université
PARIS-SACLAY

Stability Quantification of Neural Networks

Quantification de la stabilité des réseaux de neurones

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique mathématique
Graduate School : Sciences de l'ingénierie et des systèmes.
Référent : CentraleSupélec

Thèse préparée dans la unité de recherche **Centre de Vision Numérique (Université Paris-Saclay, CentraleSupélec)**, sous la direction de **Jean-Christophe Pesquet**, Professeur et la co-enradrement de **Fateh Kaakai**, expert en sécurité.

Thèse soutenue à Paris-Saclay, le 13 janvier 2023, par

Kavya GUPTA

Composition du jury

Frederic JURIE Professeur, Université de Caen Normandie	Président
Hamid KRIM Distinguished Professeur, North Carolina State University	Rapporteur & examinateur
Mathieu SERRURIER Maître de Conférences, Université Toulouse 3	Rapporteur & examinateur
Juliette MATTIOLI Experte senior en IA, Thales France	Examinatrice

THESE DE DOCTORAT

NNT : 2023UPAST004

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Quantification de la stabilité des réseaux de neurones
Mots clés: robustesse, stabilité, réseaux de neurones, attaques adverses, constante de Lipschitz

Résumé:

Dans le cadre de cette thèse, nous présentons trois contributions relatives à la stabilité des réseaux de neurones. La première contribution consiste en un attaquant adverse de style "boîte blanche", simple et facilement implémentable, ciblant les modèles de régression dans le domaine des données tabulaires. Nous utilisons les propriétés du jacobien du réseau de neurones afin de générer la pire attaque pour l'échantillon considéré. Pour quantifier le succès des attaques, nous proposons trois métriques analogues à celles présentées dans les scénarios de classification. La deuxième contribution réside en une analyse de sensibilité multivariée des entrées des réseaux de neurones, en se basant sur leur constante de Lipschitz. Une telle étude permet de mieux comprendre l'effet de chaque entrée sur la dynamique du système. La méthode présentée fonctionne sur les entrées prises individuellement ou par groupes. Nous proposons également une représentation sous forme "d'étoile de Lipschitz" pour synthétiser graphiquement cette analyse de sensibilité. Dans notre dernière contribution, nous convevons une méthode de contrôle de stabilité permettant de réaliser le meilleur compromis entre performance et stabilité, lors de la phase d'apprentissage des modèles de réseau de neurones. Cette méthode repose sur une normalization spectrale s'insérant dans une boucle d'entraînement, dont l'objectif est d'atteindre une performance et une stabilité données. Pour valider cette approche, nous l'apliquons à des données récoltées sur un drône aérien. Pour étayer nos contributions, nous les testons à la fois sur des données publiques libres d'accès et sur un jeu de données industrielles fourni par Thales LAS France, correspondant au champ des applications critiques en termes de sécurité.

Title: Stability Quantification of Neural Networks
Keywords: Robustness, stability, neural networks, adversarial attacks, constant

Abstract: This thesis contains three main contributions concerning the stability of neural networks. First, we present a simple and easily implementable white box adversarial attacker for regression models operating on tabular data. We rely upon the Jacobian properties of the neural networks to generate the worst attack on the sample. We propose three metrics for quantifying the success of the attacks analogous to metrics used in classification scenarios. In our second contribution, we propose a multivariate sensitivity analysis of the neural network inputs based on its Lipschitz properties. Such a study helps to understand better the effect of individual inputs on the dynamics of the outputs. The presented method works on individual as well as any group of inputs. We also propose a "Lipschitz star" representation to display this sensitivity analysis of the inputs. In the last contribution, we design a stability control method to achieve the best trade-off between accuracy and stability during the training phase of neural network models. This method is based on a spectral normalization inserted in a training loop that allows us to achieve given performance and stability targets. We validate this approach on an Unmanned Aerial Vehicle (UAV) application. To support our contributions, we test them both on open-source, public datasets and an industrial dataset provided by Thales LAS France that is related to the domain of safety-critical applications.

Abstract

Artificial neural networks are at the core of recent advances in Artificial Intelligence. One of the main challenges faced today, especially by companies like Thales designing advanced industrial systems is to ensure the safety of new generations of products using these technologies. In 2013 in a key observation, neural networks were shown to be sensitive to adversarial perturbations, raising serious concerns about their applicability in critically safe environments. In the last years, publications studying the various aspects of this robustness of neural networks, and rising questions such as "Why adversarial attacks occur?", "How can we make the neural network more robust to adversarial noise?", "How to generate stronger attacks?" etc., have grown exponentially. The contributions of this thesis aim to tackle such problems. The adversarial machine learning community concentrates majorly on classification scenarios, whereas studies on regression tasks are scarce. Our contributions bridge this significant gap between adversarial machine learning and regression applications. Also, all the datasets used as part of the contributions are tabular, which has not been highly researched by the deep learning community and only recently received more attention.

The first contribution in Chapter 3 proposes a white-box attackers designed to attack regression models. The presented adversarial attacker is derived from the algebraic properties of the Jacobian of the network. We show that our attacker successfully fools the neural network and measure its effectiveness in reducing the estimation performance. We present our results on various open-source and real industrial tabular datasets. Our analysis relies on the quantification of the fooling error as well as different error metrics. Another noteworthy feature of our attacker is that it allows us to optimally attack a subset of inputs, which may help to analyze the sensitivity of some specific inputs. We also, show the effect of this attacker on spectrally normalised trained models which are known to be more robust in handling attacks.

The second contribution of this thesis (Chapter 4) presents a multivariate Lipschitz constant analysis of neural networks. The Lipschitz constant is widely used in the literature to study the internal properties of neural networks. But most works do a single parametric analysis, which do not allow to quantify the effect of individual inputs on the output. We propose a multivariate Lipschitz constant-based stability analysis of fully connected neural networks allowing us to capture

the influence of each input or group of inputs on the neural network stability. Our approach relies on a suitable re-normalization of the input space, intending to perform a more precise analysis than the one provided by a global Lipschitz constant. We display the results of this analysis by a new representation designed for machine learning practitioners and safety engineers termed as a Lipschitz star. We perform experiments on various open-access tabular datasets and an actual Thales Air Mobility industrial application subject to certification requirements.

The use of spectral normalization in designing a stability control loop is discussed in Chapter 5. A critical part of the optimal model is to behave according to specified performance and stability targets while in operation. But imposing tight Lipschitz constant constraints while training the models usually leads to a reduction of their accuracy. Hence, we design an algorithm to train "stable-by-design" neural network models using our spectral normalization approach, which optimizes the model by taking into account both performance and stability targets. We focus on Small Unmanned Aerial Vehicles (UAVs). More specifically, we present a novel application of neural networks to detect in real-time elevon positioning faults to allow the remote pilot to take necessary actions to ensure safety.

Résumé (French)

Les réseaux de neurones artificiels sont au cœur des avancées récentes en Intelligence Artificielle. L'un des principaux défis auxquels on est aujourd'hui confronté, notamment au sein d'entreprises comme Thales concevant des systèmes industriels avancés, est d'assurer la sécurité des nouvelles générations de produits utilisant cette technologie. En 2013, une observation clé a révélé que les réseaux de neurones sont sensibles à des perturbations adverses. Ceci soulève de sérieuses inquiétudes quant à leur applicabilité dans des environnements où la sécurité est critique. Au cours des dernières années, des publications ont étudiées les différents aspects de la robustesse des réseaux de neurones, et des questions telles que "Pourquoi des attaques adverses se produisent?", "Comment pouvons-nous rendre les réseaux de neurones plus robustes à ces bruits ?", "Comment générer des attaques plus fortes ?", etc., se sont posées avec une acuité croissante. Cette thèse vise à apporter des réponses à de telles questions. La communauté s'intéressant aux attaques adverses en apprentissage automatique travaille principalement sur des scénarios de classification, alors que les études portant sur des tâches de régression sont rares. Nos contributions comblent le fossé existant entre les méthodes adverses en apprentissage et les applications de régression. De plus, tous les ensembles de données utilisés dans le cadre de nos travaux sont tabulaires. De telles données ont fait l'objet de peu de recherches approfondies dans la communauté de l'apprentissage et n'ont reçu que récemment une attention plus soutenue.

Notre première contribution, dans le chapitre 3, propose un algorithme de type "boîte blanche" pour attaquer les modèles de régression. L'attaquant adverse présenté est déduit des propriétés algébriques du Jacobien du réseau. Nous montrons que notre attaquant réussit à tromper le réseau de neurones et évaluons son efficacité à réduire les performances d'estimation. Nous présentons nos résultats sur divers ensembles de données tabulaires industriels en libre accès et réels. Notre analyse repose sur la quantification de l'erreur de tromperie ainsi que différentes métriques. Une autre caractéristique remarquable de notre algorithme est qu'il nous permet d'attaquer de manière optimale un sous-ensemble d'entrées, ce qui peut aider à identifier la sensibilité de certaines d'entre elles. Nous montrons également l'effet de cet attaquant sur des modèles entraînés avec une normalisation spectrale, connus pour être plus robustes aux attaques.

La deuxième contribution de cette thèse (Chapitre 4) présente une analyse de

la constante de Lipschitz multivariée des réseaux de neurones. La constante de Lipschitz est largement utilisée dans la littérature pour étudier les propriétés intrinsèques des réseaux de neurones. Mais la plupart des travaux font une analyse mono-paramétrique, qui ne permet pas de quantifier l'effet des entrées individuelles sur la sortie. Nous proposons une analyse multivariée de la stabilité des réseaux de neurones entièrement connectés, reposant sur leur propriétés Lipschitziennes. Cette analyse nous permet de saisir l'influence de chaque entrée ou groupe d'entrées sur la stabilité du réseau de neurones. Notre approche repose sur une re-normalisation appropriée de l'espace d'entrée, visant à effectuer une analyse plus précise que celle fournie par une constante de Lipschitz globale. Nous visualisons les résultats de cette analyse par une nouvelle représentation conçue pour les praticiens de l'apprentissage automatique et les ingénieurs en sécurité appelée "étoile de Lipschitz". Nous menons des expérimentations sur différents ensembles de données tabulaires en libre accès et sur une application industrielle réelle de Thales Air Mobility soumise à des exigences de certification.

L'utilisation de la normalisation spectrale dans la conception d'une boucle de contrôle de stabilité est abordée au chapitre 5. Une caractéristique essentielle du modèle optimal consiste à satisfaire aux objectifs de performance et de stabilité spécifiés pour le fonctionnement. Cependant, contraindre la constante de Lipschitz lors de l'apprentissage des modèles conduit généralement à une réduction de leur précision. Par conséquent, nous concevons un algorithme permettant de produire des modèles de réseaux de neurones "stable dès la conception" en utilisant une nouvelle approche de normalisation spectrale, qui optimise le modèle, en tenant compte à la fois des objectifs de performance et de stabilité. Nous nous concentrons sur les petits drones aériens (UAV). Plus précisément, nous présentons une nouvelle application des réseaux de neurones pour détecter en temps réel les défauts de positionnement d'aileron, afin de permettre au télépilote de prendre les mesures nécessaires pour assurer la sécurité du vol.

Acknowledgement

First of all, I would like to thank my supervisors, Prof. Jean-Christophe Pesquet, Dr. Fateh Kaakai, and Dr. Beatrice Pesquet-Popsescu, for giving me this opportunity. I have learned a lot from each of them over the years, things I will carry forward in my research career and would make me a better researcher. I want to extend my thanks to my thesis examiners Prof. Hamid Krim and Prof. Mathieu Serrurier, for going through my thesis manuscript and giving valuable feedback, as well as my jury members Prof. Frederic Jurie and Dr. Juliette Mattioli, for their time and effort towards evaluating my thesis work. I want to thank all the CVN faculty for building a lab research atmosphere - Prof. Fragkiskos Malliaros, Prof. Marc Castella, and Prof. Hughes Talbot. I also want to thank Jana for helping me immensely with administrative tasks and making life smoother. Also, I would like to thank Prof. Angshul Majumdar for encouraging to pursue research as career and teaching me everything.

I can't thank enough my friends Sanjay and Preethi for making my survival easy at the plateau and guiding me throughout for everything. I am really grateful to them. I want to thank Hyrule and Freja (cats) for helping with timely mental health support and just making life beautiful, and also to the people who live in their house Younes and Solen for being so welcoming, kind and generous. I want to thank Aymen for being my translator and Jhony for being so kind and motivating. I would like to thank my cite friends Surabhi, Shalu and Pragya for being such a great female support in Paris and also for the great food everytime and to Mihir for intently listening to my rants/stories and always providing with the best advices and solutions.

I want to thank my parents and brother for believing and not giving up on me throughout the Ph.D. journey. Their blessings and support miles apart kept me going in my low phases. I want to thank Sagar for being my constant support system, chief advisor for everything, biggest critic, and biggest cheerleader. He has encouraged me to be a better researcher and made me a stronger and braver person. This thesis is the manifestation of my late grandmother, who used to address me as a Dr. from when I was two years old. I hope I made her happy and proud.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 . Context and Motivations

Neural networks (NN) are at the core of recent advances in Artificial Intelligence (AI). One of the main challenges faced today, especially by companies like Thales designing advanced industrial systems, is ensuring the safety of new products using these technologies. In 2013, neural networks were first shown to be sensitive to adversarial perturbations [3], raising serious concerns about their suitability of use in mission-critical or safety-critical products and leading to the development of a new field of study concerning the robustness of neural networks.

In the literature, various classes of AI desirable properties have been defined by adding some objectives, components, or constraints to AI such as robustness, ethics, trustworthiness aiming at making neural networks more reliable for real-world applications as described in Table 1.1.

Concept	Definition
Robust	AI systems with the ability to cope with errors during execution and cope with erroneous input
Ethical	AI systems that do what is right, fair, and just and prevent harm.
Trustworthy	AI systems that achieve their full potential if trust can be established in the development, deployment, and use [4]
Fair	AI systems absent from any prejudice or favoritism toward an individual or a group based on their inherent or acquired characteristics [5]
Safe	AI systems deployed in ways that do not harm humanity
Dependable	AI systems that focus on reliability, verifiability, explainability, and security [6]
Human-centered	AI systems that are "continuously improving because of human input while providing an effective experience between human and robot"
Socially Responsible	AI systems that follow ethical and philanthropic responsibilities alongwith legal and performance responsibilities[7].

Table 1.1: Different classes of AI systems

Among these different classes of AI properties, the contributions presented in this thesis come under the category of Robust AI. In particular, we deal with errors (wrong predictions) during execution due to the inability of the neural network to cope with small perturbations applied to nominal inputs within the Operational Design Domain (ODD). These perturbations can result from natural system degradation, such as aging sensors, or a deliberate and malicious attack (cyber crimes).

In Szegedy et al. [3] the concept of adversarial attacks was first proposed to fool DNNs. Adding a subtle perturbation to the input of the neural network produces an incorrect output, while human eyes cannot recognize the difference in the modification of the input data. For example, after adding a slight perturbation to the original image labelled as "panda", it is classified as a "gibbon" by the same model with 99.3% confidence, while the human eyes cannot distinguish the differences between the original image and the adversarial image in Figure 1.1 [1]. This behavior of DNNs is possible even when the model is well-trained with good accuracy. The capability of handling adversarial noise is known as the robustness of the Neural Network. Even though different models have different architectures and might use different training data, the same kind of adversarial attack strategies can attack related models (*transferability of Adversarial attacks*). These attacks pose a huge threat to the performance of DNNs and cannot be ignored in safety- and security-critical AI applications. Analyzing the cause of adversarial attacks can help researchers to fix the vulnerability effectively, to provide defenses, and to guide in generating more robust and safe models.

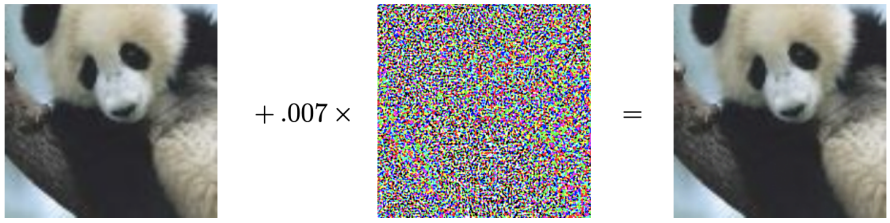

Figure 1.1: By adding an unnoticeable perturbation, an image labelled as "panda" is classified as "gibbon" [1].

.

In the recent years, the number of works devoted to understanding the instability and increasing robustness of NNs have grown exponentially. Many approaches have been proposed, more or less dedicated to some specific architectures, particularly networks using only ReLU activation functions, and grounded on more or less empirical techniques [8, 9]. Broadly we can classify the works in literature as follows:

i) Purely computational approaches consist of attacking a neural network and observing its response to such attacks [10, 11].

ii) Methods based on heuristics for testing / promoting the robustness of a neural net [12, 13, 14].

iii) Studies that aim to establish mathematical certificates and provide theoretical robustness guarantees [15, 9, 16].

The three strategies might ultimately be necessary to effectively build robust neural networks and certify them. However, we believe the only one providing strong robustness guarantees is the last one. Hence, we concentrate more on providing theoretical guarantees. In the next chapter, we give details on various state-of-the-art methods used for verifying and promoting robustness of NN.

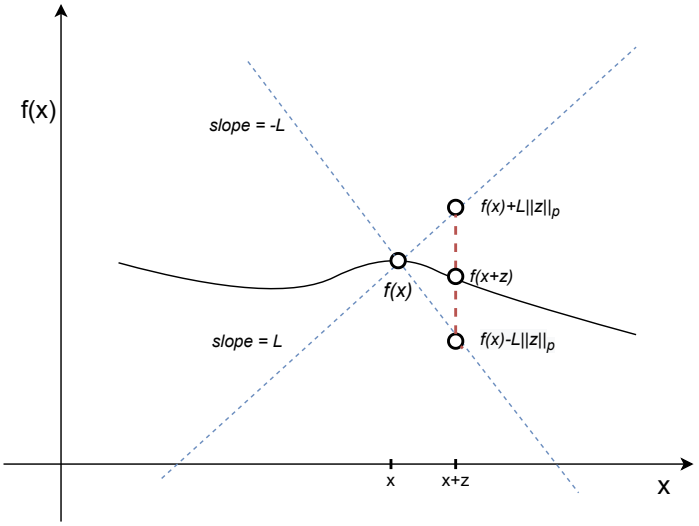

Figure 1.2: Intuition of using Lipschitz constant as a stability property of neural network [2].

Among the possible mathematical approaches, we focus on those relying upon the analysis of the Lipschitz properties of neural networks. The Lipschitz modulus of any function is defined as the upper bound on the ratio of variations in the output to the variations in the output. In the context of neural networks, it can be seen as a slope of the changes in a high dimensional space. An intuition of using Lipschitz bounds for any neural network function f is given in Figure 1.2. Very high changes or steep slope values can be sign of highly unstable models, hence not robust to input perturbations. Although neural networks not being robust can be attributed to various factors, such properties play a fundamental role in understanding the internal mechanisms governing these complex non-linear systems. Besides, these methods make very few assumptions on the type of non-linearities used and are thus valid for a wide range of networks. We base our work

and contributions on estimating and controlling the Lipschitz bounds of neural networks and hence enable engineers to develop "Stable-by-design" models which are safe under deployment phase.

1.2 . Contributions of the Thesis

The contributions presented in this thesis aim to bridge the gaps encountered in adversarial machine learning literature especially when applied to safety- and mission-critical systems. One major limitation of the existing works is related to information on the nature of adversarial attacks in regression tasks. The literature and context of adversarial attacks and their defenses move around classification tasks and notion of clear boundaries between the classes. Technically, adversarial attacks occur for neural networks when the logit of "wrong/target" class crosses the "true" class. Similarly, the defenses are designed or models are trained so that the logit of any class will be less likely to cross the logit of "true" class. More specifically, these notions are modelled using cross-entropy losses which are primarily used for classification scenarios. Regression applications cover a major chunk of industrial applications, but the adversarial perturbations are not defined as easily in this context. Also, the first industrial application presented in the thesis is a regression problem. To bridge this major gap in the literature, we provide a simple "white" box attacker for generating adversarial attacks for regression tasks. We define some metrics similar to those used in classification scenarios for checking the effectiveness of attacking the neural networks. It is worthy to note here that adversarial attacks in the literature mostly correspond to ℓ_∞ or ℓ_2 norm while the attacker designed as part of our contribution can work for various combinations of norms on input and output space. Also, we provide a way to attack only a group of inputs at a time.

The work on estimation of Lipschitz constant in the literature aims to find a single Lipschitz constant parameter that does not highlight individual contributions of the inputs in the prediction of the output. In other words, they do not provide any information on the sensitivity of the inputs. Such analysis is important so that engineers can give importance to control the perturbations arising in certain sensors over others. To assess this information on the sensitivity of the inputs, our next contribution provides a multivariate Lipschitz analysis of the neural networks by introducing the concepts of partial/weighted Lipschitz constants. As part of our contribution, we provide a representation called Lipschitz star, a graphical and practical tool to analyze the sensitivity of a neural network model during it's development, concerning different combinations of inputs.

The Lipschitz constrained neural networks often perform poorly on the clean samples if constrained to have very tight Lipschitz bounds. Such poor performance is not acceptable for a working model. For achieving the most optimally trained models it is necessary to attain a trade-off between the clean accuracy and the

Lipschitz constant value. To achieve this trade-off in our next contribution, we designed a spectral normalized stability control loop which takes care of both predefined stability and performance targets.

To support our contributions, we have validated and tested various open-source and real world applications, including a use-case provided by Thales LAS France for air traffic management and another one related to fault detection in UAVs.

1.3 . Organisation of the Thesis

This thesis is organised as follows.

i) In Chapter 2 we provide a literature review of the different strategies used to improve neural networks' stability. We have divided this chapter primarily into two parts : (a) Estimation of stability and (b) Controlling the stability of neural networks. Since the main focus of this work is based on a Lipschitz constant analysis we provide an in-depth study of works utilizing the Lipschitz properties of neural networks to improve the robustness.

ii) In Chapter 3 we focus on designing an adversarial attacker for regression tasks which are somewhat neglected in the adversarial machine learning community. We show that by utilizing the Jacobian properties of neural networks it is possible to design an attacker which generates worst attack on the sample. To achieve this, we provide some metrics for quantifying the effectiveness of the presented attacker. We show the effect of our proposed attacker on standard trained model and spectral normalized trained models which have much lower Lipschitz bounds.

iii) In Chapter 4, we present a multivariate Lipschitz analysis of neural networks. We provide the concept of partial/weighted Lipschitz constant for quantifying the sensitivity of individual inputs w.r.t to outputs. We present a new representation "Lipschitz star" to graphically represent the introduced partial Lipschitz constant. We study the effect of spectral normalization method and adversarial learning on the sensitivity of each input.

iv) In Chapter 5 we handle the issue of highly constrained Lipschitz training which leads to poor accuracy performance. Blindly training neural networks models with spectral normalized constraints to lower the Lipschitz bounds and improve stability leads to a poor performance and even sometimes the model fails to learn anything. In this part, we focus on training neural networks model within a stability-accuracy trade-off, given some specified targets. We use a spectral normalization technique for designing a stability control loop, which takes into account both accuracy and a Lipschitz target to propose an optimal model.

v) Finally, we conclude in Chapter 6 and present future work based on the contributions of this thesis.

vi) In appendix A we provide additional results related to effect of different training strategies on the robustness of neural networks for tabular data. In appendix B we provide a strategy to train robust neural networks having a specific architecture and positive weights.

Note: For this thesis we have used words stability and robustness interchangeably and they thus represent the same concept i.e. neural network's ability to handle any kind of perturbations and adversarial attacks. The stability criterion considered here highlights the fact that small perturbations in the inputs do not produce high variations of the outputs. It should be noted that the safety community has defined robustness and stability of models as two different notions. The robustness of the model is the extent to which the system can continue to operate correctly despite abnormal inputs and conditions outside the defined Operational Design Domain (ODD). In contrast, stability is the network's ability to handle small perturbations within its inputs ODD [1]. The research community working on Robust AI currently does not differentiate between the two concepts.

1.4 . List of Publications

i) Verma, S., Gupta, K. Robustness of Neural Networks used in Electrical Motor Time-Series. Workshop on Robustness in Sequence Modeling (RobustSeq), Conference on Neural Information Processing Systems (NeurIPS 2022), November 2022, New Orleans, United States [17].

ii) Gupta, K., Pesquet-Popescu, B., Kaakai, F., Pesquet, J. C. Safe Design of Stable Neural Networks for Fault Detection in Small UAVs. Workshop on Artificial Intelligence Safety Engineering (WAISE), International Conference on Computer Safety, Reliability and Security (SAFECOMP), September 2022, Munich, Germany [18]. [Runner Up Best Paper Award]

iii) Gupta, K., Kaakai, F., Pesquet-Popescu, B., Pesquet, J. C., Malliaros, F. (2022). Multivariate Lipschitz Analysis of the Stability of Neural Networks. Frontiers in Signal Processing [19].

iv) Gupta, K., Pesquet-Popescu, B., Kaakai, F.,Pesquet, J.C. , Malliaros, F. An Adversarial Attacker for Neural Networks in Regression Problems. Workshop on Artificial Intelligence Safety (AISafety), International Joint Conferences on Artificial Intelligence Organization (IJCAI), August 2021 [20]. [Runner Up Best Paper Award]

[1] This definition of model stability and robustness comes from the standardization working group EUROCAE WG-114 and SAE G34.

v) Gupta, K., Pesquet-Popescu, B., Kaakai, F., Pesquet, J. C. A Quantitative Analysis of the Robustness of Neural Networks for Tabular Data. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), June 2021 [21].

vi) Neacsu, A., Gupta, K., Pesquet, J. C., Burileanu, C. Signal denoising using a New Class of Robust Neural Networks. In European Signal Processing Conference (EUSIPCO) January 2021 [22].

1.5 . Dissemination Activities

i) Oral presentation at WAISE SAFECOMP 2022.

ii) Oral presentations at AISafety IJCAI, and ICASSP 2021.

iii) Poster presentation at DATAIA Workshop 2021 and PRAIRIE/MIAI AI summer school 2021.

1.6 . Other academic activities

i) Work on segmentation in lungs for CT-Scans of Covid-19 patients in collaboration with IGR and Owkin. The project resulted in the following publication. Lassau, N., Ammari, S., Chouzenoux, E., Gortais, H., Herent, P., Devilder, M., ...Gupta, K.. Blum, MG (2021). Integrating deep learning CT-scan model, biological and clinical variables to predict severity of COVID-19 patients. Nature communications , 12 (1), 1-11.

ii) Reviewer AISTATS 2023, AAAI 2022/2023, TCSVT, Frontiers in signal Processing, WiCV CVPR 2022, IEEE Transactions on Systems, Man and Cybernetics.

iii) Area Chair of Women in Machine Learning Workshop (WiML), NeurIPS 2022.

