82%	81%	81%	14%	11%	11%
SVHN, $\epsilon = 0.03$	94%	92%	89%	18%	8%	7%

Table 4.1: Accuracy of a vanilla trained ResNet model on MNIST, FashionMNIST, CIFAR10 and SVHN dataset against different attacks for different perturbation radius ε, from an attacker perspective the lower is the better.

considered. This result is not new [199] and simply shows that the task of perturbing models is performed better by algorithms whose aim is explicitly to fool the prediction. The performance on a perturbing set maximizing the coverage is quantitatively similar to the one of naive random noise: the increase in coverage does not appear to be perturbing the prediction of the model significantly. This does not prove that maximizing the coverage is equivalent to perturbing randomly the model, but it appears that increasing the coverage does not serve the exact purpose of changing the prediction.

4.3.1.2 Combining coverage and adversarial attacks

We have seen in section 4.3.1.1 that coverage on its own does not appear to be able to perturb the model efficiently compared to adversarial attacks. The next step is to assess if some combination of the two could provide a significant advantage. To that extent, we use the attacks defined in equations (4.2.6) and (4.2.7) that leverage the performance of adversarial attacks with some weighted contribution of coverage. By making ν vary we will explore if there exists a possible cross effect between adversarial attacks and coverage. Let us note that the magnitude of ν is not conditioning the step sizes of FGSM, PGD and C&W due to the fact that they are performing gradient steps using only the sign of the gradient (see algorithm 1). However, ν influences the prevalence of coverage over the objective: the higher the constant ν the greater the chance that the direction of the coverage will be selected for a given parameter update. The goal is to highlight if coverage can provide sensible perturbation directions that could be exploited by the adversarial attacks. We will be increasing ν progressively in order to see if any percentage of coverage insight can be beneficial or if it will simply degrade the performance of the attack. The protocol is the following: on a non adversarially trained Resnet18 on MNIST we will be measuring performance of the model under attack for varying values of ν. Since we are taking on the role of the attacker: the lower is the performance, the better is the attack.

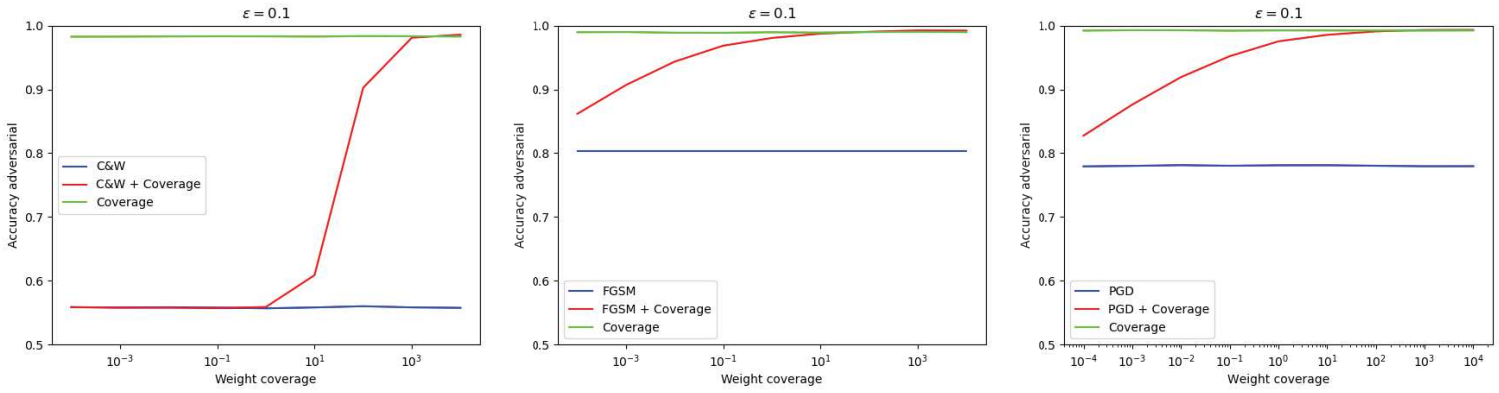

Figure 4.1: Adversarial accuracy of the model when making ν vary (the lower the curve, the more efficient the attack is) for C&W (left), FGSM (middle), PGD (right) attack augmented with coverage on MNIST dataset. Coverage attack and pure adversarial attack plotted for reference.

We display in Figure 4.1 the evolution of the adverse accuracy with respect to the coverage weight ν. To make asymptotic behaviors more visible we displayed as well the performance of the attacks without coverage and the coverage attack: they are constant in all the charts since they don't actually depend on the coverage weight ν.

Asymptotic behaviors are respected for the C&W attack (Figure 4.1, left) as the limits do converge towards the uncombined attacks when ν goes to 0 and converges towards the performance of the coverage attack when ν goes to infinity. Considering the FGSM and PGD attack (Figure 4.1, middle and right) the blue and red curves do not meet when ν goes to zero, after some extended testing we devised that this is caused by the fact that the gradient of the training loss on the training points are null due to the gradient of the loss function having a small norm. This is caused by the gradient descent method used for optimizing that enforces first order optimality conditions, whereas the coverage measure was not maximized during training. Therefore, the coverage always has gradients that surpass the one of the loss of the FGSM and PGD attack despite the very low multiplicative constant ν. The fact that it does not happen for the Carlini and Wagner attack comes from the fact that the gradients exploited by this attack concern another loss function than the one used for the training on which we did not enforce first order optimality conditions. As the C&W targets the second most probable outcome and increases its likelihood, it allows to have non-null first order information that is greater than the one from coverage when using a small ν. The main result of these graphs is that for no value of ν on the curve do we see an advantage in using a combination of adversarial attacks and coverage over using the adversarial attack on its own: it appears as if any scale of coverage gradient does not provide relevant information for perturbing the prediction of a model. On top of that, it is consistent over all three attacks: showing no improvement over the baseline (blue curve). These tests were performed on FMNIST and CIFAR10 and SVHN with similar results and we observed the same transitions between the pure attack and pure coverage and no positive effect of mixing the two approaches. These results advocate for the fact that coverage (as we defined it) does not relate conjointly with adversarial attacks to adversarial robustness.

4.3.2 Assessing coverage defenses

Maximizing coverage alongside an adversarial attack does not give any significant adversarial robustness as it has been showcased in section 4.3.1. Yet, it remains to devise whether coverage can bring a certain level of robustness when it is used as a mean for enhancing the training procedure. We would like to assess if there is a real advantage in using noise that increases coverage over naive white noise and/or adversarial noise to perform data augmentation.

4.3.2.1 The effect of vanilla training regularized with coverage

First, we test if increasing coverage during a vanilla training (equation (4.2.8)) increases robustness. We train a Resnet18 on MNIST, FMNIST, CIFAR and SVHN for 20 epochs and a batch size of 50. For the trainings, we selected two possible values of ν: $\nu = 0$ (thus making the impact of the coverage null) and $\nu = 100$. The value $\nu = 100$ has been selected by cross validation: it was the value that resulted in the best trade-off between accuracy and robustness.

If we take a look to the results in Table 4.2 we see that, indeed, when coverage is used for the training we obtain a more robust network against adversarial attacks with a consistent gain in adversarial accuracy (16% gain for MNIST, 10% for FMNIST, 3% on CIFAR10 and 12% on SVHN). We do appear to degrade the test accuracy when doing so, however, the benefits on robustness out-weight the loss on accuracy. The coverage criterion appears to act just like a regularization: in the trade-off between bias and variance we increase the bias by adding an extra term in the loss function that aims at reducing the variance of the prediction. The gains are significant , they are much smaller than usual gains when performing adversarial training.

Metric	Test Accuracy	PGD adversarial accuracy
MNIST, $\nu = 0$, $\epsilon = 0.1$	99.8%	57.5%
MNIST, $\nu = 100$, $\epsilon = 0.1$	99.5%	73.7%
FMNIST, $\nu = 0$, $\epsilon = 0.1$	92.6%	6%
FMNIST, $\nu = 100$, $\epsilon = 0.1$	91.5%	16%
CIFAR10, $\nu = 0$, $\epsilon = 0.03$	82%	11%
CIFAR10, $\nu = 100$, $\epsilon = 0.03$	74%	14%
SVHN, $\nu = 0$, $\epsilon = 0.03$	93%	22%
SVHN, $\nu = 100$, $\epsilon = 0.03$	91%	34%

Table 4.2: Test accuracy and adversarial accuracy of coverage trained ($\nu = 100$) and non coverage trained ($\nu = 0$) networks on MNIST, FMNIST, CIFAR10 and SVHN, ϵ is the perturbation amplitude used for computing the PGD adversarial accuracy.

4.3.2.2 Training on coverage attacks

In an attempt to make the most of both world, we devised a second range of experiments where we use the attacks built in section 4.3.1 in order to perform adversarial training and then we probe the adversarial accuracy using reference attacks and coverage attacks. The list of the probed approaches for training and attacking is as follows:

- **Unperturbed Coverage:** corresponds to performing the training as described in paragraph 4.2.3, this is only a defense thus it cannot be found in the range of attacks.
- **No Perturbation:** corresponds to an unperturbed test set as an attack and to vanilla training as a defense.
- **Random:** corresponds to perturbing randomly with a uniform noise as an attack and to training against white noise as a defense.
- **FGSM:** corresponds to a single step approximation of the argument of problem (2.4.1) as an attack and an adversarial training on this attack as a defense.
- **FGSM + Coverage:** corresponds to a single step approximation of the argument of problem (4.2.6) as an attack and an adversarial training on this attack as a defense.
- **PGD:** corresponds to a multi-step approximation of the argument of problem (2.4.1) as an attack and an adversarial training on this attack as a defense.
- **PGD + Coverage:** corresponds to a multi-step approximation of the argument of problem (4.2.6) as an attack and an adversarial training on this attack as a defense.
- **C&W:** corresponds to a multi-step approximation of the argument of problem (2.4.4) as an attack and an adversarial training on this attack as a defense.
- **C&W + Coverage:** corresponds to a multi-step approximation of the argument of problem (4.2.7) as an attack and an adversarial training on this attack as a defense.

We display in Figures 4.2, 4.3, 4.4 and 4.5 the evolution of the adversarial robustness (i.e. accuracy against the corresponding attack) when making ν vary between 0.0001 and 10000 over our wide range of defenses and attacks. Considering, this time, the perspective of the defender, the higher the accuracy the better as it attests for a better resilience. The values of ν chosen were selected to assess the effect among different regimes for the composed attacks: a regime at low coverage and high prevalence of the adversarial training ($\nu = 0.001$), a regime with prevalence of coverage over the adversarial training ($\nu = 1000$), and two values that aim to bridge the gap between the two ($\nu = 1.0, \nu = 100$). Intermediate values between these regimes were assessed as well, however they did not bring any additional enlightenment for the discussion hereinafter.

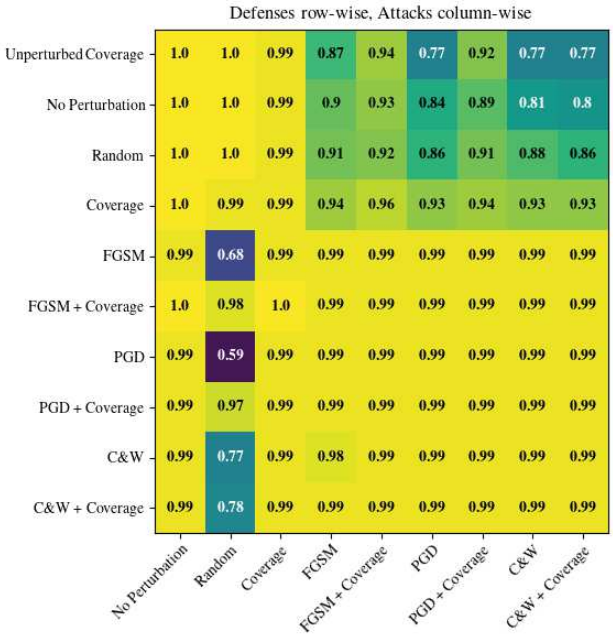

Figure 4.2: Adversarial performance of multiple adversarial trainings on MNIST, trainings horizontally and attacks vertically, $\epsilon = 0.1$, $\nu = 0.001$.

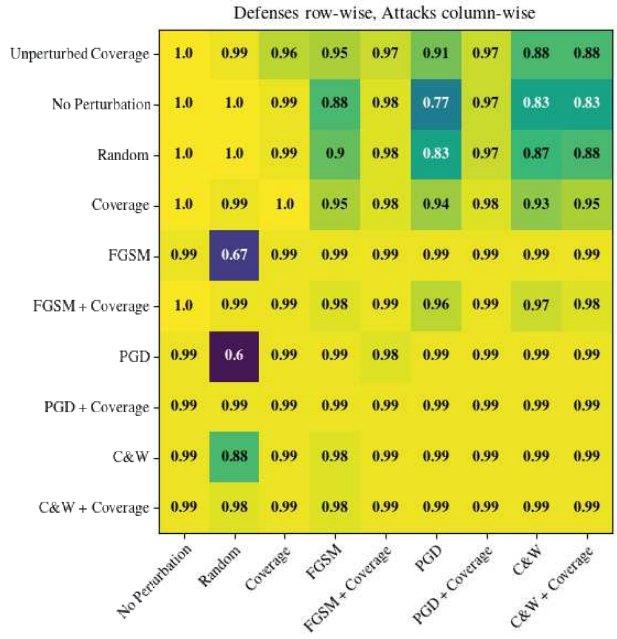

Figure 4.3: Adversarial performance of multiple adversarial trainings on MNIST, trainings horizontally and attacks vertically, $\epsilon = 0.1$, $\nu = 1$.

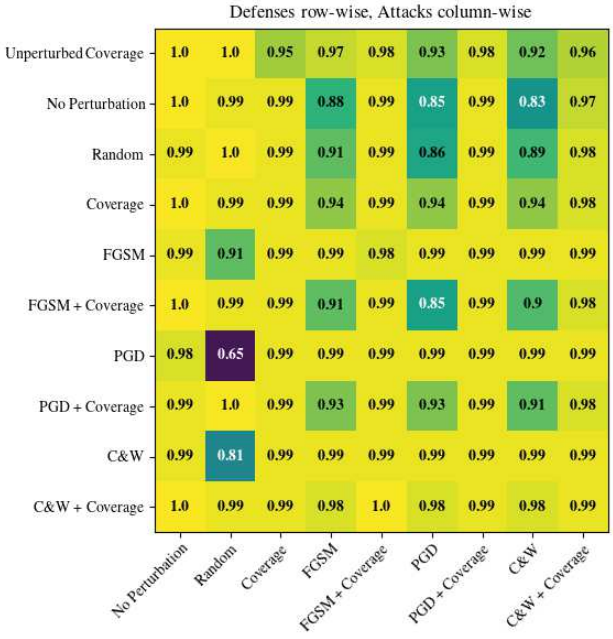

Figure 4.4: Adversarial performance of multiple adversarial trainings on MNIST, trainings horizontally and attacks vertically, $\epsilon = 0.1$, $\nu = 100$.

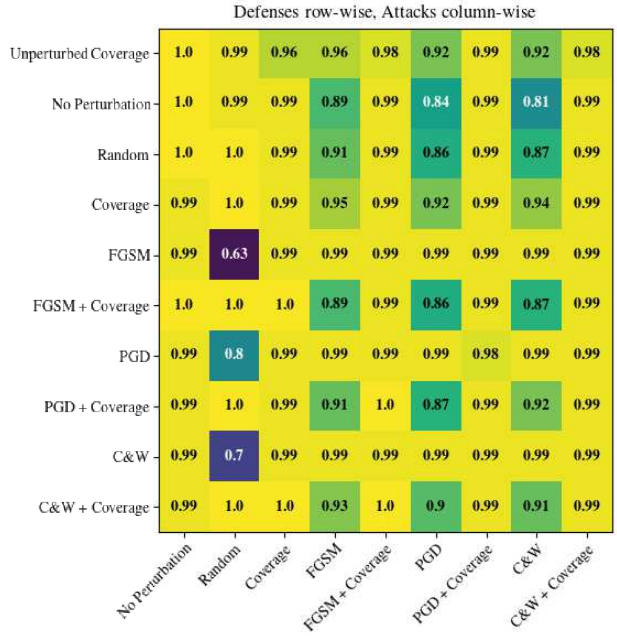

Figure 4.5: Adversarial performance of multiple adversarial trainings on MNIST, trainings horizontally and attacks vertically, $\epsilon = 0.1$, $\nu = 10000$.

On all figures we observe a similar performance between the coverage used for adversarial training and the training procedure maximizing coverage (respectively line 4 and line 1) except for the coverage attack (column 3) hinting at the fact that both methods achieve the same goal: increasing neuron coverage. However, the gain in robustness is lower than the one using adversarial trainings. Both coverage methods (line 1 and 4) slightly outperform the vanilla training (line 2) on all adversarial robustness, the results of line 1 concerning robustness were already discussed in the previous experiment: coverage training proved to be just as effective

as a regularization and does provide better results than an unconstrained training (line 2). One can also see that random noise manages to perturb pure adversarial trainings (FGSM, PGD and C&W) consistently whereas it hardly perturbs the coverage defenses. Adversarial trainings are still the best when considering adversarial robustness, since they all perform the best against all range of adversarial attacks.

Concerning low coverage contribution to adversarial attacks (Figures 4.2 and 4.3) we see no change when combining adversarial attacks and coverage compared to the base attack. Except on the random noise attack (column 2) that appears to have a greater impact on adversarial trainings (FGSM, PGD and C&W) than compared to the versions with coverage.

Concerning high coverage contribution (Figures 4.4 and 4.5) we see that all combinations between an attack and coverage get similar result that the coverage trainings with no added robustness.

Overall, the main conclusions about this experiment are:

- Our different approaches of coverage perform similarly: there appears to have little difference in making a coverage guided training or coverage guided adversarial training. They only provide a better training than the vanilla training since it is more robust to adversarial noise.
- Coverage approaches are performing better than adding white noise to input data: it provides a sensible sense of robustness compared to white noise.
- Coverage approaches are outperformed by adversarial trainings when it comes to adversarial robustness: adversarial trainings are still working the best against adversarial attacks.
- There is no happy medium between coverage and adversarial approach: mixing coverage and adversarial training never outperforms, for any value of ν, the performance of pure adversarial training except for the white noise perturbation where the added coverage seems to bring some robustness.

These experiments tend to present coverage trainings or coverage adversarial training as approaches that provide more robustness than a vanilla training however when it comes to adversarial robustness they appear to be performing less than pure adversarial trainings. Our experiments showcase coverage approaches as a regularization that provides more suitable sets of parameters than vanilla training only. There appears to have little benefit of using coverage in conjunction with adversarial training.

4.4 Conclusions

Making an analogy between code coverage and neural network coverage for justifying the lack of robustness appears to be flawed. Even if it has been showed experimentally that adversarial attacks tend to increase coverage by exploiting some blind-spots in the model, however seeking to exploit all blind-spots by using coverage does not provide efficient information for the purpose of fooling the model. Indeed, the assumed connections between robustness and coverage do not hold following our double perspective: from the perspective of an attacker using coverage is not efficient as it does not give the directions for a proper perturbation of the prediction. There appears to be no cross effect between coverage and adversarial perturbation since combining both does not result in significant increase in performance of the attack over a wide range of datasets. This hints at the fact that coverage, even in a white box setting where the attacker has access to the model, is not an efficient attack method and a non completely covered neural network is not necessarily lacking robustness. From the perspective of a defender, training with coverage acts similarly to a regularizer and is preferable to a vanilla training when considering robustness metrics. However, it does not outperform adversarial training over targeted attacks: adversarial training is still preferable on a white box settings where the attacker as access to the learned function and can tailor the attack accordingly. Our work does not contradict DeepGauge or DeepXplore that claimed that adversarial attacks are increasing coverage, we add that the opposite does not appear to hold: increasing coverage is not an effective way of perturbing a model. Robustness in neural networks is a complex problem and it appears that there is little way to circumvent the challenges ahead in robust learning using analogies with

traditional code diagnosis tools. Theoretical evidences of the presence or absence of a link between robustness and coverage are still missing but all recent experimental developments lean towards the absence of correlated behaviors.

Chapter 5

An explicit model of robustness via Lipschitz controlled networks

Main contributions of this chapter

- Providing a new general approach for designing neural networks with prescribed Lipschitz constant.
- Testing the performance and convergence of Lipschitz assessing algorithms on prescribed networks.
- Testing the validity domains of Lipschitz assessing algorithms.

The content of this chapter was presented in a workshop of ECML PKDD 2022.

Contents

5.1 Introduction

The Lipschitz constant of a neural network is a key to quantify robustness since it directly correlates the maximum discrepancy that can be caused by modification of the input data on the prediction (see section 2.5 for the

definition). The Lipschitz constant plays an important role in generalization bounds that quantify the expected discrepancy between training error and test error [8, 67, 66]: classically, the generalization gap has a linear dependency to the Lipschitz constant. However, bridging the gap between training and testing error does not attest a better generalization power since shrinking the Lipschitz constant increases robustness but inevitably takes a toll on the capacity of the network to fit complex distributions (since it shrinks the size of the class of models considered). Ideally, one should seek the best compromise between robustness and generalization, however, such a trade-off is application-dependent.

We will distinguish two branches of the literature concerning the Lipschitz constant for neural networks. In the first one, the neural network is conditioned so as to be enforced various (more or less tight) bounds on its Lipschitz constant [134, 203, 83, 42, 166]; see Section 3.1.1 for a detailed exposition. In the second type of works, authors focus on designing algorithms to compute approximations of upper/lower bounds of the Lipschitz constant on given networks (see the overview in Section 5.1.2). The focus in this chapter will be within this second group.

5.1.1 Contributions

In Section 5.2 we present a constructive approach for designing a network with a prescribed Lipschitz constant. It has the advantage of giving an exact Lipschitz constant for the network but covers a small class of models. Section 5.3 provides a min-max optimization formulation of the problem of prescribing a Lipschitz constant together with a simple algorithm to solve it, though without any convergence guarantee. This is to the best of our knowledge the first attempt at designing a prescribed Lipschitz network without the use of Lipschitz layers. We show numerically the effectiveness of our approach by prescribing neural networks with different values of Lipschitz constant. The soundness of the approach is supported by the fact that the resulting Lipschitz value is close to the objective with small error, despite the lack of convergence guarantees of our algorithm. This method is general but it lacks the guarantees of the former. The exact method is then used to appreciate the approximation properties of Lipschitz assessing algorithms: we are then able to highlight effectively their respective properties and their dependencies to either input dimension and/or parameter dimension.

5.1.2 Relation to prior work

Estimating the Lipschitz constant of a neural network is an NP-hard problem [183]. Researches started from loose estimations of the Lipschitz constant first with products of matrices norms [99]. This estimation being increasingly loose with the depth of neural nets, [183] provided tighter ones: the authors upper bound the Lipschitz constant by casting its estimation into a maximization problem on the gradient norm. However, the problem is NP-hard and their optimization algorithm is not guaranteed to provide the exact solution. [151, 71] provide a guaranteed upper bound via semi definite relaxation that are computationally demanding. [114, 37] leverage polynomial optimization that allows to compute an uncertified upper bound of the Lipschitz constant, their performance is compared to a lower bound and a selected few similar approaches. Despite these methodological differences, a common ground of all these approaches are the tests they are performing: given random or trained networks the methods are probed one against another and the one providing the lowest constant is assumed to be the one performing the best (only if the estimation is a guaranteed upper bound). Although interesting for comparing methods, this simply does not answer a basic question: how close the estimation is to the real Lipschitz constant? Usually, the authors compute a lower bound that serves the purpose of giving a reference objective: the upper bound would ideally reach the lower bound if both bounds were tight. Classic computation of a certified lower bound resort to brute force Monte Carlo sampling or gradient based optimization [27] of a differential quotient, but the resulting estimation is plagued by approximation errors. Therefore, the discrepancy between the lower bound and the upper bound can never be bridged and cannot be traced back to either the upper or lower bound estimation. Our contribution aims at isolating the sources of errors to have a better view regarding the

performance of the different algorithms computing the Lipschitz constant. The main purpose of this chapter is to provide procedures for designing diverse test cases (i.e., diverse neural networks) for which the Lipschitz constant is known. This allows to quantitatively compare and assess algorithms for estimating Lipschitz constants on test cases where the result is expected. The main difficulty is that, since computing the Lipschitz constant of a neural net is an NP-hard problem, so is building a general network with a given Lipschitz constant.

5.2 Designing specific networks with known Lipschitz constant

Let us take the notation of equation (2.5.2): we can simply upper bound the Lipschitz constant as done in [99] by the product of the spectral norm in case the activation function ϕ is 1 Lipschitz:

$$L_{f,\mathcal{X}}(\theta) \leq \prod_{i=1}^{d} |||W_i|||. \tag{5.2.1}$$

Our goal is to tailor $(W_1, ..., W_d)$ so that this inequality become an equality.

5.2.1 Construction

Let us constrain our weights with the following conditions:

- For all $i \in [d]$, $W_i = O_i S_i O_{i-1}^\top \in \mathbb{R}^{m \times m}$, $m \in \mathbb{N}$, where S_i are diagonal matrices of singular values and O_i are square orthonormal matrices also of dimension m.
- Without loss of generality we can assume that the greatest value $\lambda_i^{\max} = \max_j((S_i)_{j,j})$ is along the first dimension.

We have to make slight changes to the matrices $(O_k)_{k\in[n]}$ in order to have each of their main direction aligned along a positive dimension.

Let us write $O_i = (V_i^1, V_i^2, ...V_i^m)$ with column vectors, we have by definition that $W_i V_{i-1}^1 = \lambda_i^{\max} V_i^1$.

We would like to have that for all $i \in \{1, ..., d\}$, V_i^1 has positive or zero components (note that O_0 is not considered here) so that the ReLU activations do not change V_i^1 when it is applied element-wise. To achieve so we simply define for all $i \in \{1, ..., d\}$:

$$O_i^* \overset{\text{def}}{=} \text{diag}\left(\overline{\text{sign}}\left(V_i^1\right)\right) O_i = \text{diag}\left(\overline{\text{sign}}\left(O_i e_1\right)\right) O_i, \tag{5.2.2}$$

where e_i is the i vector of the standard base of $\mathbb{R}^m$, $\overline{\text{sign}}$ is the element-wise function defined by:

$$\overline{\text{sign}}(x) = \begin{cases} -1 & \text{if} \quad x < 0 \\ 1 & \text{otherwise.} \end{cases}$$

For all $i \in \{1, ..., d\}$, O_i^* is orthonormal since the transformation depicted in equation (5.2.2) is only a symmetry on carefully chosen axes. For the special case of O_0 we select $O_0^* = O_0$. Let us consider then the matrices $W_i^* = O_i^* S_i {O_{i-1}^*}^\top$ and $f^*(x)$ neural network composed with the W_i^* weight matrices. We will now check that $L_{f^*} = \prod_{i=1}^d |||W_i^*|||_2$ with $\sigma = \text{ReLU}$.

5.2.2 Verification

Proposition 5.2.1. *Let $(W_d^* ... W_1^*)$ be the matrices as defined in the section 5.2.1, let f^* be the network with weights $(W_d^* ... W_1^*)$ with no biases and ReLU activation function, then f^* has a Lipschitz constant of $\prod_{i=1}^d \lambda_i^{\max}$.*

Proof. We will now check that we know the value of the Lipschitz constant of the designed network and its main direction:

$$\left|\left|\left| \prod_{i=1}^{d} W_i^* \right|\right|\right| = \left|\left|\left| O_n^* \prod_{i=1}^{d} S_i O_0^\top \right|\right|\right| = \left|\left|\left| \prod_{i=1}^{d} S_i \right|\right|\right| = \prod_{i=1}^{d} \lambda_i^{\max} = \prod_{i=1}^{d} |||S_i||| \geq L_{f^*}.$$

On top of that for all $i \in \{1, ..., d\}$ we have by construction V_i^{1*} (i.e. the main direction of matrix i for the singular value $\lambda_i^{\max}$) whose components are positive or null and thus stable by the ReLU operation σ.

On the other hand, we have for $x^* = V_0^1$:

$$\begin{aligned} f^*(x^*) &= W_d^* \sigma \left(W_{d-1}^* \sigma \left(W_{d-2}^* ... \sigma \left(W_1^* x^* \right) \right) \right) \\ &= W_d^* \sigma \left(W_{d-1}^* \sigma \left(W_{d-2}^* ... \sigma \left(\lambda_1^{\max} V^{*1}_1 \right) \right) \right) \\ &= W_d^* \sigma \left(W_{d-1}^* \sigma \left(W_{d-2}^* ... \lambda_1^{\max} V^{*1}_1 \right) \right) \\ &= \prod_{i=1}^{d} \lambda_i^{\max} V^{*1}_d. \end{aligned}$$

Thus $L_{f^*} \geq \frac{\|f^*(x^*)\|_2}{\|x^*\|_2} = \prod_{i=1}^{d} \lambda_i^{\max}$.

Then $L_{f^*} = \prod_{i=1}^{d} \lambda_i^{\max}$, we have successfully constructed a neural network with a known Lipschitz constant. The subspace of maximum amplification is a half straight line of the form $x = t.V_0^1, t > 0$. □

Remark 5.2.2. We could have restricted the construction to only aligning the first singular vectors of the weight matrices and not the other ones: the verification would have remained the same. However, the construction and algorithm 2 (presented in section 5.2.3) would have been more complex for little benefit on the class of models used.

5.2.3 Algorithm for designing Lipschitz test cases

The algorithm presented here is composed of 2 main components:

- A sampling operator in the set of orthonormal matrices [131]. Named Orthonormal_Sampling in the algorithm.
- A procedure capable to change an orthonormal matrix in order to make a column indexed i positive (it is only the implementation of equation (5.2.2) for a parametrizable column i). Named Positivation_by_symmetry in the algorithm.

Algorithm 2: Building a 1-Lipschitz network with d layers

Input: Number of neurons per layer m
Input: number of layers d
$O_{old} =$ Orthonormal_Sampling(m);
$V_0 = O_{old,1}$;
for $k = 1, d$ **do**
  $O_{new} =$ Orthonormal_Sampling(m);
  $O_{new} =$ Positivation_by_symmetry$(O_{new}, 1)$;
  $D =$ Diag$((u_i)_{i \in \{1,2,...,m\}})$, $u_i \sim U(0, 1)$;
  $D_{1,1} = 1$;
  $W_k = O_{new}.D.O_{old}^{\top}$;
  $O_{old} = O_{new}$;
return $V_0, W_1, W_2, .., W_d$;

Algorithm 2 gives the main directions, and gives matrices aligned along a positive direction. This makes the ReLUs applied between each layer unable to alter the output vector of the main directions and it allows the Lipschitz constant to be equal to the product of the spectral norms of the weight matrices. The numerical values in algorithm 2 make the designed network 1-Lipschitz.

Remark 5.2.3. This algorithm can easily be changed for increasing slightly the broadness of the resulting model class:

- Having any Lipschitz constant L by changing the singular values.
- Not aligning the last layer can add an extra difficulty for Lipschitz assessing algorithms while not changing the main direction of the network, to that extent one should ensure that the singular values of the last layer do not shrink the main direction significantly compared to other directions. This feasible as there are no ReLUs after the last layer and thus the problem is linear. This does change the Lipschitz constant that can be computed by processing the main direction and checking the norm increase.
- Making different layer size on each layer to make more diverse structures.

We kept the algorithm minimal to make it more understandable but some of the improvements listed above may seem necessary for increasing the diversity of the networks generated. Nevertheless, the most general networks are the one made by solving equation (5.3.2) which we discuss in section 5.3.

Remark 5.2.4. Algorithm 2 is not an ideal case since the Lipschitz constant of a random neural network with normalized layers is actually shrunk by the nonlinearities (see [183] lemma 2). On top of that, we are creating a very specific case where finding the Lipschitz constant can easily be achieved by finding the main direction of the first weight matrix. Consequently, we are expecting any greedy algorithm seeking the main direction to perform exceptionally well on these samples. Once again, let us emphasize that models made using algorithm 2 offer a biased test case for an algorithm computing the Lipschitz constant. However, if such algorithm fails this test, it is a good indicator that the method, as it is implemented, has difficulties for approximating correctly the Lipschitz constant.

In an attempt to attend to the lack of generality of our naive construction of matrix weights, we devise in the following section a method for making more diverse structures with the same goal of controlling the Lipschitz constant of the network.

5.3 Designing general networks with known Lipschitz constant

In this section, we propose an approach that initializes randomly a model (a neural network) with a given structure. Its parameters are then optimized in order to fit a fixed target Lipschitz constant. We start by introducing notations and elementary definitions, and then present the theoretical bi-level optimization problem to solve. We subsequently provide an algorithm for approximating a solution of this bi-level optimization problem. Finally, we test the algorithm by performing some prescriptions of various values of Lipschitz constant.

5.3.1 Formulation as an optimization problem

Given a prescribed target Lipschitz constant $\bar{L}$, the objective becomes:

$$\text{Find } \theta^* \text{ such that } L_{f,\mathcal{X}}(\theta^*) = \bar{L}. \tag{5.3.1}$$

It is intended in equation (5.3.1) that such θ^* exists. Observe also that θ^* is not unique in general by obvious ambiguities (scale for instance). Thus, problem (5.3.1) can be solved only in an equivalence class.

Let us fix a distance/divergence function $d : \mathbb{R}_+^2 \to \mathbb{R}^+$ such that $d(a,b) = 0 \iff a = b$. One can then also formulate problem (5.3.1) as:

$$\min_{\theta \in \Theta} d(L_{f,\mathcal{X}}(\theta), \bar{L}). \tag{5.3.2}$$

If θ^* exists in the first problem then the minimum value in (5.3.2) is indeed 0.

Proposition 5.3.1. *In addition to the above, assume that $G_\phi(\cdot)$ is continuous (hence ϕ is $\mathcal{C}^1$), and that Θ is compact. Then the minimization problem (5.3.2) has a nonempty compact set of minimizers.*

Proof. Let us remind the expression of $L_{f,\mathcal{X}}(\theta)$ from equation (2.5.3):

$$L_{f,\mathcal{X}}(\theta) = \sup_{x\in\mathcal{X}} \left|\!\left|\!\left| \left(\prod_{i=1}^{d-1} W_i^\top \operatorname{diag}\left(G_\phi(f_i(\theta,x))\right)\right) W_d^\top \right|\!\right|\!\right|,$$

where $f_i(\theta,x)$ is the notation for intermediate layer activation from 2.2. $\left|\!\left|\!\left| \left(\prod_{i=1}^{d-1} W_i^\top \operatorname{diag}\left(G_\phi(f_i(\theta,x))\right)\right) W_d^\top \right|\!\right|\!\right|$ is a continuous function in both x and θ as we assumed $G_\phi(\cdot)$ to be continuous. The spectral norm is continuous, hence lower semicontinuous. The supremum in x is thus a lower semicontinuous function in θ. This together with compactness of Θ entails that the minimization problem (5.3.2) has a nonempty compact set of minimizers. □

The $\mathcal{C}^1$ assumption might be weakened but we will stick with this assumption for now. The compactness assumption on Θ also makes sense at least for some of the parameters to remove the ambiguities.

Remark 5.3.2. In any approach above, one has to appeal to some second-order information on ϕ, i.e., at some point to differentiate $G_\phi(z)$ with respect to z. In turn, this necessitates a higher order smoothness on ϕ compared to our work on robust optimization in Chapter 6 (which is not the case for the usual piece-wise linear activation functions).

5.3.2 Algorithm for solving equation (5.3.2)

A heuristic way to solve problem (5.3.2) is to alternatively maximize on x and then apply a generalized (sub)gradient descent step in θ. Our approach is summarized in algorithm 3, where the inner maximization step is replaced in practice by a PGD-like algorithm maximizing the spectral norm (equation (2.5.3)) with respect to the input x.

Algorithm 3: Pseudo-algorithm for approximating the prescription of a Lipschitz constant

Input: $\theta_1 = (W_i^1, b_i^1)_{i\in\{1,\dots,d\}}$ initial parameters of the network
Input: $\bar{L}$ the target Lipschitz constant
Input: $(\gamma_k)_{k\in\mathbb{N}}$ the decaying learning rate
for $k = 1, n$ **do**
  $x_k^* \in \operatorname{Argmax}_{x\in\mathcal{X}} \left|\!\left|\!\left| \left(\prod_{i=1}^{d-1} W_i^{k\top} \operatorname{diag}(G_\phi(f_i(\theta,x)))\right) W_d^{k\top} \right|\!\right|\!\right|$;
  $\mathcal{L}(\theta_k, x_k^*, \bar{L}) = \ell(\left|\!\left|\!\left| \left(\prod_{i=1}^{d-1} W_i^{k\top} \operatorname{diag}(G_\phi(f_i(\theta,x_k^*)))\right) W_d^{k\top} \right|\!\right|\!\right|, \bar{L})$;
  $u_k \in G_{\mathcal{L}(\cdot,x_k^*,\bar{L})}(\theta_k)$;
  $\theta_{k+1} = \theta_k - \gamma_k u_k$;
return θ_n;

In algorithm 3, $G_{\mathcal{L}(\cdot,x,\bar{L})}(\theta)$ is a generalized derivative of $\mathcal{L}$ with respect to variable θ evaluated at $(\theta, x, \bar{L})$.

Clearly, it is hoped that algorithm (3) provides an approximation to a solution of problem (5.3.1), though we do not provide any guarantee on this.

5.3.3 Evaluation of algorithm 3

We first evaluated the performance of algorithm 3 for enforcing a Lipschitz constant on a given network by building networks with different targets $\hat{L}$.

All the experiments were performed with a 6 layers linear neural network with 30 neurons on each layer. The input size is taken equal to the layer size: 30. We chose the ELU activation function as it is $\mathcal{C}^2$. We chose to prescribe the Lipschitz constant for the spectral norm by using power iterations for computing the operator norm. We noticed that enforcing a 1 spectral norm on all layers except the last one helps for convergence. It reduces the number of symmetries in the solution without any loss of generality. The optimization is performed using a stochastic gradient descent with a decaying learning rate. The initialization of the network [79] is the same

throughout all the configurations but in order to converge properly for large Lipschitz constants the number of epochs had to be adapted accordingly.

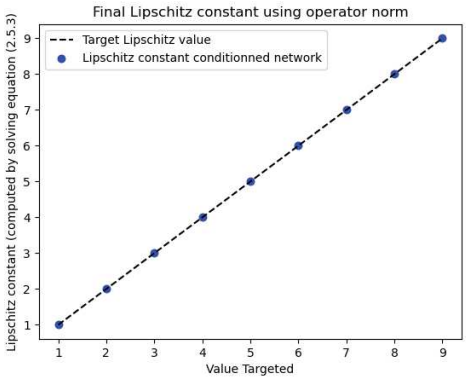

Figure 5.1: Prescription of different Lipschitz constant following algorithm 3 with objective (black) and Lipschitz constant of constrained network (blue dots).

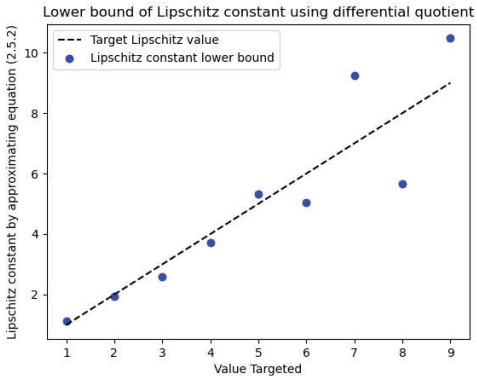

Figure 5.2: Computation of a lower bound the Lipschitz constant by maximizing the differential quotient (blue dots) with initial objective (black) on the same networks than Figure 5.1.

Figure 5.1 presents the results of using algorithm 3: the black curve is the objective that we target and the blue points are the approximation of the Lipschitz constant used for training of the resulting networks. The values of the Lipschitz constant that we prescribe in Figure 5.1 by using Algorithm 3 follow closely the target value materialized by the dashed black curve. However, further verification is needed since our approach makes several approximations. This is where Figure 5.2 comes into play: it presents a certified lower bound computed via maximization of a differential quotient (equation (2.5.2)) on the very same networks than Figure 5.1. The lower bounds computed are coherent for five values (prescribed values 2, 3, 4,6 and 8) since values lie under the target black curve, but for values 1, 5, 7 and 9 the lower bound is higher than the Lipschitz constant computed in Figure 5.1. This attests for an error in the convergence towards the target Lipschitz constant in Algorithm 3 despite the initial remarkable results in Figure 5.2. There are exactly two error-prone steps in our design process: the error coming from the PGD maximization step that approximate the argument of equation (2.5.3) and the error coming from our minimization of problem (5.3.2). The exact cause of an error on the prescription can not be traced back to either of the two sources of error and it may also be a combined effect of the two. We hypothesize that the inner maximization is to be blamed for the errors observed in Figure 5.2 since it combines a PGD-like algorithm and power iteration for the computation of the spectral norm. Similar results were obtained when using the dual form of the inner maximization of equation (2.5.4): this form allows to cast the spectral norm maximization into a single maximization problem instead of power iteration and maximization. This advocates for lingering error in the estimation of the Lipschitz constant caused by the PGD like algorithm. In order to solve exactly the inner maximization problem different approaches could have been undertaken, a first idea could have been to change the optimization algorithm for a non-deterministic one provided we have enough computational budget. This would have allowed to find a better maximizer in an unspecified execution time. Secondly we could have used entropic smoothing (this will be the subject of Chapter 6) in order to approximate and asymptotically lean towards the real maximizer, this approach is computationally demanding as well.

5.4 Comparing Lipschitz assessing algorithms

In this section we will use networks generated using algorithm 2 in order to compare Lipschitz assessing algorithms. This is motivated by two main reasons: on the one hand it is computationally cheap and allows us to run a greater variety of algorithms with an extended number of trials. On the other hand, it allows to split the sources of error as the approximation errors from the Lipschitz algorithms won't be spoiled by errors in the designing process: it ensures consistency of the result at the cost of generality. All the networks used in these experiments

have a Lipschitz constant of 1, this is an arbitrary choice but it can be changed easily with an appropriate scaling of the layers.

5.4.1 Highlighting certified/uncertified bounds

Some algorithms are giving a certified upper bound; this bound can be reasonably loose depending on the approach used. Some other algorithms provide a certified lower bound: the returned value is always lower than the actual Lipschitz constant. Finally, there exists also uncertified upper/lower bounds that aim at approximating an upper/lower bound but with no guarantees to provide a value that is above/below the Lipschitz constant. These statements can be assessed empirically if we use models from algorithm 2 to observe the magnitude of the error. We will consider the following algorithms:

- The LipSDP-network algorithm [71] in Figure 5.3: the algorithm gives a certified upper bound.
- The SeqLip algorithm [183] using a greedy approach in Figure 5.4, the algorithm gives an uncertified upper bound.
- The SeqLip algorithm using a genetic optimizer in Figure 5.5, the algorithm gives an uncertified upper bound.
- A certified lower bound computed by sampling uniformly the input space in Figure 5.6.

The protocol of the experiment is as follows: for a given layer size and depth, we design a network using algorithm 2 and assess the Lipschitz constant using all the methods enumerated above. We have chosen arbitrarily the dimension of the input to be equal to the layer size. We display the evolution of the estimation of the Lipschitz constant when making the layer size and depth vary for each Lipschitz assessing method. We decided not to average the evaluation over multiple experiments because we are trying to display the flaws in given executions of Lipschitz approximations, not an average behavior. Advocating on general performance of the algorithms on neural networks would be biased considering the specificity of networks that we are testing. Therefore, our goal resides in highlighting the dependencies between the methods and the characteristic dimensions of our designed neural networks: both dimension of the inputs and dimension of the parameters.

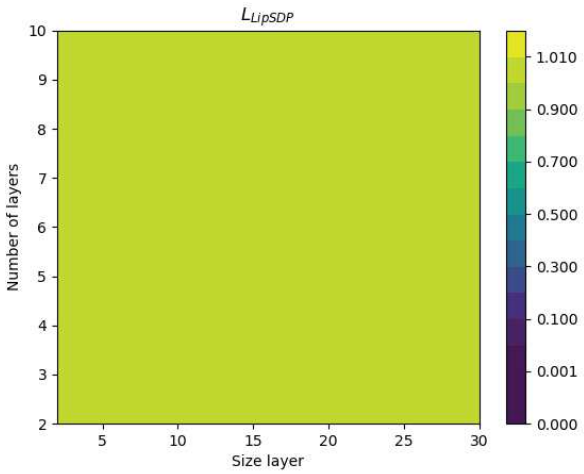

Figure 5.3: Computation of the Lipschitz constant using the LipSDP algorithm for different depths (y axis) and different layer size (x axis).

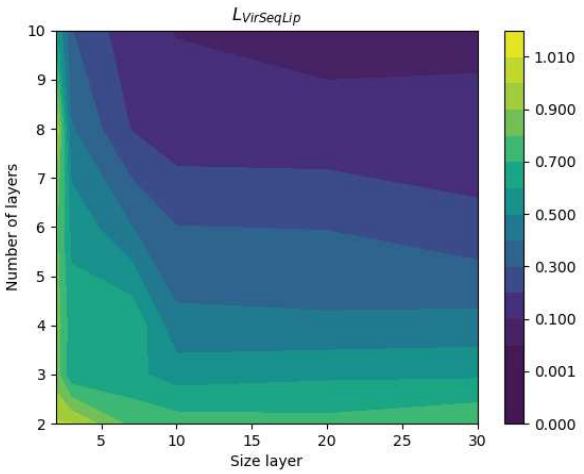

Figure 5.4: Computation of the Lipschitz constant using the SeqLip algorithm for different depths (y axis) and different layer size (x axis) with numpy optimizer.

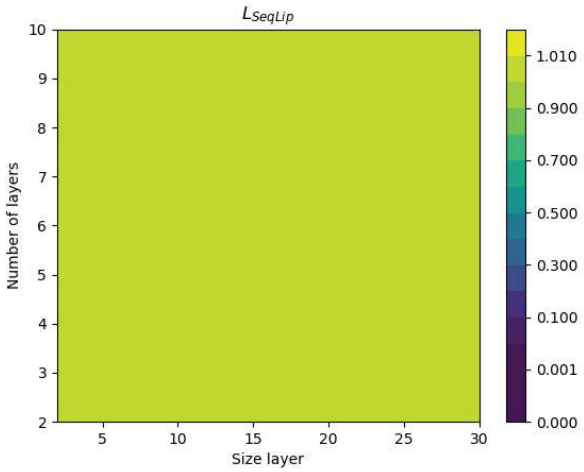

Figure 5.5: Computation of the Lipschitz constant using the SeqLip algorithm for different depths (y axis) and different layer size (x axis) with genetic optimizer.

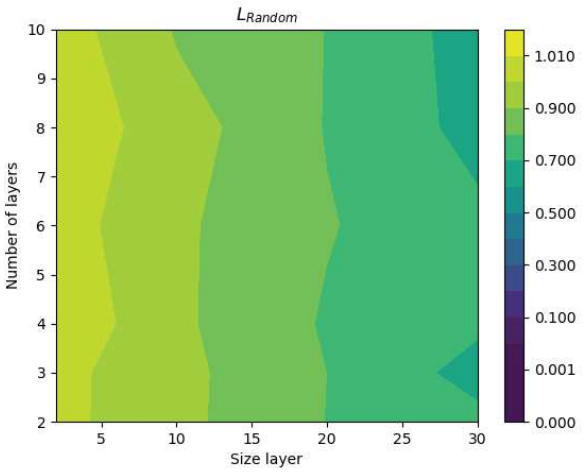

Figure 5.6: Computation of the Lipschitz constant using the random exploration for different depths (y axis) and different layer size (x axis).

Figure 5.3 and 5.5 provide the best landscape of estimation of the Lipschitz constant: according to the color scale, the methods appear to be finding accurately the value of the Lipschitz constant. One very interesting comparison to make is between Figures 5.4 and 5.5: they are both implementations of SeqLip however they are resorting to different solvers. The former solver is greedy - and therefore suboptimal for dealing with an NP-hard problem - and the latter is nondeterministic - and therefore finds the optimal solution provided it has iterated sufficiently. This shows that, although the theory ensures that SeqLip should ultimately provide an upper bound, it is not achieved if the problem is not solved suitably. On top of that, it showcases that the value given by this algorithm should be treated carefully and cannot be compared reliably to a lower bound. Figure 5.6 presents a method for computing a certified lower bound for the Lipschitz constant: a random approach to seek the maximum of the differential quotient. The random search depends only on the dimension of the sampling space (that is taken equal to the layer size in our specific case).

Our prescribed samples allow to appreciate the amplitude of the error without any other source of error plaguing the result. Our experiment is the first of its kind that quantifies the extent of the error of the Lipschitz assessing algorithm on known results. Comparing a value from an algorithm that gives a certified upper bound with an uncertified one hardly makes sense without further considerations on error bounds over the estimation. Indeed, cautions need to be taken when performing such comparisons: one should explicitly state the guarantees offered by a competing method and to which extent the two are comparable. Even though LipSDP and SeqLip are both able to provide a very good approximation on our designed use cases (see Figures 5.3 and 5.5), a poor optimizer on the former (see Figure 5.4) could have us believe it outperforms the latter if we did not know and expect the result (since its estimation of the constant is lower than the one of the two former charts). We advocate for more thorough and rigorous testing procedures and the test we are providing is a sound way to begin with. Ideally one should perform the test on more diverse structures like the ones solution to equation (5.3.2) provided it is solved exactly.

5.4.2 Highlighting computational limits

Our test cases can also serve the purpose of assessing the limits of Lipschitz assessing algorithms and their behaviors alongside validity boundaries: are their predictions still valid next to their validity limits, or are the algorithms already giving a degraded value of the Lipschitz constant? To that extent, we probe the four previous algorithms on a much larger range of values of depth and layer size that should trespass the range of validity of the algorithms. We display in hatches the zones that resulted in an error when executing the algorithms.

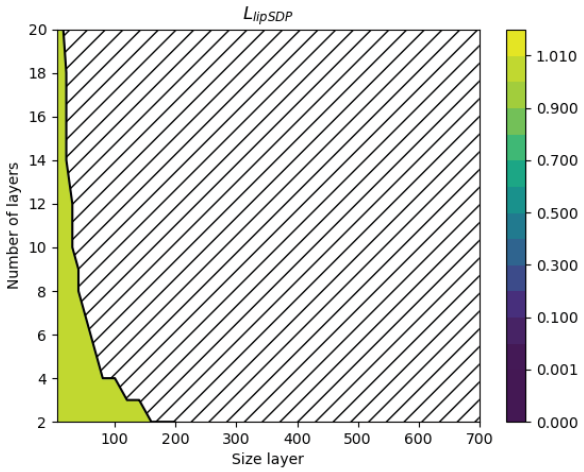

Figure 5.7: Computation of the Lipschitz constant using the LipSDP algorithm for larger depths (y axis) and larger layer size (x axis).

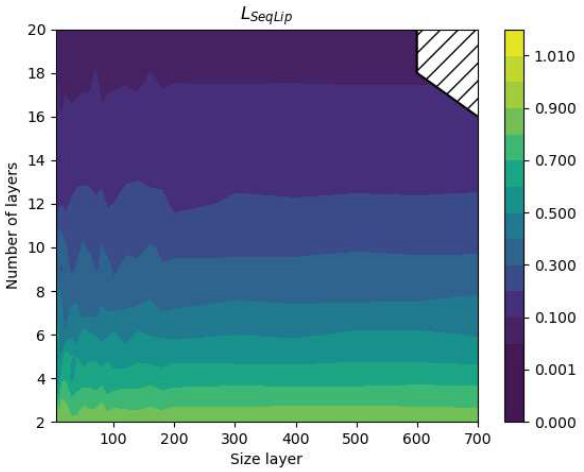

Figure 5.8: Computation of the Lipschitz constant using the SeqLip algorithm for larger depths (y axis) and larger layer size (x axis) with numpy optimizer.

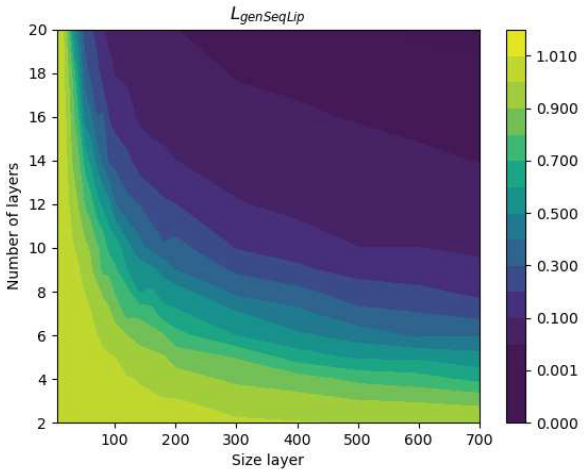

Figure 5.9: Computation of the Lipschitz constant using the SeqLip algorithm for larger depths (y axis) and larger layer size (x axis) with genetic optimizer.

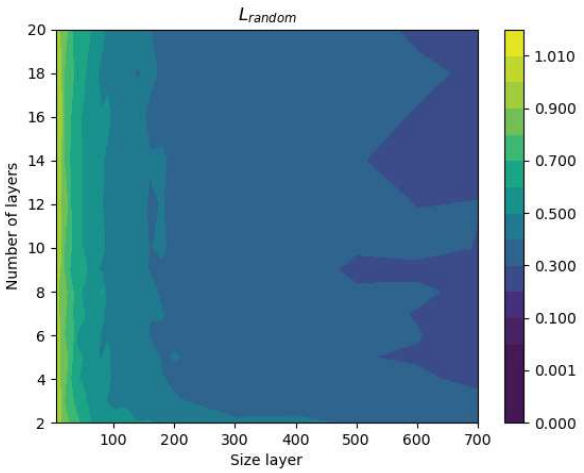

Figure 5.10: Computation of the Lipschitz constant using random exploration for larger depths (y axis) and larger layer size (x axis).

These figures can be considered as zooming out in the figures from the previous section although the object of this study is more the validity domain rather than the error in the predictions.