Chapter 2
A Survey on the Stability of Neural Networks

In this chapter we provide a literature review for various works in Robust AI domain related to verification and certification of neural networks. We concentrate on the methods utilizing the Lipschitz properties of the neural network as a formal guarantee of robustness.

2.1 . Provable Defenses

Neural networks (NNs) are being deployed in a wide range of applications, including safety- and mission-critical tasks, certifying robustness of a neural network against perturbations has become an important research topic in machine learning community. There are various approaches and strategies to test the robustness of neural networks. Traditional coverage-based approaches may be irrelevant for testing neural network systems. In such cases, code certifiability can be trivially satisfied while providing only limited guarantees on the system safe behavior when facing situations that have not been strictly met during the training process.

Deep Neural Network(DNN) verification is experimentally beyond the capabilities of tools such as Linear Programming (LP) solvers or Satisfiability Modulo Theories (SMT) solvers [23, 24]. By far, these dedicated tools have only been able to handle very small networks (a single hidden layer with 10/20 neurons [23, 25]). The difficulty in using and estimating robustness properties for DNNs is the consequence of the presence of activation functions at various parts of the architecture. An attempt towards property verification of neural networks with ReLU Activations in safety critical applications was made in [26]. The idea is to find the exact value of minimum perturbation distance (say Δ) i.e. the trained classifier is safe against any perturbations with norm less than Δ. Finding this minimum distortion/perturbation of adversarial examples with ReLU activations is known to be an NP-complete problem [27, 26]. This makes formal verification technique in Reluplex [26] computationally demanding even for small-sized NNs and hence they suffer from scalability issues. In another line of work, the piece-wise linear nature

of ReLU is exploited to efficiently compute a non-trivial certified lower bound of the minimum distortion [12, 13, 28]. Zhang et al. [14] introduced an approximation technique called CROWN for computing a certified lower bound of minimum adversarial distortion given any input data point with general activation functions beyond ReLU such as tanh, sigmoid, and arctan for larger networks.

Another popular certifiable defense against ℓ_∞ norm bounded inputs via the convex outer adversarial polytope [12] which is applicable to small networks with ReLU activation functions. Mirman et al. [29] takes a similar approach via abstract interpretation. These methods use linear relaxations of neural networks to compute an outer bound at the final layer. However, because the convex relaxations employed are relatively expensive, these methods are typically slow to train. A simple and fast certifiable defense for ℓ_∞ norm bounded inputs is Interval Bound Propagation (IBP) [30, 29]. In [30] authors demonstrated that IBP bound can be quite loose for general networks. But it can be used to train large provable NNs by using appropriate loss and clever hyper-parameters to allow the network to adapt such that the IBP bound is tight. CROWN-IBP [31] outperforms previous methods by combining IBP bound in a forward bounding pass and a tighter linear relaxation bound in a backward bound pass. Shi et al. [32] improved IBP with better initialization to accelerate training. The certification methods discussed so far provide deterministic robustness guarantees. Additionally, randomized smoothing [33, 34, 35] provides probabilistic guarantees to certify ℓ_2 norm robustness with arbitrarily high confidence. The prediction of a randomized smooth classifier is the most likely prediction returned by the base classifier that is fed by samples from a Gaussian distribution. Salman et al. [36] further improves the performance of randomized smoothing via adversarial training. All these defenses are computationally expensive and increases the run-time.

2.2 . Estimation of Stability

2.2.1 . Quantification of Lipschitz constant

An m-layered feed-forward network Figure 2.1 can be modelled by the following recursive equations:

$$(\forall i \in \{1, \ldots, m\}) \quad x_i = T_i(x_{i-1}) = R_i(W_i x_{i-1} + b_i) \tag{2.1}$$

where, at the i^{th} layer, $W_i \in \mathbb{R}^{N_i \times N_{i-1}}$ is the weight matrix, $b_i \in \mathbb{R}^{N_i}$ is the bias vector, and $R_i \colon \mathbb{R}^{N_i} \to \mathbb{R}^{N_i}$ is the activation operator. This operator may consist of the application of basic nonlinear functions, e.g. ReLU or tanh, to each component of the input. Alternatively, it may consist of a softmax operation or group sorting operations which typically arise in max pooling.

As shown in [1], the problem is related to the choice of the weight matrices. One way of quantifying the stability of the system is to calculate a Lipschitz constant of the network. A **Lipschitz constant** is an upper bound on the ratio

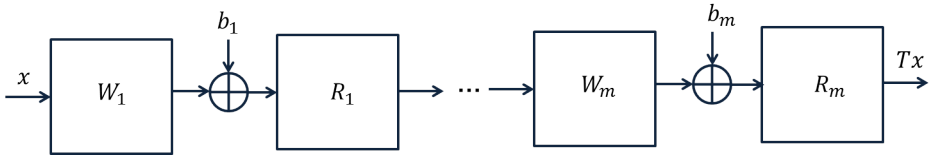

Figure 2.1: m-layered feed-forward neural network architecture. For the i^{th}-layer, W_i is the linear weight operator, b_i the bias vector, and R_i the activation operator.

between the variations of the outputs and the variations of inputs of a function T and thus it is a measure of sensitivity of the function with respect to input perturbations. This means that, if $\theta \in [0, +\infty[$ is a Lipschitz constant of T, then

$$\|T(x+z) - T(x)\| \leqslant \theta \|z\| \tag{2.2}$$

for every input $x \in \mathbb{R}^{N_0}$ and perturbation $z \in \mathbb{R}^{N_0}$. Note that the same notation is used here for the norms on $\mathbb{R}^{N_0}$ and $\mathbb{R}^{N_m}$, but different norms can be used. If not specified, the standard Euclidean norm will be used. An important assumption is that the operators $(R_i)_{1 \leqslant i \leqslant m}$ are non-expansive, i.e. 1-Lipschitz. This assumption is satisfied for all the standard choices of activation operators.

Remark 1 For every $(x, z) \in (\mathbb{R}^{N_0})^2$, if T is continuous on the segment $[x, x+z]$ and differentiable on $]x, x+z[$, it follows from the mean value inequality that

$$\|T(x+z) - T(x)\| \leqslant \|z\| \sup_{\alpha \in]0,1[} \|T'(x + \alpha(z-x))\|_S, \tag{2.3}$$

where T' is the Jacobian of T and $\|\cdot\|_S$ is the spectral norm. We deduce that, when T is differentiable on $\mathbb{R}^{N_0}$, the optimal Lipschitz constant is $\theta = \sup_{x \in \mathbb{R}^{N_0}} \|T'(x)\|_S$. According to the differentiation chain rule, if the activation operators $(R_i)_{1 \leqslant i \leqslant m}$ are differentiable, the Jacobian matrix of T at $x_0 \in \mathbb{R}^{N_0}$ is expressed as

$$T'(x_0) = R'_m(y_m) W_m R'_{m-1}(y_{m-1}) \cdots R'_1(y_1) W_1 \tag{2.4}$$

where

$$(\forall i \in \{1, \ldots, m\}) \quad y_i = W_i x_{i-1} + b_i. \tag{2.5}$$

The variables $(x_i)_{1 \leqslant i \leqslant m}$ and $(y_i)_{1 \leqslant i \leqslant m-1}$ are computed during the forward pass. It is important to note that the differentiability assumption is not satisfied by the ReLU activation function or its variants (leaky ReLU, capped ReLU,...).

Remark 2 From a practical viewpoint, it appears sometimes more relevant to determine a local Lipschitz constant of the network. Indeed, the vector of inputs of the network usually varies in a restricted subset C of $\mathbb{R}^{N_0}$. In addition, it appears reasonable to assume that the perturbations z lie in a ball $B(0, \epsilon)$ centered at 0 and with radius $\epsilon > 0$. We can thus define the local Lipschitz constant as

$$\theta_{C,\epsilon} = \sup_{x \in C} \sup_{z \in B(0,\epsilon) \setminus \{0\}} \frac{\|T(x+z) - T(x)\|}{\|z\|}. \tag{2.6}$$

From this definition, it is clear that $\theta_{C,\epsilon}$ is lower than the optimal global Lipschitz constant, whose expression is recovered as a limit case when $C = \mathbb{R}^{N_0}$ and $\epsilon \to +\infty$. Two point must however be emphasized:

i) This local Lipschitz constant may not significantly differ from the global one. To support this claim, assume that T is differentiable on $\mathbb{R}^{N_0}$ and that there exists $\widehat{x} \in C$ such that $\|T'(\widehat{x})\| = \sup_{x\in\mathbb{R}^{N_0}} \|T'(x)\|$. Then it follows from Eq. (2.3) and Eq. (2.6) that $\theta_{C,\epsilon}$ reduces to the optimal global Lipschitz constant **whatever the choice of** $\epsilon > 0$.

ii) Computing $\theta_{C,\epsilon}$ (or an upper bound of it) often is a more difficult problem than the computation of a global Lipschitz constant of T since it corresponds to solving a twice constrained maximization problem. In particular, numerical approaches can be developed where samples within the set C are drawn and adversarial examples are generated by identifying the worst choices for z for each sampled value of the input [37]. Such approaches may appear better suited for designing networks with improved robustness than for certification purposes. Alternative measures to the local Lipschitz constant have also been considered for multi-class classification problems [38].

2.2.1.1 . Trivial Upper Bound

The first upper-bound on the Lipschitz constant of a neural network was derived by analyzing the effect of each layer independently and considering a product of the resulting spectral norms [1]. This leads to the following Upper Bound:

$$\overline{\theta}_m = \|W_m\|_S \|W_{m-1}\|_S \cdots \|W_1\|_S. \tag{2.7}$$

Although easy to compute, this upper bound turns out be over-pessimistic and loose.

2.2.1.2 . Improved Bounds

In [8],the problem of computing the exact Lipschitz constant of a differentiable function is pointed out to be NP-hard. A first generic algorithm (AutoLip) for upper bounding the Lipschitz constant of any automatically differentiable function is proposed. This bound however reduces to Eq. (2.7) for standard feed-forward neural networks. Additionally, the authors proposed, an algorithm SeqLip for sequential neural networks, which shows significant improvement over AutoLip. A sequential neural network is a network for which the activation operators are separable in the sense that, for every $i \in \{1, \ldots, m\}$,

$$(\forall x_i = (\xi_{i,k})_{1\leqslant k\leqslant N_i} \in \mathbb{R}^{N_i}) \quad R_i(x) = (\rho_i(\xi_{i,k}))_{1\leqslant k\leqslant N_i}, \tag{2.8}$$

where the activation function $\rho_i \colon \mathbb{R} \to \mathbb{R}$.[1] In [8], it is assumed that the functions $(\rho_i)_{1\leqslant i\leqslant m}$ are differentiable, increasing, and their derivative are upper bounded by one. By using Eq. (2.4), one can conclude that a Lipschitz constant of the network is

$$\vartheta_m = \sup_{\Lambda_1\in\mathcal{D}_{N_1}([0,1]),\ldots,\Lambda_{m-1}\in\mathcal{D}_{N_{m-1}}([0,1])} \|W_m\Lambda_{m-1}\cdots\Lambda_1 W_1\|_S, \tag{2.9}$$

[1] More generally, a function $\rho_{i,k}$ can be applied to each component $\xi_{i,k}$ but this situation rarely happens in standard neural networks.

where $\mathcal{D}_N(I)$ designate the set of diagonal matrices of dimension $N \times N$ with diagonal values in $I \subset \mathbb{R}$. This bound simplifies as

$$\vartheta_m = \sup_{\Lambda_1 \in \mathcal{D}_{N_1}(\{0,1\}),\ldots,\Lambda_{m-1} \in \mathcal{D}_{N_{m-1}}(\{0,1\})} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\|_S, \tag{2.10}$$

which shows that 2^{N_i} values of the diagonal elements of matrix Λ_i have to be tested at each layer $i \in \{1,\ldots,m\}$, so that the global complexity amounts to $2^{N_1+\cdots+N_{m-1}}$ and thus grows exponentially as a function of the number of neurons. Estimating the Lipschitz constant using this method is intractable even for medium-size networks and authors use a greedy algorithm to compute a bound, which may under-approximate the Lipschitz constant. This does not provide tight upper bounds.

2.2.1.3 . CPLip

In [9] various bounds on the Lipschitz constant of a feed-forward network are derived by assuming that, for every $i \in \{1,\ldots,m\}$ the activation operator R_i is α_i-averaged with $\alpha_i \in]0,1]$. We recall this means that there exists a non-expansive (i.e. 1-Lipschitz) operator Q_i such that $R_i = (1-\alpha_i)\mathrm{Id} + \alpha_i Q_i$. The following inequality is then satisfied

$$(\forall (x,y) \in \mathbb{R}^{N_i}) \|R_i x - R_i y\|^2 \leqslant \|x-y\|^2 - \frac{1-\alpha_i}{\alpha_i} \|(\mathrm{Id}-R_i)x - (\mathrm{Id}-R_i)y\|^2. \tag{2.11}$$

We thus see that the smaller α_i, the more "stable" R_i. In the limit case when $\alpha_1 = 1$, R_i is non-expansive and, when $\alpha_i = 1/2$, R_i is said to be firmly non-expansive. An important sub-class of firmly non-expansive operators is the class of proximity operators of convex functions which are proper and lower-semicontinuous. Let $\Gamma_0(\mathbb{R}^N)$ the be the class of such function defined from $\mathbb{R}^N$ to $]-\infty,+\infty]$. The proximity operator of a function $f \in \Gamma_0(\mathbb{R}^N)$, at some point $x \in \mathbb{R}^N$, is the unique vector denoted by $\mathrm{prox}_f(x)$ such that

$$\mathrm{prox}_f(x) = \underset{p \in \mathbb{R}^N}{\mathrm{argmin}} \ \frac{1}{2}\|p-x\|^2 + f(p). \tag{2.12}$$

The proximity operator is a fundamental tool in convex optimization. As shown in [39], most of the activation functions (sigmoid, ReLU, leaky ReLU, ELU,...) currently used in neural networks are the proximity operators of proper lower-semi-continuous convex functions. This property is also satisfied by activation operators which are not separable like softmax or the squashing function used in capsule networks. The few activation operators which are not proximity operators (e.g. convex combinations of a max pooling and an average pooling) can be viewed as over-relaxations of proximity operators and correspond to a value of the averaging parameter greater than $1/2$.

Based on these averaging assumptions, a first estimation of the Lipschitz constant is given by

$$\theta_m = \beta_{m;\bar{p}} \|W_m \circ \cdots \circ W_1\| + \sum_{k=1}^{m-1} \sum_{(j_1,\ldots,j_k)\in\mathbb{J}_{m,k}} \beta_{m;\{j_1,\ldots,j_k\}} \sigma_{m;\{j_1,\ldots,j_k\}}, \tag{2.13}$$

where

$$(\forall\, \mathbb{J} \subset \{1,\ldots,m-1\}) \quad \beta_{m;\mathbb{J}} = \left(\prod_{j\in\mathbb{J}} \alpha_j\right) \prod_{j\in\{1,\ldots,m-1\}\smallsetminus\mathbb{J}} (1-\alpha_j), \tag{2.14}$$

for every $k \in \{1,\ldots,m-1\}$,

$$\mathbb{J}_{m,k} = \begin{cases} \big\{(j_1,\ldots,j_k) \in \mathbb{N}^k \;\big|\; 1 \leqslant j_1 < \cdots < j_k \leqslant m-1\big\}, & \text{if } k > 1; \\ \{1,\ldots,m-1\}, & \text{if } k = 1 \end{cases} \tag{2.15}$$

and, for every $(j_1,\ldots,j_k) \in \mathbb{J}_{m,k}$,

$$\sigma_{m;\{j_1,\ldots,j_k\}} = \|W_m \cdots W_{j_k+1}\|_{\mathrm{S}} \, \|W_{j_k} \cdots W_{j_{k-1}+1}\|_{\mathrm{S}} \cdots \|W_{j_1} \cdots W_1\|_{\mathrm{S}}. \tag{2.16}$$

When, for every $i \in \{1,\ldots,m-1\}$, R_i is firmly nonexpansive, the expression simplifies as

$$\theta_m = \frac{1}{2^{m-1}} \left(\|W_m \cdots W_1\|_{\mathrm{S}} + \sum_{k=1}^{m-1} \sum_{(j_1,\ldots,j_k)\in\mathbb{J}_{m,k}} \sigma_{m;\{j_1,\ldots,j_k\}} \right). \tag{2.17}$$

If, for every $i \in \{1,\ldots,m-1\}$, R_i is separable,[2] a second estimation is provided which reads

$$\vartheta_m = \sup_{\substack{\Lambda_1\in\mathcal{D}_{N_1}(\{2\alpha_1-1,1\}), \\ \vdots, \\ \Lambda_{m-1}\in\mathcal{D}_{N_{m-1}}(\{2\alpha_{m-1}-1,1\})}} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\|_{\mathrm{S}}, \tag{2.18}$$

We thus see that, when $\alpha_1 = \cdots = \alpha_{m-1} = 1/2$, we recover Eq. (2.10) without making any assumption on the differentiability of the activation functions. This estimation is more accurate than the previous one in the sense that

$$\|W_m \cdots W_1\|_{\mathrm{S}} \leqslant \vartheta_m \leqslant \theta_m. \tag{2.19}$$

It is proved in [9] that, if the network is with non-negative weights, that is $(\forall i \in \{1,\ldots,m\})$ $W_i \in [0,+\infty[^{N_i\times N_i}$, the lower bound in Eq. (2.19) is attained, i.e.

$$\vartheta_m = \|W_m \cdots W_1\|_{\mathrm{S}}. \tag{2.20}$$

[2] The result remains valid if different scalar activation functions are used in a given layer.

Another interesting result which is established in [9] is that similar results hold if other norms than the Euclidean norm are used to quantify the perturbations on the input and the output. For example, for a given $i \in \{1, \dots, m\}$, for every $p \in [1, +\infty]$, we can define the following norm:

$$\left(\forall x_i = (\xi_{i,k})_{1 \leqslant k \leqslant N_i} \in \mathbb{R}^{N_i}\right) \quad \|x\|_p = \begin{cases} \left| \sum_{k=1}^{N_i} |\xi_{i,k}|^p \right|^{1/p}, & \text{if } p < +\infty \\ \sup_{1 \leqslant k \leqslant N_i} |\xi_{i,k}|, & \text{if } p = +\infty. \end{cases} \tag{2.21}$$

If $(p, q) \in [1, +\infty]^2$, the input space $\mathbb{R}^{N_0}$ is equipped with the norm $\|\cdot\|_p$, and the output space $\mathbb{R}^{N_m}$ is equipped with the norm $\|\cdot\|_q$ a Lipschitz constant for a network with separable activation operators is

$$\vartheta_m = \sup_{\substack{\Lambda_1 \in \mathcal{D}_{N_1}([2\alpha_1 - 1, 1]), \\ \vdots, \\ \Lambda_{m-1} \in \mathcal{D}_{N_{m-1}}([2\alpha_{m-1} - 1, 1])}} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\|_{p,q} \tag{2.22}$$

$$= \sup_{\substack{\Lambda_1 \in \mathcal{D}_{N_1}(\{2\alpha_1 - 1, 1\}), \\ \vdots, \\ \Lambda_{m-1} \in \mathcal{D}_{N_{m-1}}(\{2\alpha_{m-1} - 1, 1\})}} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1\|_{p,q}, \tag{2.23}$$

where $\|\cdot\|_{p,q}$ is the subordinate $L_{p,q}$ matrix norm induced by the two previous norms. The ability to use norms other than the Euclidean one may be sometimes more meaningful in practice (especially for the ℓ_1 or the sup norm). However, computing such a subordinate norm is not always easy [40].

2.2.1.4 . SDP based approach

The work in [16] focuses on neural networks using separable activation operators. It assumes that the activation function ρ_i used at a layer $i \in \{1, \dots, m\}$ is slope-bounded, i.e. there exists non-negative parameters $\sigma_{\min}$ and $\sigma_{\max}$ such that

$$\left(\forall (\xi, \xi') \in \mathbb{R}^2\right) \qquad \xi \neq \xi' \Rightarrow \sigma_{\min} \leqslant \frac{\rho_i(\xi) - \rho_i(\xi')}{\xi - \xi'} \leqslant \sigma_{\max}.$$

As noted by the authors, most activation functions satisfy this inequality with $\sigma_{\min} = 0$ and $\sigma_{\max} = 1$. In other words, the above inequality means then that ρ_i is an increasing function and it is non-expansive. But a known result [41, Proposition 2.4] states that a function ρ_i satisfies these properties if and only if it is the proximity operator of some proper lower-semicontinuous convex function. So it turns out that we recover similar assumptions to those made in [39]. By use of the firm non-expansivity of the operators R_i

Let us thus assume that $\sigma_{\min} = 0$, $\sigma_{\max} = 1$, and $m \geqslant 2$. It is well-known that R_i is firmly nonexpansive if and only if

$$\left(\forall (x, y) \in (\mathbb{R}^{N_i})^2 \quad (x - y)^\top (R_i x - R_i y) \geqslant \|R_i x - R_i y\|^2. \right. \tag{2.24}$$

The point is that, if R_i is a separable operator, this inequality holds in a more general metric associated with a matrix

$$Q_i = \mathrm{Diag}(q_{i,1,1}, \dots, q_{i,N_i,N_i}), \tag{2.25}$$

where $(\forall k \in \{1, \dots, N_i\}^2)\ q_{i,k,k} \geqslant 0$. In the following, the set of such matrices $(Q_i)_{1 \leqslant i \leqslant m-1}$ will be denoted by $\mathcal{Q}$. This means that

$$(\forall (x,y) \in (\mathbb{R}^{N_i})^2 \quad (x-y)^\top Q_i (R_i x - R_i y) \geqslant (R_i x - R_i y)^\top Q_i (R_i x - R_i y). \tag{2.26}$$

For every $(x_i, y_i) \in (\mathbb{R}^{N_i})^2$, let $x_i = R_i(W_i x_{i-1} + b_i)$ and $y_i = R_i(W_i y_{i-1} + b_i)$. It follows from Eq. (2.26) that

$$\big(W_i(x_{i-1} - y_{i-1})\big)^\top Q_i (x_i - y_i) \geqslant (x_i - y_i)^\top Q_i (x_i - y_i). \tag{2.27}$$

Summing for the first $m-1$ layers yields

$$\sum_{i=1}^{m-1} \big(W_i(x_{i-1} - y_{i-1})\big)^\top Q_i (x_i - y_i) \geqslant \sum_{i=1}^{m-1} (x_i - y_i)^\top Q_i (x_i - y_i). \tag{2.28}$$

On the other hand, $\vartheta_m > 0$ is a Lipschitz constant of the neural network T if

$$\vartheta_m^2 \|x_0 - y_0\|^2 \geqslant \|W_m(x_{m-1} - y_{m-1})\|^2. \tag{2.29}$$

For the latter inequality to hold, it is thus sufficient to ensure that,

$$\begin{aligned} &\vartheta_m^2 \|x_0 - y_0\|^2 - \|W_m(x_{m-1} - y_{m-1})\|^2 \\ &\geqslant 2 \sum_{i=1}^{m-1} \big(W_i(x_{i-1} - y_{i-1})\big)^\top Q_i (x_i - y_i) - 2 \sum_{i=1}^{m-1} (x_i - y_i)^\top Q_i (x_i - y_i). \end{aligned} \tag{2.30}$$

This inequality can be rewritten in matrix form as

$$\begin{bmatrix} x_0 - y_0 \\ \vdots \\ x_{m-1} - y_{m-1} \end{bmatrix}^\top M(\rho_m, Q_1, \dots, Q_{m-1}) \begin{bmatrix} x_0 - y_0 \\ \vdots \\ x_{m-1} - y_{m-1} \end{bmatrix} \geqslant 0 \tag{2.31}$$

with $\rho_m = \vartheta_m^2$ and

$$M(\rho_m, Q_1, \dots, Q_{m-1}) = \begin{bmatrix} \rho_m \mathrm{Id}_{N_0} & -W_1^\top Q_1 & & & 0 \\ -Q_1 W_1 & 0 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & 0 & -W_{m-1}^\top Q_{m-1} \\ 0 & & & -Q_{m-1} W_{m-1} & 2Q_{m-1} - W_m^\top W_m \end{bmatrix}. \tag{2.32}$$

In the case of a network having just one hidden layer, which is mainly investigated in [16], the above matrix reduces to

$$M(\rho_2, Q_1) = \begin{bmatrix} \rho_2 \mathrm{Id}_{N_0} & -W_1^\top Q_1 \\ -Q_1 W_1 & 2Q_1 - W_2^\top W_2 \end{bmatrix}. \tag{2.33}$$

Condition (2.31) is satisfied, for every $(x_0, \ldots, x_{m-1})$ and $(y_0, \ldots, y_{m-1})$ if and only if

$$M(\rho_m, Q_1, \ldots, Q_{m-1}) \succeq 0. \tag{2.34}$$

It is actually sufficient that this positive semidefiniteness constraint be satisfied for any matrix $Q \in \mathcal{Q}_{N_i}$ for $\sqrt{\rho}_m$ to be a Lipschitz constant. The smallest possible value of the resulting constant can be obtained by solving the following semidefinite positive programming (SDP) problem:

$$\underset{(\rho_m, Q_1, \ldots, Q_{m-1}) \in C}{\text{minimize}} \quad \rho_m, \tag{2.35}$$

where C is the closed convex set

$$C = \big\{(\rho_m, Q_1, \ldots, Q_{m-1}) \in [0, +\infty[\times \mathcal{Q} \;\big|\; \text{Eq. (2.34) holds}\big\}. \tag{2.36}$$

Although there exists efficient SDP solvers, the method remains computationally intensive. A solution to reduce its computational complexity at the expense of a lower accuracy consists of restricting the optimization of the metric matrices $Q_1, \ldots, Q_{m-1}$ to a subset of $\mathcal{Q}$.

One limitation of this method is that it is taylored to the use of the Euclidean norm.

Remark 3 In [16], it is claimed that Eq. (2.26) is valid for every metric matrix

$$Q_i = \sum_{k=1}^{N_i} q_{i,k,k} e_k e_k^\top + \sum_{1 \leqslant k < \ell \leqslant N_i} q_{i,k,l} (e_k - e_\ell)(e_k - e_\ell)^\top, \tag{2.37}$$

where $(e_k)_{1 \leqslant k \leqslant N_i}$ is the canonical basis of $\mathbb{R}^{N_i}$ and $(\forall (k, \ell) \in \{1, \ldots, N_i\}^2)$ with $k \leqslant \ell$, $q_{i,k,\ell} \geqslant 0$ Unfortunately, this turns out to be incorrect as shown by the counterexample next. Assume that all the coefficients $(q_{i,k,\ell})_{1 \leqslant k \leqslant \ell \leqslant N_i}$ are zero, except one in the second summation set to 1. We have then

$$Q = (e_k - e_\ell)(e_k - e_\ell)^\top \tag{2.38}$$

for some $(k, \ell) \in \{1, \ldots, N_i\}^2$ with $k < \ell$. Consider the simple case when ρ_i is the ReLU function, that is

$$(\forall \xi \in \mathbb{R}) \quad \rho_i(\xi) = \max\{\xi, 0\}.$$

ρ_i is an example of slope-bounded function with $\sigma_{\min} = 0$ and $\sigma_{\max} = 1$. Then, for every $x = (\xi_k)_{1 \leqslant k \leqslant N_i} \in \mathbb{R}^{N_i}$ and $y = (v_k)_{1 \leqslant k \leqslant N_i} \in \mathbb{R}^{N_i}$,

$$\begin{aligned} (e_k - e_\ell)^\top (x - y) &= \xi_k - \xi_\ell - v_k + v_\ell \\ (e_k - e_\ell)^\top (R_i(x) - R_i(y)) &= \rho_i(\xi_k) - \rho_i(\xi_\ell) - \rho_i(v_k) + \rho_i(v_\ell). \end{aligned} \tag{2.39}$$

We have thus

$$(x-y)^\top Q(R_i x - R_i y) = (\xi_k - \xi_\ell - v_k + v_\ell)(\rho_i(\xi_k) - \rho_i(\xi_\ell) - \rho_i(v_k) + \rho_i(v_\ell))$$
$$(R_i x - R_i y)^\top Q(R_i x - R_i y) = (\rho_i(\xi_k) - \rho_i(\xi_\ell) - \rho_i(v_k) + \rho_i(v_\ell))^2. \tag{2.40}$$

If we now assume that $\xi_k < 0$, $v_k = 0$, and $v_\ell > \xi_\ell > 0$, we deduce from the expression of the ReLU function that

$$(x-y)^\top Q(R_i x - R_i y) = (\xi_k + v_\ell - \xi_\ell)(v_\ell - \xi_\ell)$$
$$(R_i x - R_i y)^\top Q(R_i x - R_i y) = (v_\ell - \xi_\ell)^2. \tag{2.41}$$

Inequality Eq. (2.26) then becomes

$$\xi_k + v_\ell - \xi_\ell \geqslant v_\ell - \xi_\ell, \tag{2.42}$$

which contradicts the fact that ξ_k has been chosen negative.

The erroneous statement comes from a flaw in the deduction of Lemma 1 from Lemma 2 in [16]. Another counterexample was also recently provided in [42].

2.2.1.5 . Polynomial optimization based approach

The approach in [43] applies to neural networks having a single output (i.e. $N_m = 1$)[3]. The authors state that their approach is restricted to differentiable activation functions, but it is actually valid for any separable firmly non-expansive activation operators. Indeed, when $N_m = 1$, the Lipschitz constant in Eq. (2.22) reduces to

$$\vartheta_m = \sup_{\substack{\Lambda_1 \in \mathcal{D}_{N_1}([0,1]), \\ \vdots, \\ \Lambda_{m-1} \in \mathcal{D}_{N_{m-1}}([0,1])}} \|W_1^\top \Lambda_1 \cdots \Lambda_{m-1} W_m^\top\|_{p^*}, \tag{2.43}$$

where $p^* \in [1, +\infty]$ is the dual exponent of p (such that $1/p + 1/p^* = 1$). Recall that $p \in [1, +\infty]$ is the exponent of the ℓ_p-norm equipping the input space. This shows that ϑ_m is equal to

$$\vartheta_m = \sup\{\Phi(x, \lambda_1, \ldots, \lambda_{m-1}) \mid \|x\|_p \leqslant 1, (\lambda_i)_{1 \leqslant i \leqslant m-1} \in [0,1]^{N_1 + \cdots + N_{m-1}}\}, \tag{2.44}$$

where, for every $x \in \mathbb{R}^{N_0}$ and $(\lambda_i)_{1 \leqslant i \leqslant m} \in \mathbb{R}^{N_1 + \cdots + N_{m-1}}$,

$$\Phi(x, \lambda_1, \ldots, \lambda_{m-1}) = x^\top W_1^\top \operatorname{Diag}(\lambda_1) \cdots \operatorname{Diag}(\lambda_{m-1}) W_m^\top. \tag{2.45}$$

Function Φ is a multivariate polynomial of the components of its vector arguments. Therefore if the unit ball associated with the ℓ_p norm can be described via polynomial inequalities, which happens when $p \in \mathbb{N} \setminus \{0\}$ and $p = +\infty$, then finding ϑ_m turns out to be a polynomial constrained optimization problem. Solving such an

[3] This can be extended to multiple output network, if the output space is equipped with the $\ell_{+\infty}$ norm.

optimization problem can be achieved by solving a hierarchy of convex problems. However, the size of the hierarchy tends to grow fast and if the order of the hierarchy is truncated to a too small value, the delivered result becomes inaccurate. Leveraging the sparsity properties that might exist for the weight matrices may be helpful numerically. Note that the approach is further improved in [44] by using Lasserre's hierarchy.

Remark 4

i) Any polynomial optimization problem can be reduced to a quadratically constrained quadratic program. This approach is followed in [45] and the problem is then solved via an SDP relaxation. This leads to an upper bound on ϑ_m.

ii) In [43], the authors argue that their method allows a local Lipschitz constant to be computed. Indeed, in the case when the activation functions are differentiable, Eq. (2.4) holds where

$$(\forall i \in \{1, \dots, m-1\}) \quad R_i'(y_i) = \mathrm{Diag}(\rho_i'(v_{i,1}), \dots, \rho_i'(v_{i,N_i})) \tag{2.46}$$

and $y_i = (v_{i,k})_{1 \leqslant k \leqslant N_i}$ is defined by Eq. (2.5). Consequently, if the input x of the network belongs to a bounded convex set C, then, for every $i \in \{1 \dots, m-1\}$ y_i varies in a bounded set too, which implies that $(\rho_i'(v_{i,k}))_{1 \leqslant k \leqslant N_i}$ belongs to a polytope C_i (say a hyperrectangle) of $[0,1]^{N_i}$. This allows us to define a local Lipschitz constant as

$$\vartheta_{m,C} = \sup\{\Phi(x, \lambda_1, \dots, \lambda_{m-1}) \mid \|x\|_p \leqslant 1, (\lambda_i)_{1 \leqslant i \leqslant m-1} \in C_1 \times \cdots \times C_{m-1}\}. \tag{2.47}$$

Computing this constant is still a a polynomial optimization problem. So, the same approach as in the global case can be followed, with an extra effort for the determination of polytopes $(C_i)_{1 \leqslant i \leqslant m-1}$.

2.2.2 . Local Lipschitz Bound

As discussed briefly in Remark 2 local definitions of Lipschitz bounds are also possible. Few works in literature utilize local Lipschitz bounds to improve the robustness of the NNs. The advantage of using a local bound is that we may theoretically expect tighter bounds since of course the local Lipschitz constant cannot be any larger than the global Lipschitz constant. However, using a local bound also has its cons. Firstly, a local bound is more computationally expensive, as each instance has its own bound, hence the required memory grows with the batch size. This in turn reduces the amount of parallelism that can be exploited when using a local bound, reducing the model throughput. Furthermore, because the local Lipschitz constant is different for every point, it must be computed every time the network sees a new point. By contrast, the global bound can be computed in advance, meaning that verification via the global bound is essentially free. This makes the global bound advantageous, assuming that it can be effectively leveraged for verification. Finding an exact local Lipschitz constant for a neural network is generally NP-hard [8], so most works focus on finding a sound upper bound. Recent work has explored methods for obtaining upper bounds on the local Lipschitz

constant [28, 14, 46]. In [2], the authors convert the robustness analysis problem into a local Lipschitz constant estimation problem, where they estimate this local constant by a set of independently and identically sampled local gradients. This algorithm is scalable but is not guaranteed to provide upper bounds. In a similar work, the authors of [28] exploit the piece-wise linear structure of ReLU activation functions to estimate the local Lipschitz constant of neural networks. In [47], the authors use quadratic constraints and semidefinite programming to analyze local (point-wise) robustness of neural networks. Hein et al. [48] derived an analytical bound for two layer neural networks and found that local Lipschitz bounds could be much tighter than the global one and provide better robustness certificates. RecurJac [49] is a recursive algorithm that analyzes the local Lipschitz constant in a neural network using a bound propagation [31] based approach. FastLip [28] is a special and weaker form of RecurJac. Jordan et al. [50] formulated the computation of the local Lipschitz constant as a Mixed Integer Linear Programming (MILP) problem and they were able to solve it. Although the authors of these approaches claim that they can obtain reasonably tight and sound local Lipschitz constants, none of them have been demonstrated effective for training a certifiably robust network, which requires high efficiency and scalability.

2.2.3 . Dealing with scalability issues

To handle scalability issues arising when estimating a Lipschitz constant of a deep or broad network, several techniques can be employed as we show next.

2.2.3.1 . Serial splitting of the network

A feed-forward network can be decomposed as a cascade of smaller sub-networks $(U_j)_{1\leqslant j\leqslant k}$, where the j-th sub-network comprises m_j weight operators (hence, $\sum_{j=1}^{k} m_j = m$). Let θ_{U_j,m_j} be a Lipschitz constant of the j-th sub-network. An overall Lipschitz constant for T is given by

$$\theta_{T,m} = \prod_{j=1}^{k} \theta_{U_j,m_j}. \tag{2.48}$$

Various decompositions can be performed which lead to more or less tight bounds. For example, if we constrain the size of the sub-networks to be equal to $m_{\max} \leqslant m$, except possibly the first and the last one, the number of possible decompositions is equal to $m_{\max}$ while the maximum number of sub-networks in a decomposition is equal to $\lfloor m/m_{\max} \rfloor + 1$. If $m = 7$ and $m_{\max} = 3$, we can choose

$$\begin{cases} U_1 = T_1 \\ U_2 = T_4 \circ T_3 \circ T_2 \\ U_3 = T_7 \circ T_6 \circ T_5 \end{cases}, \tag{2.49}$$

$$\begin{cases} U_1 = T_2 \circ T_1 \\ U_2 = T_5 \circ T_4 \circ T_3 \\ U_3 = T_7 \circ T_6 \end{cases}, \tag{2.50}$$

or

$$\begin{cases} U_1 = T_3 \circ T_2 \circ T_1 \\ U_2 = T_6 \circ T_5 \circ T_4 \\ U_3 = T_7 \end{cases} . \tag{2.51}$$

If we look at the complexity reduction based on the computation of Eq. (2.23), we see that for the j-th sub-network, the computational complexity is of the order of $2^{N_{U_j}}$ where N_{U_j} is the number of neurons in its hidden layers. For the previous example splittings, the complexity are thus reduced to $2^{N_2+N_3} + 2^{N_5+N_6}$, $2^{N_1} + 2^{N_3+N_4} + 2^{N_6}$, and $2^{N_1+N_2} + 2^{N_4+N_5}$, respectively (to be compared with $2^{N_1+\cdots+N_6}$).

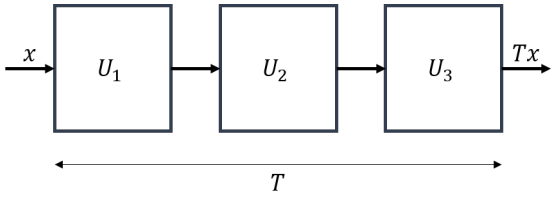

Figure 2.2: A network T is split into three sub-networks U_1, U_2, and U_3.

An additional degree of flexibility consists of factorizing the weight matrix W_i at a given layer i as

$$W_i = \widetilde{V}_i V_i \tag{2.52}$$

where $\widetilde{V}_i \in \mathbb{R}^{N_i \times M_i}$ and $V_i \in \mathbb{R}^{M_i \times N_{i-1}}$. Such a factorization may result from a singular value decomposition of W_i. In [8], it is proposed to perform such a decomposition and to consider $k = m-1$ sub-networks having one hidden layer, of the form

$$(\forall i \in \{1, \ldots, m-2\}) \quad U_i = V_{i+1} \circ R_i \circ (\widetilde{V}_i \cdot + b_i) \tag{2.53}$$

$$U_{m-1} = R_m \circ (V_m \cdot + b_m) \circ R_{m-1} \circ (\widetilde{V}_{m-1} \cdot + b_{m-1}) \tag{2.54}$$

with $\widetilde{V}_1 = W_1$ and $V_m = W_m$. This yields a Lipschitz constant of the form Eq. (2.48) where

$$(\forall i \in \{1, \ldots, m-1\}) \quad \theta_{U_i,N_i} = \sup_{\Lambda_i \in \mathcal{D}_{N_i}} \|V_{i+1} \Lambda_i \widetilde{V}_i\|_{p,q}, \tag{2.55}$$

thus inducing a global computational complexity of the order of $2^{N_1} + \cdots + 2^{N_{m-1}}$.

2.2.3.2 . Parallel splitting of the network

When the number of neurons at a given layer is large, a parallel splitting of the network may be preferable. This technique mainly applies to the computation of Eq. (2.23). The basic idea consists of block-decomposing the matrix $\Lambda_i \in \mathcal{D}_{N_i}$ associated with the activation operator used at layer $i \in \{1, \ldots, m-1\}$ in J_i

diagonal matrices $\Lambda_{i,j} \in \mathcal{D}_{N_{i,j}}$ with $j \in \{1,\ldots,J_i\}$. For every $i \in \{1,\ldots,m-1\}$, we have thus

$$\Lambda_i = \begin{bmatrix} \Lambda_{i,1} & 0 & \ldots & 0 \\ 0 & \Lambda_{i,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & \Lambda_{i,J_i} \end{bmatrix} \tag{2.56}$$

where the j-th block contains a reduced number $N_{i,j}$ of diagonal elements. Then $\sum_{j=1}^{J_i} N_{i,j} = N_i$. Set $J_m = 1$, $J_0 = 1$, $N_{m,1} = N_m$, and $N_{0,1} = N_0$. For every $i \in \{1,\ldots,m\}$, each weight matrix can be similarly block-decomposed as

$$W_i = \begin{bmatrix} W_{i,1,1} & \ldots & W_{i,1,J_{i-1}} \\ \vdots & & \vdots \\ W_{i,J_i,1} & \ldots & W_{i,J_i,J_{i-1}} \end{bmatrix} \tag{2.57}$$

where, for every $j' \in \{1,\ldots,J_i\}$ and $j \in \{1,\ldots,J_{i-1}\}$, $W_{i,j,j'}$ is a matrix of dimension $N_{i,j'} \times N_{i-1,j}$. We have then

$$\begin{aligned} &W_m \Lambda_{m-1} W_{m-1} \cdots W_2 \Lambda_1 W_1 \\ &= \sum_{j_1=1}^{J_1} \cdots \sum_{j_{m-1}=1}^{J_{m-1}} W_{m,1,j_{m-1}} \Lambda_{m-1,j_{m-1}} W_{m-1,j_{m-1},j_{m-2}} \cdots W_{2,j_2,j_1} \Lambda_{1,j_1} W_{1,j_1,1} \end{aligned} \tag{2.58}$$

By using now the triangle inequality, we deduce that

$$\begin{aligned} &\|W_m \Lambda_{m-1} W_{m-1} \cdots W_2 \Lambda_1 W_1\|_{p,q} \\ &\leqslant \sum_{j_1=1}^{J_1} \cdots \sum_{j_{m-1}=1}^{J_{m-1}} \|W_{m,1,j_{m-1}} \Lambda_{m-1,j_{m-1}} W_{m-1,j_{m-1},j_{m-2}} \cdots W_{2,j_2,j_1} \Lambda_{1,j_1} W_{1,j_1,1}\|_{p,q}. \end{aligned}$$

This leads to

$$\vartheta_m \leqslant \sum_{j_1=1}^{J_1} \cdots \sum_{j_{m-1}=1}^{J_{m-1}} \vartheta_{m,j_1,\ldots,j_{m-1}}, \tag{2.59}$$

where

$$\begin{aligned} &\vartheta_{m,j_1,\ldots,j_{m-1}} \\ &= \sup_{\substack{\Lambda_{1,j_1} \in \mathcal{D}_{N_{1,j_1}}(\{2\alpha_1-1,1\}), \\ \vdots, \\ \Lambda_{m-1,j_{m-1}} \in \mathcal{D}_{N_{m-1,j_{m-1}}}(\{2\alpha_{m-1}-1,1\})}} \|W_{m,1,j_{m-1}} \Lambda_{m-1,j_{m-1}} \cdots \Lambda_{1,j_1} W_{1,j_1,1}\|_{p,q}. \end{aligned} \tag{2.60}$$

The complexity of the computation of the latter constant is of the order of $2^{N_{1,j_1}+\cdots+N_{m-1,j_{m-1}}}$. The complexity of the Lipschitz constant defined by the upper-bound in Eq. (2.59) thus amounts to

$$\sum_{j_1=1}^{J_1} \cdots \sum_{j_{m-1}=1}^{J_{m-1}} 2^{N_{1,j_1}+\cdots+N_{m-1,j_{m-1}}}$$

For example, if for every layer $i \in \{1, \dots, m-1\}$, a decomposition in 2 blocks of equal size is performed (i.e. $J_i = 2$ and $N_{i,1} = N_{i,2} = N_i/2 \in \mathbb{N}$), the resulting complexity is $2^{(N_1+\dots+N_{m-1})/2+m-1}$.

As an illustration, when $m = 2$, we get

$$W_1 = \begin{bmatrix} W_{1,1,1} \\ W_{1,2,1} \end{bmatrix} \tag{2.61}$$

$$W_2 = [W_{2,1,1} \;\; W_{2,1,2}]. \tag{2.62}$$

By omitting redundant indices these decompositions can be rewritten as

$$W_1 = \begin{bmatrix} W_{1,1} \\ W_{1,2} \end{bmatrix} \tag{2.63}$$

$$W_2 = [W_{2,1} \;\; W_{2,2}] \tag{2.64}$$

and Eq. (2.59) becomes

$$\begin{aligned} \vartheta_m \leqslant & \sup_{\Lambda_{1,1} \in \mathcal{D}_{N_{1,1}}(\{2\alpha_1 - 1, 1\})} \|W_{2,1} \Lambda_{1,1} W_{1,1}\|_{p,q} \\ & + \sup_{\Lambda_{1,2} \in \mathcal{D}_{N_{1,2}}(\{2\alpha_1 - 1, 1\})} \|W_{2,2} \Lambda_{1,2} W_{1,2}\|_{p,q}. \end{aligned} \tag{2.65}$$

2.2.3.3 . Serial and parallel splitting of the network

A combination of the two previous splitting techniques can be used for an efficient estimation of the Lipschitz constant of a feed-forward neural network.

2.3 . Control of Stability

2.3.1 . Adversarial attacks and defenses

2.3.1.1 . Adversarial attacks

Classification: Szegedy et al. [3] proposed L-BFGS (Limited-memory; Broyden, Fletcher, Goldforb, Shanno) to construct adversarial attacks and since then, there has been a plethora of works introducing various adversarial attacks and their defenses for DNNs. Goodfellow et al. [1] proposed a simpler and faster method to construct adversarial attacks (FGSM- Fast Gradient Sign Method). The generated images are misclassified by adding perturbations and linearizing the cost function in the gradient direction. This is a non-iterative attack; hence it has a lower computation cost than the previous method. The Fast Gradient Sign Method (FGSM) is an ℓ_∞ bounded attack and is often prone to label leaking.

It may be difficult for FGSM to control the perturbation level in constructing attacks. Kurakin et al. [10] proposed an optimized FGSM, termed Iterative Gradient Sign Method (IGSM), which adds perturbations in multiple smaller steps and clips the results after each iteration ensuring that the perturbations are restricted to the neighborhood of the example. [51] added momentum to IGSM attacks.

[52] proposed the Jacobian-based Saliency Map Attack (JSMA), based on the ℓ_0 sparsity measure. The basic idea is to construct a saliency map with the gradients and model the gradients based on the impact of each image pixel.

Moosavi et al. [53] proposed a non-targeted attack method based on the ℓ_2-norm, called DeepFool. It tries to find the decision boundary closest to the sample in the image space and then uses the classification boundary to fool the classifier. FGSM, JSMA, and DeepFool are designed to generate adversarial attacks corresponding to a single image to fool the trained classifier model. Moosavi et al. [54] proposed a universal image-agnostic perturbation attack method that deceives the classifier by adding a single perturbation to all images in the dataset. Carlini et al. [55] proposed a powerful attack based on L-BFGS. The attack can be generated according to ℓ_1, ℓ_2, and ℓ_∞ norm, which can be targeted or non-targeted. Liu et al. [56] proposed an ensemble attack method combining multiple models to construct adversarial attacks. Rony et al. [57] proposed a method to generate minimally perturbed adversarial examples based on Augmented Lagrangian for various distance metrics. In [58], authors propose a general framework for generating adversarial examples in both classification and regression tasks for applications in the image domain. Most of the methods in the literature about adversarial example generation belong to the class of **white box attackers**, i.e., the attacker has access to the information related to the trained neural network model, including the model architecture and its parameters. A **black box attacker** is introduced in [59]. Such attackers do not know the model but can interact with it. [59] proposed a one-pixel attack method that only changes one pixel for each image to construct an adversarial attack to fool DNNs. A byproduct of black-box attack is **grey-box attack**, where attackers might have limited information regarding the model. Weng et al. [2] proposed a computationally feasible method called Cross Lipschitz Extreme Value for nEtwork Robustness (CLEVER), which applies extreme value theory to estimate a lower bound of the minimum adversarial perturbation required to misclassify the image. CLEVER is the first attack-independent method and can also evaluate the intrinsic robustness of neural networks.

Most of the introduced attack methods are applied digitally, where the adversary supplies input images directly to the DNN. However, this is not always the case in scenarios that use cameras, microphones, or other sensors to receive the signals as input. These systems can still be attacked by generating physical-world adversarial objects. For example, Eykholt et al. [60] attached stickers to road signs that can severely threaten autonomous car sign recognizer and in [61] introduced the generation of physical 3D adversarial objects. These kinds of adversarial objects are more destructive for deep learning models because they can directly challenge many practical applications of DNN, such as face recognition, autonomous vehicles, etc.

Regression: In regression tasks, there are no natural margins as in the case of classification tasks, and difficulties hinder adversarial learning in a regression setting

to define the adversarial attacks, their success, and evaluation metrics. Despite the number of works in adversarial attack generation, there are few articles dealing with regression tasks. Tong et al. [62] looked at adversarial attacks in the setting of an ensemble of multiple learners, investigating the interactions between these linear learners and an attacker in regression tasks, modeled as a Multi-Learner Stackelberg Game (MLSG). However, the investigated linear case cannot capture the larger class of non-linear models. The focus only on specific applications of regression is common. Ghafouri et al. [63] examined an important problem: selecting an optimal threshold for each sensor against an adversary for regression tasks in cyber-physical systems. Deng et al. [64] introduced the concept of adversarial threshold, which is related to a deviation between the original prediction and the prediction of an adversarial example, i.e., an acceptable error range in driving models. In a regression context, [65] introduced a generically useful defense to reduce the effectiveness of adversarial attacks. They consider adversarial attacks as a potential symptoms of numerical instability in the learned function.

2.3.1.2 . Adversarial Defenses

Adversarial Training : All ML models are vulnerable to adversarial attacks [66]. Therefore, defending against adversarial examples is urgent for ML security. Goodfellow et al. [1] proposed the concept of adversarial training to improve the robustness of the model. The idea is to add adversarial attacks to the training data and continuously generate new examples at each training step. The number and relative weight of adversarial examples at each batch is controlled by the loss function independently. Adversarial training can be viewed as the process of minimizing classification error rates when the data is maliciously perturbed and it can be formulated as a minimax problem. Although a model can be robust to white-box attacks after the adversarial training step, it is still vulnerable to the adversarial attacks generated from other models, i.e., the model is not robust to black-box attacks. Based on this, Tramer et al. [67] proposed the concept of ensemble adversarial training. It consists in augmenting the training data constructed not only from the model being trained but also from the other pre-trained models, increasing the diversity of adversarial attacks and improving the generalization ability.

Defensive Distillation: Adversarial training needs adversarial examples to train the model; thus, the defense is related to the process of adversarial examples construction. In Papernot et al. [52] proposed a universal defensive method for neural networks called defensive distillation. The distillation method uses a small model to simulate a large and computationally intensive model without affecting the accuracy and can solve the problem of missing information. Different from the traditional distillation technique, defensive distillation aims to smooth the model during the training process by generalizing examples outside the training data. The basic idea of defensive distillation is to generate smooth classifiers that are more resilient to adversarial examples, reducing the sensitivity of the DNN to the input

perturbation.

Strategies to improve robustness, such as adversarial training only provide empirical robustness, without any formal guarantees on the safety of trained models. Stronger and more aggressive attacks have successfully broken many existing adversarial defenses [68]. Similarly, Carlini and Wagner [55] show that the method of Defensive Distillation is still vulnerable to their adversarial examples. In contrast, certified defenses using bounds such as the Lipschitz constant provide formal robustness guarantees that any norm-bounded adversary cannot alter the prediction of a given network.

2.3.2 . Control using Lipschitz constant

Utilizing the Lipschitz constant to certify robustness has been studied at several instances in the literature. Neural networks trained without any robustness constraint usually have very large global Lipschitz constant bounds [3], so most existing works train the network with a criterion that promotes small Lipschitz bound.

Parseval Networks: [69] designed networks with orthogonal weights, whose Lipschitz constants are exactly 1. This can be too restrictive and later works mostly use the power iteration to obtain per-layer induced norms, whose product is a Lipschitz constant

Spectral Normalization: [70] showed control on the Lipschitz constant using spectral normalization for GANs. We discuss Spectral Normalization in Section 2.3.2.1.

Hinge Regularisation: The authors of [71] propose to learn 1-Lipschitz networks for binary classification using a new loss which is a hinge regularized version of the Kantorovich-Rubinstein dual formulation for the Wasserstein distance estimation. They prove that the proposed loss function has a direct interpretation in terms of adversarial robustness together with certifiable robustness bound.

Prior work seeks to use global or local Lipschitz bounds during training to promote robustness. Computing and imposing loose global Lipschitz bounds is often easy but over-regularize the network during training and decrease its accuracy on clean data.

Lipschitz Margin training(LMT): In [72] authors train models that are certifiably robust by constructing a new loss on worst logits using the global Lipschitz bounds. They add a value of $\sqrt{2}\epsilon L_{\text{glob}}$, where ϵ is the perturbation radius, to all logits other than the logit corresponding to the ground-truth class and calculate the loss thereafter.

Box constrained propagation (BCP): Lee et al. [46] achieves a tighter outer bound than global Lipschitz-based outer bound, by considering local information via interval bound (box) propagation. They also compute the worst-case logit based on the intersection of a (global) ball and a (local) box.