On Figure 5.7 we see that the algorithm gives a very good approximation of the Lipschitz constant but on a small region. This is in conformity with the description in the original work from [71]: this "network" version of LipSDP suffers greatly from the dimension. Figure 5.8 and 5.9 present two different optimizers on the same maximization problem. Figure 5.9 is clearly dependent on the dimension of the parameters as it is displaying similar quality of estimation along the iso-dimension curves (curves of the form: Depth $\times$ (Layer Size)^2 = constant) whereas Figure 5.7 displays similar performance along depth values. On the other hand, the random approach (Figure 5.10) seems to suffer drastically less from the dimension than the two previous algorithms making it a more viable choice for computing a lower bound in high parameter dimensions since it only depends on the dimension of the input.

Overall these experiments display that there is no free lunch when dealing with the complex problem of finding the Lipschitz constant: for instance, tractability of LipSDP is the cost for guarantees and accuracy in the result. For Seqlip guarantees and, to some extent, accuracy are the cost for tractability. Many characteristics

could make a method preferable over another: the non-exhaustive list of properties are the presence of guarantees, the computational cost, the scope of application (if it is applicable on CNNs, Resnets or RNNs) or if it has error bounds on its estimation. The literature is generally focused on the performance of the algorithms compared to other methods, without any regards to others key properties. Our test allows to check empirically both the computational limits and the factors influencing the error over the estimation of the Lipschitz constant. Ultimately, we believe our method paves the way for a more rigorous and extensive testing of Lipschitz assessing algorithms. Such added testing provides a better perspective over the literature and the compromises it makes.

5.5 Conclusions and future work

This chapter introduced a novel paradigm for evaluating and understanding algorithms assessing the Lipschitz constant of neural networks. This work fills a gap in the evaluation of such algorithms, as none of the methods proposed to date have been evaluated on models for which the Lipschitz constant is known. In addition, our experiments allow a better analysis of the estimates given by the algorithms. It helps highlighting the specificity of some approaches, such as the dependency to the input dimension for some of the algorithms or the choice of the optimizer for others. Overall, it allows to improve and validate the parameters chosen for a given estimation algorithm.

In future work, we plan to soften the constraints that we enforced on the neural networks, without sacrificing our knowledge of the Lipschitz constant. Extensions to convolutional layers seems the logical path, but it has its own challenges as the Lipschitz constant depends not only on the parametrization of the convolution layer but also on the input size, which can differ from one sample to another. One could also consider, in the same fashion that we did in this contribution, slightly constrained special cases where it would be easier to actually know the Lipschitz constant of the convolutions beforehand.

Chapter 6

Regularized Robust Optimization with Application to Robust Learning

Main contributions of this chapter

- Bounding the distributionally robust objective with the pointwise robustness objective, this shows that provided some hypothesis on the class of function, one can approximate the former by the latter with arbitrary small error.
- Approximating the pointwise robust problem with a smoothed objective that trades complexity for computational cost. This smoothed objective is shown to have asymptotically the same optimal point than the non smooth problem.
- Presenting an algorithm for solving the smoothed approach with arbitrary small error, almost surely.
- Showing in a use case the benefits of using smoothing in terms of performance compared to other robust training methods that do not enjoy the same guarantees.

The content of this chapter is under review in the Journal of Optimization Theory and Applications.

Contents

6.1 Introduction

The need of robust models arises when we are considering modeling in the face of uncertainties. Building a reliable decision-making system in the face of uncertain inputs is central to many critical applications: not only the system has to prove it operates correctly on its operational design setting, but it also has to remain stable under some perturbation. In the literature, stability over some kind of perturbation is referred to as robustness and is one of the main challenges in many areas of science and engineering, for instance in statistical learning. Since there are growing applications in computer vision applied to critical systems, it is crucial to prove that a statistical model can operate under a given level of uncertainty. On some critical cases the model has to remain stable given any possible point in the uncertain set, as if the perturbation was tailored by an adversary to perturb our decision-making. In this context, Robust Optimization allows to optimize under uncertainty without an explicit model of the uncertainties and thus aims at limiting the scope of actions of any adversary perturbing the model.

6.1.1 Problem statement

Let $(\mathcal{Z}, \mathsf{d})$ be a (data) metric space with $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^m$, where $\mathcal{X}$ (resp. $\mathcal{Y}$) is the input (resp. output) space. Let ρ_0 be a probability measure on $\mathcal{Z}$, $\Theta \subset \mathbb{R}^p$ the parameter/action space, $\mathcal{L} : \Theta \times \mathcal{Z} \to \mathbb{R}_+$ a loss function such that $\mathcal{L}(\cdot, \theta)$ is ρ_0-measurable and integrable for all $\theta \in \Theta$. Throughout, we assume that Θ is closed. Consider the optimization problem:

$$\min_{\theta \in \Theta} \mathbb{E}_{z \sim \rho_0} \left[\mathcal{L}(\theta, z) \right]. \tag{6.1.1}$$

Robust Optimization is one contemporary robustification approach to deal with the presence of data perturbations, adversarial attacks, or uncertainties in (6.1.1) [12, 16]. The origin of the RO approach can be traced back to the classical economic paradigm of a two-person zero-sum game formulated as a min-max problem (see e.g., [156] the recent review paper on min-max problems and their applications from a signal processing and machine learning perspective). In this framework, an agent, considered as a defender, is subject to degradation of its performance by a secondary player, the attacker. The defending agent (here θ), whose goal is to minimize an objective function under action constraints Θ, aims at guarding against the degradation of the objective by optimizing its worst value under perturbation without changing the feasibility set of the actions. Put formally, the robust counterpart of (6.1.1) reads:

$$\min_{\theta \in \Theta} \mathbb{E}_{z \sim \rho_0} \left[\max_{\mathsf{d}(z, z') \leq \varepsilon} \mathcal{L}(\theta, z') \right], \tag{6.1.2}$$

where $\varepsilon > 0$ is the size of uncertainty/perturbation/attack, and the perturbation acts pointwise on z in the adversarial risk, which justifies the terminology Pointwise Robust Optimization (PRO) for problem (6.1.2). A common choice is $\mathsf{d}(z, z') = \|z - z'\|_q$ where $\|\cdot\|_q$ is the ℓ_q norm on $\mathbb{R}^m$ with $q \geq 1$ with the usual adaptation for $q = \infty$.

PRO is one way of quantifying the impact of an adversary or perturbation, and other notions of adversarial risk have been proposed in the literature. In particular, in many areas, such as machine learning, the existence and pervasiveness of adversarial examples point to the limitations of the usual independent and identically distributed (i.i.d.) model of perturbations. This points to a more general perspective in which it is not the points themselves that are perturbed, but rather their underlying distribution ρ_0. This approach is known as Distributionally Robust Optimization (DRO) which takes the form

$$\min_{\theta \in \Theta} \max_{\mathsf{D}(\rho, \rho_0) \leq \varepsilon} \mathbb{E}_{z \sim \rho} \left[\mathcal{L}(\theta, z) \right], \tag{6.1.3}$$

where D is a discrepancy on the space of probability measures supported on $\mathcal{Z}$. Compared to PRO, DRO allows to consider a larger range of perturbations. The choice of D affects the richness of the uncertainty set and the tractability of the resulting optimization problem. Typical choices are the Wasserstein distances [168, 21, 22, 172, 36], the Maximum Mean Discrepancy (MMD) [175] or ϕ-divergences including the Kullback-Leibler divergence [11, 136, 64]. There are also other ways to parametrize the perturbation set in terms of the distribution moments, support, etc., [80, 58]. The Wasserstein distance has become very successful in this context, and unlike other distances/divergences, a Wasserstein distance enjoys the remarkable property that its ball around ρ_0 includes measures having a different support, which allows robustness to unseen data. In the rest of this paper, we focus on Wasserstein balls.

A useful observation at this stage is that the objective in the inner problem of the saddle point problem (6.1.3) is concave (actually linear) in ρ. Though this property is apparently appealing, this problem operates in infinite dimension (space of probability measures on $\mathcal{Z}$), and thus is very challenging to solve.

The natural question that arises is whether one can have a surrogate of problem (6.1.3) which operates in finite-dimension under minimal assumptions on the problem data (for instance $\mathcal{L}$), and if this can come up with provable guarantees. For instance, is there a relationship between DRO (6.1.3) and PRO (6.1.2) (or alike) and hence can we solve the latter as a surrogate for the former ? We will show later that this is indeed the case.

On the other hand, the rigorous treatment of problem (6.1.2), though in finite dimension, remains very challenging for general losses $\mathcal{L}$, especially in absence of the important properties of joint convexity-concavity in (θ, z)

and smoothness which are key to design efficient and provably convergent algorithms [12]. These assumptions are however stringent and unrealistic in applications we have in mind, for instance in adversarial training with neural networks [82, 126, 32, 180]. Iterative solvers used by many authors do not come up with any guarantee. In fact, in such applications, the inner maximization problem is generally non-concave in z and is even provably NP-hard with certain activations such as ReLU [172]. The goal pursued in this chapter is thus to design algorithms to solve problem (6.1.2), as a surrogate for problem (6.1.3) under minimal assumptions on the problem data (for instance, without need of joint convexity-concavity) while enjoying convergence guarantees.

6.1.2 Contributions

Our main contributions in this work are:

1. The DRO problem (6.1.3), when D is the Wasserstein distance with Lipschitz continuous ground cost, is approached with a PRO counterpart of the constrained form (6.1.2) with a controlled accuracy that depends on the perturbation radius.
2. To avoid solving the generally intractable inner maximization problem in equation (6.1.2), we first smooth the latter using entropic regularization and then use Monte Carlo integration to approximate integrals. We conduct an error analysis to precisely quantify these approximation errors and provide error bounds both on the objective values and its subgradients. Relying on the theory of Γ-convergence we show in particular that the minimizers of the approximate problems converge to those of PRO.
3. Capitalizing on the above results, we propose provably convergent stochastic (sub)gradient descent (SGD) algorithms to solve the PRO problem. The first algorithm supposes access to an oracle of the inner maximization problem. To avoid the latter, which can be challenging, we also provide an inexact SGD algorithm with asymptotically vanishing error/bias. The error/bias originates from the regularization and integration sampling parameters, and making them decay at an appropriate rate, convergence guarantees to critical points are established without any need of convexity-concavity assumptions on the loss $\mathcal{L}$.

6.1.3 Relation to prior work

There is a substantial body of work on robust optimization dedicated to robust learning. Here we only review those closely related to ours. Many works have studied instances of (6.1.3) for which tractable algorithms can be designed. For D chosen as a ϕ-divergence, and under some assumptions on $\mathcal{L}$, [12, 136, 65] propose convex optimization approaches. For the Wasserstein distance, and a limited class of convex losses $\mathcal{L}$ and ground costs, some authors convert (6.1.3) into a regularized empirical risk minimization problem [70, 168, 21, 22]. For a larger class of losses and ground costs c, (6.1.3) is converted in [172] to a Lagrangian form of (6.1.2). Stochastic gradient descent is then applied to this penalized form and convergence guarantees are established under the assumptions that the gradient of the loss $\mathcal{L}$ is bi-Lipschitz and the ground cost c is strongly convex. However, their algorithm resorts to an oracle corresponding to solving the inner supremum problem. This is again a challenging problem and even NP-hard. When ρ_0 is the empirical measure and $\mathcal{L}$ is Lipschitz continuous in z uniformly in θ the authors in [77, 175] convert (6.1.3) into a finite dimensional saddle point problem different from (6.1.2) (see detailed discussion in Section 6.2). Stochastic coordinate descent was advocated in [175] to solve the latter problem but without any guarantee. In this work, we treat a much larger class of losses and costs and use smoothing to translate the inner maximization problem into an integration problem that we approximate with Monte Carlo integration. Overall, this allows us to apply stochastic gradient descent while being able to prove convergence to critical points of (6.1.2). While we were finalizing this contribution, we became aware of the work of [130] who also used entropic smoothing to learn an optimally robust randomized mixture of classifiers. Their setting and motivation is however different and their algorithm does not enjoy convergence guarantees.

6.1.4 Organization of the chapter

Section 6.2 shows mild conditions under which DRO can be reasonably approximated using PRO. Section 6.3 is devoted to our smoothing approach and its key theoretical properties. In Section 6.4, we turn to studying provably convergent algorithms to solve the PRO problem. Finally, we illustrate these results on some use cases (Section 6.5) that show the advantages of using smoothing over other heuristics.

6.2 Robustness bounds

The goal here is to show how to go from the DRO problem (6.1.3) to the PRO one (6.1.2), provably, by bounding the corresponding objectives. This paves the way to using PRO as a surrogate for DRO provided that ε is not too large. In the rest of the chapter, we assume that $\mathcal{L}$ and the ground cost function c satisfy the standing assumption:

(H.1) $\mathcal{L}$ is continuous.

(H.2) $\mathsf{c} : \mathcal{Z}^2 \to \mathbb{R}_+$ is continuous, symmetric ($\mathsf{c}(z, z') = \mathsf{c}(z', z)$) and $\mathsf{c}(z, z') = 0 \Leftrightarrow z = z'$.

Proposition 6.2.1. *Suppose that (H.1)-(H.2) hold.*

(i) If

(H.3) *for every $\theta \in \Theta$, $\forall (z, z') \in \mathcal{Z}^2$, $|\mathcal{L}(\theta, z) - \mathcal{L}(\theta, z')| \leq L_{\mathcal{Z}}(\theta)\mathsf{c}(z, z')$ with $0 \leq L_{\mathcal{Z}} \overset{\text{def}}{=} \sup L_{\mathcal{Z}}(\Theta) < +\infty$.*

Then

$$0 \leq \sup_{W_{\mathsf{c}}(\rho,\rho_0)\leq\varepsilon} \mathbb{E}_{\rho}[(\mathcal{L}(\theta, z)] - \mathbb{E}_{z\sim\rho_0}\left[\sup_{\mathsf{c}(z,z')\leq\varepsilon} \mathcal{L}(\theta, z')\right] \leq L_{\mathcal{Z}}\varepsilon.$$

(ii) If $\mathsf{c}(z, z') = \|z - z'\|^q$, $q \geq 1$, where $\|\cdot\|$ is a norm on $\mathbb{R}^m$ (i.e., $W_{\mathsf{c}}^{1/q}$ is the q-Wasserstein distance W_q), and $\mathcal{L}(\theta, \cdot)$ is $L_{\mathcal{Z}}$-Lipschitz continuous with respect to $\|\cdot\|$ uniformly in θ, then

$$0 \leq \sup_{W_q(\rho,\rho_0)\leq\varepsilon^{1/q}} \mathbb{E}_{\rho}[(\mathcal{L}(\theta, z)] - \mathbb{E}_{z\sim\rho_0}\left[\sup_{\|z-z'\|\leq\varepsilon^{1/q}} \mathcal{L}(\theta, z')\right] \leq C_{q,L_{\mathcal{Z}}}\varepsilon^{1/q},$$

where $C_{q,L_{\mathcal{Z}}}$ is a non-negative constant that depends only on q and $L_{\mathcal{Z}}$.

See Appendix 6.7.1 for the proof.

In [172] (see also [21]), using Lagrangian duality arguments, it was shown that

$$\sup_{W_{\mathsf{c}}(\rho,\rho_0)\leq\varepsilon} \mathbb{E}_{\rho}[(\mathcal{L}(\theta, z)] = \inf_{\gamma\geq 0}\left(\gamma\varepsilon + \mathbb{E}_{z\sim\rho_0}\left[\sup_{z'\in\mathcal{Z}}\left(\mathcal{L}(\theta; z') - \gamma\mathsf{c}(z, z')\right)\right]\right). \tag{6.2.1}$$

For a fixed parameter $\gamma > 0$, this can be seen as a penalized form of the constrained form (6.1.2). Though (6.2.1) is an identity rather than a bound, it faces a few algorithmic challenges to solve. For instance, the joint presence of the expectation and the inner maximization problems makes minimization of the multiplier γ a difficult task. One can of course think of a simple procedure such as bisection but this will necessitate extra-smoothness assumptions and to solve for θ for each value of γ on the bisection. If $\mathcal{L}(\theta, \cdot)$ has a Lipschitz continuous gradient, and c is strongly convex in its second argument, then it can be easily shown that for γ large enough, $\mathcal{L}(\theta, \cdot) - \mathsf{c}(z, \cdot)$ is strongly concave. This has been leveraged by [172] to use gradient descent to minimize over θ, but only for a fixed (large enough) parameter γ. But still, choosing γ is not easy.

In [77, 175], taking c as the ℓ_p cost, and ρ_0 the empirical measure on n points, the following bound was established

$$0 \leq \sup_{W_p(\rho,\rho_0)\leq\varepsilon} \mathbb{E}_{\rho}[(\mathcal{L}(\theta, z)] - \sup_{(z'_i)_i:\frac{1}{n}\sum_{i=1}^n \|z'_i - z_i\|^p\leq\varepsilon^p} \frac{1}{n}\sum_{i=1}^n \mathcal{L}(\theta, z'_i) \leq L_{\mathcal{Z}}/n. \tag{6.2.2}$$

As in our case, this gives access to a uniform bound, but which depends now on n rather than ε (and thus gets tighter as n increases). However, the price to pay is that, unlike problem (6.2.1), the inner maximization in the surrogate problem (6.2.2) is coupled between all variables in the objective and constraints, necessitating to optimize on a variable in $\mathbb{R}^{mn}$ rather than $\mathbb{R}^m$.

6.3 Entropic regularization

Despite formulating surrogates as devised in (6.2.1) and (6.2.2) and discussed above, solving the resulting min-max problems remains a very challenging task unless stringent joint convexity/concavity assumptions are made. This is the motivation behind our smoothing hereafter.

From now on, we will denote $\mathcal{C}_z^\varepsilon \overset{\text{def}}{=} \{z' : \mathsf{c}(z', z) \leq \varepsilon\}$. The PRO problem now reads

$$\min_{\theta\in\Theta} \mathbb{E}_{z\sim\rho_0} \left\{ \max_{z'\in\mathcal{C}_z^\varepsilon} \mathcal{L}(\theta, z') \right\}. \tag{6.3.1}$$

We will use the shorthand notation[1]

$$g(\theta) \overset{\text{def}}{=} \max_{z'\in\mathcal{C}_z^\varepsilon} \mathcal{L}(\theta, z').$$

Provided that $\mathcal{C}_z^\varepsilon$ is bounded, hence compact since it is closed by assumption on c, and recalling the continuity assumption (H.1), the set of maximizers in g is a non-empty compact set. The function g can also be equivalently rewritten as

$$g(\theta) = \max_{\mu\in\mathcal{P}(\mathcal{C}_z^\varepsilon)} \int_{\mathcal{C}_z^\varepsilon} \mathcal{L}(\theta, z')\mathrm{d}\mu(z'), \tag{6.3.2}$$

and the integral is a duality pairing between $\mathcal{P}(\mathcal{C}_z^\varepsilon)$ and $\mathcal{C}(\mathcal{C}_z^\varepsilon)$. We will show that (6.3.2) and its subgradients can be provably approximated following a two-step strategy: first, an *entropic regularization* followed by *Monte-Carlo sampling* to approximate the integrals.

6.3.1 Regularized objective

Problem (6.3.2) is concave in μ, but operates in infinite-dimension and is thus hard to solve. Approximating (6.3.2) by an atomic measure supported on a finite set (i.e., replace $\mathcal{P}(\mathcal{C}_z^\varepsilon)$ by a finite-dimensional simplex by sampling N points at random in $\mathcal{C}_z^\varepsilon$), as done by some authors (see e.g., [140]), suffers an exponential dependence in $1/m$. Indeed, an analysis using Lipschitzianity of $\mathcal{L}$ shows that this method achieves an approximation rate of $O(N^{-1/m})$, and essentially, this cannot be improved. Rather, we will consider the following regularized version of (6.3.2), namely:

$$g_\tau(\theta) \overset{\text{def}}{=} \max_{\mu\in\mathcal{P}(\mathcal{C}_z^\varepsilon)} \int_{\mathcal{C}_z^\varepsilon} \mathcal{L}(\theta, z')\mathrm{d}\mu(z') - \tau\mathrm{KL}(\mu, \nu), \tag{6.3.3}$$

where $\tau > 0$ is the regularization parameter, and ν is a reference measure supported on $\mathcal{C}_z^\varepsilon$. Entropic regularization has been used in several fields including optimal transport [146] and semi-infinite programming [187].

Observe that $\mathcal{C}_z^\varepsilon$ is Lebesgue measurable by continuity of c. In the sequel we also suppose that $\mathcal{C}_z^\varepsilon$ is of full dimension, and set ν as the uniform measure $\mu_\mathcal{U}$ on $\mathcal{C}_z^\varepsilon$, that is $\nu(z') = \mu_\mathcal{U}(z') \overset{\text{def}}{=} \mu_\mathcal{L}(\mathcal{C}_z^\varepsilon)^{-1} < +\infty$ for all $z' \in \mathcal{C}_z^\varepsilon$. The KL regularization term then prevents solutions to be atomic measures as such measures are not absolutely continuous with respect to the uniform one.

Remarkably, (6.3.3) is well-posed under mild conditions and has a unique solution taking an explicit form.

[1] g depends on z but we drop this in the notation to lighten the latter.

Proposition 6.3.1. *Assume that (H.1)-(H.2) hold and that $\mathcal{C}_z^\varepsilon$ is bounded and full dimensional. Then (6.3.3) has a unique solution and*

$$g_\tau(\theta) = \tau \log\left(\mathbb{E}_{z'\sim\mu_{\mathcal{U}}}\left[\exp\left(\frac{\mathcal{L}(\theta,z')}{\tau}\right)\right]\right) = \tau\log\left(\frac{\int_{\mathcal{C}_z^\varepsilon}\exp\left(\frac{\mathcal{L}(\theta,z')}{\tau}\right)\mathrm{d}z'}{\mu_{\mathcal{L}}(\mathcal{C}_z^\varepsilon)}\right). \tag{6.3.4}$$

The proof can be found in Appendix 6.7.2. Note that this is a generalization of the standard log-sum-exp formula for softmax smoothing of the maximum of a finite number of functions, where now the sum is replaced by an expectation wrt to the base measure $\mu_{\mathcal{U}}$.

Remark 6.3.2. The boundedness assumption on $\mathcal{C}_z^\varepsilon$ is very mild and verified in most applications we have in mind, for instance in robust training in machine learning. For instance, when $\mathsf{c}(z,z') = \varphi(\|z-z'\|)$ where $\|\cdot\|$ is any norm on $\mathbb{R}^m$, and $\varphi : \mathbb{R}_+ \to \mathbb{R}_+$ is a continuous increasing function.

Remark 6.3.3. There have been recently very few papers replacing the Wasserstein distance by its entropic (KL) regularization in the DRO uncertainty set, see [20, 185, 5]. In all these papers, the focus is on establishing strong duality and deriving the dual problem of DRO with the regularized Wasserstein distance, hence extending the dual problem (6.2.1) under various more or less stringent assumptions. The key property of the dual problem after entropic regularization of the Wasserstein distance is that the inner supremum problem in (6.2.1) is replaced by a smoothed approximation of the LIE (Log-Integration-Exp) type. This was used in [20] and [185] for computational purposes, using for instance bisection on γ but fixed entropic regularization parameter, but with little/no guarantees. The LIE smoothing is reminiscent of ours but it has also distinctive properties. Our uniform base measure in the integration is the same as in [20], but different from [185]. We also embark directly from the PRO problem where the uncertainty is in a constrained form rather than a penalized in the above papers. We thus do not need to optimize or choose γ. Moreover, we let our smoothing parameter decrease (this can be also done in those papers) and we provide strong convergence guarantees of SGD which is lacking in those papers.

6.3.2 Consistency of the regularization

We now turn to describing how g_τ in equation (6.3.3) is a good surrogate for g in (6.3.1). We provide both a qualitative result as the regularization parameter τ vanishes, and a quantitative convergence rate.

Theorem 6.3.4. *Under the assumptions of Proposition 6.3.1, the following statements hold:*

(i) $g_\tau(\theta) \nearrow g(\theta)$ *as* $\tau \searrow 0^+$. *In turn,* g_τ Γ*-converges to* g *as* $\tau \searrow 0^+$.

(ii) If, moreover, $\mathcal{C}_z^\varepsilon$ is convex and full dimensional, and (H.3) holds with $\mathsf{c}(z,z') = \|z-z'\|$, where $\|\cdot\|$ is a norm on $\mathbb{R}^m$. Then for any $\tau \in]0,1]$:

$$g(\theta) - h(\tau) \le g_\tau(\theta) \le g(\theta), \tag{6.3.5}$$

where

$$h(\tau) = m\tau\log(\tau^{-1}) + \tau\log\left(\frac{\mu_{\mathcal{L}}(\mathcal{C}_z^\varepsilon)}{\mu_{\mathcal{L}}(\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(0))}\right) + \tau L_{\mathcal{Z}}(R_{\mathcal{C}_z^\varepsilon} + D_{\mathcal{C}_z^\varepsilon}), \tag{6.3.6}$$

and $R_{\mathcal{C}_z^\varepsilon}$ is the radius of the largest ball[2] contained in $\mathcal{C}_z^\varepsilon$ and $D_{\mathcal{C}_z^\varepsilon}$ is the diameter of $\mathcal{C}_z^\varepsilon$.

See Appendix 6.7.3 for the proof.

Remark 6.3.5.

1. The convexity of $\mathcal{C}_z^\varepsilon$ is again usual in robust training in machine learning. It holds if the robustness cost c on $\mathcal{Z}$ is convex.

[2] In the norm $\|\cdot\|$ of course.

2. The bound (6.3.5) yields uniform convergence. The Γ-convergence claim in this case also follows from [128, Proposition 5.2 and Remark 5.3].
3. The dependence of the convergence rate function $h(\tau)$ on τ is nearly linear (up to a logarithmic term).
4. For the dependence on the dimension of the convergence rate, the first term grows linearly. So does also the second term since it can be upper-bounded by $m\tau \log\left(\bar{R}_{\mathcal{C}_z^\varepsilon}/R_{\mathcal{C}_z^\varepsilon}\right)$ where $\bar{R}_{\mathcal{C}_z^\varepsilon}$ is the radius of the smallest ball containing $\mathcal{C}_z^\varepsilon$. For the last term, the Lipschitz constant L_z of the loss function may also depend on the dimension. This emphasizes the role of the Lipschitz constant of the loss in controlling the robustness of the model.

In view of Theorem 6.3.4, we obtain the following key result, which relates the minimizers of the smoothed problem to those of the original PRO problem.

Theorem 6.3.6. *Suppose that the assumptions of Proposition 6.3.1 hold. Assume also that Θ is compact. Let $\theta_\tau^\star \in \operatorname{Argmin}_\Theta(g_\tau)$. Then the following holds:*

(i) $\lim_{\tau\to 0^+} \min_{\theta\in\Theta} g_\tau(\theta) = \min_{\theta\in\Theta} g(\theta)$.

(ii) Each cluster point of $\theta_\tau^\star$, as $\tau \to 0^+$, lies in $\operatorname{Argmin}_\Theta(g)$.

(iii) In particular, if $\operatorname{Argmin}_\Theta(g) = \{\theta^\star\}$, *then* $\lim_{\tau\to 0^+} \theta_\tau^\star = \theta^\star$.

See Appendix 6.7.4 for the proof.

6.3.3 Monte Carlo approximation of the integral

Computing the values $g_\tau(\theta)$ in (6.3.4) necessitates to compute a possibly high dimensional integral. Our goal is to approximate the latter with Monte Carlo integration by uniformly drawing independent samples $(z_k')_{k=1}^N$ in the set $\mathcal{C}_z^\varepsilon$. This gives the approximation

$$g_{\tau,N}(\theta) \stackrel{\text{def}}{=} \tau \log\left(\sum_{k=1}^N \frac{\exp\left(\frac{\mathcal{L}(\theta,z_k')}{\tau}\right)}{N}\right). \tag{6.3.7}$$

We now provide an error bound for such an approximation.

Theorem 6.3.7. *Suppose that the assumptions of Theorem 6.3.4(ii) hold. Then, the following holds.*

(i) For any $t > 0$ and every $\theta \in \Theta$,

$$|g_{\tau,N}(\theta) - g(\theta)| \le h(\tau) + \tau e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau}} \sqrt{\frac{t\log N}{2N}}, \tag{6.3.8}$$

with probability at least $1 - 2N^{-t}$, where h is given in (6.3.6), $\underline{\mathcal{L}} \stackrel{\text{def}}{=} \inf \mathcal{L}(\Theta, \mathcal{C}_z^\varepsilon)$ and $\overline{\mathcal{L}} \stackrel{\text{def}}{=} \sup \mathcal{L}(\Theta, \mathcal{C}_z^\varepsilon)$.

(ii) Suppose that τ is a function of N, say τ_N, with $h(\tau_N) + \tau_N e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau_N}} \sqrt{\frac{\log N}{N}} \to 0$ as $N \to +\infty$, then

(a) for every $\theta \in \Theta$

$$g_{\tau_N,N}(\theta) \underset{N\to+\infty}{\longrightarrow} g(\theta) \quad \text{almost surely.}$$

(b) If moreover,

(H.4) *for every $z \in \mathcal{Z}$, $|\mathcal{L}(\theta, z) - \mathcal{L}(\theta', z)| \le L_\Theta(z)\,\|\theta - \theta'\|$, $\forall (\theta, \theta') \in \Theta^2$, with $0 \le L_\Theta \stackrel{\text{def}}{=} \sup L_\Theta(\mathcal{Z}) < +\infty$.*

Then, almost surely,

$$g_{\tau_N,N}(\theta) \underset{N\to+\infty}{\longrightarrow} g(\theta) \quad \text{for all} \quad \theta \in \Theta.$$

The proof can be found in Appendix 6.7.5.

Remark 6.3.8.

- Observe that since $\mathcal{L}$ is continuous by (H.1) and $\mathcal{C}_z^\varepsilon$ is compact, $\underline{\mathcal{L}}$ and $\overline{\mathcal{L}}$ are well-defined as minimal and maximal values as soon as Θ is compact.
- If $\mathcal{L}$ also verifies assumption (H.5), the bound (6.3.8) can be extended to hold uniformly over θ on any convex compact subset $\mathcal{C} \subset \Theta$. Indeed, it can be shown, combining our proof in 6.7.5 with a covering argument, that

$$\sup_{\theta\in\mathcal{C}} |g_{\tau,N}(\theta) - g(\theta)| \leq h(\tau) + \tau e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau}} \sqrt{\frac{(p+t)\log N}{2N}} + 8\frac{L_{\Theta,\mathcal{Z}} D_{\mathcal{C}}}{N},$$

  with probability at least $1 - 2N^{-t}$.
- It is important to realize that the claim of Theorem 6.3.7(ii)(b) is different (and stronger) than that of Theorem 6.3.7(ii)(a); observe the order of quantifiers. In the latter, the set of events of probability one on which Theorem 6.3.7(ii)(a) holds actually depends on θ, while it is does not for claim (b). On the other hand getting this uniform claim requires some additional regularity. This discussion is very important when it will come to showing the convergence result of our SGD algorithm.

For fixed τ, the convergence rate in (6.3.8) is nearly $O(N^{-1/2})$. But one has to keep in mind that we have not used any smoothness property of $\mathcal{L}(\theta,\cdot)$ and used a very simple (uniform) Monte Carlo integration. This could be possibly improved using the rich theory of Monte Carlo integration, see e.g., [29, 60], but probably at the price of a higher computation cost. Such improvements may also potentially necessitate more sophisticated deviation bounds in the proof instead of Hoeffding's inequality that we use here.

Note that the variance of the samples appears implicitly in (6.3.8) through the exponential term. One can alternatively use Bernstein's inequality instead of Hoeffding to show that

$$|g_{\tau,N}(\theta) - g_\tau(\theta)| = O\left(\tau \frac{\sigma}{\bar{S}} \sqrt{\frac{t\log N}{N}}\right)$$

where

$$\bar{S} = \frac{1}{\mu_{\mathcal{L}}(\mathcal{C}_z^\varepsilon)} \int_{\mathcal{C}_z^\varepsilon} e^{\frac{\mathcal{L}(\theta,z')}{\tau}} \mathrm{d}z' \quad \text{and} \quad \sigma^2 = \frac{1}{\mu_{\mathcal{L}}(\mathcal{C}_z^\varepsilon)} \int_{\mathcal{C}_z^\varepsilon} e^{2\frac{\mathcal{L}(\theta,z')}{\tau}} \mathrm{d}z' - \bar{S}^2.$$

The error bound (6.3.8) reveals an exponential dependence in τ, and thus for the right-hand side to vanish as $\tau \to 0^+$ and $N \to +\infty$, there is trade-off between N and τ to enforce the term $e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau}} \sqrt{\frac{t\log N}{2N}}$ to converge to 0. Taking $\tau = O\left(\frac{1}{\kappa \log N}\right)$, for any $\kappa > 0$ such that $\kappa(\overline{\mathcal{L}} - \underline{\mathcal{L}}) < 1/2$, the convergence rate in (6.3.8) is dominated by the first term $h(\tau)$ which scales as $O\left((\log N)^{-1}\right)$. This is obviously a slow convergence rate but reflects the difficulty of approximating the function g in (6.3.1).

6.3.4 Consistency of subgradient estimates

Equipped with the above results, a natural strategy now is to solve the PRO problem (6.3.1) by using $g_{\tau,N}$ in (6.3.7) as a (provably controlled) approximation of g. Towards this goal, we would like to apply a first-order scheme, typically (sub)gradient descent. Such a scheme will involve a first-order oracle on $g_{\tau,N}$, the gradient

$$\nabla g_{\tau,N}(\theta) = \sum_{k=1}^N \nabla_\theta \mathcal{L}(\theta, z_k') \frac{\exp\left(\frac{\mathcal{L}(\theta,z_k')}{\tau}\right)}{\sum_{j=1}^N \exp\left(\frac{\mathcal{L}(\theta,z_j')}{\tau}\right)}. \tag{6.3.9}$$

The natural question that arises is whether (6.3.9) behaves well as τ vanishes and is a consistent approximation of a Clarke subgradient of g (the latter being accessible via the formula (2.10.2) in Proposition 2.10.1 under mild assumptions). The result in Theorem 6.3.9 shows that this is indeed the case under appropriate assumptions that are stronger than those required for consistency of the (zero-th order oracle) function values established in Theorem 6.3.7.

To show some of our results, in particular Theorem 6.3.9(ii), we will need some regularity properties on the set of maximizers $\mathcal{M} \overset{\text{def}}{=} \operatorname{Argmax} \mathcal{L}(\theta, \mathcal{C}) = \operatorname{Argmax}_{z \in \mathcal{C}} \mathcal{L}(\theta, z)$ (we drop the dependence of $\mathcal{M}$ on θ to lighten notation). We say that a set $\mathcal{S} \subset \mathbb{R}^m$ is $\mathcal{C}^r$-stratifiable, for some integer $r \geq 1$, if there is a finite partition of $\mathcal{S}$ into disjoint $\mathcal{C}^r$ submanifolds $(\mathcal{M}_i)_{i \in I}$ of $\mathbb{R}^m$, called strata, with $m \geq \dim(\mathcal{M}_1) > \dim(\mathcal{M}_2) > \cdots > \dim(\mathcal{M}_{|I|}) \geq 0$.

Theorem 6.3.9. *Suppose that the assumptions of Proposition 6.3.1 hold and that*

(H.5) *$\mathcal{L}(\cdot, z')$ is differentiable with $\nabla_\theta \mathcal{L}(\cdot, \cdot)$ continuous in its both arguments (θ, z') and uniformly bounded by $L_{\Theta,\mathcal{Z}}$ on $\Theta \times \mathcal{Z}$.*

Then $\mathcal{L}$ and $g_{\tau,N}$ are continuously differentiable.

(i) If (H.3) also holds, then for every $\theta \in \Theta$ and any $t > 0$, with probability at least $1 - 2(p+1)N^{-t}$

$$\operatorname{dist}\left(\nabla g_{\tau,N}(\theta), \partial^C g(\theta)\right) \leq \max(1, 2L_{\Theta,\mathcal{Z}}\sqrt{p}) e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau}} \sqrt{\frac{t \log N}{2N}} + o_\tau(1). \tag{6.3.10}$$

where

$$\partial^C g(\theta) = \left\{ \int_{\mathcal{C}_z^\varepsilon} \nabla_\theta \mathcal{L}(\theta, z') \mathrm{d}\mu(z') : \ \mu \in \mathcal{P}\Big(\operatorname{Argmax} \mathcal{L}(\theta, \mathcal{C}_z^\varepsilon) \Big) \right\}. \tag{6.3.11}$$

(ii) Suppose in addition that

(H.6) *$\mathcal{L}(\theta, .)$ is $\mathcal{C}^3$ on an open set containing $\mathcal{C}_z^\varepsilon$ with Hölder continuous third-order derivative;*

(H.7) *for $r \geq 3$*

(a) $\mathcal{M}$ is $\mathcal{C}^r$-stratifiable with closed strata;

(b) for each $i \in I$, the Hessian $\nabla^2_{z'} \mathcal{L}(\theta, z')$ is negative semidefinite for any $z' \in \mathcal{M}_i$ with constant rank $m - \dim(\mathcal{M}_i)$.

Then for every $\theta \in \Theta$ and any $t > 0$, with probability at least $1 - 2(p+1)N^{-t}$

$$\operatorname{dist}\left(\nabla g_{\tau,N}(\theta), \partial^C g(\theta)\right) \leq \|\nabla g_{\tau,N}(\theta) - \eta(\theta)\| \leq \max(1, 2L_{\Theta,\mathcal{Z}}\sqrt{p}) e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau}} \sqrt{\frac{t \log N}{2N}} + o_\tau(1), \tag{6.3.12}$$

where $\eta(\theta) \overset{\text{def}}{=} \int_{\mathcal{M}_1} \nabla_\theta \mathcal{L}(\theta, z') \mathrm{d}\mu(z') \subset \partial^C g(\theta)$ and $\mu \in \mathcal{P}(\mathcal{M}_1)$.

(iii) Under the assumptions of either statement (i) or (ii), if τ is a function of N, say τ_N, with $\tau_N \to 0$ and $e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau_N}} \sqrt{\frac{\log N}{N}} \to 0$ as $N \to +\infty$, then

(a) for every $\theta \in \Theta$

$$\operatorname{dist}\left(\nabla g_{\tau_N,N}(\theta), \partial^C g(\theta)\right) \underset{N \to +\infty}{\longrightarrow} 0 \quad \textit{almost surely.}$$

(b) Let Ξ be any closed convex subset of Θ. If moreover $\operatorname{Argmax} \mathcal{L}(\theta, \mathcal{C}_\varepsilon^z)$ is a singleton for each $\theta \in \Xi$, then almost surely,

$$\operatorname{dist}\left(\nabla g_{\tau_N,N}(\theta), \partial^C g(\theta)\right) \underset{N \to +\infty}{\longrightarrow} 0 \quad \textit{for all} \quad \theta \in \Xi.$$

The proof of this result follows from by a simple triangle inequality and using the following two lemmas whose proofs are deferred to Appendix 6.7.6.

Lemma 6.3.10. *(i) Under the assumptions of Theorem 6.3.9(i), we have*

$$\operatorname{dist}\left(\nabla g_\tau(\theta), \partial^C g(\theta)\right) \to 0 \quad \textit{as} \quad \tau \to 0^+. \tag{6.3.13}$$

(ii) Under the assumptions of Theorem 6.3.9(ii),

$$\nabla g_\tau(\theta) \underset{\tau \to 0^+}{\longrightarrow} \eta(\theta) \overset{\text{def}}{=} \int_{\mathcal{M}_1} \nabla_\theta \mathcal{L}(\theta, z) \mathrm{d}\mu(z) \subset \partial^C g(\theta),$$

where $\mu \in \mathcal{P}(\mathcal{M}_1)$.

Lemma 6.3.11. *Suppose that the assumptions of Proposition 6.3.1 hold, and that $\mathcal{L}(\cdot, z')$ is differentiable with $\nabla_\theta \mathcal{L}(\theta, z')$ uniformly bounded by $L_{\Theta,\mathcal{Z}}$ on $\Theta \times \mathcal{Z}$.*

(i) For any $t > 0$

$$\|\nabla g_\tau(\theta) - \nabla g_{\tau,N}(\theta)\| \leq \max(1, 2L_{\Theta,\mathcal{Z}}\sqrt{p}) e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau}} \sqrt{\frac{t \log N}{2N}} \tag{6.3.14}$$

with probability at least $1 - 2(p+1)N^{-t}$.

(ii) Suppose that τ is a function of N, say τ_N, with $\tau_N e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau_N}} \sqrt{\frac{\log N}{N}} \to 0$ as $N \to +\infty$, then

(a) for every $\theta \in \Theta$

$$\nabla g_{\tau_N}(\theta) - \nabla g_{\tau_N,N}(\theta) \underset{N\to+\infty}{\longrightarrow} 0 \quad \textit{almost surely.}$$

(b) Let Ξ be any closed convex subset of Θ. If moreover $\operatorname{Argmax} \mathcal{L}(\theta, \mathcal{C}_\varepsilon^z)$ is a singleton for each $\theta \in \Xi$, then almost surely,

$$\nabla g_{\tau_N}(\theta) - \nabla g_{\tau_N,N}(\theta) \underset{N\to+\infty}{\longrightarrow} 0 \quad \textit{for all} \quad \theta \in \Xi.$$

To show Lemma 6.3.10, one observes that under our assumptions,

$$\nabla g_\tau(\theta) = \int_{\mathcal{C}_z^\varepsilon} \nabla_\theta \mathcal{L}(\theta, z') \frac{\exp\left(\frac{\mathcal{L}(\theta,z')}{\tau}\right)}{\int_{\mathcal{C}_z^\varepsilon} \exp\left(\frac{\mathcal{L}(\theta,v)}{\tau}\right) \mathrm{d}v} \mathrm{d}z', \tag{6.3.15}$$

which is an expectation with respect to a Gibbs measure indexed by τ (and θ) and supported on $\mathcal{C}_z^\varepsilon$. In view of the rule (2.10.2), the proof will then amount to showing that as $\tau \to 0^+$, the familty of such Gibbs measures has all its cluster points in the narrow topology (equivalent to the weak–* topology) in $\mathcal{P}\Big(\operatorname{Argmax} \mathcal{L}(\theta, \mathcal{C}_z^\varepsilon)\Big)$ (first claim of Lemma 6.3.10), or that it even converges in the weak–* topology to a measure supported on $\operatorname{Argmax}_{z\in\mathcal{C}} \mathcal{L}(\theta, z)$ (second claim of Lemma 6.3.10)[3].

Clearly, Theorem 6.3.9 tells us that, with high probability, $\nabla g_{\tau,N}(\theta)$ is at most within a ball around the Clarke subdifferential of g at θ. The result also quantifies its radius and how it vanishes with N and τ, and shows the influence of p, the dimension of Θ. Arguing as above, this radius vanishes as $N \to +\infty$ by taking $\tau = O\left(\frac{1}{\kappa \log N}\right)$, for any κ such that $\kappa(\overline{\mathcal{L}} - \underline{\mathcal{L}}) \in]0, 1/2 - \alpha]$, $\alpha \in]0, 1/2[$. The convergence rate of the first term in (6.3.10) or (6.3.12) is then nearly $O\left(N^{-\alpha}\right)$ (up to logarithmic factors). As far as the $o_\tau(1)$ term is concerned, we do not have any quantitative estimate for the corresponding rate in the case of (6.3.10). For (6.3.12), a close inspection of the proof Lemma 6.3.10(ii) reveals that the convergence rate in τ is at least

$$O\left(\tau^{-\frac{m-m_1}{2}} e^{-\frac{\kappa}{\tau}} + \sum_{i>1} \tau^{\frac{m_1-m_i}{2}} + \tau^{1/2}\right) = O\left(\tau^{1/2}\right).$$

Remark 6.3.12. It is worth noting that the statement of Lemma 6.3.10(i), hence Theorem 6.3.9(i), requires less stringent assumptions than claim (i). However, it only ensures subsequential convergence of the gradient whose cluster points are Clarke subgradients of g. On the other hand, Lemma 6.3.10(ii) not only shows global convergence of the gradient but also gives the precise form of the limit Clarke subgradient, and characterizes the corresponding measure. This in turn necessitates the extra regularity assumptions above.

Remark 6.3.13. Similarly to the discussion in Remark 6.3.8, we again need a little bit more to ensure almost sure convergence simultaneously for all θ. This observation is very important when it will come to using almost sure unbiasedness of the (sub)gradient estimate in our SGD algorithm, which in turn will be crucial to prove our convergence result in Theorem 6.4.2.

[3] This is reminiscent of works on simulated annealing where τ is the temperature parameter; see e.g. [100, 50]. Our context is however different and in particular, $\mathcal{C}_z^\varepsilon$ is not the whole space nor it is a finite set nor a compact submanifold.

Remark 6.3.14. The assumption that the partition of $\mathcal{M}$ is disjoint can be removed by assuming in addition that each intersecting pair of submanifolds $(\mathcal{M}_i, \mathcal{M}_j)$, $i \neq j$, do so transversely [116, Theorem 6.30]. Therefore, $\mathcal{M}_i \cap \mathcal{M}_j$ is also a submanifold whose dimension strictly smaller than that of $\mathcal{M}_i$ and $\mathcal{M}_j$. The main change in our proof will lie in subtracting the contribution of these intersections in (6.7.5), and then use that their dimensions are strictly smaller than that of the largest submanifold.

6.4 Robust optimization algorithm via SGD

We are now ready to describe our algorithmic framework to solve the PRO problem (6.3.1) based on stochastic (sub)gradient descent. To make the presentation easier, we assume that $\Theta = \mathbb{R}^p$ though our algorithmic framework can be extended to the case where Θ is a convex closed set by including a projection step onto Θ (see e.g., [57, 112] in different settings). Moreover, as considered in most applications, we take in this section

$$\mathsf{c}(z, z') = \varphi(z - z'),$$

where $\varphi : \mathbb{R}^m \to \mathbb{R}_+$ is a continuous coercive function. It is also even-symmetric and $\varphi(0) = 0$ by assumption (H.2). We now consider the standard setting where ρ_0 in (6.3.1) is the empirical measure on $\mathcal{Z}$, hence leading to the finite sum minimization problem

$$\min_{\theta\in\Theta} \left\{ G(\theta) \overset{\text{def}}{=} \frac{1}{M}\sum_{i=1}^{M} g_i(\theta) \right\} \quad \text{where} \quad g_i(\theta) \overset{\text{def}}{=} \max_{u\in\mathcal{C}^\varepsilon} \mathcal{L}(\theta, z_i + u), \tag{6.4.1}$$

where $\mathcal{C}^\varepsilon \overset{\text{def}}{=} \{u \in \mathbb{R}^m : \varphi(u) \leq \varepsilon\}$ is obviously a nonempty compact and full dimensional set by the assumptions on φ. It is worth observing that convexity of φ is not needed in this section.

6.4.1 Without smoothing

As revealed by Proposition 2.10.1, the Clarke subdifferential has only inclusion rules under finite sum and pointwise maximization. Thus, if in addition to (H.1), is $\mathcal{L}(\cdot, z_i + u)$ is assumed locally Lipschitz continuous for each (z_i, u), (2.10.1) gives us the inclusion

$$\partial^C G(\theta) = \partial^C \left(\frac{1}{M}\sum_{i=1}^{M} g_i(\theta) \right) \subset \frac{1}{M}\sum_{i=1}^{M} \partial^C g_i(\theta) \subset \frac{1}{M}\sum_{i=1}^{M} \mathcal{D}_i(\theta), \tag{6.4.2}$$

where

$$\mathcal{D}_i(\theta) = \operatorname{conv}\left\{ \lim_{k\to\infty} \nabla_\theta \mathcal{L}(\theta_k, z_i + u_k) : \theta_k \to \theta, \theta_k \in S, u_k \in \mathcal{C}^\varepsilon, \mathcal{L}(\theta, z_i + u_k) \to g_i(\theta) \right\}.$$

Remark 6.4.1. The inclusion above, for instance the one of the sum rule, is strict in many situations of interest in applications. For the sum rule, one may consider other generalized derivatives or even other (but closely related) fields than the Clarke subdifferential, e.g. the conservative fields proposed in [25, 24]. These fields enjoy nice sum and chain rules and coincide with the Clarke subdifferential almost everywhere. After a first version of this work was posted, we became aware of the recent work of [144] who established that conservative fields deriving from definable potentials also have a calculus rule under pointwise maximization involving again an inner maximization oracle. Thus in the rest of this subsection, we could just use the formula of that conservative field instead of $\mathcal{D}_i$ and our convergence result will remain true. The advantage of using conservative fields is their rich calculus, including the sum and chain rules. Nevertheless, we will not elaborate more on this to keep the presentation simpler and since anyway, this would necessitate to have the inner maximization oracle.

We are now naturally led to consider the set of critical points:

$$\text{crit–}G \overset{\text{def}}{=} \left(\frac{1}{M}\sum_{i=1}^{M} \mathcal{D}_i \right)^{-1}(0). \tag{6.4.3}$$

Clearly, this set is larger than the set of critical points $(\partial^C G)^{-1}(0)$.

Let $(B_k)_{k\in\mathbb{N}}$ be a sequence of nonempty mini-batches sampled independently, uniformly at random in $[M]$. We can then devise the following iteration

$$\theta_{k+1} = \theta_k - \gamma_k d_k, \quad \text{where} \quad d_k \in \frac{1}{|B_k|}\sum_{i\in B_k} \mathcal{D}_i(\theta_k), \tag{6.4.4}$$

and $(\gamma_k)_{k\in\mathbb{N}}$ is a positive step sequence decaying at an appropriate rate. A natural question now is whether the sequence $(\theta_k)_{k\in\mathbb{N}}$ in (6.4.4) enjoys some convergence guarantees to the set of critical points in crit–G. For this, we will rely on the stochastic approximation method for differential inclusions with compact and convex-valued operators developed in [14], and used recently in [25, 33]. The idea is to view $(\theta_k)_{k\in\mathbb{N}}$ as a discrete-time stochastic process which asymptotically behaves as the (absolutely continuous) solution trajectories of the differential inclusion

$$\begin{cases} 0 \in \dot\theta(t) + \frac{1}{M}\sum_{i=1}^M \mathcal{D}_i(\theta(t)) & \text{for almost every } t \in \mathbb{R}, \\ \theta(0) = \theta_0, \end{cases} \tag{6.4.5}$$

whose stationary solutions are the critical points in (6.4.3). A key argument to invoke the results of [14], is to build an appropriate Lyapunov function and show that the function G is path differentiable, that is, it obeys the chain rule

$$\frac{\mathrm{d}}{\mathrm{d}t} G(\theta(t)) = \langle \dot\theta(t), v\rangle, \quad \forall v \in \frac{1}{M}\sum_{i=1}^M \mathcal{D}_i(\theta(t)). \tag{6.4.6}$$

While path differentiability can be shown (see later) for the finite sum under tameness/definability for the Clarke subdifferential, it seems very difficult to deal with the pointwise maximization and to prove path differentiability of g_i with the field $\mathcal{D}_i$.

The situation however changes if we work under (a part of) assumption (H.5), in which case (6.3.11) applies and (6.4.2) becomes

$$\partial^C G(\theta) = \partial^C\left(\frac{1}{M}\sum_{i=1}^M g_i(\theta)\right) \subset \frac{1}{M}\sum_{i=1}^M \partial^C g_i(\theta), \tag{6.4.7}$$

where

$$\partial^C g_i(\theta) = \left\{ \int_{\mathcal{C}^\varepsilon} \nabla_\theta \mathcal{L}(\theta, z_i + u)\mathrm{d}\mu(u) : \ \mu \in \mathcal{P}\Big(\operatorname{Argmax} \mathcal{L}(\theta, z_i + \mathcal{C}^\varepsilon)\Big)\right\}. \tag{6.4.8}$$

This gives the scheme in Algorithm 4.

Algorithm 4: SGD for PRO without smoothing.

Input: Step-sizes $(\gamma_k)_{k\in\mathbb{N}}$;
Input: Initialization θ_0;
for $k = 0, \dots$ **do**
  Draw independently uniformly at random a mini-batch $B_k \subset [M]$;
  for $i \in B_k$ **do**
    Solve $\bar u_i \in \operatorname{Argmax}_{u\in\mathcal{C}^\varepsilon} \mathcal{L}(\theta_k, z_i + u)$.
  $d_k = \frac{1}{|B_k|}\sum_{i\in B_k} \nabla_\theta \mathcal{L}(\theta_k, z_i + \bar u_i)$;
  $\theta_{k+1} = \theta_k - \gamma_k d_k$.

This algorithm enjoys the following guarantees.

Theorem 6.4.2. *Assume that (H.1) holds, that $\mathcal{L}(\cdot, z_i + u)$ is locally Lipschitz continuous for each $(z_i, u) \in \mathcal{Z}\times\mathcal{C}^\varepsilon$, and that $\mathcal{L}(\cdot, z)$ is differentiable with $\nabla_\theta\mathcal{L}(\theta, z)$ continuous in (θ, z). Suppose moreover that φ and $\mathcal{L}$ are definable, and that the step-sizes satisfy $\sum_{k\in\mathbb{N}}\gamma_k = +\infty$ and $\gamma_k = o\left(\frac{1}{\log k}\right)$. Consider the sequence $(\theta_k)_{k\in\mathbb{N}}$ generated by Algorithm 4 and suppose that there exists a constant $C > 0$ such that $\sup_{k\in\mathbb{N}} \|\theta_k\| \le C$ almost*

surely. Then, almost surely, the set of cluster points of $(\theta_k)_{k\in\mathbb{N}}$ *belong to* crit–$G = \left(\frac{1}{M}\sum_{i=1}^M \partial^C g_i\right)^{-1}(0)$. *Moreover* $(G(\theta_k))_{k\in\mathbb{N}}$ *converges and* G *is constant on* crit–G.

See Appendix 6.7.7 for the proof.

A caveat of Algorithm 4 is that one has to solve the inner maximization problems to compute the subgradient approximation d_k as dictated by (6.4.8). We recall that this can be computationally challenging in general and iterative schemes do not come with any guarantees unless stringent assumptions are imposed on $\mathcal{L}$. To avoid this, one can appeal to the smoothing strategy as we develop now.

6.4.2 With smoothing

In this section, we work under the assumptions of Theorem 6.3.9 and capitalize on the results there. This gives the scheme summarized in Algorithm 5.

Algorithm 5: SGD for PRO with smoothing.