GloRo: Leino et al. [15] bounds the upper bounds on the worst margins using the global Lipschitz constant. It constructs a new logit with a newly constructed

class called the bottom class and determines if the sample can be certified.

Local-Lip: [73] utilizes the interactions between activation functions (e.g. ReLU, MaxMin) and weight matrices. By eliminating the corresponding rows and columns where the activation function output is constant, they guarantee a lower provable local Lipschitz bound than the global Lipschitz bound for the neural network.

LipNet1: Bethune et al. [74] concentrate on classifiers constructed with 1-Lipschitz networks. They use DeelLip Library [71] for constructing 1-Lipschitz neural networks. They show that the classifiers trained with 1-Lipschitz constrained can be as expressive as those trained without any Lipschitz constraints and are certifiably robust. They introduce an implicit parameter τ, which controls the network expressiveness. A careful examination of the algorithm reveals that controlling this parameter is equivalent to relaxing the Lipschitz constant of the neural network, which is the approach we follow in a more direct manner in this work.

Known Lipschitz Constants: In Piat et al. [75], authors learn general neural networks for which the target Lipschitz constants are known before hand. They propose a PGD-like algorithm to approximate the target fixed Lipschitz constant by maximizing a spectral norm.

2.3.2.1 . Spectral Normalization

Spectral normalization stabilizes DNN training by constraining the Lipschitz constant of the objective function. Spectrally normalized DNNs have also been shown to generalize well [76], which is an indication of stability in machine learning. For fully-connected layers, the spectral norm of a given weight matrix W can be computed by using the approach described by [70] using the power iteration method. For each W, we initialize a vector $\tilde{u}$ and approximate both the left and right singular vectors by iterating the update rules:

$$\tilde{v} \leftarrow W\tilde{u}/\|W\tilde{u}\|_2$$
$$\tilde{u} \leftarrow W^\top\tilde{v}/\|W^\top\tilde{v}\|_2$$

The final singular value can be approximated with $\sigma(W) \approx \tilde{v}^T W\tilde{u}$.

The final weight matrix after spectral normalization W_{SN} is given as

$$W_{\text{SN}} = W/\sigma(W).$$

In the case of a CNN consisting of convolutional layers, dense layers, and ReLU activations, the spectral norm of each of the ReLU layers can be calculated using the power iteration method [11, 77]. Gouk et al. [77] also give a procedure for bounding the spectral norm of skip connections, and batch normalization layers enable this approach on ResNet architectures. The effectiveness of spectral normalization in training safety-critical systems was shown in [78, 79]. The DNNs are trained with layer-wise spectrally normalized weight matrices. In practice, we can apply spectral

normalization to the weight matrices in each layer during training as follows:

$$W_{SN} = \frac{W}{\sigma(W)} \gamma^{1/m} \tag{2.66}$$

where γ is the intended global Lipschitz constant of the DNN having m layers.

2.3.2.2 . Conclusion

Several developments related to various state-of-art approaches have been presented in this chapter. In the following chapters, we present some contributions and discuss the limitations of state-of-the-art techniques. We show how to tackle a few of these shortcomings so as to better understand and improve the stability of neural networks. We evaluate our proposed methods on diverse open-source and industrial datasets in the field of safety-critical applications.

Chapter 3
Adversarial Attacks on Regression Tasks

3.1 . Introduction

Adversarial attacks against neural networks and their defenses have been primarily investigated in classification tasks. However, adversarial attacks in a regression setting remain understudied, although regression tasks are an important part of mission-critical and safety-critical use cases in many domains like aviation, space, transportation, and defense. In this work, we present an adversarial attacker for regression tasks derived from the algebraic properties of the Jacobian of the neural network. We show that our attacker successfully fools the neural network and we measure its effectiveness in reducing the estimation performance. This white-box adversarial attacker aims to support engineers in designing safety-critical regression machine learning models. We present our results on various open-source and real industrial tabular datasets. In particular, the proposed adversarial attacker outperforms attackers based on random perturbations of the inputs. Our analysis relies on the quantification of the fooling error as well as various error metrics. A noteworthy feature of our attacker is that it allows us to optimally attack a subset of inputs, which may help to analyze the sensitivity of some specific inputs, as we will see in the next chapter.

3.2 . Limitations of Previous Works

As pointed out in Section 2.3.1, adversarial machine learning has received increased attention in the past decade. Adversarial attacks cause vulnerability in model deployment and especially need to be considered in the deployment of security-critical AI applications. Despite the newfound interest of the research community in trustworthy and explainable AI, only a few works are investigating adversaries in the case of regression tasks.

Current advances in the adversarial machine learning field evolve around the issue of designing attacks and defenses with a focus on the use of neural networks

in image analysis and computer vision [1], [10]. Much fewer works concern tabular data. However, most machine learning tasks in the industry rely on tabular data, e.g., fraud detection, product failure prediction, anti-money laundering, recommendation systems, click-through rate prediction, or flight arrival time prediction. To the best of our knowledge, the only work dealing with adversarial attacks in white box settings for tabular data has been proposed in [80], and this work handles only classification tasks. In this work, we focus on generating adversarial attacks for neural networks in the specific scenario when *i*) a regression task is performed and *ii*) tabular data are employed.

3.3 . Proposed Attacker

3.3.1 . Objective

As already mentioned, the problem of adversarial attacks is closely related to the robustness issue of a neural network, i.e., its sensitivity to perturbations. Let $T\colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_m}$ be the considered neural network having N_0 scalar inputs and N_m scalar outputs. If $x \in \mathbb{R}^{N_0}$ is a given vector of inputs for some data for which y is the associated target output, the network has been trained to produce an output $T(x)$ close to y. If the input is now perturbed by an additive vector $e \in \mathbb{R}^{N_0}$, the perturbed output is $T(x+e)$. Attacking the network then amounts to finding a perturbation e of preset magnitude, which makes the output of the network maximally deviate from a reference output. This reference output may be the model output $T(x)$ or the ground truth output y. Since our purpose is to develop an efficient approach even if the network accuracy is not very high, we choose y as the reference output when available. In this context, measures of deviation and magnitude of the perturbation play an important role in the mathematical formulation of the problem. As a common choice, the measure of perturbation magnitude will be here an ℓ_p-norm where $p \in [1, +\infty]$. For measuring the output deviation, we will similarly consider an ℓ_q-norm where $q \in [1, +\infty]$. It must be emphasized that this choice makes sense when dealing with regression problems. In this context, the ℓ_2 or the ℓ_1 norms are frequently used as training loss functions. On the other hand, the $\ell_{+\infty}$ norm is also a popular measure when dealing with reliability issues.

3.3.2 . Optimization formulation

In the described setting, the design of the attacker can be formulated as the problem of finding the "worst perturbation" $\widehat{e}$ such that

$$\widehat{e} \in \underset{e \in C_{p,\delta}}{\operatorname{Argmax}} \ \|T(x+e) - y\|_q, \tag{3.1}$$

where $C_{p,\delta}$ is the closed and convex set defined as

$$C_{p,\delta} = \{e \in \mathbb{R}^{N_0} \mid \|\Sigma^{-1/2} e\|_p \leqslant \delta\}. \tag{3.2}$$

$\Sigma \in \mathbb{R}^{N_0 \times N_0}$ is symmetric positive definite matrix. δ is a parameter that controls the maximum allowed perturbation, and Σ is a weighting matrix typically corresponding to the covariance matrix of the inputs. For instance, if we assume it to be a diagonal matrix, it simply introduces a normalization of the perturbation components with respect to the standard deviations of the associated inputs.

For standard choices of activation functions, T is a continuous function. Using the Weierstrass theorem, the existence of a solution (not necessarily unique) to Problem 3.1 is then ensured. Although $C_{p,\delta}$ is a relatively simple convex set, this problem appears as a difficult non-convex problem due to the fact that *i*) T is a complex nonlinear operator, *ii*) we maximize an ℓ_q measure which, in addition, leads to a non-smooth cost function when $q = 1$ or $q = +\infty$. A further difficulty is that we usually need to attack a large dataset to evaluate the robustness of a network, and the provided optimization algorithm should therefore be fast.

3.3.3 . Algorithm

We propose to implement a two-step approach.

- **Step 1.** We first perform a linearization based on the following first-order Taylor expansion:

$$T(x+e) \simeq T(x) + J(x)e, \tag{3.3}$$

where $J(x) \in \mathbb{R}^{N_m \times N_0}$ is the Jacobian of the network at x.[1] Note that $J(x)$ can be computed by classical back-propagation techniques. We will make a second approximation, that is $y \simeq T(x)$. Based on these two approximations and after the variable change $e' = \delta^{-1}\Sigma^{-1/2}e$, Problem 3.1 simplifies to

$$\underset{e' \in B_p}{\text{maximize}} \ \|J(x)\Sigma^{1/2}e'\|_q, \tag{3.4}$$

where B_p is the closed ℓ_p ball centered at 0 and with unit radius. Note that the optimal cost value in Eq. (3.4) is the subordinate norm of matrix $J(x)\Sigma^{1/2}$ when the input space is equipped with the ℓ_p norm and the output space with the ℓ_q one. We recall that this subordinate norm is defined, for every matrix $M \in \mathbb{R}^{N_m \times N_0}$, as

$$\|M\|_{p,q} = \sup_{z \in \mathbb{R}^{N_0} \setminus \{0\}} \frac{\|Mz\|_q}{\|z\|_p}. \tag{3.5}$$

Problem 3.4 is thus equivalent to find a vector $\widehat{e}'$ for which the value of the cost function is equal to $\|J(x)\Sigma^{1/2}\|_{p,q}$. For values of (p,q) listed below the expression of such vector has an explicit form.

  - If $p = q = 2$, $\widehat{e}'$ is any unit ℓ_2 norm eigenvector of $\Sigma^{1/2}J(x)^\top J(x)\Sigma^{1/2}$ associated with the maximum eigenvalue of this matrix. This vector

[1] We assume that $J(x)$ is defined at x, see [81] for a justification of this assumption in the non-smooth case.

can be computed by performing a singular value decomposition of $J(x)\Sigma^{1/2}$.

- If $p = 2$ and $q = +\infty$, $\widehat{e}'$ is any unit ℓ_2 norm vector colinear with a row of $J(x)\Sigma^{1/2}$ having maximum ℓ_2 norm.
- If $p = +\infty$ and $q = +\infty$, $\widehat{e}'$ is a unit norm vector whose elements are equal to $(\epsilon^{(i)})_{1 \leqslant i \leqslant N_0}$ where, for every $i \in \{1, \ldots, N_0\}$, $\epsilon_i \in \{-1, 0, 1\}$ is the sign of the i-th element of a row of $J(x)\Sigma^{1/2}$ with maximum ℓ_1 norm.
- If $p = 1$ and $q = 1$, $\widehat{e}'$ is a vector which has only one nonzero component equal to ± 1, the index of this component corresponds to the column of $J(x)\Sigma^{1/2}$ with maximum ℓ_1 norm.
- If $p = 1$ and $q = 2$, $\widehat{e}'$ is a vector with only one nonzero component equal to ± 1. The index of this component corresponds to a column of $J(x)\Sigma^{1/2}$ with maximum ℓ_2 norm.
- If $p = 1$ and $q = +\infty$, $\widehat{e}'$ is again a vector with only one nonzero component equal to ± 1. The index of this component corresponds to a column of $J(x)\Sigma^{1/2}$ where is located an element of maximum absolute value.

- **Step 2.** In the previous optimization step, the optimal solution is not unique. Indeed if $\widehat{e} = \delta\Sigma^{1/2}\widehat{e}'$ is a solution to Problem 3.4, then $-\widehat{e}$ is also a solution. In addition, there may exist other reasons for the multiplicity of the solutions. For example, there may be several maximum norm rows for matrix $J(x)\Sigma^{1/2}$. Among all the possible choices, we propose to choose the solution $\widehat{e}$ leading to the maximum deviation w.r.t. the ground truth, that is such that $\|T(x + \widehat{e}) - y\|_q$ is maximum. This requires to perform a search on a small number of possible candidates. Note that no approximation error is involved in this step. If the ground truth for the output is not available, it can be replaced by the model output.

- **Post-optimization.** If $1 < q < +\infty$ and T is assumed to be differentiable, $e \mapsto \|T(x + e) - y\|_q^q$ is a differentiable function. A further refinement consists of minimizing this function over $C_{p,\delta}$ by using a projected gradient algorithm with Armijo search for the step-size. The previous estimates of $\widehat{e}$ can then be used to initialize the algorithm. According to our numerical tests, implementing this strategy when $q = 2$ only brings a marginal improvement. Moreover, this approach cannot be used when $q = 1$ or $q = +\infty$.

3.3.4 . Attacking a group of inputs

It can also be interesting to attack only a selected subset of inputs. It may help in identifying the more sensitive inputs of the network. Also, for some inputs like unsorted categorical ones, attacks are often meaningless since they introduce

a main change in the informative contents of the dataset, which can be easily detected. Our proposed approach can be adapted to generate such partial attacks. In Problem 3.4, it is indeed sufficient to replace matrix $\Sigma^{1/2}$ by $D\Sigma^{1/2}D$, where D a masking diagonal matrix whose diagonal elements are equal to 1 when the input is attacked and 0 otherwise. The optimal solutions $\widehat{e}'$ and $\widehat{e} = \delta D\Sigma^{1/2}D\widehat{e}'$have then their components equal to 0 for the non-attacked inputs. Note that the naive approach which would consist in solving Eq. (3.4) and setting to zero the resulting perturbation components for non-attacked inputs would be sub-optimal.

3.3.5 . Error Metrics

Analogous to metrics such as fooling rates/success rates in classification tasks we propose three metrics to quantify the performance of the proposed adversarial attacker in regression tasks. These metrics formulations are given in Table 3.1. Let K be the number of samples in the dataset.

- Mean Accuracy Error (MAE) : It is the average of the distance between the ground truth of the sample and the prediction of the attacked sample over K samples.

- Fooling Error (E) : It is the average of the distance between the prediction of the original sample and the prediction of the attacked sample over K samples.

- Symmetric Mean Accuracy Error (SMAPE) : This measure takes into account relative error distances of perturbed samples and original samples from the ground truth. This value is averaged over samples where distance of ground truth is farther from perturbed samples than the original sample. For all the above metrics, a high value implies a strong attack.

Mean Accuracy Error	MAE	=	$\frac{1}{K}\sum_{k=1}^{K}\|T(x_k+e_k)-y_k\|_q$
Fooling Error	E	=	$\frac{1}{K}\sum_{k=1}^{K}\|T(x_k+e_k)-T(x_k)\|_q$
Symmetric Mean Accuracy Percentage Error	SMAPE	=	$\frac{2}{K_+}\sum_{k=1}^{K_+}\frac{\|T(x_k+e_k)-y_k\|_q-\|T(x_k)-y_k\|_q}{\|T(x_k+e_k)-y_k\|_q+\|T(x_k)-y_k\|_q}$

Table 3.1: Error metrics used for evaluation. The mean value computed for SMAPE is limited to the K_+ positive values of the elements in the summation. e_k is the perturbation generated by the adversarial attacker on the k-th sample in the dataset of length K.

3.4 . Experiments

3.4.1 . Open Source Datasets

We run our experiments on three open source regression datasets. The *Combined Cycle Power Plant [82] dataset* has 4 features with 9,568 instances. The task is to predict the net hourly electrical energy output using hourly average ambient variables. The *Red Wine Quality* dataset [83] contains 1,599 total samples and each instance has 11 features. The features are physio-chemical and sensory measurements for wine. The output variable is a quality score ranging from 0 to 10, where 10 represents for best quality and 0 for least quality. For the *Abalone* dataset, the task is to model an Abalone's age based purely on its physical measurements. This would allow Abalone's age estimation without cutting its shell. There are in total 4,177 instances with 8 input variables including one categorical variable. The datasets are divided with a ratio of 4:1 between training and testing data. The categorical attributes are dealt with by using one hot encoding based on the number of categories. The input attributes are normalized by removing their mean and scaling to unit variance.

We train fully connected networks for the estimation of variables from the datasets. The network architecture for the dataset are given below. The values represent the number of hidden neurons in the layers. The activation function at each layer is ReLU except for the last layer.

- Combined cycle Power Plant dataset - $(10, 6, 1)$
- Red Wine Quality dataset - $(100, 100, 100, 10, 1)$
- Abalone Data set - $(256, 256, 256, 256, 1)$

3.4.2 . Industrial Dataset – Safety Critical Application

An industrial application dataset is also considered with 2,219,097 training, 739,639 validation, and 739,891 test samples. The description of the input/output variables of the dataset is given in Table 3.2. The variable to be predicted is the Estimation of Arrival time (ETE) of a flight, given variables including the distance and speed, and also an initial estimate of ETE. The dataset is related to flight control, an activity area where safety is critical. The input attributes are normalized by removing their mean and scaling to unit variance. For models, we build fully connected networks with ReLU activation function on all the hidden layers except the last one. The network architecture is shown in the Figure 3.1.

3.4.3 . Adversarial attacks on Standard Training

For checking the efficacy of the proposed adversarial attacker, we first train the neural networks without any constraints using the network architecture presented in the previous section with the aim of reducing the prediction/performance loss on the train dataset. This is refered to as a standard training procedure.

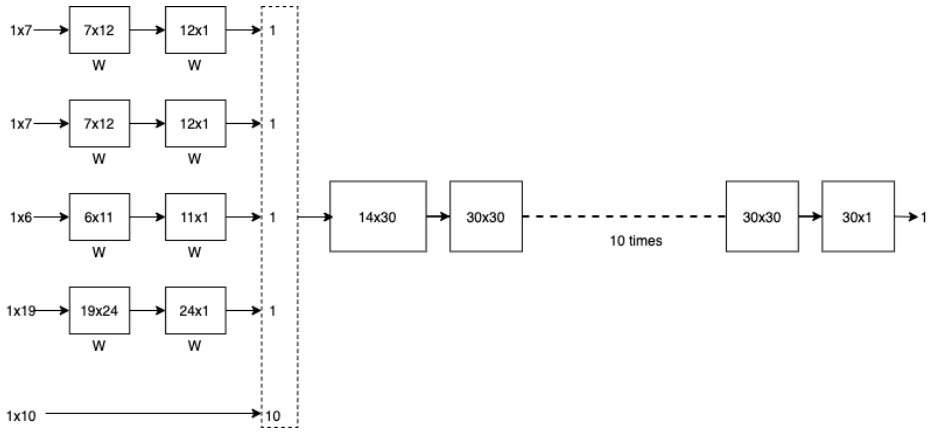

Figure 3.1: Network Architecture.

Input	0	Speed	continuous
	1	Flight Distance	
	2	Departure Delay	
	3	Initial ETE	
	4	Latitude Origin	
	5	Longitude Origin	
	6	Altitude Origin	
	7	Latitude Destination	
	8	Longitude Destination	
	9	Altitude Destination	
	10	Arrival Time Slot	7 slots (categorical)
	11	Departure Time Slot	7 slots (categorical)
	12	Aircraft Category	6 classes (categorical)
	13	Airline Company	19 classes (categorical)
Output	3	Refinement ETE	continuous

Table 3.2: Input and output variables description for the Thales Air Mobility industrial application dataset.

We compare the proposed adversarial attacker with random noise attackers generated by i.i.d. perturbations. We use three additive noise distributions: Gaussian, uniform, and binary for comparisons. The output of these attackers have been normalized so as to meet the desired bound on the norm of the perturbation. The metrics (MAE, E and SMAPE) are computed on the test samples where K is the total number of samples in the test set. The results on the 4 datasets for varying noise levels for ℓ_2 perturbations are shown in Table 3.3. Our proposed attacker

successfully attacks the neural network models, and is significantly better than random noise attackers in terms of distance metrics.

We also show the histograms of $(\|T(x_k + e_k) - y_k\|_q - \|T(x_k) - y_k\|_q)_{1 \leqslant k \leqslant K}$ in Figures 3.2-3.5, where $(e_k)_{1 \leqslant k \leqslant K}$ have been generated from various noise distributions and the proposed adversarial attacker. From the histograms, we observe the proposed attacker is guaranteed to attack the sample as it generates the worst positive deviation, while the random noise attackers are symmetric around zero. This implies that random attackers sometimes make the prediction better rather than making it deviate from the ground truth predictions. We observe similar trends for all four test datasets in the histograms.

Noise	$\mathbf{1 \times 10^{-1}}$	$\mathbf{2 \times 10^{-1}}$	$\mathbf{1 \times 10^{-1}}$	$\mathbf{2 \times 10^{-1}}$	$\mathbf{5 \times 10^{-2}}$	$\mathbf{1 \times 10^{-1}}$	$\mathbf{1 \times 10^{-1}}$	$\mathbf{2 \times 10^{-1}}$
Dataset	Power Plant		Wine		Abalone		Industrial	
MAE_{std}	6.4×10^{-3}	6.4×10^{-3}	0.47	0.47	1.68	1.68	9.2×10^{-3}	9.2×10^{-3}
$\text{MAE}_{\text{gauss}}$	6.5×10^{-3}	6.8×10^{-3}	0.46	0.47	1.68	1.68	9.6×10^{-3}	10.7×10^{-3}
MAE_{uni}	6.5×10^{-3}	6.8×10^{-3}	0.46	0.47	1.68	1.68	9.6×10^{-3}	10.7×10^{-3}
MAE_{bin}	6.5×10^{-3}	6.9×10^{-3}	0.47	0.48	1.68	1.68	9.6×10^{-3}	10.7×10^{-3}
MAE_{adv}	10.3×10^{-3}	14.2×10^{-3}	0.58	0.66	2.04	2.40	20.9×10^{-3}	32.5×10^{-3}
E_{guass}	1.3×10^{-3}	2.5×10^{-3}	0.04	0.09	0.02	0.05	2.6×10^{-3}	5.1×10^{-3}
E_{uni}	1.3×10^{-3}	2.5×10^{-3}	0.05	0.09	0.02	0.05	2.6×10^{-3}	5.2×10^{-3}
E_{bin}	1.4×10^{-3}	2.7×10^{-3}	0.04	0.09	0.03	0.05	2.7×10^{-3}	5.4×10^{-3}
E_{adv}	4.0×10^{-3}	8.0×10^{-3}	0.12	0.21	0.36	0.72	11.8×10^{-3}	24.0×10^{-3}
$\text{SMAPE}_{\text{gauss}}$	0.33	0.34	0.34	0.49	0.05	0.09	0.45	0.65
$\text{SMAPE}_{\text{uni}}$	0.34	0.52	0.33	0.51	0.04	0.08	0.46	0.66
$\text{SMAPE}_{\text{bin}}$	0.36	0.56	0.29	0.48	0.05	0.09	0.47	0.67
$\text{SMAPE}_{\text{adv}}$	0.62	0.87	0.41	0.56	0.38	0.58	0.96	1.24

Table 3.3: Comparison on evaluation metrics random attacker vs. proposed adversarial attacker with variation in perturbation level (ℓ_2 attack).

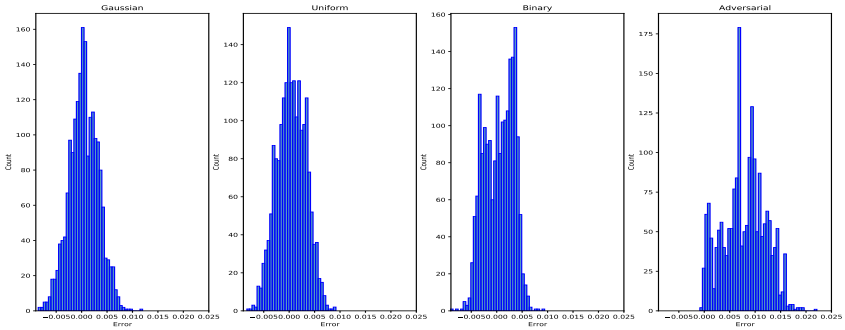

Figure 3.2: Error distribution of random attacks and proposed adversarial attack on Combined cycle Power Plant dataset for perturbation level of 2×10^{-1} for ℓ_2 attack.

3.4.4 . Attacks on Spectral Normalization training

In these experiments, we train our networks while using a spectral normalization technique as shown in section 2.3.2.1 which has been proven to be very effective

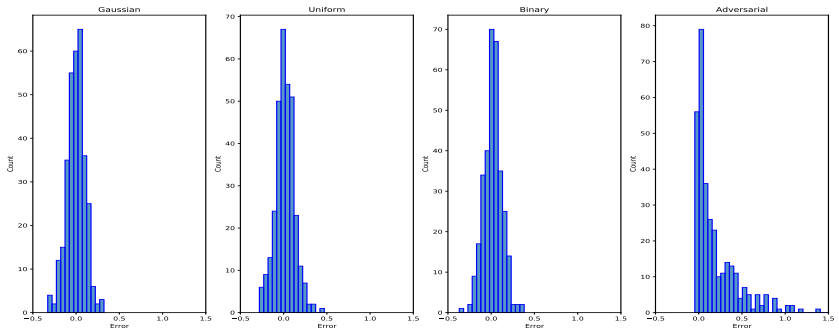

Figure 3.3: Error distribution of random attacks and proposed adversarial attack on Red-wine dataset for perturbation level of 2×10^{-1} for ℓ_2 attack.

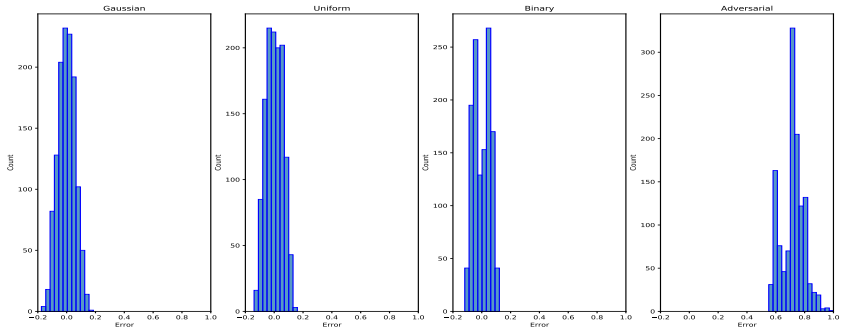

Figure 3.4: Error distribution of random attacks and proposed adversarial attack on Abalone dataset for perturbation level of 1×10^{-1} for ℓ_2 attack.

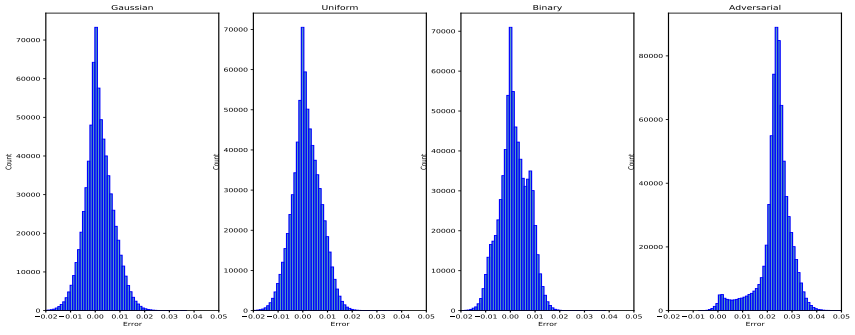

Figure 3.5: Error distribution of random attacks and proposed adversarial attack on industrial dataset for perturbation level of 2×10^{-1} for ℓ_2 attack.

in controlling Lipschitz properties in GANs. Recall that, given an m layer fully connected architecture and a Lipschitz target θ, we constrain the spectral norm of each layer to be less than $\sqrt[m]{\theta}$. This ensures that the upper bound on the global Lipschitz constant is less than θ. We keep the network architectures exactly the same as the standard training procedures in the previous section. The performance of adversarial attacker on standard and spectrally normalized trained model in terms

of Fooling Error (E) and Symmetric Mean Accuracy Percentage error (SMAPE) for various datasets and varying perturbation magnitude under ℓ_2 perturbations is given in Table 3.4. We observe that the spectral normalized models are more resilient to attacks than the standard trained models. Thus spectral normalization constraints help to generate more robust and safer models.

Noise	E_{adv}	E_{spec}	$SMAPE_{adv}$	$SMAPE_{spec}$
Power Plant				
$\mathbf{1 \times 10^{-1}}$	4.0×10^{-3}	3.9×10^{-3}	0.62	0.60
$\mathbf{2 \times 10^{-1}}$	8.0×10^{-3}	7.7×10^{-3}	0.87	0.84
Wine				
$\mathbf{1 \times 10^{-1}}$	0.12	0.02	0.41	0.12
$\mathbf{2 \times 10^{-1}}$	0.21	0.03	0.56	0.17
Abalone				
$\mathbf{5 \times 10^{-2}}$	0.36	0.11	0.38	0.12
$\mathbf{1 \times 10^{-1}}$	0.72	0.23	0.58	0.21
Industrial				
$\mathbf{1 \times 10^{-1}}$	11.8×10^{-3}	9.1×10^{-3}	0.91	0.49
$\mathbf{2 \times 10^{-1}}$	24.0×10^{-3}	17.5×10^{-3}	1.24	0.72

Table 3.4: Standard training vs Spectral Normalization training on ℓ_2 attacks.

3.4.5 . Attacking only continuous variables

All the previous results have been obtained with attack and noise addition on all the input features present in the datasets. As pointed in Section 3.3.4, the introduced adversarial attacker is capable of attacking a group of inputs. While generating an adversarial attack, we avoid attacking the categorical input variables [80]. Attacking categorical inputs may introduce a major change in the informative contents of the dataset, which is often easily identifiable and detectable. This is clearly in contradiction with the purpose of generating adversarial attacks which are hardly noticeable by the user. Hence in Abalone and industrial datasets, we attack only the continuous variables. For the Combined Power plant dataset, we attack 3 out of 4 continuous variables since it does not contain any categorical variables. Similarly, for the Red-wine dataset we attack 8 continuous variables out of 11. The performance of the adversarial attacker, when attacking only few inputs, is shown in Table 3.5. We observe that, from the attacker viewpoint, attacking only few inputs has an advantage since some inputs can be more sensitive than others and can contribute more in generating perturbed outputs. Attacking all the inputs might add redundancy to the generated perturbation and hence render the

attack more easily detectable.

3.4.6 . Comparison on different measures of deviations

As emphasized in Section 3.3.3, our adversarial attacker is applicable for various measures of perturbation on input and output deviations. The previous results have been obtained for the value $p = q = 2$ termed as ℓ_2 attacks here. We further show results for $p = q = 1$ termed as ℓ_1 attacks and for $p = q = +\infty$ termed as ℓ_∞ attacks in Table 3.6. We observe that our proposed attacker us successful in attacking under different measures of perturbations and remains better than random noise attackers in all cases and for all metrics.

Noise	E_{adv}	E_{inp}	$SMAPE_{adv}$	$SMAPE_{inp}$
Power Plant				
$\mathbf{1 \times 10^{-1}}$	4.0×10^{-3}	3.4×10^{-3}	0.62	0.58
$\mathbf{2 \times 10^{-1}}$	8.0×10^{-3}	7.0×10^{-3}	0.87	0.82
Wine				
$\mathbf{1 \times 10^{-1}}$	0.12	0.13	0.41	0.44
$\mathbf{2 \times 10^{-1}}$	0.21	0.22	0.56	0.60
Abalone				
$\mathbf{5 \times 10^{-2}}$	0.36	0.36	0.38	0.38
$\mathbf{1 \times 10^{-1}}$	0.72	0.71	0.58	0.59
Industrial				
$\mathbf{1 \times 10^{-1}}$	12.0×10^{-3}	12.0×10^{-3}	0.91	0.96
$\mathbf{2 \times 10^{-1}}$	24.0×10^{-3}	24.0×10^{-3}	1.24	1.24

Table 3.5: Standard training attacking all inputs vs standard training attacking few inputs on ℓ_2 attacks.

3.5 . Summary

The contributions of this work are summarised as follows:

- We propose a simple, novel, and easily implementable Jacobian-based adversarial attacker for regression tasks in the context of white box attacks.
- We define three error metrics: Mean Accuracy Error, Fooling Error, and Symmetric Mean Accuracy Percentage Error that help to analyze the attacker's effectiveness.

Noise	$\mathbf{1 \times 10^{-1}}$	$\mathbf{2 \times 10^{-1}}$	$\mathbf{1 \times 10^{-1}}$	$\mathbf{2 \times 10^{-1}}$	$\mathbf{1 \times 10^{-1}}$	$\mathbf{2 \times 10^{-1}}$
Attacks	ℓ_2		ℓ_1		ℓ_∞	
$\mathrm{MAE}_{\mathrm{gauss}}(\times 10^{-3})$	9.6	10.7	9.2	9.4	10.5	13.5
$\mathrm{MAE}_{\mathrm{uni}}(\times 10^{-3})$	9.6	10.7	9.2	9.3	11.1	15.3
$\mathrm{MAE}_{\mathrm{bin}}(\times 10^{-3})$	9.6	10.7	9.2	9.3	13.0	22.0
$\mathrm{MAE}_{\mathrm{adv}}(\times 10^{-3})$	20.9	32.5	18.5	28.1	31.1	52.5
$\mathrm{E}_{\mathrm{gauss}}(\times 10^{-3})$	2.6	5.1	0.84	1.7	4.7	9.4
$\mathrm{E}_{\mathrm{uni}}(\times 10^{-3})$	2.6	5.2	0.81	1.6	6.0	12.0
$\mathrm{E}_{\mathrm{bin}}(\times 10^{-3})$	2.7	5.4	0.72	1.4	9.9	19.0
$\mathrm{E}_{\mathrm{adv}}(\times 10^{-3})$	11.8	24.0	9.5	20.0	22.0	45.0
$\mathrm{SMAPE}_{\mathrm{gauss}}$	0.45	0.65	0.22	0.35	0.63	0.84
$\mathrm{SMAPE}_{\mathrm{uni}}$	0.46	0.66	0.21	0.34	0.71	0.92
$\mathrm{SMAPE}_{\mathrm{bin}}$	0.47	0.67	0.20	0.32	0.87	1.08
$\mathrm{SMAPE}_{\mathrm{adv}}$	0.96	1.24	0.87	1.15	1.22	1.47

Table 3.6: Comparison on industrial dataset for ℓ_2, ℓ_1 and ℓ_∞ attacks with variation in perturbation levels. $\mathrm{MAE}_{\mathrm{std}}$(MAE for Standard training) $= 9.2 \times 10^{-3}$

- Our attacker is generic in the sense that it can handle any measure (ℓ_1, ℓ_2, ℓ_∞) on input or output perturbations according to the target application.
- We show that the proposed attacker allows us to attack any given subset of input features optimally. This feature may be helpful when handling specific tabular datasets and also be insightful when information regarding sensitivity or the ability to control some inputs is available.
- We evaluate our results on open-source regression datasets and an industrial dataset (output and input features described in Table 3.2) which lies in the domain of safety-critical applications.
- Our tests concentrated on fully connected networks, but it is worth pointing out that the proposed approach can be applied to any network architecture.

Chapter 4
Multivariate Estimation of Stability of Neural Networks

4.1 . Introduction

In this chapter, we introduce a multivariate Lipschitz constant-based stability analysis of fully connected neural networks allowing us to capture the influence of each input or group of inputs on the neural network stability. Our approach relies on a suitable re-normalization of the input space, with the objective to perform a more precise analysis than the one provided by a global Lipschitz constant. We investigate the mathematical properties of the proposed multivariate Lipschitz analysis and show its usefulness in better understanding the sensitivity of the neural network with regard to groups of inputs. We display the results of this analysis by a new representation designed for machine learning practitioners and safety engineers termed as a Lipschitz star. The Lipschitz star is a graphical and practical tool to analyze the sensitivity of a neural network model during its development, with regard to different combinations of inputs. By leveraging this tool, we show that it is possible to build stable-by-design models using spectral normalization techniques for controlling the stability of a neural network.Thanks to our multivariate Lipschitz analysis, we can also measure the efficiency of adversarial training in inference tasks. We used the adversarial attacker proposed in Chapter 3 to generate adversarial attack and perform adversarial training. We perform experiments on various open access tabular datasets, and also on a real Thales Air Mobility industrial application subject to certification requirements, also considered in the previous chapter.

4.2 . Limitations of Previous Works

The main limitations of the Lipschitz constant, defined in either global or local context, is that it only provides a single parameter to quantify the stability of a neural network. Such a single-parameter analysis does not facilitate the understanding of potential sources of instability. In particular, it may be insightful to identify

the inputs which have the highest impact in terms of sensitivity on the output. In the context of tabular data mining, the inputs often have quite heterogeneous characteristics. Some of them are categorical data, often encoded in a specific way (e.g., one-hot encoder [84]) and among them, one can usually distinguish those which are unsorted (like labels identifying countries) or those which are sorted (like severity scores in a disease). So, it may appear useful to analyze in a specific manner each type of inputs of a NN and even sometimes to exclude some of these inputs (e.g., unsorted categorical data for which the notion of small perturbation may be meaningless) from the performed sensitivity analysis. A comparison of the state-of-the-art and proposed approach is presented in Table 4.1.

Method	Properties	Sensitivity of Inputs
Naive upper Bound [1]	spectral bound, loose bound, univariate	No
SDPLip [16]	ℓ_2 norm, more scalable to broad networks, univariate	No
CPLip [9]	$\ell_p \in [1, +\infty]$, not scalable to broad networks, univariate	No
LipOpt-k [43]	$\ell_p \in [1, +\infty]$, univariate	No
Proposed	scalable to broad networks,multivariate	Yes

Table 4.1: Comparison of state-of-the-art Lipschitz estimation approaches vs the proposed one.

4.3 . Partial/Weighted Lipschitz Constant

We present in this section a new approach based on a suitable weighting operation performed in the computation of Lipschitz constants. This enables a multivariate sensitivity analysis of the neural network stability for individual inputs or groups of inputs. We will start by motivating this weighting from a statistical standpoint. Then we will define it in a more precise manner, before discussing its resulting mathematical properties.

4.3.1 . Statistical motivations

For tractability, assume that the perturbation at the network input is a realization of a zero-mean Gaussian distributed random vector z with $N_0 \times N_0$ covariance matrix $\Sigma \succ 0$. Then, its density upper level sets are defined as

$$C_\eta = \{z \in \mathbb{R}^{N_0} \mid z^\top \Sigma^{-1} z \leqslant \eta\}, \tag{4.1}$$

for every $\eta \in]0, +\infty[$. The set C_η defines an ellipsoid where the probability density takes its highest values. More precisely, the probability for z to belong to this set is independent of Σ and is equal to

$$\mathrm{P}(\mathrm{z} \in C_\eta) = \frac{\gamma(N_0/2, \eta/2)}{\Gamma(N_0/2)}, \tag{4.2}$$

where Γ is the gamma function and γ the lower (unnormalized) incomplete gamma function.

Proof of Eq. (4.2):
Let $B(0,\sqrt{\eta})$ be the closed ball of $\mathbb{R}^{N_0}$ with center 0 and radius $\sqrt{\eta}$ and let S_{N_0-1} be the surface of the unit-radius sphere in $\mathbb{R}^{N_0-1}$. We have

$$\begin{aligned}
\mathrm{P}(\mathrm{z}\in C_\eta) &= \frac{1}{(2\pi)^{N_0/2}(\det(\Omega))^{1/2}}\int_{C_\eta}\exp\Big(-\frac{1}{2}z^\top\Omega^{-1}z\Big)\,dz\\
&= \frac{1}{(2\pi)^{N_0/2}}\int_{B(0,\sqrt{\eta})}\exp\Big(-\frac{1}{2}\|u\|^2\Big)\,du\\
&= \frac{1}{(2\pi)^{N_0/2}}S_{N_0-1}\int_0^{\sqrt{\eta}}\rho^{N_0-1}\exp\Big(-\frac{\rho^2}{2}\Big)\,d\rho\\
&= \frac{1}{(2\pi)^{N_0/2}}\frac{2(\pi)^{N_0/2}}{\Gamma(N_0/2)}\int_0^{\sqrt{\eta}}\rho^{N_0-1}\exp\Big(-\frac{\rho^2}{2}\Big)\,d\rho\\
&= \frac{1}{\Gamma(N_0/2)}\int_0^{\frac{\eta}{2}}\zeta^{N_0/2-1}\exp(-\zeta)\,d\zeta\\
&= \frac{\gamma(N_0/2,\eta/2)}{\Gamma(N_0/2)}.
\end{aligned}\tag{4.3}$$

On the other hand, let us assume that the maximum standard deviation $\sigma_{\max}$ of the components of z (i.e., square root of the maximum diagonal element of matrix Σ) is small enough. If we suppose that the network T is differentiable in the neighborhood of a given input $x\in\mathbb{R}^{N_0}$, as the input perturbation is small enough, we can approximate the network output by the following first-order expansion:

$$T(x+z)\simeq T(x)+T'(x)z.\tag{4.4}$$

Let us focus our attention on perturbations in C_η. By doing so, we impose some norm-bounded condition, which may appear realistic for adversarial perturbations. Then, we will be interested in calculating

$$\sup_{z\in C_\eta}\|T(x+z)-T(x)\|\simeq\sup_{z\in C_\eta}\|T'(x)z\|.\tag{4.5}$$

By making the variable change $z'=z/\sqrt{\eta}$ and using Eq. (4.1),

$$\begin{aligned}
\sup_{z\in C_\eta}\|T(x+z)-T(x)\| &\simeq \sqrt{\eta}\sup_{z'\in C_1}\|T'(x)z'\|\\
&= \sqrt{\eta}\sup_{\substack{z\in\mathbb{R}^{N_0}\\ z'\neq 0}}\frac{\|T'(x)z'\|}{\|z'\|_{\Sigma^{-1}}}\\
&= \sqrt{\eta}\sigma_{\max}\sup_{\substack{z'\in\mathbb{R}^{N_0}\\ z'\neq 0}}\frac{\|T'(x)z'\|}{\|z'\|_{\Omega^{-1}}},
\end{aligned}\tag{4.6}$$

where $\Omega=\Sigma/\sigma_{\max}^2$ and $\|\cdot\|_{\Omega^{-1}}=\sqrt{(\cdot)^\top\Omega^{-1}(\cdot)}$. This suggests that, in this context, the suitable subordinate matrix norm for computing the Lipschitz constant

is obtained by weighting the Euclidean norm in the input space with Ω^{-1}. We can also deduce from Eq. (4.6), by setting $z'' = \Omega^{-1/2} z'$, that

$$\sup_{z \in C_\eta} \|T(x+z) - T(x)\| \simeq \sqrt{\eta}\sigma_{\max} \sup_{\substack{z'' \in \mathbb{R}^{N_0} \\ z'' \neq 0}} \frac{\|T'(x)\Omega^{1/2} z''\|}{\|z''\|}$$
$$= \sqrt{\eta}\|T'(x)\Sigma^{1/2}\|_{\mathrm{S}}. \tag{4.7}$$

On the other hand, based on the first-order approximation in Eq. (4.4), $T(x+z)$ is approximately Gaussian with mean $T(x)$ and covariance matrix $T'(x)\Sigma T'(x)^\top$. As $\|T'(x)\Sigma T'(x)^\top\|_{\mathrm{S}} = \|T'(x)\Sigma^{1/2}\|_{\mathrm{S}}^2$, we see that another insightful interpretation of Eq. (4.7) is that, up to the scaling factor $\sqrt{\eta}$, it approximately delivers the square root of the spectral norm of the covariance matrix of the output perturbations.

4.3.2 . New definition of a weighted Lipschitz constant

Based on the previous motivations, we propose to employ a weighted norm to define a Lipschitz constant of the network as follows:

Definition 5 *Let Ω be an $N_0 \times N_0$ symmetric positive definite real-valued matrix. We say that θ_m^Ω is an Ω-weighted norm Lipschitz constant of T as described in Fig. 2.1 if*

$$(\forall (x,z) \in (\mathbb{R}^{N_0})^2) \quad \|T(x+z) - T(z)\| \leqslant \theta_m^\Omega \|z\|_{\Omega^{-1}}. \tag{4.8}$$

The above definition can be extended to non Euclidean norms by making use of exponents $(p,q) \in [1,+\infty]^2$ and by replacing inequality Eq. (4.8) with

$$(\forall (x,z) \in (\mathbb{R}^{N_0})^2) \quad \|T(x+z) - T(z)\|_q \leqslant \theta_m^\Omega \|\Omega^{-1/2} z\|_p. \tag{4.9}$$

By changes of variable, this inequality can also be rewritten as

$$(\forall (x',z') \in (\mathbb{R}^{N_0})^2) \quad \|T(\Omega^{1/2}(x'+z')) - T(\Omega^{1/2} z')\|_q \leqslant \theta_m^\Omega \|z'\|_p. \tag{4.10}$$

Therefore, we see that calculating θ_m^Ω is equivalent to deriving a Lipschitz constant of the network T where an additional first linear layer $\Omega^{1/2}$ has been added. Throughout the rest of this section, it will be assumed that, for every $i \in \{1, \ldots, m-1\}$ the activation operator R_i is separable and α_i-averaged. It then follows from Eq. (2.23) that an Ω-weighted norm Lipschitz constant of T is

$$\vartheta_m^\Omega = \sup_{\substack{\Lambda_1 \in \mathcal{D}_{N_1}(\{2\alpha_1 - 1, 1\}), \\ \vdots \\ \Lambda_{m-1} \in \mathcal{D}_{N_{m-1}}(\{2\alpha_{m-1} - 1, 1\})}} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 \Omega^{1/2}\|_{p,q}. \tag{4.11}$$

Although all our derivations were based on the fact that Ω is positive definite, from the latter expression we see that, by continuous extension, ϑ_m^Ω can be defined when Ω is a singular matrix.

4.3.3 . Sensitivity with respect to a group of inputs

In this section, we will be interested in a specific family of weighted norms associated with the set of matrices

$$\{\Omega_{\epsilon,\mathbb{K}} \mid \varnothing \neq \mathbb{K} \subset \{1,\dots,N_0\}, \epsilon \in]0,1]\},$$

defined, for every nonempty subset $\mathbb{K}$ of $\{1,\dots,N_0\}$ and for every $\epsilon \in]0,1]$, as

$$\Omega_{\epsilon,\mathbb{K}} = \mathrm{Diag}(\sigma^2_{\epsilon,\mathbb{K},1},\dots,\sigma^2_{\epsilon,\mathbb{K},N_0}), \tag{4.12}$$

where

$$(\forall \ell \in \{1,\dots,N_0\}) \quad \sigma_{\epsilon,\mathbb{K},\ell} = \begin{cases} 1 & \text{if } \ell \in \mathbb{K} \\ \epsilon & \text{otherwise.} \end{cases} \tag{4.13}$$

If we come back to the statistical interpretation in Section 4.3.1, $\Omega_{\epsilon,\mathbb{K}}$ is then (up to a positive scale factor) the covariance matrix of a Gaussian random vector z with independent components.[1] The components with indices in $\mathbb{K}$ have a given variance $\sigma^2_{\max}$ while the others have variance $\epsilon^2\sigma^2_{\max}$. Such a matrix thus provides a natural way of putting emphasis on the group of inputs with indices in $\mathbb{K}$. Thus, variables $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}$ will be termed partial Lipschitz constants in the following.

The next proposition lists the main properties related to the use of such weighted norms for calculating Lipschitz constants.

Proposition 6 *Let $(p,q) \in [1,+\infty]^2$. For every nonempty subset $\mathbb{K}$ of $\{1,\dots,N_0\}$ and for every $\epsilon \in]0,1]$, let $\Omega_{\epsilon,\mathbb{K}}$ be defined as above and let $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}$ be defined by Eq. (4.11). Let $\mathbb{K}_0$ and $\mathbb{K}_1$ be nonempty subsets of $\{1,\dots,N_0\}$. Then the following hold:*

i) *As $\epsilon \to 0$, $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}}$ converges to the Lipschitz constant of a network where all the inputs with indices out of $\mathbb{K}_0$ are kept constant.*

ii) *$\vartheta_m^{\Omega_{1,\mathbb{K}_0}}$ is equal to the global Lipschitz constant ϑ_m defined by Eq. (2.23).*

iii) *Let $(\epsilon,\epsilon') \in]0,1]^2$. If $\Omega_{\epsilon,\mathbb{K}_0} \preceq \Omega_{\epsilon',\mathbb{K}_1}$,[2] then $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}} \leqslant \vartheta_m^{\Omega_{\epsilon',\mathbb{K}_1}}$.*

iv) *Function $\vartheta_m^{\Omega_{\cdot,\mathbb{K}_0}} :]0,1] \to [0,+\infty[: \epsilon \mapsto \vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}}$ is monotonically increasing.*

v) *Let $\epsilon \in]0,1]$. If $\mathbb{K}_0 \subset \mathbb{K}_1$, then $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}} \leqslant \vartheta_m^{\Omega_{\epsilon,\mathbb{K}_1}}$.*

vi) *Let $\epsilon \in]0,1]$, let $K \in \mathbb{N} \setminus \{0\}$, and let*

$$\omega_{K,\epsilon} = \binom{N_0-1}{K-1}\left(1+\left(\frac{N_0}{K}-1\right)\epsilon\right). \tag{4.14}$$

We have

$$\max_{\substack{\mathbb{K}\subset\{1,\dots,N_0\} \\ \mathrm{card}\,\mathbb{K}=K}} \vartheta_m^{\Omega_{\epsilon,\mathbb{K}}} \leqslant \vartheta_m \leqslant \frac{1}{\omega_{K,\epsilon}} \sum_{\substack{\mathbb{K}\subset\{1,\dots,N_0\} \\ \mathrm{card}\,\mathbb{K}=K}} \vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}. \tag{4.15}$$

[1] Recall that this interpretation is valid when $p = 2$ in Eq. (4.11).