Input: Step-sizes $(\gamma_k)_{k\in\mathbb{N}}$; number of integration points $(N_k)_{k\in\mathbb{N}}$; smoothing parameters $(\tau_k)_{k\in\mathbb{N}}$;
Input: Initialization θ_0;
for $k = 0, \dots$ **do**
Draw independently uniformly at random a mini-batch $B_k \subset [M]$;
Draw N_k samples $(u_j)_{j\in[N_k]}$ independently uniformly at random in $\mathcal{C}^\varepsilon$;
$d_k = \frac{1}{|B_k|}\sum_{i\in B_k} \nabla g^i_{\tau_k,N_k}(\theta_k)$, with $\nabla g^i_{\tau_k,N_k}(\theta_k) \overset{\text{def}}{=} \sum_{j=1}^{N_k} \nabla_\theta \mathcal{L}(\theta, z_i + u_j) \frac{e^{\frac{\mathcal{L}(\theta,z_i+u_j)}{\tau_k}}}{\sum_{l=1}^{N_k} e^{\frac{\mathcal{L}(\theta,z_i+u_l)}{\tau_k}}}$;
$\theta_{k+1} = \theta_k - \gamma_k d_k$.

The direction d_k in Algorithm 5 can also be written as

$$d_k = v_k + e_k + \zeta_k,$$

where

- $v_k = \frac{1}{M}\sum_{i=1}^M \mathrm{P}_{\partial^C g_i(\theta_k)}(\nabla g^i_{\tau_k,N_k}(\theta_k))$;
- $e_k = \frac{1}{|B_k|}\sum_{i\in B_k}\left(\nabla g^i_{\tau_k,N_k}(\theta_k) - \mathrm{P}_{\partial^C g_i(\theta_k)}(\nabla g^i_{\tau_k,N_k}(\theta_k))\right)$;
- $\zeta_k = \frac{1}{|B_k|}\sum_{i\in B_k} \mathrm{P}_{\partial^C g_i(\theta_k)}(\nabla g^i_{\tau_k,N_k}(\theta_k)) - \frac{1}{M}\sum_{l=1}^M \mathrm{P}_{\partial^C g_l(\theta_k)}(\nabla g^l_{\tau_k,N_k}(\theta_k))$;

where all projectors above are well-defined since the Clarke subdifferential is closed (in fact compact) and convex-valued. We then have the following convergence result.

Theorem 6.4.3. *Assume that the assumptions of Theorem 6.3.9(ii)(b) hold. Suppose moreover that* φ *and* $\mathcal{L}$ *are definable, that the step-sizes satisfy* $\sum_{k\in\mathbb{N}}\gamma_k = +\infty$ *and* $\gamma_k = o\left(\frac{1}{\log k}\right)$, *and that* $(\tau_k, N_k)_{k\in\mathbb{N}}$ *are such that* $\tau_k \to 0$ *and* $e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau_k}}\sqrt{\frac{\log N_k}{N_k}} \to 0$ *as* $k \to +\infty$, *and* $\sum_{k\in\mathbb{N}} N_k^{-t} < +\infty$ *for some* $t > 0$. *Consider the sequence* $(\theta_k)_{k\in\mathbb{N}}$ *generated by Algorithm 5 and suppose that there exists a constant* $C > 0$ *such that* $\sup_{k\in\mathbb{N}}\|\theta_k\| \le C$ *almost surely. Then, almost surely, the set of cluster points of* $(\theta_k)_{k\in\mathbb{N}}$ *belong to* crit–$G = \left(\frac{1}{M}\sum_{i=1}^M \partial^C g_i\right)^{-1}(0)$. *Moreover* $(G(\theta_k))_{k\in\mathbb{N}}$ *converges and* G *is constant on* crit–G.

See Appendix 6.7.8 for the proof.

From the discussion after Lemma 6.3.10, the vanishing assumption on the bias holds provided one chooses N_k an increasing function of k, and τ_k decreasing as $O\left(\frac{1}{\kappa\log N_k}\right)$ for $\kappa > 0$ small enough. The algorithm remains simple and effective at making the learning robust, the initial value of the parameters N and T do not matter as long as we increase and decrease them respectively. However in practice it is better to select their initial

values considering the dimension of the problem. We suggest reading the following sources for considerations on Monte Carlo integration or Quasi Monte Carlo [29] [60]. Other algorithms can be used instead of SGD provided they are proven to converge with appropriate generalized subgradients (see [160] for instance).

6.5 Numerical results

6.5.1 Dataset, model and metrics

The following experiments were all performed on the Avila dataset, introduced in [59]. This dataset is representative of a classification task – writer identification in medieval manuscripts through page layout features – with 8 input features and 12 classes. We centred and normalized the input features in a preprocessing step. The label distribution is uneven for the twelves classes (A:41%, B:0.048%, C:0.99%, D:3.4%, E:10%, F:19%, G:4.3%, H:5.0%, I:8.0%, W:0.43%, X:5.0%, Y:2.6%), with a class A that is far more present than the other labels. This dataset was selected for its moderate input dimension to keep Monte Carlo sampling needed for computing (6.3.7) reasonable, and because it has unevenly distributed labels which will help highlighting the compromise between generalization and robust learning. In these experiments, we build a 3 layer MLP network $f : \Theta \times \mathbb{R}^8 \to \mathbb{R}^{12}$ with two hidden layers of 200 neurons each and an output layer, resulting in $p = 44000$ parameter vector θ; i.e., $\Theta \subset \mathbb{R}^{44000}$. To comply with our regularity assumptions, we used the ELU activation function [44]. The loss used for training the model is the cross-entropy loss after a softmax step on the network output, i.e., for a training example $z = (x, y)$, where $x \in \mathbb{R}^8$ is a feature vector and $y \in [12]$ is the (true) label, the loss is given by

$$\mathcal{L}(\theta, z) = -f(\theta, x)_y + \log\left(\sum_{j=1}^{12} e^{f(\theta,x)_j}\right),$$

where the subscript here stands for the corresponding entry of a vector. Note that this loss also verifies our assumptions.

All experiments were carried out with the same number of epochs, batch size and therefore the same number of updates (see Table 6.1 for details).

For comparison purposes, we trained the MLP network in 3 different ways: with a vanilla (non-robust) training, an adversarial training using PGD as a heuristic to solve the inner maximization problem [126], and robust training using Algorithm 5. Our aim is to show that we can provably train a robust model using our algorithm, while being competitive with current state of the art procedures for robustifying neural networks such as adversarial training, for different values of the perturbation radius ε. Throughout this section, we take $\mathcal{C}^\varepsilon = \mathbb{B}_\varepsilon^q(0)$ with typically $q = +\infty$ (see Table 6.1).

For a pair $(x, y) \in \mathbb{R}^8 \times [12]$, let $F_\theta(x) = \mathrm{Argmax}_{j\in[12]}(f(\theta, x))_j$ be the predicted label. We denote $S : \mathbb{R}^2 \to \{0, 1\}$ the mapping that returns 1 if its arguments are equal and 0 otherwise. In the numerical results, we will report three different performance metrics.

- **Test Accuracy**: We define it as the accuracy on a test dataset $\{(x_i, y_i) : i \in [N_{\text{test}}]\}$:

$$\text{Test Accuracy} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} S(F_\theta(x_i), y_i).$$

- **Adversarial Accuracy**: This metric is meant to represent the accuracy on an adversarial set given by applying a white-box PGD attack [126]. The attack depends on the loss function $\mathcal{L}$, a ball $\mathbb{B}_\varepsilon^q(0)$ in which the attack is constrained, and a tuple (x_i, y_i) of data points to be attacked. It reads

$$\text{Adversarial Accuracy} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} S(F_\theta(\hat{x}_i), y_i) \quad \text{where} \quad \hat{x}_i = \text{PGD}(\mathcal{L}, \mathbb{B}_\varepsilon^q(0), x_i, y_i).$$

- **Worst-case Robustness Accuracy**: This is defined as the worst-case accuracy when the data points undergo perturbations within a ball $\mathbb{B}_\varepsilon^q(0)$ (the same ball as for the PGD attack). More precisely, recall that $\mu_\mathcal{U}$ is the uniform measure on $\mathbb{B}_\varepsilon^q(0)$. For N perturbation samples, this metric is defined as

$$\text{Robust Accuracy} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \min_{(u_j)_{j=1}^N, u_j \sim_{\text{i.i.d.}} \mu_\mathcal{U}} S(F_\theta(x_i + u_j), y_i).$$

  For large enough number of samples N, this metric is intended to show robustness as promoted when solving the PRO problem during the training.

We added to these metrics estimates of an upper-bound of the Lipschitz constant of the network. To this end, we chose the LipSDP method [71] which is efficient, accurate and adequate for the size and structure of the networks used.

These metrics are to be evaluated on the three kinds of training, for different values of ε. We made 100 runs for each configuration with different initializations to account for statistical variability. The training and test sets have been sampled once for all runs to ensure that the variance of the results would come only from differences in the initialization of the model and dynamics of the optimization. The plots we will display show the median value and the quantiles at 0.1, 0.25, 0.75 and 0.9.

Remark 6.5.1. Ultimately, the ideal metric for quantifying robustness is the population intractable robust 0-1 gain. It is closely linked to the adversarial frequency [9]

$$\mathbb{E}_{(x,y)\sim\rho_0}\left[\min_{u\in\mathbb{B}_\varepsilon^q(x)} S(F_\theta(u), y)\right].$$

The robust accuracy attempts to estimate this quantity by drawing random samples in both the min and expectation. This however may suffer the curse of dimensionality when estimating the min value. The adversarial accuracy uses a heuristic in the form of an adversarial attack obtained by PGD, but the latter does not enjoy any convergence guarantee to the minimal value. The robust accuracy metric appears as a better representative metric of the robust behavior of a model with the proviso that N is large.

6.5.2 Dependence of smoothing factor and number of samples for robust optimization

Following Theorem 6.3.9, we advocated that N, the number of samples used by Monte Carlo integration must increase during the execution of Algorithm 5 and that $\tau = O(\frac{1}{\kappa \log N})$, the regularization parameter, goes to zero. However in practice, making the sampling infinite is impossible due to hardware memory limitations. Some workarounds can be found to extend the batch size, however they drastically increase the computation time. Therefore we need to check experimentally how the algorithm behaves with different values of τ and N.

The sampling for the robust training is performed in $\mathbb{B}_{0.05}^\infty(0)$ for various values of τ and N. The inner maximization problem is assessed and averaged on the test base resorting to a fixed sampling of the perturbation set. This sampling is chosen larger (10^6) than the biggest value explored in the set so as to maximize the chance that the metric is evaluated accurately when testing.

Figure 6.1 shows the value of the loss for different values of τ and N, after a given number of iterations. The observed behavior is indeed the expected one with respect to the sampling: the more we shrink the temperature and increase the sampling the smaller the error on the robust problem we have.

6.5.3 Robustness to noise

In this section, we present a few experiments illustrating the robustness to noise a model trained with our method, and provide comparisons with alternative methods.

For these experiments, the adversarial training was performed in $\mathbb{B}_\varepsilon^\infty(0)$ and so was the sampling for the robust training: we kept the same fixed sampling of $N = 5.10^5$ points and a fixed temperature of $\tau = 10^{-4}$.

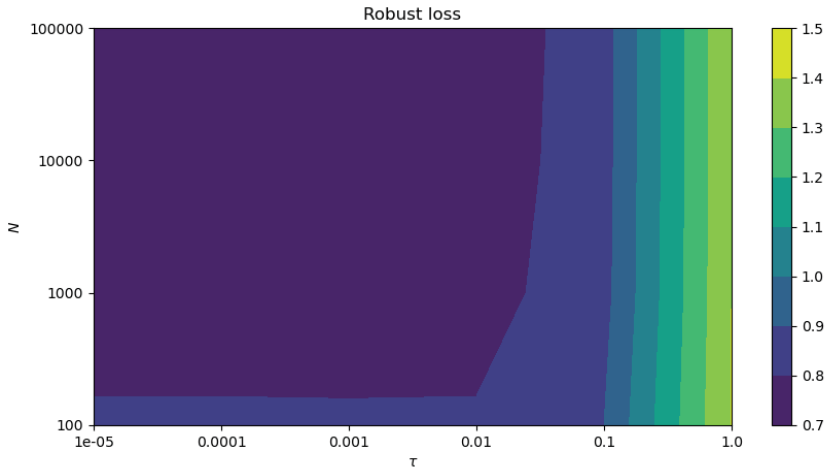

Figure 6.1: Influence of τ (x axis) and N (y axis) on the robust loss on test set (color scale). The darker the better.

Fixing the sampling to the highest possible value ensures the smallest error possible on the estimation of the robust loss. We give in Section 6.7.8 more information about the parameters for these experiments.

We plot the evolution of the test accuracy (Figure 6.2), adversarial accuracy (Figure 6.3) and worst case accuracy (Figure 6.4) against the radius of robustness with median and quantiles. On top of this we also present the evolution of upper bounds of the Lipschitz constant of the learned network estimated using LipSDP [71] (Figure 6.5) and SeqLip [183] (Figure 6.6): the former is guaranteed but overestimates the upper bound, the later is not guaranteed but is closer to the real value.

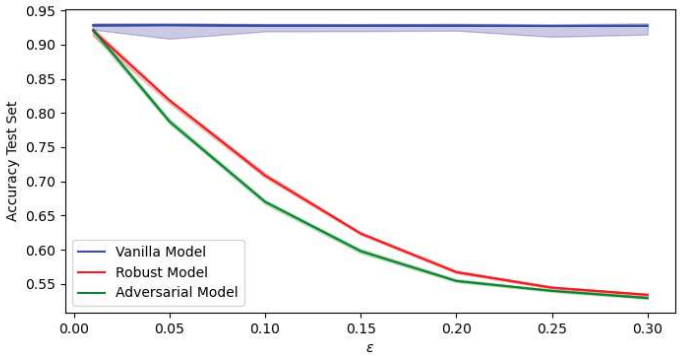

Figure 6.2: Test accuracy of the three trainings on Avila dataset: vanilla (blue), adversarial (green), robust (red) with median (plain line) and 10% and 90% quantiles for 50 trials.

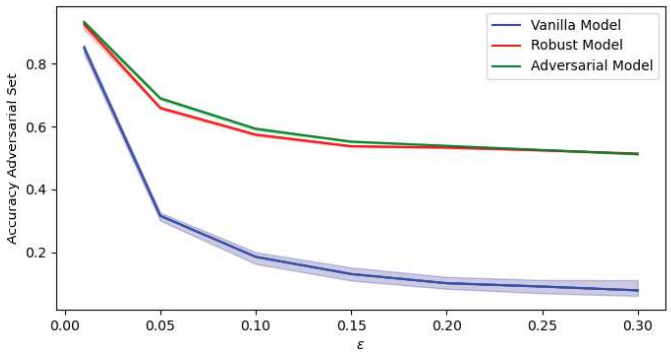

Figure 6.3: Adversarial accuracy of three trainings on Avila dataset: vanilla (blue), adversarial (green), robust (red) with median (plain line) and 10% and 90% quantiles for 50 trials.

For the sake of completeness, and due to the fact that the loss values do not account for the complexity of the distribution of the predictions, we also provide three confusion matrices (i.e., percentage of predicted labels per each class in the test set). We normalized the confusion matrices column-wise as it allows to assess which percentage of the real label was split to which predicted label. The confusion matrices were computed for the vanilla training, for the robust training with $\varepsilon = 0.3$ and the adversarial training with the same value of ε. The results are displayed in Figure 6.7.

On all performance metric (Figures 6.2,6.3 and 6.4) the behavior of our robust training is competitive compared to the popular PGD-based adversarial training, and differences are in general small. We note that robust training is better on the test accuracy for moderate perturbations (Figure 6.2), while adversarial training appears to be slightly better in terms of adversarial accuracy (Figure 6.3). We note, as expected, a decrease in accuracy on the test dataset as the perturbation radius ε increases, but a better tolerance to perturbations/attacks since

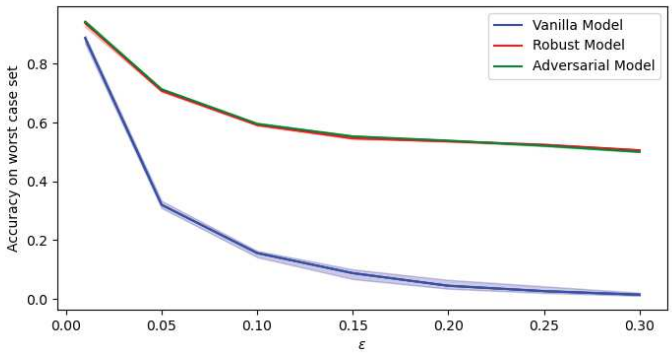

Figure 6.4: Worst-case robustness accuracy of three trainings on Avila dataset: vanilla (blue), adversarial (green), robust (red) with median (plain line) and 10% and 90% quantiles for 50 trials.

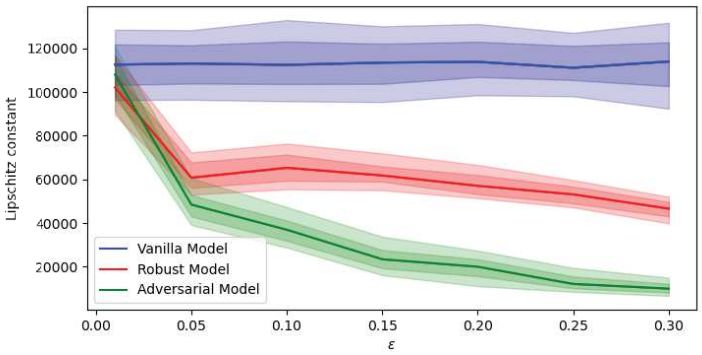

Figure 6.5: Lipschitz constant upper bound of the learned network of the three training methods on Avila dataset computed using LipSDP: vanilla (blue), adversarial (green), robust (red) with median (plain line) and 10%, 25%, 75% and 90% quantiles for 50 trials.

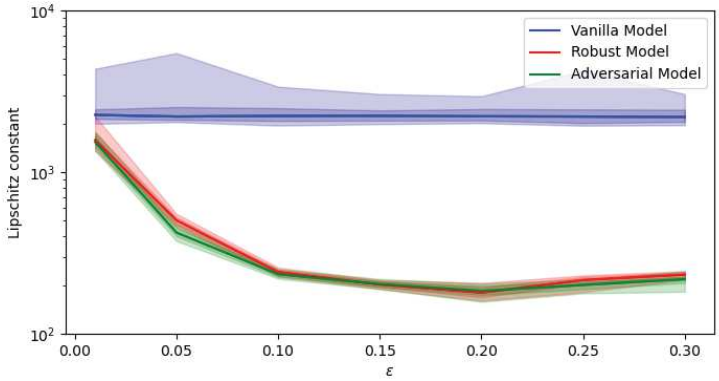

Figure 6.6: Lipschitz constant upper bound of the learned network of the three training methods on Avila dataset computed using SeqLip: vanilla (blue), adversarial (green), robust (red) with median (plain line) and 10%, 25%, 75% and 90% quantiles for 50 trials.

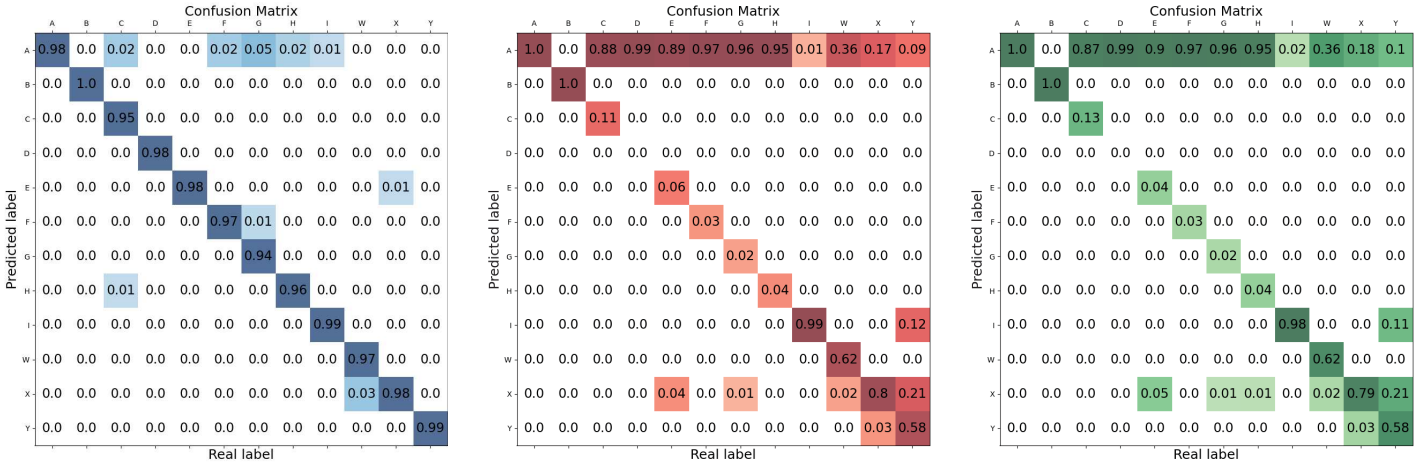

Figure 6.7: Confusion matrix for the vanilla training (left), for the robust training ($\varepsilon = 0.3$) (middle) and for the adversarial training ($\varepsilon = 0.3$) (right). All matrices are normalized column-wise to display the percentage of predicted labels per each class in the test set.

increasing the perturbation radius ε, we make the learned model stable to larger adversarial attacks. This is symptomatic of the trade-off between robustness and generalization, see e.g., [181, 153, 202, 61]. Note also that the variability across runs is small (and highest for the vanilla training), confirming that the performance of all the trainings is reproducible from run to run and for different initializations.

The decrease in accuracy of the robust and adversarial training on the test dataset can be further understood when considering the confusion matrices in Figure 6.7. Indeed, we see that these two robust learning methods aggregated the labels that were close to label A, namely labels C, D, E, F, G and H: this is due to class A being overly represented as it leans toward aggregating samples that are close. The labelling performed by the robustly trained networks on a test point assigns the label that has the greatest mass in the ε perturbation ball around the test point. This clearly means that some feature vectors originally in classes C, D, E, F, G and H are in fact within a ball $\mathbb{B}_\varepsilon^q(0)$ around those of label A. Class B has a very limited number of samples (5, only 0.048% of the dataset), however, it appears to be far enough from the other classes not to be confused with them.

As far as Figures 6.5 and 6.6 are concerned, one can clearly (and unsurprisingly) see that the adversarial and robust training methods tend to monotonically reduce the Lipschitz constant of the learned networks as ε increases. The fact that, on an adversarial set, the vanilla model performs poorly compared to the adversarial one is expected (Figure 6.3), but the performance of the robust model is similar to the adversarial one on this dataset. Of course, adversarial attacks favor those models that have already been trained in an adversarial setting. Our experiments confirm that our approach via smoothing and Monte Carlo sampling on the perturbation set provably converges to a critical point of the PRO problem. This is in contrast to PGD-based adversarial training for which convergence guarantees are not available in such general setting. The main drawback of the robust training remains its computational cost compared to adversarial training. For instance, under the hyperparameters in Appendix 6.7.8 with $\varepsilon = 0.3$, both training methods were performed on the same GPU A100 with the same number of epochs and updates. Adversarial training took 47 min whereas robust training took 11 hours. Adversarial training remains a very effective method for solving empirically the PRO problem. Let us stress that, despite this increased cost, robust training can be more effectively parallelized as it only requires one expensive forward pass and one expensive backward pass whereas the adversarial training requires multiple iterations of both.

6.6 Conclusions

Solving numerically the DRO problem beyond stringent assumptions on the loss remains a challenging open problem. Here, we have shown that the DRO problem with sufficiently small error can be approached with a PRO problem. In order to solve the latter, we designed SGD-type algorithms hinging on smoothing of the inner maximization problem and Monte Carlo sampling. Our approach is one of the few that enjoys provable convergence guarantees at the expense of an overall higher computational cost. Our robust training has given performances similar to those of adversarial training on practical examples. This showcases the soundness of using robust training with an adequate sampling. However, in machine leaning applications with overparametrized models involving a very large number of parameters and high dimensional input space, scalability of our robust training framework is still a challenge due in particular to our simple Monte Carlo sampling step. More sophisticated sampling strategies, for instance those based on Langevin diffusion, is one direction that is worth investigating in a future work.

6.7 Appendix: Proofs

6.7.1 Proof of Proposition 6.2.1

(i) For any $z \in \mathcal{Z}$ and $\varepsilon \geq 0$, we have

$$\begin{aligned}\sup_{z' \in \mathcal{Z}} \big(\mathcal{L}(\theta, z') - \gamma \mathsf{c}(z, z')\big) &\geq \sup_{\mathsf{c}(z,z') \leq \varepsilon} \big(\mathcal{L}(\theta, z') - \gamma \mathsf{c}(z, z')\big) \\ &\geq \sup_{\mathsf{c}(z,z') \leq \varepsilon} \mathcal{L}(\theta, z') - \gamma\varepsilon.\end{aligned}$$

Taking the expectation on both sides, we get

$$\mathbb{E}_{z \sim \rho_0}\left[\sup_{\mathsf{c}(z',z) \leq \varepsilon} \mathcal{L}(\theta, z')\right] \leq \gamma\varepsilon + \mathbb{E}_{z \sim \rho_0}\left[\sup_{z' \in \mathcal{Z}} \big(\mathcal{L}(\theta, z') - \gamma \mathsf{c}(z', z)\big)\right].$$

In turn, since $\gamma \geq 0$ was arbitrary, we take the infimum on the right-hand side and use the identity (6.2.1), which holds under assumptions (H.1) and (H.2), to get the lower bound.
Let us turn to the upper-bound. We embark from (6.2.1) and consider the case where $\gamma = L_{\mathcal{Z}}$:

$$\begin{aligned}\sup_{W_{\mathsf{c}}(\rho,\rho_0) \leq \varepsilon} \mathbb{E}_{z \sim \rho}[\mathcal{L}(\theta, z)] &\leq L_{\mathcal{Z}}\varepsilon + \mathbb{E}_{z \sim \rho_0}\left[\sup_{z' \in \mathcal{Z}} \big(\mathcal{L}(\theta, z') - L_{\mathcal{Z}} \mathsf{c}(z', z)\big)\right] \\ &\leq L_{\mathcal{Z}}\varepsilon + \mathbb{E}_{z \sim \rho_0}\left[\mathcal{L}(\theta, z)\right] \\ &\leq L_{\mathcal{Z}}\varepsilon + \mathbb{E}_{z \sim \rho_0}\left[\sup_{\mathsf{c}(z,z') \leq \varepsilon} \mathcal{L}(\theta, z')\right]\end{aligned}$$

where we used the uniform Lipschitz continuity assumption (H.3) in the second inequality: $\mathcal{L}(\theta, z') - \mathcal{L}(\theta, z) \leq |\mathcal{L}(\theta, z') - \mathcal{L}(\theta, z)| \leq L_{\mathcal{Z}} \mathsf{c}(z', z)$ and that z is a feasible solution in the last constrained supremum since $\mathsf{c}(z, z') = 0$ when $z = z'$.