[2] $\preceq$ designates the Loewner order.

vii) *Let $\epsilon \in]0,1]$, let $\mathcal{P}$ be a partition of $\{1,\dots,N_0\}$, and let $\omega_{\mathcal{P},\epsilon} = 1 + (\operatorname{card}\mathcal{P} - 1)\epsilon$. We have*

$$\max_{\mathbb{K}\in\mathcal{P}} \vartheta_m^{\Omega_{\epsilon,\mathbb{K}}} \leqslant \vartheta_m \leqslant \frac{1}{\omega_{\mathcal{P},\epsilon}} \sum_{\mathbb{K}\in\mathcal{P}} \vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}. \tag{4.16}$$

viii) *Let $\mathbb{K}_2$ be such that $\mathbb{K}_1 \cap \mathbb{K}_2 \neq \tilde{p}$ and $\mathbb{K}_1 \cup \mathbb{K}_2 = \mathbb{K}_0$. Let $p^* \in [1,+\infty]$ be such that $1/p + 1/p^* = 1$ Then*

$$\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}} \leqslant \left((\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_1}})^{p^*} + (\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_2}})^{p^*} \right)^{1/p^*} + o(\epsilon). \tag{4.17}$$

Proofs:

i): As $\epsilon \to 0$, the diagonal elements of $\Omega_{\epsilon,\mathbb{K}_0}$ with index $\ell \notin \mathbb{K}$ tend to zero, which in Eq. (4.11) amounts to assuming that the corresponding components of the input perturbation are zero. The existence of the limit $\Omega_{0,\mathbb{K}_0}$ is secured based on the remark at the end of Section4.3.2.

ii): If $\epsilon = 1$, then $\Omega_{\epsilon,\mathbb{K}_0} = \mathrm{Id}_{N_0}$ and Eq. (4.11) reduces to Eq. (2.23).

iii): For every $i \in \{1,\dots,m-1\}$, let $\Lambda_i \in \mathcal{D}_{N_i}(\{2\alpha_i - 1, 1\})$. Then, by using the triangle inequality,

$$\begin{aligned} \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 (\Omega_{\epsilon,\mathbb{K}_0})^{1/2}\|_{p,q} \leqslant \\ \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 (\Omega_{\epsilon',\mathbb{K}_1})^{1/2}\|_{p,q} \|(\Omega_{\epsilon',\mathbb{K}_1})^{-1} \Omega_{\epsilon,\mathbb{K}_0}\|_{p,p}^{1/2}. \end{aligned} \tag{4.18}$$

By taking the supremum of both sides with respect to the $(\Lambda_i)_{1\leqslant i\leqslant m-1}$ matrices, we deduce that

$$\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}} \leqslant \|(\Omega_{\epsilon',\mathbb{K}_1})^{-1} \Omega_{\epsilon,\mathbb{K}_0}\|_{p,p}^{1/2} \, \vartheta_m^{\Omega_{\epsilon',\mathbb{K}_1}}. \tag{4.19}$$

On the other hand,

$$(\Omega_{\epsilon',\mathbb{K}_1})^{-1} \Omega_{\epsilon,\mathbb{K}_0} = \mathrm{Diag}\left(\frac{\sigma^2_{\epsilon,\mathbb{K}_0,1}}{\sigma^2_{\epsilon',\mathbb{K}_1,1}}, \dots, \frac{\sigma^2_{\epsilon,\mathbb{K}_0,N_0}}{\sigma^2_{\epsilon',\mathbb{K}_1,N_0}} \right), \tag{4.20}$$

where, by using the fact that $\Omega_{\epsilon,\mathbb{K}_0} \preceq \Omega_{\epsilon',\mathbb{K}_1}$,

$$(\forall \ell \in \{1,\dots,N_0\}) \quad \frac{\sigma_{\epsilon,\mathbb{K}_0,\ell}}{\sigma_{\epsilon',\mathbb{K}_1,\ell}} \leqslant 1. \tag{4.21}$$

Since $(\Omega_{\epsilon',\mathbb{K}_1})^{-1} \Omega_{\epsilon,\mathbb{K}_0}$ is a diagonal matrix with elements lower than or equal to 1, $\|(\Omega_{\epsilon',\mathbb{K}_1})^{-1} \Omega_{\epsilon,\mathbb{K}_0}\|_{p,p} \leqslant 1$ and it follows from Eq. (4.19) that

$$\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}} \leqslant \vartheta_m^{\Omega_{\epsilon',\mathbb{K}_1}}. \tag{4.22}$$

iv): Let $(\epsilon,\epsilon') \in]0,1]^2$ with $\epsilon < \epsilon'$. We have $\Omega_{\epsilon,\mathbb{K}_0} \preceq \Omega_{\epsilon',\mathbb{K}_0}$ and, according to iii),

$$\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}} \leqslant \vartheta_m^{\Omega_{\epsilon',\mathbb{K}_0}}. \tag{4.23}$$

v): If $\mathbb{K}_0 \subset \mathbb{K}_1$, then $\Omega_{\epsilon,\mathbb{K}_0} \preceq \Omega_{\epsilon,\mathbb{K}_1}$ and the result follows from iii).

vi): We have

$$\sum_{\substack{\mathbb{K}\subset\{1,\dots,N_0\}\\ \operatorname{card}\mathbb{K}=K}} \Omega_{\epsilon,\mathbb{K}}^{1/2} = \left(\binom{N_0-1}{K-1} + \left(\binom{N_0}{K} - \binom{N_0-1}{K-1}\right)\epsilon\right)\mathrm{Id}_{N_0}. \tag{4.24}$$

By using the relation

$$\binom{N_0}{K} = \frac{N_0}{K}\binom{N_0-1}{K-1}, \tag{4.25}$$

we deduce that

$$\sum_{\substack{\mathbb{K}\subset\{1,\dots,N_0\}\\ \operatorname{card}\mathbb{K}=K}} \Omega_{\epsilon,\mathbb{K}}^{1/2} = \omega_{K,\epsilon}\mathrm{Id}_{N_0}. \tag{4.26}$$

For every $i \in \{1,\dots,m-1\}$, let $\Lambda_i \in \mathcal{D}_{N_i}(\{2\alpha_i - 1, 1\})$. Then,

$$\begin{aligned}\|W_m\Lambda_{m-1}\cdots\Lambda_1 W_1\|_{p,q} &= \frac{1}{\omega_{K,\epsilon}}\left\|W_m\Lambda_{m-1}\cdots\Lambda_1 W_1\Big(\sum_{\substack{\mathbb{K}\subset\{1,\dots,N_0\}\\ \operatorname{card}\mathbb{K}=K}}\Omega_{\epsilon,\mathbb{K}}^{1/2}\Big)\right\|_{p,q}\\ &\leqslant \frac{1}{\omega_{K,\epsilon}}\sum_{\substack{\mathbb{K}\subset\{1,\dots,N_0\}\\ \operatorname{card}\mathbb{K}=K}}\|W_m\Lambda_{m-1}\cdots\Lambda_1 W_1\Omega_{\epsilon,\mathbb{K}}^{1/2}\|_{p,q}.\end{aligned} \tag{4.27}$$

We deduce that

$$\begin{aligned}\vartheta_m = &\sup_{\substack{\Lambda_1\in\mathcal{D}_{N_1}(\{2\alpha_1-1,1\}),\\ \vdots,\\ \Lambda_{m-1}\in\mathcal{D}_{N_{m-1}}(\{2\alpha_{m-1}-1,1\})}} \|W_m\Lambda_{m-1}\cdots\Lambda_1 W_1\|_{p,q}\\ &\leqslant \frac{1}{\omega_{K,\epsilon}}\sum_{\substack{\mathbb{K}\subset\{1,\dots,N_0\}\\ \operatorname{card}\mathbb{K}=K}}\sup_{\substack{\Lambda_1\in\mathcal{D}_{N_1},\\ \vdots,\\ \Lambda_{m-1}\in\mathcal{D}_{N_{m-1}}}}\|W_m\cdots\Lambda_1 W_1\Omega_{\epsilon,\mathbb{K}}^{1/2}\|_{p,q}\\ &= \frac{1}{\omega_{K,\epsilon}}\sum_{\substack{\mathbb{K}\subset\{1,\dots,N_0\}\\ \operatorname{card}\mathbb{K}=K}}\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}.\end{aligned} \tag{4.28}$$

Furthermore, according to ii) and iv), for every $\mathbb{K} \subset \{1, \ldots, N_0\}$,

$$\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}} \leqslant \vartheta_m^{\Omega_{1,\mathbb{K}}} = \vartheta_m. \tag{4.29}$$

This yields

$$\max_{\substack{\mathbb{K}\subset\{1,\ldots,N_0\} \\ \operatorname{card}\mathbb{K}=K}} \vartheta_m^{\Omega_{\epsilon,\mathbb{K}}} \leqslant \vartheta_m. \tag{4.30}$$

vii): The proof is similar to that of vi) by noticing that, if $\mathcal{P}$ is a partition of $\{1, \ldots, N_0\}$, then

$$\sum_{\mathbb{K}\in\mathcal{P}} \Omega_{\epsilon,\mathbb{K}}^{1/2} = \omega_{\mathcal{P},\epsilon} \mathrm{Id}_{N_0}. \tag{4.31}$$

viii): For every $\varnothing \neq \mathbb{K} \subset \{1, \ldots, N_0\}$, $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}} = \vartheta_m^{\Omega_{0,\mathbb{K}}} + o(\epsilon)$. It is thus sufficient to prove this inequality in the limit case when $\epsilon \to 0$. For every $i \in \{1, \ldots, m-1\}$, let $\Lambda_i \in \mathcal{D}_{N_i}(\{2\alpha_i - 1, 1\})$. Let $x \in \mathbb{R}^{N_0}$ and let $x_{\mathbb{K}}$ be the projection of x onto the space of vectors whose components indexed by $\{1, \ldots, N_0\} \setminus \mathbb{K}$ are zero. We have thus $x_{\mathbb{K}_0} = x_{\mathbb{K}_1} + x_{\mathbb{K}_2}$. Then,

$$\begin{aligned}
&\|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_0}^{1/2} x\|_q \\
&= \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 x_{\mathbb{K}_0}\|_q \\
&\leqslant \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 x_{\mathbb{K}_1}\|_q + \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 x_{\mathbb{K}_2}\|_q \\
&= \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_1}^{1/2} x_{\mathbb{K}_1}\|_q + \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_2}^{1/2} x_{\mathbb{K}_2}\|_q \\
&\leqslant \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_1}^{1/2}\|_{p,q} \|x_{\mathbb{K}_1}\|_p + \|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_2}^{1/2}\|_{p,q} \|x_{\mathbb{K}_2}\|_p.
\end{aligned} \tag{4.32}$$

By using Hölder's inequality, we deduce that

$$\begin{aligned}
\|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_0}^{1/2} x\|_q \leqslant \\
\left(\|W_m \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_1}^{1/2}\|_{p,q}^{p^*} + \|W_m \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_2}^{1/2}\|_{p,q}^{p^*}\right)^{1/p^*} \\
(\|x_{\mathbb{K}_1}\|_p^p + \|x_{\mathbb{K}_2}\|_p^p)^{1/p}.
\end{aligned} \tag{4.33}$$

Since $\|x_{\mathbb{K}_1}\|_p^p + \|x_{\mathbb{K}_2}\|_p^p = \|x_{\mathbb{K}_0}\|_p^p$, it follows that

$$\begin{aligned}
\|W_m \Lambda_{m-1} \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_0}^{1/2}\|_{p,q} \leqslant \\
\left(\|W_m \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_1}^{1/2}\|_{p,q}^{p^*} + \|W_m \cdots \Lambda_1 W_1 \Omega_{0,\mathbb{K}_2}^{1/2}\|_{p,q}^{p^*}\right)^{1/p^*}
\end{aligned} \tag{4.34}$$

Taking the supremum with respect to $(\Lambda_i)_{1\leqslant i\leqslant m-1}$ and majorizing the supremum of the sum in the right-hand side by the sum of the suprema yield Eq. (4.17).

Let us comment on these results. According to Property i) in the limit case when $\epsilon \to 0$, only the inputs with indices in $\mathbb{K}_0$ are used in the computation of

the associated Lipschitz constant. In turn, Property ii) states that, when $\epsilon = 1$, we recover the classical expression of a Lipschitz constant where the perturbations on all the inputs are taken into account. In addition, based on Property iv), the evolution of $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}_0}}$ when ϵ varies from 1 to 0 provides a way of assessing how the group of inputs indexed by $\mathbb{K}_0$ contributes progressively to the overall Lipschitz behaviour of the network. Although one would expect that summing the Lipschitz constants obtained for each group of inputs would yield the global Lipschitz constant, Properties vi) and vii) show that this does not hold in general whatever the way the entries are split (possibly overlapping groups of given size K or disjoint groups of arbitrary size). Instead, after suitable normalization, such sums provide upper bounds on ϑ_m. Furthermore, it follows from ii), Eq. (4.15), and Eq. (4.16) that the difference between these normalized sums and ϑ_m tends to vanish when ϵ increases.
Note that, when looking at the sensitivity with respect to individual inputs, i.e., when the considered set of indices are singletons, both vi) (with $K = 1$) and vii) (with $\mathcal{P} = \{\{k\} \mid k \in \{1, \dots, N_0\}\}$ lead to the same inequality

$$\max_{k\in\{1,\dots,N_0\}} \vartheta_m^{\Omega_{\epsilon,\{k\}}} \leqslant \vartheta_m \leqslant \frac{1}{1+(N_0-1)\epsilon} \sum_{k=1}^{N_0} \vartheta_m^{\Omega_{\epsilon,\{k\}}}. \tag{4.35}$$

4.3.4 . Lipschitz Star

We propose a new representation for displaying the results of the Lipschitz analysis of a neural network utilizing radial plots. More precisely, we plot the values of $(\vartheta_m^{\Omega_{\epsilon,\{k\}}})_{1\leqslant k\leqslant N_0}$ on a star or radar chart where each branch of the star corresponds to the index k of an input. For each value of ϵ, a new plot is obtained which is displayed in a specific color. Note that, according to Proposition 6iv), the plots generated for different ϵ values cannot cross. When $\epsilon = 1$, we obtain an "isotropic" representation whose "radius" corresponds to the global Lipschitz constant ϑ_m of the network. This representation is called a Lipschitz star. All the results of our analysis will be displayed using this representation.

4.4 . Experiments

4.4.1 . Validation on Synthetic Datasets

4.4.1.1 . Context

To verify the correctness of the partial Lipschitz computation, we first study simple synthetic examples of polynomial systems for which we can calculate explicitly the partial Lipschitz constants. We generate input-output data for the defined systems, and train a fully connected model using a standard training, i.e., without any constraints. We compare this approach with a training subject to a spectral norm constraint on the layers.

Spectral Normalization: For safety critical tasks, Lipschitz constant and performance targets can be specified as engineering requirements, prior to network

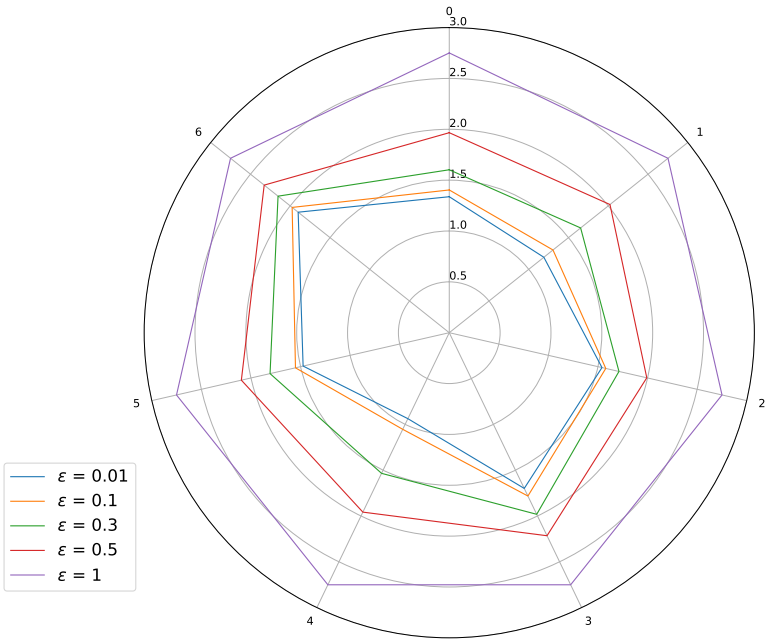

Figure 4.1: Lipschitz Star: Representing Partial Lipschitz constants w.r.t. to the inputs as the vertices of the radial plot.

training. A Lipschitz target can be defined by a safety analysis of the acceptable perturbations for each output knowing the input range and it constitutes a current practice in many industries. As mentioned in the previous chapter, imposing this Lipschitz target can be done either by controlling the Lipschitz constant for each layer or for the whole network depending on the application at hand. In our experiments, we train networks while using a spectral normalization technique. We recall from section 2.3.2.1 for practical application of spectral normalization, given an m layer fully connected architecture and a Lipschitz target θ, we can constrain the spectral norm of each layer to be less than $\sqrt[m]{\theta}$. According to Eq. (2.7), this ensures that the upper bound on the global Lipschitz constant is less than θ.

For each training, we study the effect of input variables on the stability of the networks. As proposed in Section 4.3.3, for a given group of inputs with indices in $\mathbb{K}$, we will quantify the partial Lipschitz constant $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}$. The obtained value of $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}$ allows us to evaluate how the corresponding group of variables may potentially affect the stability of the network. For simplicity, in the next section, we will focus on the limit case when $\epsilon = 0$ (see the last remark in Section 4.3.2).

Partial Lipschitz constant values $\vartheta_m^{\Omega_{0,\mathbb{K}}}$, for all possible choices for $\mathbb{K}$, are computed using the numerical method described in Section 2.2.1.4 and compared with

the theoretical values derived in the following subsection. More details on the models are also provided in these sections.

4.4.1.2 . Polynomial systems

We consider regression problems where the data is synthesized by a second-order multivariate polynomial. The system to be modelled is thus described by the following function:

$$(\forall(\xi_1,\dots,\xi_{N_0}) \in \mathbb{R}^{N_0}) \quad f(\xi_1,\dots,\xi_{N_0}) = \sum_{k=1}^{N_0} a_k\xi_k + \sum_{k=1}^{N_0}\sum_{l=1}^{N_0} b_{k,l}\xi_k\xi_l, \quad (4.36)$$

where $(a_k)_{k\in N_0}$ and $(b_{k,l})_{1\leqslant k,l\leqslant N_0}$ are the real-valued polynomial coefficients. Note that, such a polynomial system is generally not Lipschitz-continuous. The Lipschitz-continuity property only holds on every compact set. Subsequently, we will thus study this system on the hypercube $[-M,M]^{N_0}$ with $M > 0$.

The explicit values of the partial Lipschitz constant on this domain can be derived as follows. We first calculate the gradient of f

$$\nabla f(\xi_1,\dots,\xi_{N_0}) = \big(\partial_k f(\xi_1,\dots,\xi_{N_0})\big)_{1\leqslant k\leqslant N_0}, \quad (4.37)$$

where, for every $k \in \{1,\dots,N_0\}$, $\partial_k f$ denotes the partial derivative w.r.t. the k-th variable given by

$$\partial_k f(\xi_1,\dots,\xi_{N_0}) = a_k + \sum_{l=1}^{N_0}(b_{k,l} + b_{l,k})\xi_l. \quad (4.38)$$

For every $\mathbb{K} \subset \{1,\dots,N_0\}$, the partial Lipschitz constant $\mathring{\vartheta}^{\Omega_{0,\mathbb{K}}}$ of the polynomial system (restricted to $[-M,M]^{N_0}$) w.r.t. the group of variables with indices in $\mathbb{K}$ is then equal to

$$\mathring{\vartheta}^{\Omega_{0,\mathbb{K}}} = \sup_{(\xi_1,\dots,\xi_{N_0})\in[-M,M]^{N_0}} \sqrt{\lambda_{\Omega_{0,\mathbb{K}}}(\xi_1,\dots,\xi_{N_0})}, \quad (4.39)$$

where, for every diagonal matrix $\Lambda = \mathrm{Diag}(\varepsilon_1^2,\dots,\varepsilon_{N_0}^2)$ with $(\varepsilon_1,\dots,\varepsilon_{N_0}) \in [0,+\infty[^{N_0}$,

$$\begin{aligned}\lambda_\Lambda(\xi_1,\dots,\xi_{N_0}) &= \|(\nabla f(\xi_1,\dots,\xi_{N_0}))^\top \Lambda^{1/2}\|^2 \\ &= \sum_{k=1}^{N_0} \varepsilon_k\big(\partial_k f(\xi_1,\dots,\xi_{N_0})\big)^2. \end{aligned} \quad (4.40)$$

Since the partial derivatives in Eq. (4.38) are affine functions of the variables $(\xi_1,\dots,\xi_{N_0})$, λ_Λ is a convex function. We deduce that the supremum in Eq. (4.39) is attained when $\xi_1 = \pm M,\dots,\xi_{N_0} = \pm M$, so that $\mathring{\vartheta}^{\Omega_{0,\mathbb{K}}}$ can be computed by looking for the maximum of a finite number (2^{N_0}) of values.

4.4.1.3 . Numerical results

In our numerical experiments, we consider a toy example corresponding to $N_0 = 3$ and

$$(\forall(\xi_1,\xi_2,\xi_3) \in \mathbb{R}^3) \quad f(\xi_1,\xi_2,\xi_3) = \xi_1 + 100\xi_3 - \xi_2^2 + \gamma\xi_1\xi_3, \tag{4.41}$$

where $\gamma \in [0,+\infty[$. We deduce from Eq. (4.40) that

$$\lambda_\Lambda(\xi_1,\xi_2,\xi_3) = \varepsilon_1(1+\gamma\xi_3)^2 + 4\varepsilon_2\xi_2^2 + \varepsilon_3(100+\gamma\xi_1)^2 \tag{4.42}$$

and, consequently,

$$\sup_{(\xi_1,\xi_2,\xi_3)\in[-M,M]^3} \lambda_\Lambda(\xi_1,\xi_2,\xi_3) = \varepsilon_1(1+\gamma M)^2 + 4\varepsilon_2 M^2 + \varepsilon_3(100+\gamma M)^2. \tag{4.43}$$

By looking at the seven possible binary values of $(\varepsilon_1,\varepsilon_2,\varepsilon_3) \neq (0,0,0)$, we thus calculate the Lipschitz constant of f with respect to each group of inputs. For example,

- if $\varepsilon_1 = 1, \varepsilon = 0, \varepsilon_3 = 0$, we calculate $\mathring{\vartheta}^{\Omega_{0,\mathbb{K}}}$ with $\mathbb{K} = \{1\}$, i.e., evaluate the sensitivity w.r.t. the first variable;
- if $\varepsilon_1 = \varepsilon_2 = 1, \varepsilon_3 = 0$, we calculate $\mathring{\vartheta}^{\Omega_{0,\mathbb{K}}}$ with $\mathbb{K} = \{1,2\}$, i.e., evaluate the joint sensitivity w.r.t. the first and second variables;
- if $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 1$, we calculate $\mathring{\vartheta}^{\Omega_{0,\mathbb{K}}}$ with $\mathbb{K} = \{1,2,3\}$, i.e., evaluate the sensitivity w.r.t. all the variables (global Lipschitz constant).

These Lipschitz constants allow us to evaluate the intrinsic dynamics of the system, that is how it responds when its inputs vary.

Our interest will be now to evaluate how this dynamics is modified when the system is modelled by a neural network. To do so, three systems are studied by choosing $\gamma \in \{0, \frac{1}{10}, 1\}$ and $M = 50$. We generate 5,000 data samples from each system, the input values being drawn independently from a random uniform distribution. While training the neural networks, the dataset is divided with a ratio of 4:1 into training and testing samples. The input is normalized using its mean and standard deviation, while the output is max-normalized. We build neural networks for approximating the systems using two hidden layers ($m = 3$) with a number of hidden neurons equal to 30 in each layer and ReLU activation functions. The training loss is the mean square error.

For different values of γ, we report the values of the partial Lipschitz constants in Tables 4.2-4.4. The variable $\theta_{\mathbb{K}}$ corresponds to $\mathring{\vartheta}^{\Omega_{0,\mathbb{K}}}$ for the analytical value we derived from previous formulas, whereas it corresponds to the Lipschitz constant $\vartheta_3^{\Omega_{0,\mathbb{K}}}$, when computed for the neural network trained either in a standard manner or with a spectral normalization constraint. The value of θ used in the spectral normalization was adjusted to obtain a similar global Lipschitz constant to the

polynomial system. In the caption, we also indicate the accuracy in terms of normalized mean square error (NMSE) and normalized mean absolute error (NMAE). These values are slightly higher for constrained training, but remain quite small.

Partial LC	Analytical	Standard	Spectral Normalized
$\theta_{\{1\}}$	1	133.9	6.75
$\theta_{\{2\}}$	100	211.7	76.3
$\theta_{\{3\}}$	100	299.7	136.0
$\theta_{\{1,2\}}$	100.0	229.0	102.2
$\theta_{\{1,3\}}$	100.0	303.1	136.0
$\theta_{\{2,3\}}$	141.4	314.2	141.2
$\theta_{\{1,2,3\}}$	141.4	315.3	141.2

Table 4.2: Comparison of Lipschitz constant values when $\gamma = 0$. Test performance for standard training: NMSE = 0.007, NMAE = 0.005, for spectral normalization: NMSE = 0.011, NMAE = 0.009.

Partial LC	Analytical	Standard	Spectral Normalized
$\theta_{\{1\}}$	6	138.7	10.1
$\theta_{\{2\}}$	100	219.2	90.0
$\theta_{\{3\}}$	105	302.7	138.9
$\theta_{\{1,2\}}$	100.2	231.6	108.1
$\theta_{\{1,3\}}$	105.2	306.3	139.0
$\theta_{\{2,3\}}$	145	316.4	147.2
$\theta_{\{1,2,3\}}$	145.1	316.5	147.2

Table 4.3: Comparison of Lipschitz constant values when $\gamma = 1/10$. Test performance for standard training: NMSE = 0.006, NMAE = 0.005, for spectral normalization: NMSE = 0.009, NMAE = 0.007.

Comments on the results:

- In general, ξ_3 impacts the output of this system the most, and (ξ_2, ξ_3) mainly account for the global dynamics of the system.
- With standard training, we see that there exists a significant increase of the

Partial LC	Analytical	Standard	Spectral Normalized
$\theta_{\{1\}}$	51	274.7	59.5
$\theta_{\{2\}}$	100	298.9	80.3
$\theta_{\{3\}}$	150	388.7	183.7
$\theta_{\{1,2\}}$	112.6	337.0	119.4
$\theta_{\{1,3\}}$	158.4	392.2	183.7
$\theta_{\{2,3\}}$	180.3	400.1	188.9
$\theta_{\{1,2,3\}}$	187.4	400.5	189.0

Table 4.4: Comparison of Lipschitz constant values when $\gamma = 1$. Test performance for standard training: NMSE = 0.006, MAE =0.005, for spectral normalization: NMSE = 0.014, NMAE = 0.009.

sensitivity with respect to the input variations, so making the neural network vulnerable to adversarial perturbations.

- By using spectral normalization, it is possible to constrain the global Lipschitz constant of the system to be close to the analytical global value while keeping a good accuracy. One may however notice an increase of the sensitivity to ξ_1 and ξ_3, and a decrease of the sensitivity to ξ_2 with respect to the original system.

- For all the three models, the values obtained with neural networks follow the same trend, for different groups of inputs, as those observed with the analytical values.

- Although the Lipschitz constant of the neural networks is computed on the whole space and the one of the system on $[-50, 50]^3$, our Lipschitz estimates appear to be consistent without resorting to a local analysis.

These observations emphasize the importance of controlling the Lipschitz constant of neural network models through specific training strategies. In addition, we see that evaluating the Lipschitz constant with respect to groups of inputs allow us to have a better understanding of the behaviour of the models.

In this section, we have discussed the proposed method for synthetic datasets. In the next section, the sensitivity analysis will be made on widely used open source datasets and an industrial dataset.

4.4.2 . Open-source use-cases

We study regression problems involving tabular datasets to showcase our proposed multivariate analysis of the stability of neural networks. Tabular data take

advantage of heterogeneous sources of information coming from different sensors or data collection processes. We apply our methods on widely used tabular datasets: *i)* Combined Cycle Power Plant dataset [3] which has 4 attributes with 9,568 instances; *ii)* Auto MPG dataset [4] consists of 398 instances with 7 attributes; *iii)* Boston Housing dataset [5] consists of 506 instances with 13 attributes. For Combined Power Plant and Auto MPG datasets, we solve a regression problem with a single output, whereas for Boston Housing dataset we consider a two-output regression problem with "price" and "ptratio :pupil-teacher ratio by town" as the output variables. The attributes in the dataset are a combination of continuous and categorical ones. The datasets are divided with a ratio of 4:1 between training and test data. Information on the input and output attributes are provided in Tables 4.5-4.7, respectively.

Input	0 1 2 3	Temperature Ambient Pressure Relative Humidity Exhaust Vacuum	continuous
Output	4	Net hourly electrical energy output	continuous

Table 4.5: Input and output attributes of Combined Cycle Power Plant dataset.

Input	0 1 2 3 4 5 6	Cylinders Displacement Horsepower Weight Acceleration Model Year Origin	continuous unsorted categorical
Output	7	MPG	continuous

Table 4.6: Input and output attributes of Auto MPG dataset.

4.4.3 . Industrial Dataset

Thales Air Mobility industrial application represents the prediction of the Estimated Time En-route (ETE), meaning the time spent by an aircraft between the take-off and landing, considering a number of variables as described in Table 3.2.

[3] https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
[4] https://archive.ics.uci.edu/ml/datasets/auto+mpg
[5] https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

Input	0	CRIM	continuous
	1	ZN	
	2	INDUS	
	3	CHAS	
	4	NOX	
	5	RM	
	6	AGE	
	7	DIS	
	8	RAD	
	9	TAX	
	10	B	
	11	LSTAT	
Output	12	MEDV	continuous
	13	PTRATIO	continuous

Table 4.7: Input and output attributes of Boston Housing dataset.

The application is important in air traffic flow management, which is an activity area where safety is critical. The purpose of the proposed sensitivity analysis is thus to help engineers in building safe by design models complying with given safety stability targets. The dataset consists of 2,219,097 training, 739,639 validation, and 739,891 test samples.

For all the models, we build fully connected networks with ReLU [6] activation function on all the hidden layers, except the last one. The models are trained on Keras with Tensorflow backend. The initializers are set to Glorot uniform. The network architecture of the different models, number of layers, and neurons are tabulated in Table 4.8. Combined Cycle Power Plant dataset with $(10, 6)$ network architecture is trained with two hidden layers having 10 and 6 hidden neurons, respectively. For Thales Air Mobility industrial application $(10 \times (30))$ implies that the neural network has 10 hidden layers with 30 neurons each.

4.4.4 . Sensitivity analysis with respect to each input

In this section we study the effect of input variables on the stability of the networks. More specifically, we study the effect of input variations on the stability of the networks by quantifying $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}$ with $\epsilon \in]0, 1]$, for various choices of $\mathbb{K}$, instead of a global Lipschitz constant accounting for the influence of the whole set of inputs. The obtained value of $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}$ allows us to evaluate how the corresponding group of variables may potentially affect the stability of the network. By performing this analysis for several choices of $\mathbb{K}$, we thus generate a multivariate analysis of

[6] We present the results only for ReLU, but we tested our approach with other activation functions such as tanh as well and found the trends in sensitivity of inputs to be similar.

Dataset	layers & neurons	EPOCHS	Optimizer	lr
Combined Cycle Power Plant	$(10, 6)$	100	Adam	0.01
Auto MPG	$(16, 8)$	1000	RMSprop	0.001
Boston Housing	$(10, 5)$	500	RMSprop	0.001
Thales Air Mobility App.	$(10 \times (30))$	100	Adam	0.01

Table 4.8: Network Architecture and training setup for different datasets.

the Lipschitz regularity of the network.

As shown by Proposition 6, varying the ϵ parameter is also insightful since it allows us to measure how the network behaves when input perturbations are gradually more concentrated on a given subset of inputs.

Although our approach can be applied to groups of inputs, for simplicity in this section, we will focus on the case when the sets $\mathbb{K}$ reduce to singletons.

For each dataset, we first perform a standard training when designing the network. To facilitate comparisons, the Lipschitz star of the network trained in such standard manner is presented as the first subplot of all the figures. Next, we show the variation in terms of input sensitivity, when *i)* a Lipschitz target is imposed, and *ii)* when an adversarial training of the networks is performed. The network architecture remains unchanged, for all our experiments and each dataset. All the Lipschitz constants for each value of ϵ are calculated using LipSDP-Neuron [16]. Since an increased stability may come at the price of a loss of accuracy [85], we also report the performance of the networks on test datasets in terms of MAE (Mean Absolute Error) for each of the Lipschitz star plot.

4.4.5 . Effect of training with specified Lipschitz target

Spectral norm constrained training is performed as explained in Section 4.4.1.1. The results are shown for our three datasets in Figures 4.2-4.5. On these plots, we can observe a shrinkage of the Lipschitz stars following the reduction of the target Lipschitz value. Interestingly, improving stability does not affect significantly the performance of the networks. Let us comment on the last use case in light of the obtained results.

Comments on the Thales Air Mobility industrial application: From the star plots, it is clear that the various variables have a quite different effect on the Lipschitz behavior of the network. This is an expected outcome since these variables carry a different amount of information captured by learning. From Figure 4.5 we observe that variables 1 – Flight Distance and 3 – Initial ETE play a prominent role, while variables 5 – Longitude Origin, and 8 – Longitude Destination are also sensitive. Some plausible explanations for these facts are mentioned below.

- Flight distance: The impact of a change of this input can be significant since, as a result of air traffic management separation rules, commercial aircrafts

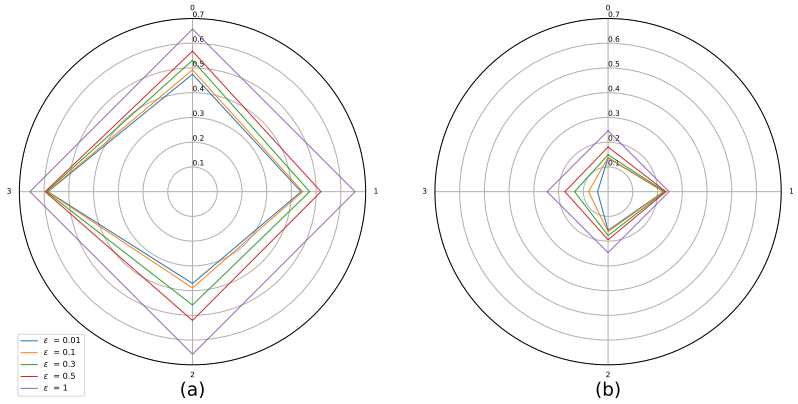

Figure 4.2: Sensitivity w.r.t. to each input on Combined Cycle Power Plant dataset. Influence of a spectral normalization constraint. a) Standard training: Lipschitz constant = 0.66, MAE = 0.007 , b) With spectral normalization: Lipschitz constant = 0.25, MAE = 0.0066.

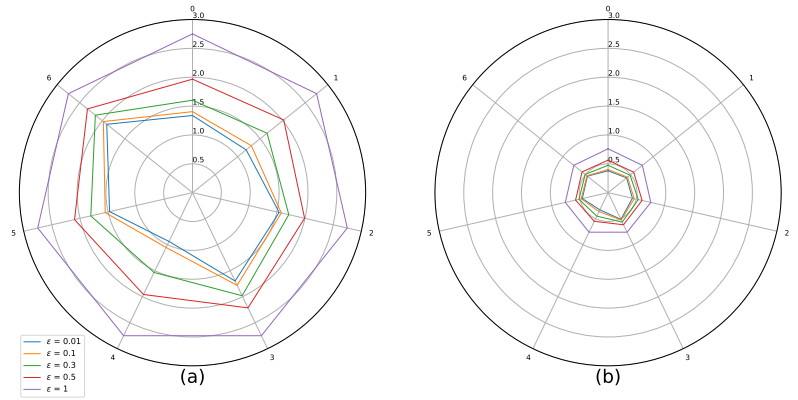

Figure 4.3: Sensitivity w.r.t. to each input on Auto MPG dataset. Influence of a spectral normalization constraint. a) Standard training : Lipschitz constant = 2.75, MAE = 0.05 , b) With spectral normalization: Lipschitz constant = 0.76, MAE = 0.04.

cannot freely increase their speed to minimize the impact of a longer flight distance.

- Initial ETE: Modifying this input is equivalent to changing the initial conditions, which will have a significant impact. It is possible, in the worst case scenario, to accumulate other perturbations coming from other cou-

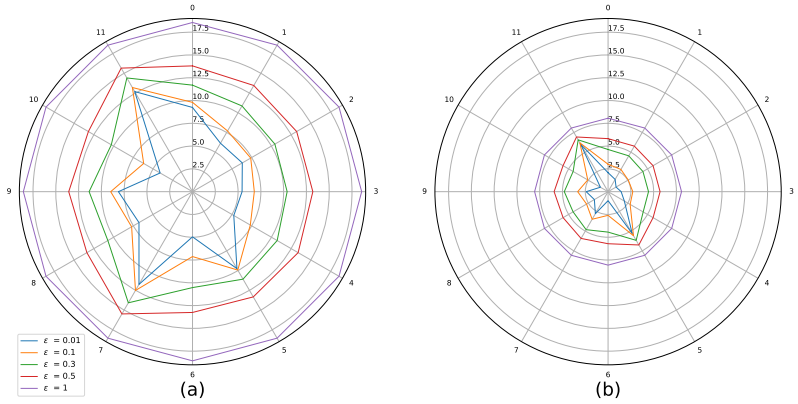

Figure 4.4: Sensitivity w.r.t. to each input on Boston Housing dataset. Influence of a spectral normalization constraint. a) Standard training : Lipschitz constant = 18.56, MAE(y_1) = 2.45, MAE(y_2) = 1.41, b) With spectral normalization: Lipschitz constant = 8.06 , MAE(y_1) = 2.96, MAE(y_2) = 1.35.

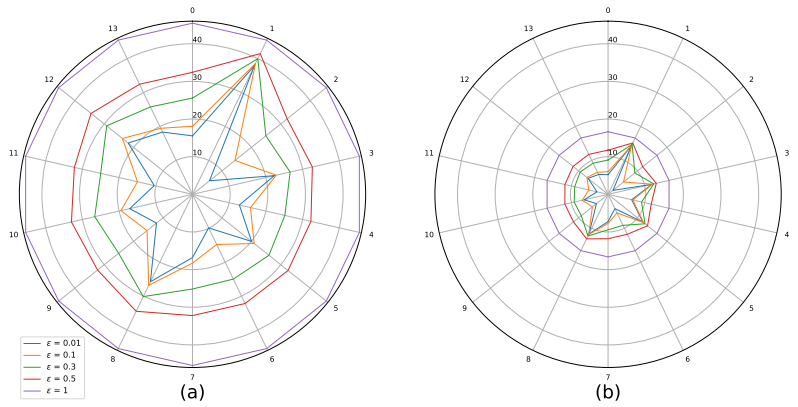

Figure 4.5: Sensitivity w.r.t. to each input on Thales Air Mobility industrial application. Influence of a spectral normalization constraint. a) Standard training: Lipschitz constant = 45.46, MAE = 496.37 (s), b) With spectral normalization constraint: Lipschitz constant = 16.62, MAE = 478.88 (s).

pled inputs and parameters (e.g., weather conditions) and this is probably the reason why the partial Lipschitz constant is very high, and close to the global Lipschitz constant.

- Longitude origin and destination parameters: These parameters are related

to different continents and even countries of the origin and destination airports, probably with different qualities of air traffic equipment.

4.4.6 . Effect of adversarial training

Generating adversarial attacks and performing adversarial training constitute popular methods in designing robust neural networks. However, these techniques have received less attention for regression tasks, since most of the works deal with classification tasks [60, 1, 86]. Also, most of the existing works in the deep learning literature are for standard signal/image processing problems, whereas there are only few works handling tabular data [87, 88]. One noticeable exception is [80] which investigates problems related to adversarial attacks for classification tasks involving tabular data.

Since our applications are related to regression problems for which few existing works are directly applicable, we designed a specific adversarial training method. More specifically, for a given amplitude of the adversarial noise and for each sample in the training set, we generate the worst attack based on the spectral properties of the Jacobian of the network, computed by backpropagation at this point. At each epoch of the adversarial training procedure, we solve the underlying minmax problem [89]. More details on the generation of adversarial attacks for regression attacks can be found in [20].

The generated adversarial attacks from the trained model at the previous epoch are successively concatenated to the training set for the next training epoch, much like in standard adversarial training practices using FGSM [1] and Deepfool [53] attacks. While generating adversarial attacks on tabular data, some of the variables may be more susceptible to attacks than others. The authors of [80] take care of this aspect by using a feature importance vector. They also only attack the continuous variables, disregarding categorical ones while generating attacks. For the Power plant and Boston Housing datasets, we attack all the four input variables, while on the MPG dataset, we attack only the continuous variables. For the industrial dataset, we generate attacks for the five most sensitive input variables. We also tried attacking all the variables of the dataset but this was not observed to be more efficient. The results in form of Lipschitz star are given in Figures 4.6-4.9.

As expected, adversarial training leads to a shrinkage of the star plots, which indicates a better control on the stability of the trained models, while also improving slightly the MAE. In the test we did, we observe however that our adversarial training procedure is globally less efficient than the spectral normalization technique.

4.4.7 . Sensitivity w.r.t. pair of variables

We now consider the case when the set $\mathbb{K}$ contains pairs of elements. We first show the corresponding *Partial Lipschitz constants* using a *Lipschitz star*, for the different datasets we have discussed in the article. Vertices in the Lipschitz star represent the obtained Lipschitz constant value $\vartheta_m^{\Omega_{\epsilon,\mathbb{K}}}$ for all possible combinations

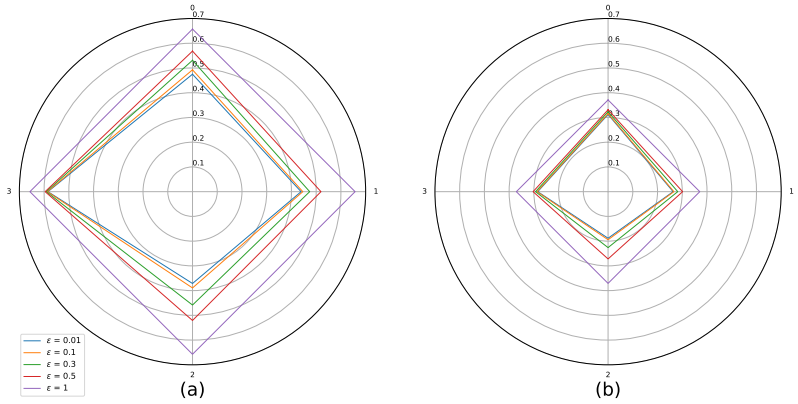

Figure 4.6: Sensitivity w.r.t. to each input on Combined Cycle Power Plant dataset. Effect of adversarial training. a) Standard training: Lipschitz constant = 0.657, MAE = 0.007, b) Adversarial training: Lipschitz constant = 0.37, MAE = 0.0068.

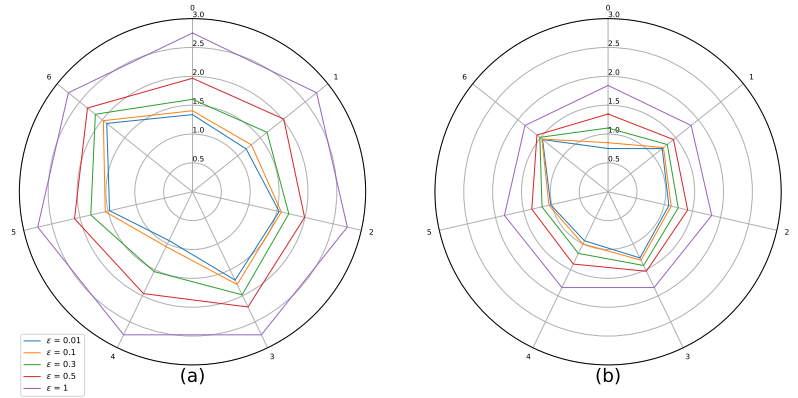

Figure 4.7: Sensitivity w.r.t. to each input on Auto MPG dataset. Effect of adversarial training. a) Standard training: Lipschitz constant = 2.75, MAE = 0.05, b) Adversarial training: Lipschitz constant = 1.84, MAE = 0.042.

of pair of variables with varying values of ϵ, i.e., it represents the sensitivity w.r.t. to that particular pair.

As shown by Figure 4.10-4.13, this Lipschitz star representation can be useful for displaying the influence of groups of variables instead of single ones. This may be of high interest when the number of inputs is large, especially if they can be grouped into variables belonging to a given class having a specific physical meaning (e.g., electrical variables versus mechanical ones). Such Lipschitz star

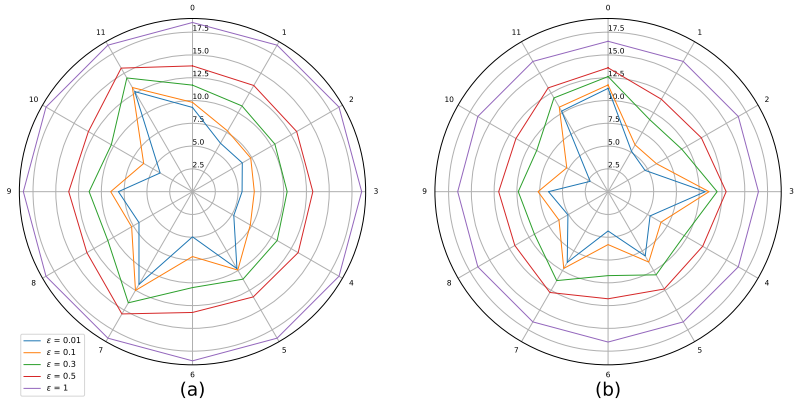

Figure 4.8: Sensitivity w.r.t. to each input on Boston Housing dataset. Effect of adversarial training. a) Standard training : Lipschitz = 18.56, MAE(y_1) = 2.45, MAE(y_2) = 1.41, b) Adversarial training: Lipschitz constant = 16.50, MAE(y_1) = 2.35 MAE(y_2) = 1.32.

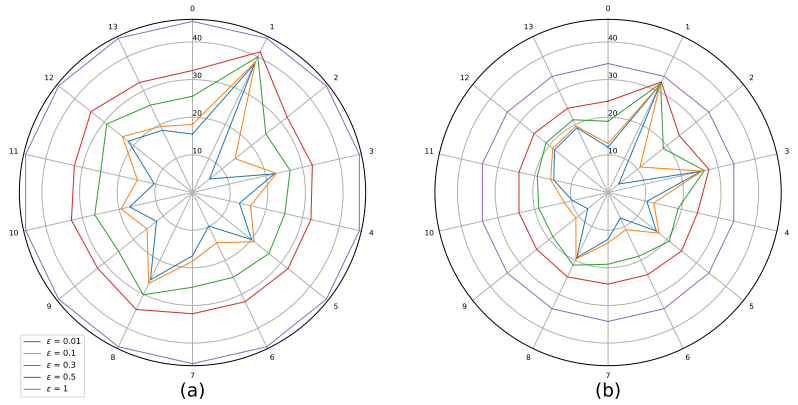

Figure 4.9: Sensitivity w.r.t. to each input on Thales Air Mobility industrial application. Effect of adversarial training. a) Standard training: Lipschitz = 45.47, MAE = 496.37 (s), Adversarial training. b) Lipschitz = 34.26, MAE = 494.7 (s).

representation might however not be very insightful for identifying the coupling that may exist between the variables within a given group. For example, it may happen that, considered together, two variables yield an increased sensitivity than the sensitivity of each of them individually. The reason why we need to find a better way for highlighting these coupling effects is related to Proposition 6v) which states

that, for every $\epsilon \in]0,1]$ and $(k,\ell) \in \{1,\dots,N_0\}^2$,

$$\max\{\vartheta_m^{\Omega_{\epsilon,\{k\}}}, \vartheta_m^{\Omega_{\epsilon,\{\ell\}}}\} \leqslant \vartheta_m^{\Omega_{\epsilon,\{k,\ell\}}}. \tag{4.44}$$

This property means that, when considering a pair of inputs, the one with the highest partial Lipschitz constant will "dominate" the other. To circumvent this difficulty and make our analysis more interpretable, we can think of normalizing the Lipschitz constant in a suitable manner. Such a strategy is a common practice in statistics when, for example, the covariance of a pair of variables is normalized by the product of their standard deviations to define their correlation factor. Once again, we can take advantage of the properties established in Proposition 6 to provide us a guideline to perform this normalization. In addition to Eq. (4.44), according to Property viii),

$$\begin{aligned}\vartheta_m^{\Omega_{\epsilon,\{k,\ell\}}} &\leqslant \left((\vartheta_m^{\Omega_{\epsilon,\{k\}}})^{p^*} + (\vartheta_m^{\Omega_{\epsilon,\{\ell\}}})^{p^*}\right)^{1/p^*} + o(\epsilon) \\ &\leqslant 2^{1/p^*} \max\{\vartheta_m^{\Omega_{\epsilon,\{k\}}}, \vartheta_m^{\Omega_{\epsilon,\{\ell\}}}\} + o(\epsilon).\end{aligned} \tag{4.45}$$

The two previous inequalities suggest to normalize the Lipschitz constant for pairs of inputs by defining

$$\widetilde{\vartheta}_m^{\Omega_{\epsilon,\{k,\ell\}}} = \frac{1}{2^{1/p^*}-1}\left(\frac{\vartheta_m^{\Omega_{\epsilon,\{k,\ell\}}}}{\max\{\vartheta_m^{\Omega_{\epsilon,\{k\}}}, \vartheta_m^{\Omega_{\epsilon,\{\ell\}}}\}} - 1\right). \tag{4.46}$$

Indeed, when ϵ is close to zero, Eq. (4.44)-Eq. (4.46) show that $\widetilde{\vartheta}_m^{\Omega_{\epsilon,\{k,\ell\}}} \in [0,1]$. Note that, for the diagonal terms, $\widetilde{\vartheta}_m^{\Omega_{\epsilon,\{k,k\}}} = 0$. The higher $\widetilde{\vartheta}_m^{\Omega_{\epsilon,\{k,\ell\}}}$, the higher the gain in sensitivity due to the coupling between k and ℓ. The normalized values for the different datasets are reported in Table 4.9.

4.4.8 . Interpretation of the results

We summarize some observations/properties concerning the stability of the NNs which can be drawn from training on different datasets and leveraging the quantitative tools we have proposed in this article.

(a) Combined Power Plant Dataset

- '3 – Exhaust Vacuum' is the most sensitive variable out of the four variables.

- We observe for any variable coupled with '3' gives a higher partial Lipschitz constant.

- From Table 4.9(a), we see that the effect is mostly caused by the sensitivity of '3' and there is no gain when coupled with other variables. Hence, '3' dominates the overall sensitivity of the NN.

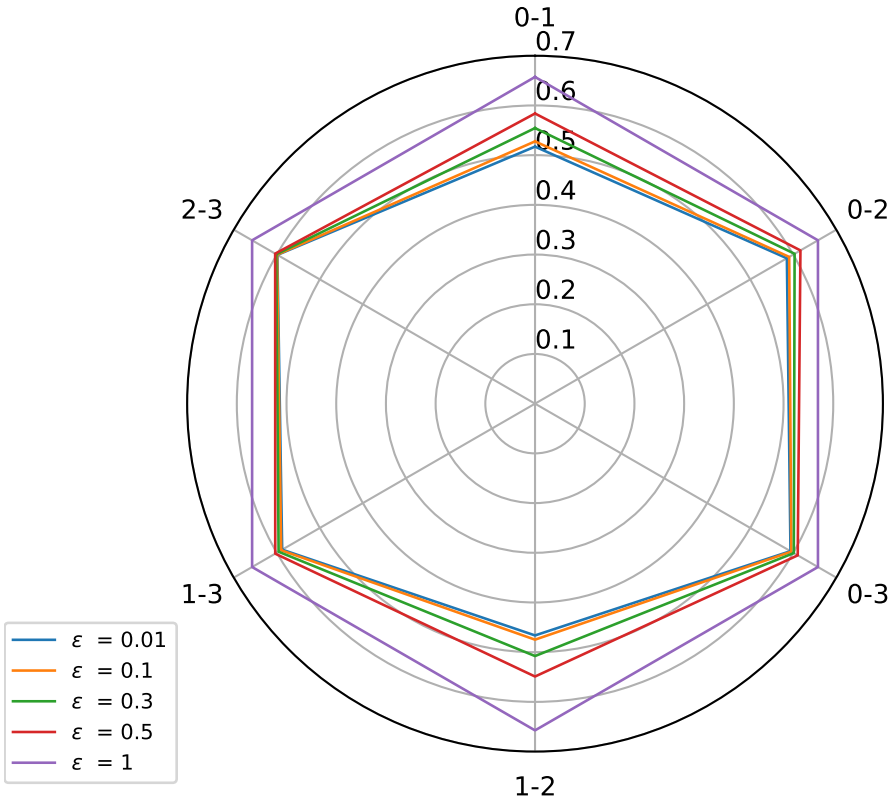

Figure 4.10: Sensitivity w.r.t to pair of variables on Combined Power Plant dataset

• On the other hand, we observe that, '0' when coupled with '1' and '2' becomes more sensitive as evidenced by the gain in Table 4.9(a).

(b) Auto MPG Dataset

• Variable '6 – Origin' and '3 – Weight' are the most sensitive variables.

• The values of partial Lipschitz constant peak when the other variables are coupled with '3' or '6'.

• From Table 4.9(b), we see that most of the values coupled with either '3' or '6' are close to zero, except when '3' and '6' are coupled together. Also, we see an exception when '5' is coupled with either '3' or '6'. This suggests that altogether '3', '5', and '6' have a higher impact on the stability of the network.

(c) Boston Housing Dataset

• Variable '7 – DIS' and '11 – LSTAT' are the most sensitive variables.

• We observe a high partial Lipschitz constant when coupling any variables with '7' or '11'.

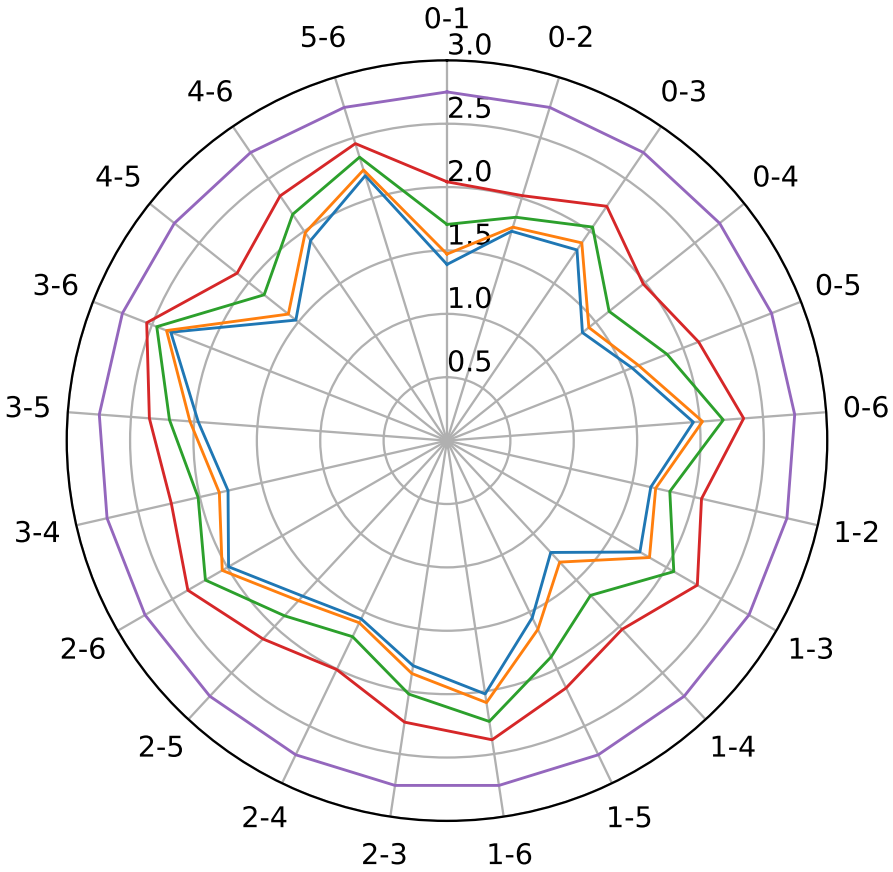

Figure 4.11: Sensitivity w.r.t to pair of variables on Auto MPG dataset.

• From Table 4.9(c), we see that all the values for both '7' and '11' coupled with other variables are close to zero, except when '7' and '11' are jointly considered. Hence, '7' and '11' dominate the sensitivity of the NN.