(ii) The proof of the lower-bound part is the same as in the first claim above. Let us turn to the upper-bound. We use again (6.2.1) and Lipschitz continuity of $\mathcal{L}(\theta, \cdot)$ to get that for any $\gamma \geq 0$,

$$\begin{aligned}\sup_{W_q(\rho,\rho_0) \leq \varepsilon^{1/q}} \mathbb{E}_{z \sim \rho}[\mathcal{L}(\theta, z)] &\leq \gamma\varepsilon + \mathbb{E}_{z \sim \rho_0}\left[\sup_{z' \in \mathcal{Z}} \big(\mathcal{L}(\theta, z') - \gamma \|z' - z\|^q\big)\right] \\ &\leq \gamma\varepsilon + \mathbb{E}_{z \sim \rho_0}\left[\sup_{z' \in \mathcal{Z}} \big(L_{\mathcal{Z}}\|z' - z\| - \gamma \|z' - z\|^q\big)\right] + \mathbb{E}_{z \sim \rho_0}\left[\mathcal{L}(\theta, z)\right] \\ &= \gamma\varepsilon + \mathbb{E}_{z \sim \rho_0}\left[\sup_{t \geq 0} \sup_{\|z' - z\| = t} \big(L_{\mathcal{Z}}\|z' - z\| - \gamma \|z' - z\|^q\big)\right] + \mathbb{E}_{z \sim \rho_0}\left[\mathcal{L}(\theta, z)\right] \\ &= \gamma\varepsilon + \sup_{t \geq 0} \left(L_{\mathcal{Z}} t - \gamma t^q\right) + \mathbb{E}_{z \sim \rho_0}\left[\mathcal{L}(\theta, z)\right] \\ &\leq \gamma\varepsilon + \sup_{t \geq 0} \left(L_{\mathcal{Z}} t - \gamma t^q\right) + \mathbb{E}_{z \sim \rho_0}\left[\sup_{\|z - z'\| \leq \varepsilon^{1/q}} \mathcal{L}(\theta, z')\right].\end{aligned}$$

Optimizing for t and after basic algebra, we get

$$\sup_{W_q(\rho,\rho_0) \leq \varepsilon^{1/q}} \mathbb{E}_{z \sim \rho}[\mathcal{L}(\theta, z)] \leq \gamma\varepsilon + (q-1)\left(\frac{L_{\mathcal{Z}}}{q}\right)^{\frac{q}{q-1}} \gamma^{-\frac{1}{q-1}} + \mathbb{E}_{z \sim \rho_0}\left[\sup_{\|z - z'\| \leq \varepsilon^{1/q}} \mathcal{L}(\theta, z')\right].$$

This upper-bound is minimal for $\gamma = c_{q,L_{\mathcal{Z}}} \varepsilon^{-\frac{q-1}{q}}$, for an explicit constant $c_{q,L_{\mathcal{Z}}}$. Plugging this value of γ in the upper-bound, we get the claim. □

6.7.2 Proof of Proposition 6.3.1

We provide a concise self-contained proof as the arguments are standard. We equip $\mathcal{P}(\mathcal{C}_z^\varepsilon)$ with the weak–* topology. Continuity of c implies closedness of $\mathcal{C}_z^\varepsilon$. This together with its boundedness assumption imply

compactness of $\mathcal{C}_z^\varepsilon$ as it is finite dimensional. Thus $\mathcal{P}(\mathcal{C}_z^\varepsilon)$ is weak–* compact by [1, Theorem 15.11]. It is also convex. In addition, recall that the weak–* topology is the weakest topology which makes the integration against continuous bounded functions a continuous linear form. In then follows from continuity of $\mathcal{L}(\theta,\cdot)$ and compactness of $\mathcal{C}_z^\varepsilon$ that $\mu \mapsto \int_{\mathcal{C}_z^\varepsilon} \mathcal{L}(\theta,z')\mathrm{d}\mu(z')$ is weak–* continuous. It is known that $\mathrm{KL}(\cdot,\mu_{\mathcal{U}})$ is convex and lower semicontinuous in the weak–* topology on $\mathcal{P}(\mathcal{C}_z^\varepsilon)$. Thus, since $\tau > 0$, the objective in (6.3.3) is convex and upper semicontinuous. This together with convex and weak–* compactness of $\mathcal{P}(\mathcal{C}_z^\varepsilon)$ entail that (6.3.3) has a non-empty convex and weak–* compact set of solutions. Uniqueness of the minimizer then follows from strong convexity of $\mathrm{KL}(\cdot,\mu_{\mathcal{U}})$ on $\mathcal{P}(\mathcal{C}_z^\varepsilon)$ thanks to the celebrated Pinsker's inequality. The closed form solution follows from standard calculus of variations and Lagrangian duality; see e.g., [187, Lemma 6.6]. □

6.7.3 Proof of Theorem 6.3.4

(i) Define $\psi_\tau(\theta) \overset{\text{def}}{=} \int_{\mathcal{C}_z^\varepsilon} \mathcal{L}(\theta,z')\mathrm{d}\mu(z') - \tau\mathrm{KL}(\mu,\mu_{\mathcal{U}})$. The function $\tau \mapsto \psi_\tau(\theta)$ obviously increases as τ decreases and so is g_τ. Continuity of $\mathcal{L}$, compactness of $\mathcal{C}_z^\varepsilon$ and Proposition 6.3.1 entail that g_τ is continuous and converges pointwise to g. The Γ-convergence claim in this case then follows from [128, Proposition 5.4 and Remark 5.5].

(ii) The upper bound in (6.3.5) is immediate by definition of g_τ. Let us turn to the lower bound. Let us denote $z^\star \in \mathrm{Argmax}_{z'\in\mathcal{C}_z^\varepsilon} \mathcal{L}(\theta,z')$, where the latter is a non-empty compact set thanks to continuity of $\mathcal{L}(\theta,\cdot)$ and compactness of $\mathcal{C}_z^\varepsilon$. We then have

$$\begin{aligned}
\tau\log\left(\int_{\mathcal{C}_z^\varepsilon} e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'\right) &= \max_{\hat{z}\in\mathcal{C}_z^\varepsilon}\mathcal{L}(\theta,\hat{z}) + \tau\log\int_{\mathcal{C}_z^\varepsilon} e^{\frac{\mathcal{L}(\theta,z')-\mathcal{L}(\theta,z^\star)}{\tau}}\mathrm{d}z' \\
&= g(\theta) + \tau\log\int_{\mathcal{C}_z^\varepsilon} e^{\frac{\mathcal{L}(\theta,z')-\mathcal{L}(\theta,z^\star)}{\tau}}\mathrm{d}z' \\
&\geq g(\theta) + \tau\log\int_{\mathcal{C}_z^\varepsilon} e^{\frac{-L_{\mathcal{Z}}\|z'-z^\star\|}{\tau}}\mathrm{d}z'.
\end{aligned}$$

By definition of $R_{\mathcal{C}_z^\varepsilon}$, there exists $\bar{z}$ such that $\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(\bar{z}) \subset \mathcal{C}_z^\varepsilon$. Convexity of $\mathcal{C}_z^\varepsilon$ entails that:

$$\tau(\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(\bar{z}) - z^\star) + z^\star = (1-\tau)z^\star + \tau\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(\bar{z}) \subset \mathcal{C}_z^\varepsilon,$$

and thus:

$$\begin{aligned}
\tau\log\left(\int_{\mathcal{C}_z^\varepsilon} e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'\right) &\geq g(\theta) + \tau\log\int_{\tau(\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(\bar{z})-z^\star)+z^\star} e^{\frac{-L_{\mathcal{Z}}\|z'-z^\star\|}{\tau}}\mathrm{d}z' \\
&= g(\theta) + \tau\log\left(\tau^m\int_{\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(\bar{z})} e^{-L_{\mathcal{Z}}\|z'-z^\star\|}\mathrm{d}z'\right) \\
&\geq g(\theta) + \tau\log\left(\tau^m\int_{\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(\bar{z})} e^{-L_{\mathcal{Z}}(\|z'-\bar{z}\|+\|z^\star-\bar{z}\|)}\mathrm{d}z'\right) \\
&\geq g(\theta) + \tau\log\left(\tau^m\int_{\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(0)} e^{-L_{\mathcal{Z}}(R_{\mathcal{C}_z^\varepsilon}+D_{\mathcal{C}_z^\varepsilon})}\mathrm{d}z'\right) \\
&= g(\theta) - m\tau\log(\tau^{-1}) + \tau\log(\mu_{\mathcal{L}}(\mathbb{B}_{R_{\mathcal{C}_z^\varepsilon}}(0))) - \tau L_{\mathcal{Z}}(R_{\mathcal{C}_z^\varepsilon}+D_{\mathcal{C}_z^\varepsilon}).
\end{aligned} \tag{6.7.1}$$

Inserting this into the expression of g_τ (see (6.3.4)), we get the upper-bound. □

6.7.4 Proof of Theorem 6.3.6

Compactness of Θ entails that g_τ and g are equi-coercive (see [128, Definition 7.6 and Proposition 7.7]. The first claim on convergence of the minimal values follows by combining the first claim in Theorem 6.3.4 and [128,

Theorem 7.8]. The second claim is a consequence of Γ-convergence of g_τ (Theorem 6.3.4), compactness of Θ and [128, Corollary 7.20]. The last claim is immediate from the second as the cluster point is unique. □

6.7.5 Proof of Theorem 6.3.7

We have

$$|g_{\tau,N}(\theta) - g(\theta)| \leq |g_\tau(\theta) - g(\theta)| + |g_{\tau,N}(\theta) - g_\tau(\theta)| \leq h(\tau) + |g_{\tau,N}(\theta) - g_\tau(\theta)|$$

where we used (6.3.5). It remains to bound the last term. This is the subject of the following lemma.

Lemma 6.7.1. *Under the assumptions of Theorem 6.3.4, the following holds.*

(i) For any $t > 0$ and fixed $\theta \in \Theta$,

$$|g_{\tau,N}(\theta) - g_\tau(\theta)| \leq \tau e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau}} \sqrt{\frac{t \log N}{2N}},$$

with probability at least $1 - 2N^{-t}$.

(ii) Suppose that τ is a function of N, say τ_N, with $\tau_N e^{\frac{\overline{\mathcal{L}}-\underline{\mathcal{L}}}{\tau_N}} \sqrt{\frac{\log N}{N}} \to 0$ as $N \to +\infty$, then

(a) for every $\theta \in \Theta$

$$g_{\tau_N,N}(\theta) - g_{\tau_N}(\theta) \underset{N\to+\infty}{\longrightarrow} 0 \quad \text{almost surely.}$$

(b) If moreover (H.4) holds then, almost surely,

$$g_{\tau_N,N}(\theta) - g_{\tau_N}(\theta) \underset{N\to+\infty}{\longrightarrow} 0 \quad \text{for all} \quad \theta \in \Theta.$$

Proof. To lighten the notation, denote

$$S_N \overset{\text{def}}{=} \frac{1}{N} \sum_{k=1}^{N} e^{\frac{\mathcal{L}(\theta, z'_k)}{\tau}}.$$

(i) Since the z'_k's are independent samples from the uniform distribution supported on $\mathcal{C}_z^\varepsilon$, we have

$$\mathbb{E}[S_N] = \frac{1}{\mu_{\mathcal{L}}(\mathcal{C}_z^\varepsilon)} \int_{\mathcal{C}_z^\varepsilon} e^{\frac{\mathcal{L}(\theta,z')}{\tau}} \mathrm{d}z'.$$

We then have

$$g_{\tau,N}(\theta) - g_\tau(\theta) = \tau \log\left(\frac{S_N}{\mathbb{E}[S_N]}\right).$$

Using the standard inequality $\log(1+t) \leq t$ for $t \geq 0$, we can write, for any $\epsilon \geq 0$

$$\begin{aligned}
&\Pr\left(|g_{\tau,N}(\theta) - g_\tau(\theta)| \geq \epsilon\right) = \Pr\left(\left|\log\left(\frac{S_N}{\mathbb{E}[S_N]}\right)\right| \geq \epsilon/\tau\right) \\
&= \Pr\left(\log\left(\frac{S_N}{\mathbb{E}[S_N]}\right) > \epsilon/\tau\right) \mathbb{1}(S_N \geq \mathbb{E}[S_N]) + \Pr\left(\log\left(\frac{\mathbb{E}[S_N]}{S_N}\right) > \epsilon/\tau\right) \mathbb{1}(S_N \leq \mathbb{E}[S_N]) \\
&\leq \Pr\left(\frac{S_N - \mathbb{E}[S_N]}{\mathbb{E}[S_N]} > \epsilon/\tau\right) \mathbb{1}(S_N \geq \mathbb{E}[S_N]) + \Pr\left(\frac{\mathbb{E}[S_N] - S_N}{S_N} > \epsilon/\tau\right) \mathbb{1}(S_N \leq \mathbb{E}[S_N]) \\
&\leq \Pr\left(S_N - \mathbb{E}[S_N] > e^{\underline{\mathcal{L}}/\tau}\epsilon/\tau\right) \mathbb{1}(S_N \geq \mathbb{E}[S_N]) + \Pr\left(S_N - \mathbb{E}[S_N] < -e^{\underline{\mathcal{L}}/\tau}\epsilon/\tau\right) \mathbb{1}(S_N \leq \mathbb{E}[S_N]) \\
&= \Pr\left(|S_N - \mathbb{E}[S_N]| > e^{\underline{\mathcal{L}}/\tau}\epsilon/\tau\right) \mathbb{1}(S_N \geq \mathbb{E}[S_N]) + \Pr\left(|S_N - \mathbb{E}[S_N]| > e^{\underline{\mathcal{L}}/\tau}\epsilon/\tau\right) \mathbb{1}(S_N \leq \mathbb{E}[S_N]) \\
&= \Pr\left(|S_N - \mathbb{E}[S_N]| > e^{\underline{\mathcal{L}}/\tau}\epsilon/\tau\right).
\end{aligned}$$

Since the random variables $e^{\frac{\mathcal{L}(\theta,z_i')}{\tau}}$ are independent and bounded (they live in the interval $[e^{\underline{\mathcal{L}}/\tau}, e^{\overline{\mathcal{L}}/\tau}]$), we are in position to invoke Hoeffding's inequality to obtain

$$\begin{aligned}\Pr\left(|g_{\tau,N}(\theta) - g_\tau(\theta)| \geq \epsilon\right) &\leq 2\exp\left(-\frac{2N^2 e^{2\underline{\mathcal{L}}/\tau}\epsilon^2}{N\left(e^{\overline{\mathcal{L}}/\tau} - e^{\underline{\mathcal{L}}/\tau}\right)^2\tau^2}\right) \\ &\leq 2\exp\left(-\frac{2Ne^{-2(\overline{\mathcal{L}}-\underline{\mathcal{L}})/\tau}\epsilon^2}{\tau^2}\right). \qquad (6.7.2)\end{aligned}$$

Taking

$$\epsilon = \tau e^{(\overline{\mathcal{L}}-\underline{\mathcal{L}})/\tau}\sqrt{\frac{t\log N}{2N}},$$

we get

$$\Pr\left(|g_{\tau,N}(\theta) - g_\tau(\theta)| > \epsilon\right) \leq 2e^{-t\log N} = 2N^{-t}$$

(ii) Let $\epsilon_N \stackrel{\text{def}}{=} \tau_N e^{(\overline{\mathcal{L}}-\underline{\mathcal{L}})/\tau_N}\sqrt{\frac{\log N}{N}}$.

(a) We have from the first claim above that

$$\Pr\left(|g_{\tau_N,N}(\theta) - g_{\tau_N}(\theta)| > \epsilon_N\right) \leq 2N^{-2}.$$

Since the right-hand side above is summable in N, we conclude by the (first) Borel-Cantelli lemma that with probability one

$$\limsup_{N\to+\infty} |g_{\tau_N,N}(\theta) - g_{\tau_N}(\theta)| = 0,$$

whence almost sure convergence is immediate.

(b) Since $\mathbb{R}^p$ is separable, there exists a countable set T whose closure is Θ. According to claim (a), for every $\theta \in \Theta$ there exists a set of events Ω_θ of probability one and, for every $\omega \in \Omega_\theta$, $g_{\tau_N,N}(\theta,\omega) - g_{\tau_N}(\theta,\omega) \to 0$ as $N \to +\infty$. Set $\widetilde{\Omega} = \bigcap_{\theta\in\mathsf{T}} \Omega_\theta$. Since T is countable, a union bound immediately shows that $\widetilde{\Omega}$ is also of probability one. For fixed $\theta \in \Theta$, there exists a sequence $(\theta_k)_{k\in\mathbb{N}}$ in T such that $\theta_k \to \theta$. Let $\omega \in \widetilde{\Omega}$. We have

$$\begin{aligned}|g_{\tau_N,N}(\theta,\omega) - g_{\tau_N}(\theta)| \leq |g_{\tau_N,N}(\theta_k,\omega) - g_{\tau_N,N}(\theta,\omega)| + |g_{\tau_N,N}(\theta_k,\omega) - g_{\tau_N}(\theta_k)| \\ + |g_{\tau_N}(\theta_k) - g_{\tau_N}(\theta)|.\end{aligned}$$

Since $\theta_k \in \mathsf{T}$, $\widetilde{\Omega} \subset \Omega_{\theta_k}$, and as just seen hereabove, the second term in the last inequality vanishes as $N \to +\infty$. Let us turn to the first term. Let

$$I^\theta_{\max}(\omega) = \left\{i \in [N] :\ \mathcal{L}(\theta, z_i(\omega)) = \max_{j\in[N]} \mathcal{L}(\theta, z_j(\omega))\right\}.$$

We have

$$g_{\tau_N,N}(\theta,\omega) = \max_{i\in[N]} \mathcal{L}(\theta, z_i(\omega)) + \tau_N \log\left(\frac{|I^\theta_{\max}(\omega)|}{N} + \frac{1}{N}\sum_{i\notin I^\theta_{\max}(\omega)} e^{\frac{\mathcal{L}(\theta,z_i(\omega)) - \max_{i\in[N]}\mathcal{L}(\theta,z_i(\omega))}{\tau_N}}\right).$$

Lipschitz continuity of a family of functions implies that of their maximum. It then follows that

$$\begin{aligned}|g_{\tau_N,N}(\theta_k,\omega)-g_{\tau_N,N}(\theta,\omega)| &\leq |\max_{i\in[N]}\mathcal{L}(\theta_k,z_i(\omega))-\max_{i\in[N]}\mathcal{L}(\theta,z_i(\omega))| \\ &\quad+\tau_N\left|\log\left(|I^{\theta}_{\max}(\omega)|+\sum_{i\notin I^{\theta}_{\max}(\omega)}e^{\frac{\mathcal{L}(\theta,z_i(\omega))-\max_{i\in[N]}\mathcal{L}(\theta,z_i(\omega))}{\tau_N}}\right)\right. \\ &\quad\left.-\log\left(|I^{\theta_k}_{\max}(\omega)|+\sum_{i\notin I^{\theta_k}_{\max}(\omega)}e^{\frac{\mathcal{L}(\theta_k,z_i(\omega))-\max_{i\in[N]}\mathcal{L}(\theta_k,z_i(\omega))}{\tau_N}}\right)\right| \\ &\leq L_\Theta\left\|\theta_k-\theta\right\|+2\tau_N\log N.\end{aligned}$$

Passing to the limit as $N\to+\infty$ we get

$$\limsup_{N\to+\infty}|g_{\tau_N,N}(\theta_k,\omega)-g_{\tau_N,N}(\theta,\omega)|\leq L_\Theta\left\|\theta_k-\theta\right\|.$$

We now invoke (6.3.5) to infer that

$$|g_{\tau_N}(\theta_k)-g_{\tau_N}(\theta)|\leq|g(\theta_k)-g(\theta)|+h(\tau_N)\leq L_\Theta\left\|\theta_k-\theta\right\|+h(\tau_N),$$

and thus

$$\limsup_{N\to+\infty}|g_{\tau_N}(\theta_k)-g_{\tau_N}(\theta)|\leq L_\Theta\left\|\theta_k-\theta\right\|.$$

Collecting the above estimates we get

$$\limsup_{N\to+\infty}|g_{\tau_N,N}(\theta,\omega)-g_{\tau_N}(\theta)|\leq 2L_\Theta\left\|\theta_k-\theta\right\|.$$

Taking the limit as $k\to+\infty$, we obtain $g_{\tau_N,N}(\theta,\omega)-g_{\tau_N}(\theta)\to 0$. This completes the proof.

□

6.7.6 Proof of Theorem 6.3.9

6.7.6.1 Proof of Lemma 6.3.10

To lighten notation in the proof, we drop the super- and subscript in $\mathcal{C}^{\varepsilon}_{z}$. Let the (Gibbs) probability measure[4]

$$\mathrm{d}\mu_\tau(z')\overset{\text{def}}{=}\frac{e^{\frac{\mathcal{L}(\theta,z')}{\tau}}}{\int_{\mathcal{C}}e^{\frac{\mathcal{L}(\theta,v)}{\tau}}\mathrm{d}v}\mathrm{d}z'.$$

Compactness of $\mathcal{C}$ and continuity of $\mathcal{L}(\theta,\cdot)$ imply that $\mathcal{M}$ is a non-empty compact set. Without loss of generality, we assume that $\max\mathcal{L}(\theta,\mathcal{C})=\mathcal{L}(\theta,\mathcal{M})=0$ (otherwise, one can use a simple translation argument).

(i) The proof of this claim is inspired by standard arguments in the literature of simulated annealing and Markov chains (see e.g. [34, Proposition 1.2] or [100, Corollary 2.1 and Proposition 2.3])[5]. We provide a self-contained proof adapted to our setting.
Given $\epsilon>0$, we define

$$\mathcal{U}^{\epsilon}=\{u\in\mathbb{R}^m:\ \mathcal{L}(\theta,u)\geq-\epsilon\}.$$

By assumption (H.3), $\mathcal{L}(\theta,\cdot)$ is $L_{\mathcal{Z}}$-Lipschitz continuous, and thus $\mathcal{U}^{\epsilon}$ is contained in the open tubular

[4] Strictly speaking, we should also index it with θ. In this proof, we will drop this to lighten notation.
[5] We thank the reviewer for raising similar arguments.

neighborhood of radius $\epsilon/L_{\mathcal{Z}}$ around $\operatorname{Argmax}\mathcal{L}(\theta,\mathcal{C})$. This implies that $\mu_{\mathcal{L}}(\mathcal{U}^\epsilon)>0$. We then have

$$\begin{aligned}\mu_\tau(\mathcal{C}\setminus\mathcal{U}^\epsilon)&=\frac{\int_{\mathcal{C}\setminus\mathcal{U}^\epsilon}e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'}{\int_{\mathcal{C}}e^{\frac{\mathcal{L}(\theta,v)}{\tau}}\mathrm{d}v}\\&\leq\frac{e^{\frac{-\epsilon}{\tau}}\mu_{\mathcal{L}}(\mathcal{C}\setminus\mathcal{U}^\epsilon)}{\int_{\mathcal{U}^{\epsilon/2}}e^{\frac{\mathcal{L}(\theta,v)}{\tau}}\mathrm{d}v}\\&\leq\frac{e^{\frac{-\epsilon}{\tau}}\mu_{\mathcal{L}}(\mathcal{C}\setminus\mathcal{U}^\epsilon)}{e^{\frac{-\epsilon}{2\tau}}\mu_{\mathcal{L}}(\mathcal{U}^\epsilon)}\leq e^{\frac{-\epsilon}{2\tau}}\frac{\mu_{\mathcal{L}}(\mathcal{C})}{\mu_{\mathcal{L}}(\mathcal{U}^\epsilon)}.\end{aligned}$$

Passing to the limit as $\tau\to0^+$ we get

$$\mu_\tau(\mathcal{C}\setminus\mathcal{U}^\epsilon)\to0.$$

Compactness of $\mathcal{C}$ implies that $\mathcal{P}(\mathcal{C})$ is weak–* compact by [1, Theorem 15.11]. The family $(\mu_\tau)_{\tau\geq0}$ is then sequentially precompact by Prokhorov's theorem. Let $(\mu_{\tau_k})_{k\in\mathbb{N}}$ be a subsequence with weak–* cluster point $\bar\mu$. We then have $\bar\mu(\mathcal{C}\setminus\mathcal{U}^\epsilon)=0$, and since ϵ is arbitrary, we get that $\bar\mu$ is supported on $\mathcal{U}^0=\operatorname{Argmax}\mathcal{L}(\theta,\mathcal{C})$. We thus infer, in view of continuity of $\nabla_\theta\mathcal{L}(\theta,\cdot)$ (by (H.5)) that

$$\nabla g_{\tau_k}(\theta)=\int_{\mathcal{C}}\nabla_\theta\mathcal{L}(\theta,z')\mathrm{d}\mu_{\tau_k}(z')\xrightarrow[k\to+\infty]{}\int_{\mathcal{C}}\nabla_\theta\mathcal{L}(\theta,z')\mathrm{d}\bar\mu(z')\in\partial^C g(\theta),$$

where we used (2.10.2) in the last inclusion. This is being true for any subsequence $(\mu_{\tau_k})_{k\in\mathbb{N}}$, we conclude that all cluster points of $(\nabla g_{\tau_k}(\theta))_{k\in\mathbb{N}}$ belong to $\partial^C g(\theta)$ which is equivalent to (6.3.13).

(ii) For any Borel set $\mathcal{C}\subset\mathbb{R}^m$ and $k\in\mathbb{N}$, $\mathcal{H}^k(\mathcal{C})$ is the k-dimensional Hausdorff measure. It is normalized to coincide with the Lebesgue measure on $\mathbb{R}^k$. For a k-dimensional smooth submanifold of $\mathbb{R}^m$, its k-dimensional Hausdorff measure coincides with the Riemannian volume measure.
By (H.7)(a), for any $r\geq3$, $\mathcal{M}$ is $\mathcal{C}^r$-stratifiable and thus the strata $(\mathcal{M}_i)_{i\in I}$ are $\mathcal{C}^r$-smooth compact submanifolds.

Given $\epsilon>0$, for each $\mathcal{M}_i$, we define its open neighborhood

$$\mathcal{U}_i=\{u\in\mathbb{R}^m:\ \operatorname{dist}(u,\mathcal{M}_i)<\epsilon\}.$$

Let $\mathcal{U}\overset{\text{def}}{=}\bigcup_{i\in I}\mathcal{U}_i$. We then have for $f\in\mathcal{C}$

$$\int_{\mathcal{C}}f(z')e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'=\int_{\mathcal{C}\cap\mathcal{U}}f(z')e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'+\int_{\mathcal{C}\setminus(\mathcal{C}\cap\mathcal{U})}f(z)e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'.\tag{6.7.3}$$

Since $f(\mathcal{C})$ is compact by compactness of $\mathcal{C}$ and continuity of f, and $\exists\kappa>0$ such that $\forall z'\in\mathcal{C}\setminus(\mathcal{C}\cap\mathcal{U})$, $\mathcal{L}(\theta,z')\leq-\kappa<\max\mathcal{L}(\theta,\mathcal{C})=0$, the second integral in (6.7.3) verifies, for any $s\geq0$,

$$\tau^{-s}\left|\int_{\mathcal{C}\setminus(\mathcal{C}\cap\mathcal{U})}f(z)e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'\right|\leq(\mu_{\mathcal{L}}(\mathcal{C})\sup|f(\mathcal{C})|)\,\tau^{-s}e^{-\kappa/\tau}\to0\quad\text{uniformly as }\tau\to0^+.\tag{6.7.4}$$

Let us now turn to the first integral. We have, for ϵ sufficiently small

$$\int_{\mathcal{C}\cap\mathcal{U}}f(z')e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'=\sum_{i\in I}\int_{\mathcal{C}\cap\mathcal{U}_i}f(z)e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'.\tag{6.7.5}$$

Since, for any $i\in I$, $\mathcal{M}_i$ is a compact $\mathcal{C}^r$-smooth submanifold with $r\geq2$, it is a set with positive reach thanks to [72, Theorem 4.12] (see [72, Definition 4.1] for definition of sets of positive reach). Thus, it follows from [72, Theorem 4.8] that $\mathrm{P}_{\mathcal{M}_i}$ is single-valued and Lipschitz continuous on $\mathcal{U}_i$, hence $\mathcal{C}\cap\mathcal{U}_i$, for some $\epsilon>0$ small enough. This together with rectifiability and measurability of the sets $\mathcal{C}\cap\mathcal{U}_i$ and $\mathcal{M}_i$ allows to apply the coarea change of variable formula [73, Theorem 3.2.22(3)] to get

$$\int_{\mathcal{C}\cap\mathcal{U}_i}f(z')e^{\frac{\mathcal{L}(\theta,z')}{\tau}}\mathrm{d}z'=\int_{\mathcal{M}_i}\left(\int_{\mathrm{P}_{\mathcal{M}_i}^{-1}(v)}f(u)e^{\frac{\mathcal{L}(\theta,u)}{\tau}}(\mathbf{J}_{m_i}(\mathrm{P}_{\mathcal{M}_i})(u))^{-1}d\mathcal{H}^{m-m_i}(u)\right)d\mathcal{H}^{m_i}(v),$$

where $m_i = \dim(\mathcal{M}_i)$, $\mathbf{J}_{m_i}(\mathrm{P}_{\mathcal{M}_i})$ is the m_i-dimensional Jacobian of $\mathrm{P}_{\mathcal{M}_i}$, i.e.,

$$\mathbf{J}_{m_i}(\mathrm{P}_{\mathcal{M}_i})(u) = \sqrt{\det\left(\mathbf{D}(\mathrm{P}_{\mathcal{M}_i})(u)\,\mathbf{D}(\mathrm{P}_{\mathcal{M}_i})(u)^\top\right)}$$

and $\mathbf{D}$ is the derivative operator. In addition, since $\mathcal{M}_i$ is $\mathcal{C}^r$-smooth, we have from [161, Proposition 5.1] that $\mathrm{P}_{\mathcal{M}_i}$ is $\mathcal{C}^{r-1}$-smooth with Lipschitz derivative on $\mathcal{U}_i$ (taking ϵ smaller if necessary). This entails that the key estimates of [50, Lemma 6.1] hold in our case. The rest of our argument follows then similar lines to those of [50, Theorem 3.1, starting from (9.3)]. This allows us to show that

$$\lim_{\tau\to 0^+} \tau^{-\frac{m-m_i}{2}} \int_{\mathcal{C}\cap\mathcal{U}_i} f(z) e^{\frac{\mathcal{L}(\theta,z)}{\tau}} \mathrm{d}z' = 2^{\frac{m-m_i}{2}} (m-m_i)\alpha_{(m-m_i)}\beta_{(m-m_i)} \int_{\mathcal{M}_i} f(v) \left(\prod_{j=1}^{m-m_i} \lambda_j(v)^{-\frac{1}{2}}\right) d\mathcal{H}^{m_i}(v), \quad (6.7.6)$$

where $(-\lambda_j(v))_j$ are the $m-m_i$ eigenvalues of the Hessian $\nabla^2_{z'}\mathcal{L}(\theta,v)$ for $v\in\mathcal{M}_i$, which are negative by (H.7)(b), α_k is the k-dimensional Lebesgue measure of the unit ball in $\mathbb{R}^k$, and

$$\beta_k \overset{\text{def}}{=} \begin{cases} 2^{-\frac{k}{2}}(k-2)(k-4)\cdot(2) & \text{for } k \text{ even} \\ 2^{-\frac{k}{2}}(k-2)(k-4)\cdot(3)\sqrt{\pi} & \text{for } k \text{ odd,} \end{cases}$$

Since the strata are ordered by strictly decreasing dimension, we have from (6.7.6) that for any $i>j$,

$$\lim_{\tau\to 0^+} \tau^{-\frac{m-m_j}{2}} \int_{\mathcal{C}\cap\mathcal{U}_i} f(z) e^{\frac{\mathcal{L}(\theta,z)}{\tau}} \mathrm{d}z' = \lim_{\tau\to 0^+} \tau^{\frac{m_j-m_i}{2}} \left(\tau^{-\frac{m-m_i}{2}} \int_{\mathcal{C}\cap\mathcal{U}_i} f(z) e^{\frac{\mathcal{L}(\theta,z)}{\tau}} \mathrm{d}z'\right) = 0. \quad (6.7.7)$$

Combining (6.7.7) (for $j=1$) and (6.7.6) (for $i=1$) with (6.7.3), (6.7.4) and (6.7.5), we get

$$\lim_{\tau\to 0^+} \tau^{-\frac{m-m_1}{2}} \int_{\mathcal{C}} f(z') e^{\frac{\mathcal{L}(\theta,z')}{\tau}} \mathrm{d}z' = \lim_{\tau\to 0^+} \tau^{-\frac{m-m_1}{2}} \int_{\mathcal{C}\cap\mathcal{U}_1} f(z) e^{\frac{\mathcal{L}(\theta,z)}{\tau}} \mathrm{d}z'$$
$$= 2^{\frac{m-m_1}{2}} (m-m_1)\alpha_{(m-m_1)}\beta_{(m-m_1)} \int_{\mathcal{M}_1} f(v) \left(\prod_{j=1}^{m-m_1} \lambda_1(v)^{-\frac{1}{2}}\right) d\mathcal{H}^{m_1}(v).$$

Applying this with $f\equiv 1$ and arbitrary $f\in\mathcal{C}$, we get that μ_τ converges in the narrow topology to the probability measure supported on $\mathcal{M}_1 \subset \operatorname{Argmax}\mathcal{L}(\theta,\mathcal{C})$

$$\mathrm{d}\mu(v) = \frac{1}{\int_{\mathcal{M}_1}\left(\prod_{j=1}^{m-m_1}\lambda_1(u)^{-\frac{1}{2}}\right) d\mathcal{H}^{m_1}(u)} \left(\prod_{j=1}^{m-m_1}\lambda_1(v)^{-\frac{1}{2}}\right) d\mathcal{H}^{m_1}(v).$$

By the continuity assumption (H.5) on $\nabla_\theta\mathcal{L}(\theta,z')$, we deduce that

$$\lim_{\tau\to 0^+} \nabla g_\tau(\theta) = \lim_{\tau\to 0^+} \int_{\mathcal{C}} \nabla_\theta\mathcal{L}(\theta,z')\mathrm{d}\mu_\tau(z') = \int_{\mathcal{M}_1} \nabla_\theta\mathcal{L}(\theta,v)\mathrm{d}\mu(v) \subset \partial^C g(\theta),$$

where we used (2.10.2) in the inclusion. This concludes the proof.

□

6.7.6.2 Proof of Lemma 6.3.11

To lighten notation in the proof, we drop the super- and subscript in $\mathcal{C}^\varepsilon_z$. Denote the probability measures

$$\mathrm{d}\mu^\theta_\tau(z') \overset{\text{def}}{=} \frac{1}{\mu_{\mathcal{L}}(\mathcal{C})} \frac{e^{\frac{\mathcal{L}(\theta,z')}{\tau}}}{S^\theta_\tau} \mathrm{d}z' \quad \text{and} \quad \mathrm{d}\mu^\theta_{\tau,N}(z') \overset{\text{def}}{=} \frac{1}{N}\sum_{k=1}^N \frac{e^{\frac{\mathcal{L}(\theta,z'_k)}{\tau}}}{S^\theta_{\tau,N}} \delta_{z'_k}$$

where

$$S^\theta_\tau \overset{\text{def}}{=} \frac{1}{\mu_{\mathcal{L}}(\mathcal{C})} \int_{\mathcal{C}} e^{\frac{\mathcal{L}(\theta,z')}{\tau}} \mathrm{d}z' \quad \text{and} \quad S^\theta_{\tau,N} \overset{\text{def}}{=} \frac{1}{N}\sum_{k=1}^N e^{\frac{\mathcal{L}(\theta,z'_k)}{\tau}}.$$

We have made here the dependence on θ, τ and N explicit as it will make our reasoning clearer especially for proving the last claim of the lemma.

(i) It follows from (6.3.9) and (6.3.15) that

$$\nabla g_\tau(\theta) - \nabla g_{\tau,N}(\theta) = \int_{\mathcal{C}} \nabla_\theta \mathcal{L}(\theta, z') \mathrm{d}\mu_\tau^\theta(z') - \int_{\mathcal{C}} \nabla_\theta \mathcal{L}(\theta, z') \mathrm{d}\mu_{\tau,N}^\theta(z')$$

and thus, by assumption (H.5), we get

$$\|\nabla g_\tau(\theta) - \nabla g_{\tau,N}(\theta)\| \leq L_{\Theta,\mathcal{Z}} \left| 1 - \frac{S_\tau^\theta}{S_{\tau,N}^\theta} \right| + \left\| \int_{\mathcal{C}} \nabla_\theta \mathcal{L}(\theta, z') \left(\mathrm{d}\mu_\tau^\theta(z') \frac{S_\tau^\theta}{S_{\tau,N}^\theta} - \mathrm{d}\mu_{\tau,N}^\theta(z') \right) \right\|. \tag{6.7.8}$$

For the first term, since $S_\tau^\theta = \mathbb{E}[S_{\tau,N}^\theta]$, we get from the proof of Lemma 6.7.1 that for any $t > 0$,

$$\left| 1 - \frac{S_\tau^\theta}{S_{\tau,N}^\theta} \right| \leq e^{\frac{\overline{\mathcal{L}} - \underline{\mathcal{L}}}{\tau}} \sqrt{\frac{t \log N}{2N}}$$

with probability at least $1 - 2N^{-t}$.

Let us now turn to the second term in (6.7.8). Denote

$$G_\tau^\theta \overset{\text{def}}{=} \frac{1}{\mu_{\mathcal{L}}(\mathcal{C})} \int_{\mathcal{C}} \nabla_\theta \mathcal{L}(\theta, z') e^{\frac{\mathcal{L}(\theta, z')}{\tau}} \mathrm{d}z' \quad \text{and} \quad G_{\tau,N}^\theta \overset{\text{def}}{=} \frac{1}{N} \sum_{k=1}^N \nabla_\theta \mathcal{L}(\theta, z_k') e^{\frac{\mathcal{L}(\theta, z_k')}{\tau}}.$$

We then have

$$\left\| \int_{\mathcal{C}} \nabla_\theta \mathcal{L}(\theta, z') \left(\mathrm{d}\mu_\tau^\theta(z') \frac{S_\tau^\theta}{S_{\tau,N}^\theta} - \mathrm{d}\mu_{\tau,N}^\theta(z') \right) \right\| = \frac{\left\| G_{\tau,N}^\theta - G_\tau^\theta \right\|}{S_{\tau,N}^\theta} \leq e^{\frac{-\underline{\mathcal{L}}}{\tau}} \left\| G_{\tau,N}^\theta - G_\tau^\theta \right\|.$$

Since $\mathbb{E}[G_{\tau,N}^\theta] = G_\tau^\theta$ and the random vectors $\nabla \mathcal{L}(\theta, z_k') e^{\frac{\mathcal{L}(\theta, z_k')}{\tau}}$ are independent and bounded, we apply Hoeffding's inequality and the union bound to obtain

$$\begin{aligned} \Pr\left(\left\| G_{\tau,N}^\theta - G_\tau^\theta \right\| > \epsilon \right) &\leq \Pr\left(\max_j |(G_{\tau,N}^\theta)_j - (G_\tau^\theta)_j| > \epsilon/\sqrt{p} \right) \\ &\leq p \max_j \Pr\left(|(G_{\tau,N}^\theta)_j - (G_\tau^\theta)_j| > \epsilon/\sqrt{p} \right). \end{aligned}$$

Taking $\epsilon = L_{\Theta,\mathcal{Z}} e^{\overline{\mathcal{L}}/\tau} \sqrt{\frac{2tp \log N}{N}}$, we infer that

$$\left\| \int_{\mathcal{C}} \nabla_\theta \mathcal{L}(\theta, z') \left(\mathrm{d}\mu_\tau^\theta(z') \frac{S_\tau^\theta}{S_{\tau,N}^\theta} - \mathrm{d}\mu_{\tau,N}^\theta(z') \right) \right\| \leq L_{\Theta,\mathcal{Z}} e^{\frac{\overline{\mathcal{L}} - \underline{\mathcal{L}}}{\tau}} \sqrt{\frac{2tp \log N}{N}}$$

with probability larger than $1 - 2pN^{-t}$. Combining the above bounds with the union bound, we get the claim.

(ii) Let $\epsilon_N \overset{\text{def}}{=} \max(1, 2L_{\Theta,\mathcal{Z}}\sqrt{p}) e^{\frac{\overline{\mathcal{L}} - \underline{\mathcal{L}}}{\tau_N}} \sqrt{\frac{\log N}{N}}$.

(a) We argue as in the proof of Lemma 6.7.1(ii)(a). We have from claim (i) that

$$\Pr\left(\|\nabla g_{\tau_N}(\theta) - \nabla g_{\tau_N,N}(\theta)\| > \epsilon_N \right) \leq 2(p+1)N^{-2}.$$

Since the right-hand side above is summable in N, we conclude by the (first) Borel-Cantelli lemma that with probability one

$$\limsup_{N \to +\infty} \|\nabla g_{\tau_N}(\theta) - \nabla g_{\tau,N}(\theta)\| = 0.$$

(b) We will follow a reasoning similar to the proof of Lemma 6.7.1(ii)(b) using separability of $\mathbb{R}^p$ and a density argument. There exists a countable set T whose closure is Ξ. According to claim (a), for every $\theta \in \Xi \subset \Theta$, there exists a set of events Ω_θ of probability one and, for every $\omega \in \Omega_\theta$, $\nabla g_{\tau_N,N}(\theta, \omega) - \nabla g_{\tau_N}(\theta, \omega) \to 0$ as $N \to +\infty$. Set $\widetilde{\Omega} = \bigcap_{\theta \in \mathsf{T}} \Omega_\theta$. Countability of T and a union

bound show that $\widetilde{\Omega}$ is also of probability one. Moreover, for fixed $\theta \in \Xi$, there exists a sequence $(\theta_k)_{k\in\mathbb{N}}$ in T such that $\theta_k \to \theta$. Let $\omega \in \widetilde{\Omega}$. We have

$$\|\nabla g_{\tau_N,N}(\theta,\omega) - \nabla g_{\tau_N}(\theta)\| \le \|\nabla g_{\tau_N}(\theta_k) - \nabla g_{\tau_N}(\theta)\| + \|\nabla g_{\tau_N,N}(\theta_k,\omega) - \nabla g_{\tau_N}(\theta_k)\| + \|\nabla g_{\tau_N,N}(\theta_k,\omega) - \nabla g_{\tau_N,N}(\theta,\omega)\|. \quad (6.7.9)$$

Since $\theta_k \in \mathsf{T}$, $\widetilde{\Omega} \subset \Omega_{\theta_k}$, claim (a) gives us that the second term in the right hand side of (6.7.9) vanishes as $N \to +\infty$.
Let us turn to the first term. We have

$$\nabla g_{\tau_N}(\theta_k) - \nabla g_{\tau_N}(\theta) = \int_{\mathcal{C}} \nabla\mathcal{L}(\theta,z')\mathrm{d}\mu^{\theta_k}_{\tau_N}(z') - \int_{\mathcal{C}} \nabla\mathcal{L}(\theta,z')\mathrm{d}\mu^{\theta}_{\tau_N}(z').$$

By Lemma 6.3.10(i), each weak–* cluster point of $(\mu^{\theta_k}_{\tau_N})_{N\in\mathbb{N}}$ belongs to $\operatorname{Argmin}\mathcal{L}(\theta_k,\mathcal{Z})$. Since the latter reduces to a single element $\widehat{z}^{\theta_k}$ by uniqueness of the maximizer, we get by Prokhorov's theorem that $(\mu^{\theta_k}_{\tau_N})_{N\in\mathbb{N}}$ converges in the weak–* topology to the Dirac measure supported on $\widehat{z}^{\theta_k}$. Similarly, $(\mu^{\theta}_{\tau_N})_{N\in\mathbb{N}}$ converges in the weak–* topology to the Dirac measure supported on the unique maximizer $\widehat{z}^{\theta}$ of $\operatorname{Argmin}\mathcal{L}(\theta,\mathcal{Z})$. Consequently,

$$\nabla g_{\tau_N}(\theta_k) - \nabla g_{\tau_N}(\theta) \xrightarrow[N\to+\infty]{} \nabla_\theta\mathcal{L}(\theta_k,\widehat{z}^{\theta_k}) - \nabla_\theta\mathcal{L}(\theta,\widehat{z}^{\theta}). \quad (6.7.10)$$

Now, observe that by (H.5) and convexity of Ξ, the mean value theorem yields

$$\sup_{z'\in\mathcal{C}} |\mathcal{L}(\theta_k,z') - \mathcal{L}(\theta,z')| \le \sup_{z'\in\mathcal{C},\xi\in\Xi} \left\|\nabla_\theta\mathcal{L}(\xi,z')\right\| \|\theta_k - \theta\| \le L_{\Theta,\mathcal{Z}} \|\theta_k - \theta\|,$$

and taking the limit as $k \to +\infty$, we see that $\mathcal{L}(\theta_k,\cdot)$ converges uniformly to $\mathcal{L}(\theta,\cdot)$, and thus by [128, Proposition 5.2 and Remark 5.3] $\mathcal{L}(\theta_k,\cdot)$ Γ-converges to $\mathcal{L}(\theta,\cdot)$. Compactness of $\mathcal{C}$ also entails equi-coercivity of $-\mathcal{L}(\theta_k,\cdot)$ on $\mathcal{C}$. This together with Γ-convergence of $\mathcal{L}(\theta_k,\cdot)$ seen just above allows to apply [128, Corollary 7.20] to infer that $\widehat{z}^{\theta_k} \to \widehat{z}^{\theta}$ as $k \to +\infty$. In view of this, taking the limit as $k \to +\infty$ in (6.7.10), using continuity of $\nabla_\theta\mathcal{L}$ in both arguments (see (H.5)), we get that

$$\lim_{k\to+\infty}\lim_{N\to+\infty} \|\nabla g_{\tau_N}(\theta_k) - \nabla g_{\tau_N}(\theta)\| = 0.$$

A similar reasoning can be applied to arrive at the same conclusion for the third term in (6.7.9). Passing to the limit in N and then in k in (6.7.9), we have proved that for every $\omega \in \widetilde{\Omega}$

$$\lim_{N\to+\infty} \|\nabla g_{\tau_N,N}(\theta,\omega) - \nabla g_{\tau_N}(\theta)\| = 0, \quad \text{for all} \quad \theta \in \Theta.$$

This completes the proof. □

6.7.7 Proof of Theorem 6.4.2

We first show that G is definable on an o-minimal structure. Indeed, o-minimal structures enjoy powerful stability results under many operations: for instance sublevel sets of definable functions are definable, finite sums of definable functions are definable, and functions of the type $\sup_{v\in\mathcal{S}} F(u,v)$ (resp. $\inf_{v\in\mathcal{S}} F(u,v)$) where F and $\mathcal{S}$ are definable, are definable. Thus since φ is definable, so is $\mathcal{C}^\varepsilon$. This together with definability of $\mathcal{L}$ implies that g_i is definable for each i. In turn, we get definability of G as a finite sum of definable functions.

Consider an absolutely continuous curve $\theta : \mathbb{R}_+ \to \mathbb{R}^p$. The function G being locally Lipschitz continuous, $t \mapsto G(\theta(t))$ is also absolutely continuous and thus

$$\frac{\mathrm{d}}{\mathrm{d}t}G(\theta(t)) = \frac{1}{M}\sum_{i=1}^{M}\frac{\mathrm{d}}{\mathrm{d}t}g_i(\theta(t)) = \langle\frac{1}{M}\sum_{i=1}^{M} v_i, \dot{\theta}(t)\rangle, \quad \text{for all } v_i \in \partial^C g_i(\theta(t)) \text{ and for a.e. } t \ge 0,$$

where we used that the functions g_i are path differentiable for the Clarke subdifferential by [57, Theorem 5.8]. Therefore, G is a Lyapunov function for the set crit–G. Moreover, by [25, Theorem 6], $G(\text{crit–}G)$ has empty interior.

By the almost sure boundedness assumption, $\max_i \left\|\partial^C g_i(\theta_k)\right\|$ is also uniformly bounded almost surely. Moreover, the direction d_k is such that

$$d_k = v_k + \zeta_k,$$

with $v_k \in \frac{1}{M}\sum_{i=1}^M \partial^C g_i(\theta_k)$ and the random process ζ_k is a zero-mean uniformly bounded martingale difference noise. These uniform boundedness properties and the choice of the sequence γ_k allows to apply [14, Remark 1.5(ii) and Proposition 1.4] to get by [14, Proposition 1.3] that the continuous-time affine interpolant of $(\theta_k)_{k\in\mathbb{N}}$ is almost surely an asymptotic pseudotrajectory of the flow (6.4.5). Combining this with [14, Theorem 3.6 and Proposition 3.27] gives the claimed results. □

6.7.8 Proof of Theorem 6.4.3

We obviously have $v_k \in \frac{1}{M}\sum_{i=1}^M \partial^C g_i(\theta_k)$. Moreover, the bias term e_k obeys

$$\|e_k\| \leq \frac{1}{|B_k|}\sum_{i\in B_k} \operatorname{dist}\left(\nabla g^i_{\tau_k,N_k}(\theta_k), \partial^C g_i(\theta_k)\right) \leq \max_{\theta\in\mathbb{B}_C(0)} \operatorname{dist}\left(\nabla g^i_{\tau_k,N_k}(\theta), \partial^C g_i(\theta)\right). \tag{6.7.11}$$

In view of Theorem 6.3.9(ii)(b), if the sequence $(\tau_k, N_k)_{k\in\mathbb{N}}$ is as devised, then almost surely, $\lim_{k\to+\infty}\|e_k\| = 0$ for all $(\theta_k)_{k\in\mathbb{N}} \subset \mathbb{B}_C(0)$. By independent and uniform sampling of the mini-batches, ζ_k is a zero-mean martingale difference noise. We are then in position to invoke [13, Remark 4.5] to get that the conclusions of [14, Remark 1.5(ii) and Proposition 1.4] still hold provided that γ_k decays as devised. The rest of the proof is then same as that of Theorem 6.4.2. □

Appendix: Additional information on the experiments

Following the recommendations of the paper that introduced the dataset [59] we removed the columns with the modular ratios and the data was centered and normalized. The model trained is a Multi layer Perceptron with 2 layers of 200 neurons each and an output layer of 12 neurons for the 12 classes with ELU activation function.

Parameter	Value	Description
General parameters for all trainings		
Epochs	1500	Number of epochs
Optimizer	SGD-type	SGD for vanilla training, Algorithm 5 for robust training
Learning rate	0.01	Initial learning rate
Learning rate decay	0.1 every 300 epochs	Multiplicative decay for learning rate
Batch size	100	Batch size input data
Train set size	10430	
Test set size	10437	
Robustness radius	range between 0. and 0.3	Only relevant for adversarial and robust training
Loss function	Cross Entropy Loss	
Weight initialization	Xavier Glorot's [79]	Default initialization for Pytorch modules
Parameters for adversarial training		
Adversarial Loss	Cross Entropy	
Iteration number	40	Iterations for adversarial attack
Attack norm	ℓ_∞	Norm of the attack, taken accordingly to the Sampling ball
Parameters for robust training		
Monte-Carlo sampling	150 000	Number of samples for computing LSE
Sampling ball	$\mathbb{B}_r^\infty$	Ball for uniform MC sampling, taken accordingly to the attack norm
Temperature	0.0001	Fixed temperature for LSE computation

Table 6.1: Parameters for the trainings on Avila dataset

Chapter 7

Conclusion and perspectives

Robustness in deep learning is a hot topic of interest as regulators and industrials seek ways to open up the scope of applications of these algorithms. Enlarging to critical applications is a double-edged sword: on the one hand, it allows for real gains in performance and more accurate processes. On the other hand, consequences could be dire in the case of wrong prediction making or possible alteration of the decision process by an adversary. There are multiple notions revolving around robustness "certified", "verified", "robust","explainable" and "reliable" that may confound an inexperienced reader. To that extent, we offered clear distinctions that help to grasp the subtle shading between the different contributions and give an easier way to tap into the field of robustness. Approaches to find an optimal robust model spurred experimentally at first: this had the benefit of providing user-friendly tools and unified frameworks that concentrated major developments and recent researches, however it produced a very large pool of papers that turned out to be outperformed on a short term by other, more accurate, experimental approaches. This trend will be pursuing while no theoretical framework would be able to guarantee and provide an optimal procedure. The theory has been lagging behind due to the rather complex setting of robust neural network learning. Indeed, the current problems considered in robust learning of neural networks are, in complete generality, bi-level optimization problems, non-smooth (and thus non-differentiable), non-convex in their parameters, non-concave in their input variables, thus partially explaining the slower pace of theoretical search on the topic. The results that we provided within the scope of this thesis follow also an experimental-theoretical pattern: we have shown that analogies with other systems can be useful to provide new axis of research but that they do not guarantee results, such assumptions can easily be proven wrong by a single counter example. We provided ways of evaluating in a more standard way algorithms meant for computing the Lipschitz constant of deep learning algorithms, in all generality, this requires solving a robust minmax problem that we approximated using common heuristics. Finally, for a more theoretical result, we presented an algorithm that can, in the limit, approximate arbitrary closely the robust problem against local perturbations. There are still open leads for improvement:

Finding a suitable distance for high dimensions. The main problem of nowadays deep learning comes from the fact that the distance metric that we are using are the exact same one that in small dimension, namely ℓ_2 or ℓ_∞. Because metrics do not behave similarly with respect to the dimension, having an appropriate distance suited for images might be an asset that many learning procedures could exploit. This is linked to the notion of effective sampling in high dimensions as currently we are sampling on ℓ_2 or ℓ_∞ balls.

Develop Lipschitz constrained networks. We would like to emphasize on the word network as currently the Lipschitz constant is generally bounded layer-wise, this has an overall impact on the freedom that we give to the layers. This could speed up computation as Lipschitz layers are known to be slow to train. Of course scaling from the linear layer or convolution layer to the network itself increases the difficulty of the problem significantly but the overall gains could be drastic.

Scale robust learning to high dimensions. The Monte Carlo sampling that we used for our robust learning has the convenient property of having a sampling error independent to the dimension. However, according to our different tests in higher dimension the sampling needs to grow within unreasonable bounds to ensure the overall convergence. We were unsuccessful at having the same level of performance that we had on small dimensions to state of the art structures on common picture datasets. On top of that random perturbations cannot represent unrestricted attacks (i.e. transformations that do not alter the content of an image such as translations, slight rotations or brightness changes for instance): making robust learning with distances insensitive to transformations such as the Gromov-Wassertein distance [146] could be beneficial for taking into account restricted and unrestricted attacks altogether.

The challenges ahead of modern deep learning have not changed ever since adversarial attacks emerged as a serious threat to certification. Yet, the progress has been steady and ever since the discovery of such vulnerabilities, the literature has introduced methods to verify robustness properties on training sets and developed robust training methods. However, a methodology for certifying the use of neural networks in critical applications is still missing. Our contribution strives towards the understanding and the making of robust models in a both theoretical and practical way. We believe this theoretical and empirical tandem can still make great advances as it brings relevant advances to the robustness problem.

List of Figures

List of Notations

General definitions

$\mathbb{R}$: the set of real numbers
$\mathbb{R}_+$: nonnegative real numbers
$\mathbb{N}$: set of nonnegative integers
$\mathbb{N}_+$: set of positive integers
$[k]$: set of k first positive integers
$\mathbb{R}^n, \mathbb{R}^m$: finite dimensional real Euclidean spaces
$\mathbb{B}_r^q(x)$: ball of center x of norm q of radius r
Id: identity operator on $\mathbb{R}^n$
$\mathbb{I}_A$: Indicator function of set A: $\mathbb{I}_A(x) \mapsto 1 \quad \text{if} \quad x \in A \quad \text{else} \quad 0$
P_A: Projection function onto a set $A \in \mathcal{X}$: $\mathrm{P}_A(x) : \mathcal{X} \to A, X \mapsto \mathrm{Argmin}_{\tilde{x} \in A} \|x - \tilde{x}\|$
x_i: ith component of the vector x
$\mathbb{1}$: vector of all 1s
$\mathcal{M}_+(S)$: ensemble of positive measures on a measurable set S
$\mathcal{P}(S)$: ensemble of probability measures supported on a measurable set S
$\mathcal{C}^n$: set of n-continuously differentiable function

List of Acronyms

General appellations

AI:	Artificial Intelligence
CNN:	Convolutional Neural Network
C&W:	Carlini and Wagner (attack)
DNN:	Deep Neural Network
DRO:	Distributional Robust Optimization
ELU:	Exponential Linear Unit
FGSM:	Fast Gradient Sign Method
GAN:	Generative Adversarial Network
GPU:	Graphical processing unit
KDE:	Kernel Density Estimation
KL:	Kullback Leibler (divergence)
MLP:	Multi-Layer Perceptron
MMD:	Maximum Mean discrepancy
OOD:	Out Of Distribution
PCA:	Principal Component Analysis
PGD:	Projected Gradient Descent
PRO:	Pointwise Robust Optimisation
ReLU:	Rectified Linear Unit
RNN:	Recurrent Neural Network
SDP:	Semi Definite Programming
SVM:	Support Vector Machine

Bibliography

[1] Charalambos D. Aliprantis and Kim C. Border. *Infinite Dimensional Analysis: a Hitchhiker's Guide*. Springer, Berlin; London, 2006.

[2] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[3] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 233–242. PMLR, 2017.

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 284–293. PMLR, 2018.

[5] Waïss Azizian, Franck Iutzeler, and Jérôme Malick. Regularization for wasserstein distributionally robust optimization, 2022.

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12):2481–2495, 2017.

[7] Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6240–6249, 2017.

[8] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. In David Helmbold and Bob Williamson, editors, *Computational Learning Theory*, volume 2111, pages 224–240. Springer Berlin Heidelberg, 2001. Series Title: Lecture Notes in Computer Science.

[9] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2613–2621, 2016.

[10] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning and the bias-variance trade-off. *CoRR*, abs/1812.11118, 2018.

[11] Aharon Ben-Tal, Dick den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Manag. Sci.*, 59(2):341–357, 2013.

[12] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*, volume 28 of *Princeton Series in Applied Mathematics*. Princeton University Press, 2009.

[13] Michel Benaïm. Dynamics of stochastic approximation algorithms. In *Séminaire de probabilités XXXIII*, volume 1709 of *Lecture Notes in Mathematics*, pages 1–68. Springer, 1999.

[14] Michel Benaïm, Josef Hofbauer, and Sylvain Sorin. Stochastic Approximations and Differential Inclusions. *SIAM J. Control. Optim.*, 44(1):328–348, 2005.

[15] DP Bertsekas. *Control of uncertain systems with a set-membership description of the uncertainty*. PhD thesis, Massachusetts Institute of Technology, 1971.

[16] Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and Applications of Robust Optimization. *SIAM Rev.*, 53(3):464–501, 2011.

[17] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. Data-driven robust optimization. *Math. Program.*, 167(2):235–292, 2018.

[18] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *CoRR*, abs/1704.02654, 2017.

[19] Alberto Bietti, Grégoire Mialon, and Julien Mairal. On Regularization and Robustness of Deep Neural Networks. *CoRR*, abs/1810.00363, 2018. arXiv: 1810.00363.

[20] Jose Blanchet and Yang Kang. *Semi-supervised Learning Based on Distributionally Robust Optimization*, chapter 1, pages 1–33. John Wiley & Sons, Ltd, 2020.

[21] Jose H. Blanchet, Yang Kang, and Karthyek Rajhaa A. M. Robust Wasserstein profile inference and applications to machine learning. *J. Appl. Probab.*, 56(3):830–857, 2019.

[22] Jose H. Blanchet and Karthyek R. A. Murthy. Quantifying Distributional Model Risk via Optimal Transport. *Math. Oper. Res.*, 44(2):565–600, 2019.

[23] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *CoRR*, abs/1601.00670, 2016.

[24] Jérôme Bolte and Edouard Pauwels. A mathematical model for automatic differentiation in machine learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[25] Jérôme Bolte and Edouard Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Math. Program.*, 188(1):19–51, 2021.

[26] Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alexander J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. In *Proceedings 14th International Conference on Intelligent Systems for Molecular Biology 2006, Fortaleza, Brazil, August 6-10, 2006*, pages 49–57, 2006.

[27] Leon Bungert, René Raab, Tim Roith, Leo Schwinn, and Daniel Tenbrinck. CLIP: Cheap Lipschitz Training of Neural Networks. In *SSVM*, 2021.

[28] Louis Béthune, Alberto González-Sanz, Franck Mamalet, and Mathieu Serrurier. The Many Faces of 1-Lipschitz Neural Networks. *CoRR*, abs/2104.05097, 2021. arXiv: 2104.05097.

[29] Russel E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, January 1998. Publisher: Cambridge University Press.

[30] G. C. Calafiore and L. El Ghaoui. On Distributionally Robust Chance-Constrained Linear Programs. *Journal of Optimization Theory and Applications*, 130(1):1–22, December 2006.

[31] Nicholas Carlini and David A. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In Bhavani Thuraisingham, Battista Biggio, David Mandell Freeman, Brad Miller, and Arunesh Sinha, editors, *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, pages 3–14. ACM, 2017.

[32] Nicholas Carlini and David A. Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[33] C. Castera, J. Bolte, C. A. Sing-Long Févotte, and E. Pauwels. An inertial newton algorithm for deep learning. *Journal of Machine Learning Research*, 22(134):1–31, 2021.

[34] Olivier Catoni. Simulated annealing algorithms and markov chains with rare transitions. In Jacques Azéma, Michel Émery, Michel Ledoux, and Marc Yor, editors, *Séminaire de Probabilités XXXIII*, pages 69–119, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[35] Junjie Chen, Ming Yan, Zan Wang, Yuning Kang, and Zhuo Wu. Deep Neural Network Test Coverage: How Far Are We? *CoRR*, abs/2010.04946, 2020. arXiv: 2010.04946.

[36] Ruidi Chen. *Distributional robust learning under the Wasserstein metric*. PhD thesis, Boston University, 2019.

[37] Tong Chen, Jean-Bernard Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic Optimization for Lipschitz Constants of ReLU Networks. *arXiv:2002.03657 [cs, math]*, October 2020. arXiv: 2002.03657.

[38] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings*, volume 10482 of *Lecture Notes in Computer Science*, pages 251–268. Springer, 2017.

[39] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014.

[40] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5(2):153–163, 2017.

[41] Emilie Chouzenoux, Henri Gerard, and Jean-Christophe Pesquet. General risk measures for robust machine learning. *Foundations of Data Science*, 1, 01 2019.

[42] Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann N. Dauphin, and Nicolas Usunier. Parseval Networks: Improving Robustness to Adversarial Examples. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 854–863. PMLR, 2017.

[43] Frank H. Clarke. *Optimization and Nonsmooth Analysis*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1990.

[44] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[45] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 2019.

[46] Patrick L. Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators. *SIAM J. Math. Data Sci.*, 2(2):529–557, 2020.

[47] European Commission. Proposal for a regulation laying down harmonised rules on artificial intelligence | https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence.

[48] Michel Coste. *An introduction to o-minimal geometry*. Dottorato di ricerca in matematica / Università di Pisa, Dipartimento di Matematica. Istituti Editoriali e Poligrafici Internazionali, Pisa, 2000.

[49] Michel Coste. *An introduction to semialgebraic geometry*. Dottorato di ricerca in matematica / Università di Pisa, Dipartimento di Matematica. Istituti Editoriali e Poligrafici Internazionali, Pisa, 2000.

[50] Dennis D. Cox, Robert M. Hardt, and Petr Klouček. Convergence of Gibbs Measures Associated with Simulated Annealing. *SIAM Journal on Mathematical Analysis*, 39(5):1472–1496, January 2008.

[51] Kate Crawford and Ryan Calo. There is a blind spot in AI research. *Nature*, 538(7625):311–313, October 2016.

[52] Antonia Creswell and Anil Anthony Bharath. Denoising Adversarial Autoencoders. *IEEE Trans. Neural Networks Learn. Syst.*, 30(4):968–984, 2019.

[53] George Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 2(4):303–314, 1989.

[54] Thomas Daniel, Fabien Casenave, Nissrine Akkari, and David Ryckelynck. Model order reduction assisted by deep neural networks (ROM-net). *Adv. Model. Simul. Eng. Sci.*, 7(1):16, 2020.

[55] George Dasoulas, Kevin Scaman, and Aladin Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2456–2466. PMLR, 2021.

[56] Ingrid Daubechies, Ronald A. DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova. Nonlinear Approximation and (Deep) ReLU Networks. *CoRR*, abs/1905.02199, 2019. arXiv: 1905.02199.

[57] Damek Davis, Dmitriy Drusvyatskiy, Sham M. Kakade, and Jason D. Lee. Stochastic Subgradient Method Converges on Tame Functions. *Found. Comput. Math.*, 20(1):119–154, 2020.

[58] E. Delage and Y. Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):95–612, 2010.

[59] C. De Stefano, M. Maniaci, F. Fontanella, and A. Scotto di Freca. Reliable writer identification in medieval manuscripts through page layout features: The "Avila" Bible case. *Engineering Applications of Artificial Intelligence*, 72:99–110, June 2018.

[60] Josef Dick, Frances Y. Kuo, and Ian H. Sloan. High-dimensional integration: The quasi-Monte Carlo way. *Acta Numerica*, 22:133–288, May 2013.

[61] Elvis Dohmatob and Alberto Bietti. On the (non-)robustness of two-layer neural networks in different learning regimes. arXiv:2203.11864, Mar 2022.

[62] Yizhen Dong, Peixin Zhang, Jingyi Wang, Shuang Liu, Jun Sun, Jianye Hao, Xinyu Wang, Li Wang, Jin Song Dong, and Dai Ting. There is Limited Correlation between Coverage and Robustness for Deep Neural Networks. *CoRR*, abs/1911.05904, 2019. arXiv: 1911.05904.

[63] Lou van den Dries and Chris Miller. Geometric categories and o-minimal structures. *Duke Mathematical Journal*, 84(2):497 – 540, 1996.

[64] John C. Duchi, Peter W. Glynn, and Hongseok Namkoong. Statistics of robust optimization: A generalized empirical likelihood approach. *Math. Oper. Res.*, 46(3):946–969, 2021.

[65] John C. Duchi and Hongseok Namkoong. Variance-based Regularization with Convex Objectives. *J. Mach. Learn. Res.*, 20:68:1–68:55, 2019.

[66] Weinan E, Chao Ma, and Qingcan Wang. Rademacher complexity and the generalization error of residual networks. *Communications in Mathematical Sciences*, 18(6):1755–1774, 2020.

[67] Weinan E, Chao Ma, and Lei Wu. The generalization error of the minimum-norm solutions for over-parameterized neural networks.

[68] Laurent El Ghaoui and Hervé Lebret. Robust Solutions to Least-Squares Problems with Uncertain Data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, October 1997.

[69] Gamaleldin F. Elsayed, Ian J. Goodfellow, and Jascha Sohl-Dickstein. Adversarial Reprogramming of Neural Networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[70] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations. *Math. Program.*, 171(1-2):115–166, 2018.

[71] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11423–11434, 2019.

[72] H. Federer. Curvature measures. *Trans. Amer. Math. Soc.*, 93:418–491, 1959.

[73] Herbert Federer. *Geometric Measure Theory*. Classics in Mathematics. Springer Berlin Heidelberg, 1996.

[74] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting adversarial samples from artifacts. *CoRR*, abs/1703.00410, 2017.

[75] Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints An Int. J.*, 23(3):296–309, 2018.

[76] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[77] Rui Gao and Anton J. Kleywegt. Distributionally robust stochastic optimization with wasserstein distance. arXiv:1604.02199 [math.OC], 2016.

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

[79] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[80] J. Goh and M. Sim. Distributionally robust optimization and its tractable approximations. *Operations Research*, 58(4):902–917, 2010.

[81] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[82] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[83] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing Lipschitz continuity. *Mach. Learn.*, 110(2):393–416, 2021.

[84] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *CoRR*, abs/1810.12715, 2018.

[85] Alex Graves. Practical Variational Inference for Neural Networks. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2348–2356, 2011.

[86] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, 2012.

[87] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, abs/1702.06280, 2017.

[88] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018.

[89] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. Dlfuzz: differential fuzzing testing of deep learning systems. In Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu, editors, *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, pages 739–743. ACM, 2018.

[90] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. Is neuron coverage a meaningful measure for testing deep neural networks? In Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann, editors, *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pages 851–862. ACM, 2020.

[91] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009.

[92] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[93] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[94] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2266–2276, 2017.

[95] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[96] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Comput. Sci. Rev.*, 37:100270, 2020.

[97] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 3–29. Springer, 2017.

[98] Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA, January 2009.

[99] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the Lipschitz Constant as a Defense Against Adversarial Examples. In Carlos Alzate, Anna Monreale, Haytham Assem, Albert Bifet, Teodora Sandra Buda, Bora Caglayan, Brett Drury, Eva García-Martín, Ricard Gavaldà, Stefan Kramer, Niklas Lavesson, Michael Madden, Ian M. Molloy, Maria-Irina Nicolae, and Mathieu Sinn, editors, *ECML PKDD 2018 Workshops - Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings*, volume 11329 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2018.

[100] Chii-Ruey Hwang. Laplace's method revisited: Weak convergence of probability measures. *The Annals of Probability*, 8(6):1177–1182, 1980. Publisher: Institute of Mathematical Statistics.

[101] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[102] M. Chris Jones, J. S. Marron, and Simon J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91:401–407, 1996.

[103] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Mach. Learn.*, 37(2):183–233, 1999.

[104] Kyle D. Julian, Mykel J. Kochenderfer, and Michael P. Owen. Deep Neural Network Compression for Aircraft Collision Avoidance Systems. *CoRR*, abs/1810.04240, 2018. arXiv: 1810.04240.

[105] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2017.

[106] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle, editors, *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pages 1039–1049. IEEE / ACM, 2019.

[107] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[108] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Master's thesis, Dept. of Comp. Sci., University of Toronto, 2009.

[109] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017.

[110] Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. Wasserstein Distributionally Robust Optimization: Theory and Applications in Machine Learning. *CoRR*, abs/1908.08729, 2019. arXiv: 1908.08729.

[111] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[112] H. J. Kushner and G. G. Yin. *Stochastic Approximation Algorithms and Applications*, volume 35 of *Applications of Mathematics*. Springer, 1997.

[113] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1096, December 2019.

[114] Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of Neural Networks via sparse polynomial optimization. *arXiv:2004.08688 [cs, stat]*, April 2020.

[115] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten Digit Recognition with a Back-Propagation Network. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 396–404. Morgan Kaufmann, 1989.

[116] J. M. Lee. *Introduction to smooth manifolds*. Springer, 2003.

[117] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7167–7177, 2018.

[118] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[119] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9459–9469, 2019.

[120] Jianlin Li, Jiangchao Liu, Pengfei Yang, Liqian Chen, Xiaowei Huang, and Lijun Zhang. Analyzing deep neural networks with symbolic propagation: Towards higher precision and faster verification. In Bor-Yuh Evan Chang, editor, *Static Analysis - 26th International Symposium, SAS 2019, Porto, Portugal, October 8-11, 2019, Proceedings*, volume 11822 of *Lecture Notes in Computer Science*, pages 296–319. Springer, 2019.

[121] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. Structural coverage criteria for neural networks could be misleading. In Anita Sarma and Leonardo Murta, editors, *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE (NIER) 2019, Montreal, QC, Canada, May 29-31, 2019*, pages 89–92. IEEE / ACM, 2019.

[122] Shiyu Liang, Yixuan Li, and R. Srikant. Principled detection of out-of-distribution examples in neural networks. *CoRR*, abs/1706.02690, 2017.

[123] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019. Special Issue: Deep Learning in Medical Physics.

[124] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. DeepGauge: multi-granularity testing criteria for deep learning systems. In Marianne Huchard, Christian Kästner, and Gordon Fraser, editors, *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pages 120–131. ACM, 2018.

[125] David J C MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1992.

[126] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[127] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial*

Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 4536–4543. AAAI Press, 2019.

[128] G.D. Maso. *An Introduction to Γ-Convergence*. Progress in Nonlinear Differential Equations and Their Applications. Birkhäuser Boston, 2012.

[129] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[130] Laurent Meunier, Meyer Scetbon, Rafael Pinot, Jamal Atif, and Yann Chevaleyre. Mixed Nash Equilibria in the Adversarial Examples Game. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 7677–7687. PMLR, 2021.

[131] Francesco Mezzadri. How to generate random matrices from the classical compact groups. *arXiv:math-ph/0609050*, February 2007. arXiv: math-ph/0609050.

[132] Joan C. Miller and Clifford J. Maloney. Systematic mistake analysis of digital computer programs. *Communications of the ACM*, 6(2):58–63, February 1963.

[133] Matthew Mirman, Timon Gehr, and Martin T. Vechev. Differentiable abstract interpretation for provably robust neural networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3575–3583. PMLR, 2018.

[134] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[135] Michel Moukari, Sylvaine Picard, Loïc Simon, and Frédéric Jurie. Deep multi-scale architectures for monocular depth estimation. In *2018 IEEE International Conference on Image Processing, ICIP 2018, Athens, Greece, October 7-10, 2018*, pages 2940–2944. IEEE, 2018.

[136] Hongseok Namkoong and John C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2208–2216, 2016.

[137] Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, 1996.

[138] Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[139] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[140] Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D. Lee, and Meisam Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 14905–14916, Vancouver, BC, Canada, 2019.

[141] Augustus Odena and Ian Goodfellow. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing. *arXiv:1807.10875 [cs, stat]*, July 2018. arXiv: 1807.10875.

[142] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 582–597. IEEE Computer Society, 2016.

[143] Andrea Paudice, Luis Muñoz-González, András György, and Emil C. Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection. *CoRR*, abs/1802.03041, 2018.

[144] E. Pauwels. The ridge method for tame min-max problems. hal-03186676, February 2022.

[145] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, October 2017. arXiv: 1705.06640.

[146] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[147] Oskar Pfungst. *Clever Hans (The horse of Mr. Von Osten): a contribution to experimental animal and human psychology (classic reprint).* FORGOTTEN Books, 2015. OCLC: 975964540.

[148] Edouard Pineau, Sébastien Razakarivony, and Thomas Bonald. Unsupervised ageing detection of mechanical systems on a causality graph. In M. Arif Wani, Feng Luo, Xiaolin Andy Li, Dejing Dou, and Francesco Bonchi, editors, *19th IEEE International Conference on Machine Learning and Applications, ICMLA 2020, Miami, FL, USA, December 14-17, 2020*, pages 270–277. IEEE, 2020.

[149] Luca Pulina and Armando Tacchella. Challenging SMT solvers to verify neural networks. *AI Commun.*, 25(2):117–135, 2012.

[150] Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean. Barrage of random transforms for adversarially robust defense. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6528–6537. Computer Vision Foundation / IEEE, 2019.

[151] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[152] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10900–10910, 2018.

[153] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Adversarial Training Can Hurt Generalization. *CoRR*, abs/1906.06032, 2019. arXiv: 1906.06032.

[154] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Understanding and Mitigating the Tradeoff between Robustness and Accuracy. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7909–7919. PMLR, 2020.

[155] Hamed Rahimian and Sanjay Mehrotra. Distributionally Robust Optimization: A Review, August 2019. arXiv:1908.05659 [cs, math, stat].

[156] Meisam Razaviyayn, Tianjian Huang, Songtao Lu, Maher Nouiehed, Maziar Sanjabi, and Mingyi Hong. Nonconvex min-max optimization: Applications, challenges, and recent theoretical advances. *IEEE Signal Processing Magazine*, 37(5):55–66, 2020.

[157] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 779–788. IEEE Computer Society, 2016.

[158] R. T. Rockafellar and R. Wets. *Variational analysis*, volume 317. Springer Verlag, 1998.

[159] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[160] Andrzej Ruszczynski. Convergence of a stochastic subgradient method with averaging for nonsmooth nonconvex constrained optimization. *Optim. Lett.*, 14(7):1615–1625, 2020.

[161] D. Salas and L. Thibault. On characterizations of submanifolds via smoothness of the distance function in Hilbert spaces. *Journal of Optimization Theory and Applications*, 182(1):189–210, 2019.

[162] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis. The Primer*. Wiley, 1 edition, December 2007.

[163] John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3:417 – 424, 1980.

[164] Jasmine Sekhon and Cody H. Fleming. Towards improved testing for deep learning. In Anita Sarma and Leonardo Murta, editors, *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE (NIER) 2019, Montreal, QC, Canada, May 29-31, 2019*, pages 85–88. IEEE / ACM, 2019.

[165] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Zídek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nat.*, 577(7792):706–710, 2020.

[166] Mathieu Serrurier, Franck Mamalet, Alberto González-Sanz, Thibaut Boissin, Jean-Michel Loubes, and Eustasio del Barrio. Achieving Robustness in Classification Using Optimal Transport With Hinge Regularization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 505–514. Computer Vision Foundation / IEEE, 2021.

[167] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[168] Soroosh Shafieezadeh-Abadeh, Peyman Mohajerin Esfahani, and Daniel Kuhn. Distributionally Robust Logistic Regression. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1576–1584, 2015.

[169] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *CoRR*, abs/1712.01815, 2017. arXiv: 1712.01815.

[170] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 1 edition, February 2018.

[171] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL):41:1–41:30, 2019.

[172] Aman Sinha, Hongseok Namkoong, and John C. Duchi. Certifying Some Distributional Robustness with Principled Adversarial Training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[173] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[174] Matthew Staib and Stefanie Jegelka. Distributionally robust deep learning as a generalization of adversarial training. In *NIPS workshop on Machine Learning and Computer Security*, 2017.

[175] Matthew Staib and Stefanie Jegelka. Distributionally Robust Optimization and Generalization in Kernel Methods. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9131–9141, 2019.

[176] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is Robustness the Cost of Accuracy? - A Comprehensive Study on the Robustness of 18 Deep Image Classification Models. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, volume 11216 of *Lecture Notes in Computer Science*, pages 644–661. Springer, 2018.

[177] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Structural test coverage criteria for deep neural networks. *ACM Trans. Embed. Comput. Syst.*, 18(5):94:1–94:23, 2019.

[178] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[179] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. DeepTest: automated testing of deep-neural-network-driven autonomous cars. In Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman, editors, *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pages 303–314. ACM, 2018.

[180] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[181] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[182] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[183] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3839–3848, 2018.

[184] Abraham Wald. Statistical Decision Functions Which Minimize the Maximum Risk. *The Annals of Mathematics*, 46(2):265, April 1945.

[185] Jie Wang, Rui Gao, and Yao Xie. Sinkhorn distributionally robust optimization. *CoRR*, abs/2109.11926, 2021.

[186] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 1599–1614. USENIX Association, 2018.

[187] Bo Wei, William B. Haskell, and Sixiang Zhao. An inexact primal-dual algorithm for semi-infinite programming. *Math. Methods Oper. Res.*, 91(3):501–544, 2020.

[188] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. Towards Fast Computation of Certified Robustness for ReLU Networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5273–5282. PMLR, 2018.

[189] Eric Wong and J. Zico Kolter. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5283–5292. PMLR, 2018.

[190] Eric Wong, Frank R. Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8410–8419, 2018.

[191] Zuxuan Wu, Ser-Nam Lim, Larry S. Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part IV*, volume 12349 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2020.

[192] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating Adversarial Examples with Adversarial Networks. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 3905–3911. ijcai.org, 2018.

[193] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[194] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, September 2017. arXiv:1708.07747 [cs, stat].

[195] Xiaofei Xie, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu, and Yang Liu. NPC: neuron path coverage via characterizing decision logic of deep neural networks. *ACM Trans. Softw. Eng. Methodol.*, 31(3):47:1–47:27, 2022.

[196] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. DeepHunter: a coverage-guided fuzz testing framework for deep neural networks. In Dongmei Zhang and Anders Møller, editors, *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, Beijing, China, July 15-19, 2019*, pages 146–157. ACM, 2019.

[197] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and Regularization of Support Vector Machines. *J. Mach. Learn. Res.*, 10:1485–1510, 2009.

[198] Huan Xu and Shie Mannor. Robustness and generalization. *Mach. Learn.*, 86(3):391–423, 2012.

[199] Shenao Yan, Guanhong Tao, Xuwei Liu, Juan Zhai, Shiqing Ma, Lei Xu, and Xiangyu Zhang. Correlations between deep neural network model coverage criteria and model quality. In Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann, editors, *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pages 775–787. ACM, 2020.

[200] Yan Yan, Rómer Rosales, Glenn Fung, Subramanian Ramanathan, and Jennifer G. Dy. Learning from multiple annotators with varying expertise. *Mach. Learn.*, 95(3):291–327, 2014.

[201] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. A Closer Look at Accuracy vs. Robustness. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[202] Y.Y. Yang, C. Rashtchian, H. Zhang, R.R Salakhutdinov, and K Chaudhur. A closer look at accuracy vs. robustness. In *Advances in neural information processing systems*, volume 33, pages 8588–8601, 2020.

[203] Yuichi Yoshida and Takeru Miyato. Spectral Norm Regularization for Improving the Generalizability of Deep Learning. In *ICML Workshop on Implicit Models*, 2017. arXiv:1705.10941.

[204] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically Principled Trade-off between Robustness and Accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019.

[205] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane S. Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[206] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Trans. Software Eng.*, 48(2):1–36, 2022.

[207] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In Marianne Huchard, Christian Kästner, and Gordon Fraser, editors, *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pages 132–142. ACM, 2018.

[208] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Trans. Neural Networks Learn. Syst.*, 30(11):3212–3232, 2019.