• We observe from the table of normalized values, that '2-9' have a higher impact on the sensitivity of the NN when coupled. Similar observation can be made for pairs '2-8','1-4','3-4'.

(d) Thales Air Mobility industrial application

• Variable '1 – Flight distance','3 – Initial ETE', and '8 – Longitude Destination' are the most sensitive variables.

• We see peaks in the partial Lipschitz constant values when these highly sensitive variables are coupled with other variables.

• But when analyzing the normalized tables, it becomes clear that the gain is mostly due to these sensitive variables.

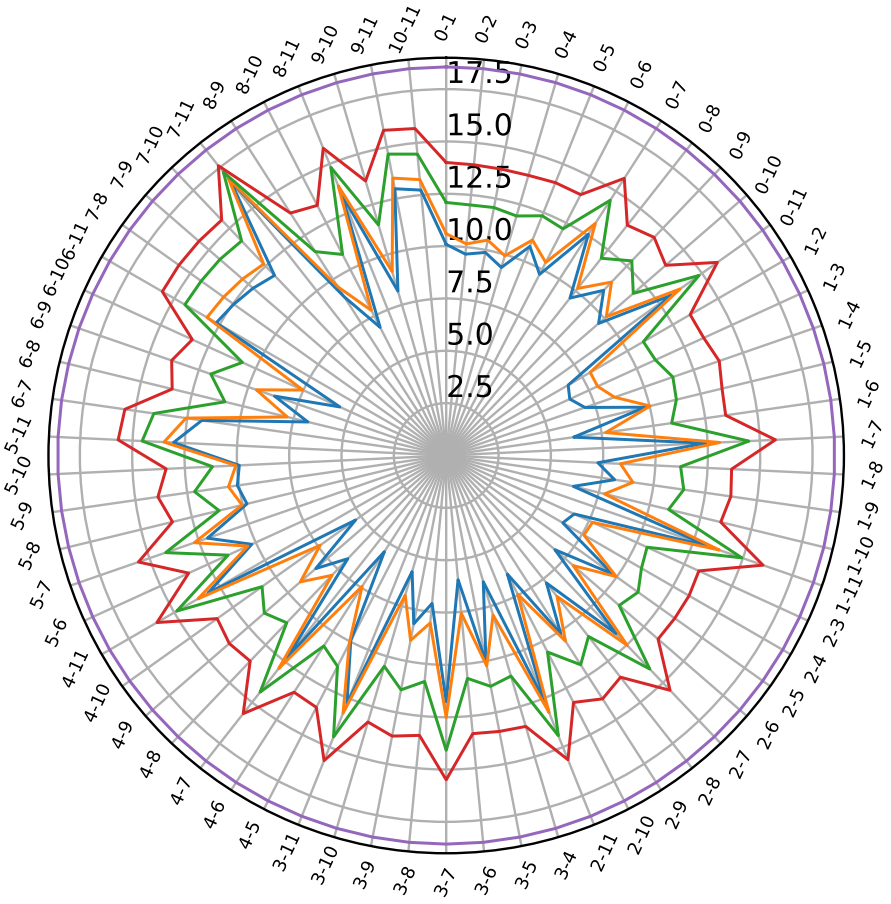

Figure 4.12: Sensitivity w.r.t to pair of variables on Boston Housing Dataset.

- We also observe from Table 4.9(d), an increased sensitivity of '0' when coupled with other variables '5', '7', '10', '11', and '13'.

4.5 . Summary

The contributions of this work are summarised as follows:

- This chapter proposes a new multivariate analysis of the Lipschitz regularity of a neural network. The theoretical foundations of our approach are given in Section 4.3.

- A multivariate analysis of the Lipschitz properties of NNs is performed by generating a set of partial Lipschitz constants. This opens a new dimension to studying the stability of NNs.

- Our sensitivity analysis allows us to capture the behaviour of an individual input or group of inputs on the output of the neural network.

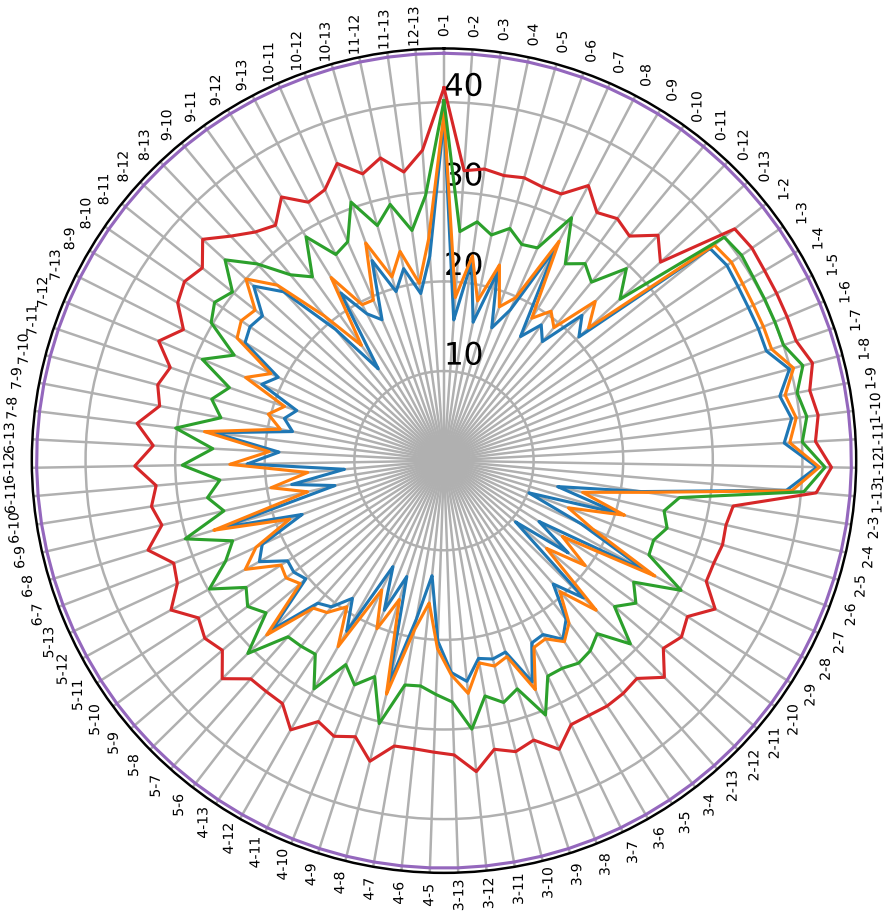

Figure 4.13: Sensitivity w.r.t to pair of variables on Thales Air Mobility industrial application.

- The results of this analysis are displayed by a new graphical representation termed as a Lipschitz star. This representation is helpful in displaying how each input or group of inputs contributes to the global Lipschitz behaviour of a network.
- Using the proposed analysis, we also study quantitatively the effect of spectral normalization constraint and adversarial training on the stability of NNs.
- We showcase our results on various open-source datasets along with a real industrial application in the domain of Air Traffic Management.

Variable	1	2	3
0	0.22	0.57	0.04
1		0.15	0.01
2			0.06

(a)

Variable	1	2	3	4	5	6
0	0.1	0.29	0.18	0.06	0.16	0.05
1		0.17	0.08	0.03	0.12	0.15
2			0.14	0.03	0.20	0.11
3				0.11	0.39	0.56
4					0.08	0
5						0.34

(b)

Variable	1	2	3	4	5	6	7	8	9	10	11
0	0.22	0.11	0.16	0.03	0.25	0.12	0.17	0.09	0.43	0.11	0.10
1		0.17	0.16	0.37	0.00	0.00	0.14	0.18	0	0.05	0.11
2			0.17	0.05	0	0.23	0	0.35	0.61	0.02	0.00
3				0.35	0.07	0.21	0	0.12	0.02	0.16	0.01
4					0.05	0.11	0.11	0.01	0.04	0.06	0.08
5						0.11	0.07	0.01	0.08	0.04	0.07
6							0.01	0	0.16	0.35	0
7								0.15	0.1	0.03	0.76
8									0.27	0.04	0.14
9										0.02	0.06
10											0.01

(c)

Variable	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.01	0.03	0.01	0.03	0.21	0.03	0.23	0.09	0.11	0.21	0.27	0.06	0.28
1		0	0.03	0.01	0	0	0	0.12	0.01	0.07	0	0.24	0.03
2			0	0.04	0.01	0.05	0.02	0	0.06	0.01	0.02	0	0.01
3				0.01	0.19	0	0.03	0.08	0.01	0.05	0.01	0.26	0.15
4					0	0.02	0.17	0	0.13	0.11	0.06	0.01	0.01
5						0.06	0.13	0.11	0.02	0.08	0.07	0.19	0.27
6							0.03	0.01	0.04	0	0.19	0.02	0.01
7								0.01	0.07	0.19	0.09	0.02	0.32
8									0.01	0.06	0.02	0.29	0.03
9										0.27	0.06	0.02	0.03
10											0	0.21	0.16
11												0.01	0.07
12													0.12

(d)

Table 4.9: 2^{nd} order normalized coupling matrix with $\epsilon = 0.001$ on a) Combined Power Plant Dataset b) Auto MPG Dataset c) Boston Housing Dataset and d) Thales Air Mobility industrial application.

Chapter 5
Spectral Normalization Loop for controlling stability

5.1 . Introduction

As discussed in the previous chapter, a form of formal measure of stability of neural network is the Lipschitz constant of the model which allows to evaluate how small perturbations in the inputs impact the output variations. With this chapter we move to controlling and tightening the Lipschitz bounds of the neural networks to promote robustness. We show the efficiency of a formal method based on the Lipschitz constant for quantifying the stability of neural network models. We present how this formal method can be coupled with spectral normalization constraints at the design phase to control the internal parameters of the model and make it more stable while keeping a high level of performance **(accuracy-stability trade-off)**. We first, show the efficacy of the proposed control loop on a public tabular dataset related to German Credit risk and then move to safety critical application. We evaluate our approach on an application related to Small Unmanned Aerial Vehicles (sUAVs) which are in the frontage of new technology solutions for intelligent systems. However, real-time fault detection/diagnosis in such UAVs remains a challenge from data collection to prediction tasks. This work presents a novel application of neural networks to detect in real-time elevon positioning faults to allow the remote pilot to take necessary actions to ensure the safety (e.g.remote triggering of the drone emergency parachute). In both scenarios, we show the effectiveness of our proposed algorithm in reaching a optimal model following both accuracy and stability targets that are user specified.

5.2 . Limitations of Previous Works

As introduced in the Section 2.3.2, the Lipschitz constant can be used to train more robust models. The existing methods are hindered with several limitations. One being the lack of accuracy-vs-stability trade-off. Randomly constraining neural networks to have very tight Lipschitz bounds or maybe making them 1-Lipschitz

as in the case of Spectral Normalization will lead the accuracy to drop drastically on clean data. It is possible that, under very tight constraints, the networks fails to learn anything. Such a reduction in the performance is not acceptable in applications such as UAVs. Engineers need to maintain respectable levels of accuracy and stability to deploy any solution in that case. To mitigate this effect, we propose a clever use of Spectral Normalization to design a stability control loop allowing us to lower the Lipschitz bounds while maintaining a good performance.

5.3 . Proposed Design method

For safety critical tasks, Lipschitz constant and performance targets can be specified as engineering requirements, prior to network training. A Lipschitz constant target can be defined by a safety analysis of the acceptable perturbations for each output knowing the input ranges and it constitutes a current practice in many industrial processes. The Lipschitz constant of a neural network can be arbitrarily high when unconstrained, making it sensitive to adversarial attacks. Imposing this Lipschitz target can be done either by controlling the Lipschitz constant for each layer or for the whole network, depending on the application at hand. Spectral normalization techniques [70, 90] have been proved to be very effective in controlling stability properties of DNNs and counter adversarial attacks. But they require an exact computation of the spectral norm in Eq. (2.18), which is computationally expensive, and therefore it becomes intractable for deep neural networks. On the other hand, it has been shown that constraining the Lipschitz constant of DNNs enhances their stability, but makes the minimization of the loss function hard. Hence, a trade-off between stability and prediction performance needs to be reached [91]. We subsequently propose an iterative procedure allowing us to find the best trade-off.

Consider an m-layer fully connected NN architecture (FCN) denoted by T with a desired Lipschitz constant θ_{target} and an accuracy target $\text{acc}_{\text{target}}$. We can constrain the spectral norm of each layer to be less than $\sqrt[m]{\theta_{\text{target}}}$. According to Eq. (2.7), this ensures that the upper bound on the global Lipschitz constant is less than θ_{target}. This bound is however over-pessimistic, which means that, by monitoring accurately the value of the actual Lipschitz constant of the network, we can relax the bound on individual linear layers so as to get an improved accuracy. In our proposed approach, this process is done in a closed loop with two targets : good performance ($\text{acc}_{\text{target}}$) and desired stability θ_{target}.

The approach is summarized in Algorithm 1. Module $\text{Train}_{\text{SN}}(\theta, \ldots)$ is a spectral normalization training module which trains the defined neural network model with the Lipschitz constant constraint, i.e. each weight matrices W_i with $i \in \{1, \ldots, m\}$ of the neural network is divided by $(\sigma_i / \sqrt[m]{\theta})$, where σ_i is the largest singular value of W_i. Module LipEst estimates accurately the Lipschitz constant of the trained neural network using LipSDP-neuron[16]. Module Predict checks

the performance (classification accuracy) of the trained model on the validation dataset. If θ is small, the training will be over-constrained and the accuracy of the model will suffer. To relax this constraint, we iteratively increase the value of θ through a factor α close to 1 to find the best value θ_{opt} which would be less than θ_{target} and ensure an increase in the performance on the validation dataset to achieve acc_{opt} greater than $\text{acc}_{\text{target}}$. We stop the iterations when both conditions are met i.e. the accuracy of the trained model is greater than $\text{acc}_{\text{target}}$ and the Lipschitz constant is less than θ_{target}.

Algorithm 1 Spectral Normalization-based Stability Control Loop

Inputs : training dataset $(X_{\text{train}}, Y_{\text{train}})$, validation dataset $(X_{\text{valid}}, Y_{\text{valid}})$, target Lipschitz constant θ_{target}, target accuracy $\text{acc}_{\text{target}}$, neural network T, multiplicative factor $\alpha > 1$, maximum iteration number $n_{\max}$
Output : optimal neural network T_{opt}, Lipschitz constant θ_{opt}, and accuracy acc_{opt}
Algo :
$L \leftarrow \theta_{\text{target}}$, $n \leftarrow 0$
$T \leftarrow \text{Train}_{\text{SN}}(\theta, X_{\text{train}}, Y_{\text{train}})$
$\theta_{\text{est}} \leftarrow \text{LipEst}(T)$
$\text{acc} \leftarrow \text{Predict}(T, X_{\text{valid}}, Y_{\text{valid}})$
while $(n < n_{\max})$ and $(\theta_{\text{est}} < \theta_{\text{target}})$ and $(\text{acc} < \text{acc}_{\text{target}})$ **do**
    $\theta \leftarrow \alpha\,\theta$
    $T \leftarrow \text{Train}_{\text{SN}}(L, X_{\text{train}}, Y_{\text{train}})$
    $\theta_{\text{est}} \leftarrow \text{LipEst}(T)$
    $\text{acc} \leftarrow \text{Predict}(T, X_{\text{valid}}, Y_{\text{valid}})$
    $n \leftarrow n + 1$
end while
$T_{\text{opt}} \leftarrow T$
$\theta_{\text{opt}} \leftarrow \theta_{\text{est}}$
$\text{acc}_{\text{opt}} \leftarrow \text{acc}$
if $n = n_{\max}$ **then**
    Notification: No trade-off found between Stability and Accuracy
else
    Notification: Trade-off found between Stability and Accuracy
end if

θ_{target}	θ	$acc(\%)$	$\text{acc}_{\text{target}}(\%)$	θ_{prop}	$acc_{\text{prop}}(\%)$
1	1.02	69.4	70	1.00	70.0
2	1.56	70.4	70	1.95	71.2
3	1.90	70.9	70	2.73	72.8
4	2.49	71.2	70	3.50	74.2
5	2.78	72.3	70	4.73	74.4
8	3.42	72.6	75	7.59	75.2
12	5.2	73.1	75	10.11	76.4
14	5.8	73.7	75	13.29	75.6

Table 5.1: Results on German Credit dataset using the proposed algorithm.

5.4 . Experiments

5.4.1 . Open source dataset

We first show the efficacy of the proposed stability control loop on an Open source dataset. We use Statlog (German Credit Data) Data Set. It has a classification setting of 1,000 instances with 23 continuous features per instance. The goal is to determine credit-worthiness of customers (Good or bad) . We build fully connected networks with ReLU activation function on all the hidden layers, except the last one. The architecture has two hidden layers with 50 and 20 neurons. The input attributes are scaled between 0 and 1.

If we train the neural network model without any stability constraint, i.e., perform standard training, the network performance is 75.2% with an estimate for the Lipschitz constant of 17.58. Table 5.1 tabulates the results for basic spectral normalization (with a Lipschitz target) in columns 2 and 3. The value of θ is determined using LipSDP-neuron. The results with our proposed method, adding an accuracy target, are given in columns 5 and 6. For the first set of results, we train this model without any accuracy target and perform a simple Spectral normalization to constrain the Lipschitz norm of each layer to 1; we see a reduction in the accuracy by more than 5% with respect to the standard trained model. The accuracy improves slowly as the Lipschitz constraint is relaxed, but the values are much lower than the target value. For the second set of results, we provide both accuracy and Lipschitz targets so as to obtain a better solution by using Algorithm 1. We see an increase in the overall performance of the models. We see a similar increase of the accuracy as the stability constraints are relaxed and it can even become better than the baseline in some cases. We also observe that the upper bounds for the Lipschitz constant provided by LipSDP are much closer to the targeted values. For an accuracy target of 75% with a Lipschitz target less

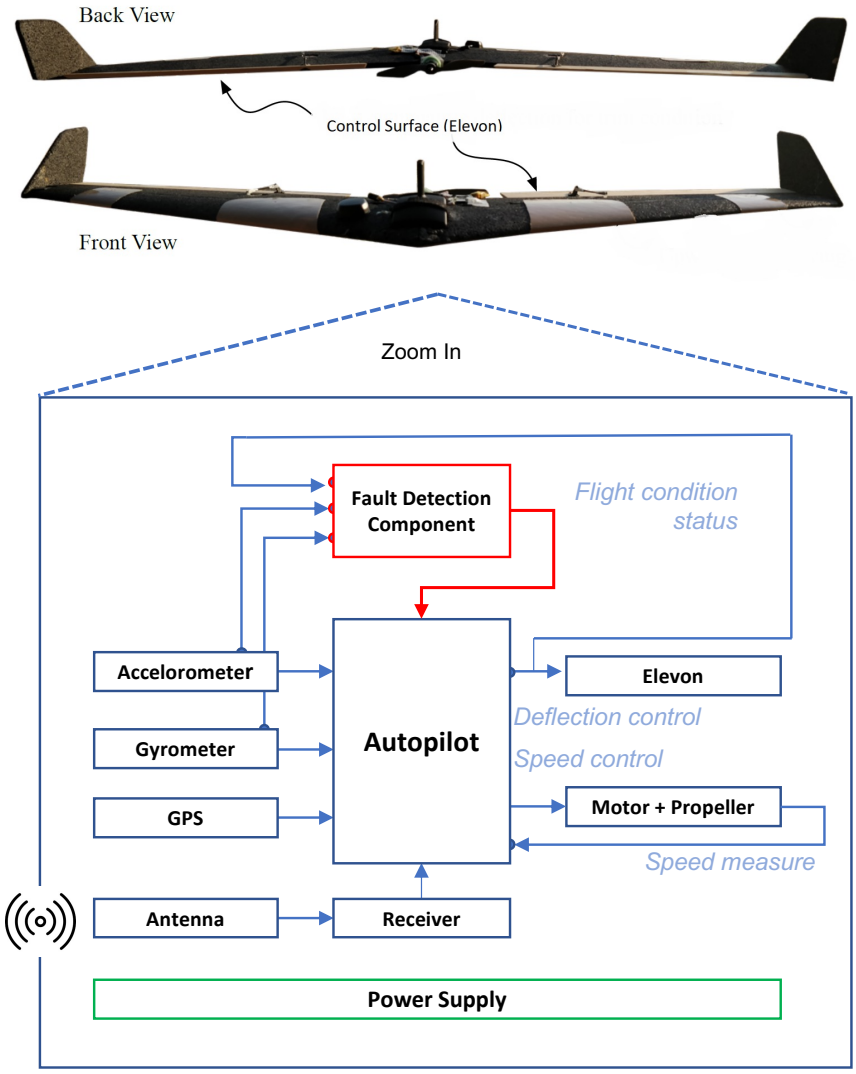

Figure 5.1: Fault Detection in drones/UAVs : Inertial sensors of the drone: accelerometer measures translational acceleration, and the gyrometer measures angular velocities (rotational motion). By combining these measurements the flight controller is able to calculate the drone current attitude (angle of flying) and perform necessary corrections to the measurements. The component called "Fault Detection Component" computes a flight condition status (nominal or faulty) using the outputs of the inertial sensors and the elevon deflection control variable produced by the autopilot

.

than 5, the algorithm fails to find a trade-off between stability and accuracy. We report the results for θ_{target} less than 5 with $\text{acc}_{\text{target}}$ of 70% in Table 5.1. The best solution in terms of accuracy is obtained with $\theta_{\text{target}} = 12$ and $\text{acc}_{\text{target}} = 75\%$.

5.4.2 . UAV use case

Unmanned Aerial Vehicles (UAVs) are currently being used in various applications such as surveillance, home delivery of products by e-commerce websites, disaster management, emergency services, efficient farming, insurance underwriting, etc. UAVs are capable of replacing humans in potentially dangerous situations, but also reduce the cost of operations in some businesses, these being the main advantages of using drones. But systems of this kind need to be continuously monitored to detect in real time any problem (called a "fault") that could lead to a failure of the system, to the potential loss of the UAV, and thus an interruption of the mission. Broadly, a fault analysis algorithm comprises of a continuously monitoring a system during its operation for detecting faults (Fault Detection), locate faults (Fault Isolation) and predict their temporal evolution (Fault Identification). This projected surge of use in the future by all industries makes real-time fault diagnosis of these systems a priority to avoid further critical losses i.e there can be many mission or safety critical implication to the faults if left undetected. Fault diagnosis techniques are becoming increasingly important in order to ensure high levels of safety and reliability in aeronautical systems. This gives rise for the need for more intelligent and reliable systems in the future. Hence, the advances in UAV or aerial vehicle technology is accompanied by the concern of safety. Designing a system that is fault tolerant is a recognized approach in the field of dependability engineering. Formally, fault tolerance is the ability of a system to perform its intended function despite the presence or occurrence of faults, whether physical damage to the hardware, software defects [92], or malicious attacks. In this work, we will only deal with the fault detection capability that is depicted in Figure 1. It is designed to detect mechanical faults like an elevon stucked or locked in a given position. Other dimensions of fault tolerance involving error recovery mechanisms or other safety measures are not addressed here.

5.4.3 . Limitation of AI in UAV use cases

Machine Learning methods and in particular neural networks are rapidly paving their way in various mission-critical and safety-critical domains such as aeronautics, aerospace, automotive, railways, and nuclear plants. In particular, UAVs are currently being used in various applications such as surveillance of critical infrastructures, delivery of goods in densely populated urban areas, disaster management, emergency services, etc. UAVs are capable of replacing humans in potentially dangerous situations, but also reduce the cost of operations in some businesses. But these kind of systems need to be continuously monitored to detect in real-time any problem ("fault") that could lead to a failure of the UAV and with a potential interruption of the mission, a potential collision with a manned aircraft (air risk), a potential collision with people on ground or with a critical infrastructure like a nuclear plant (ground risk).

AI based data-driven techniques are gaining more and more interest in the field

of fault detection. Fault detection in UAVs has been studied by using different sensors and/or for detecting faults in different fault prone parts of the vehicle. Freeman et al.[93] compares model-based and data-driven fault detection methods by using data from small, low-cost UAVs. Both approaches were shown to be capable of detecting different elevon faults in real data during maneuvers and in the presence of environmental variations. Guo et al.[94] proposed a hybrid feature model and deep learning-based fault diagnosis for UAV sensors. The model detects different sensor faults, such as GPS, IMU and ADS. The authors of [95] modelled fault detection using Generalized Regression Neural Network with a particle swarm optimization algorithm. Eroglu et al. [96] proposed a data-driven fault estimation method for estimating actuator faults from aircraft state trajectories. The authors propose to use an architecture with 1-D CNN followed by LSTM layers for learning state trajectories on simulated data of fault injected aircraft. Sadhu et al. [97] proposed a deep learning based real-time fault detection and identification on UAV IMU sensor data using CNN and Bi-LSTM. Iannace et al. [98] presented a single layered neural network for detecting unbalanced blades in a UAV propeller. Bronz et al. [99] uses an SVM classifier with RBF kernel to classify the different faults for small fixed-wing UAVs. We use the same real flight dataset for our analysis and make comparisons with the reported results for SVM classifier.

5.4.4 . Dataset Description

For our fault detection and stability analysis, we use the real-flight dataset provided in [99]. The authors use Paparazzi Autopilot system [100] for possible flight trajectories. The flight test data is captured in outdoor setting on different days with varying environments especially wind speeds and also imperfections in the geometry due to manufacturing of the UAV.

The actuator fault model (F) as proposed in [101] is given by

$$u_{\text{app}} = F u_{\text{com}} + E \tag{5.1}$$

where u_{com} is the commanded control deflection from the autopilot to the actuators, u_{app} is the applied control deflection (i.e. final movement of the elevon), and E is the offset. In vector form, it can be expressed as

$$\begin{bmatrix} u_{\text{app}_\text{r}} \\ u_{\text{app}_\text{l}} \end{bmatrix} = \begin{bmatrix} f_\text{r} & 0 \\ 0 & f_\text{l} \end{bmatrix} \begin{bmatrix} u_{\text{com}_\text{r}} \\ u_{\text{com}_\text{l}} \end{bmatrix} + \begin{bmatrix} e_\text{r} \\ e_\text{l} \end{bmatrix}, \tag{5.2}$$

where $()_\text{r}$ and and $()_\text{l}$ stand for the right and left elevon respectively. The feature set is a vector of length 8 consisting of linear accelerations at three coordinates (a_x, a_y, a_z), angular rates $(\omega_x, \omega_y, \omega_z)$, and auto-pilot commanded controls $(u_{\text{com}_\text{r}}, u_{\text{com}_\text{l}})$ for the two aerodynamic actuators. Depending on the values of f_r and f_l, the fault detection problem can be translated into two kinds of classification problems: 1) Binary Class Classification and 2) Multi-class Classification. Each fault category is labeled according to its fault code in order to generate the

Input	0	a_x
	1	a_y
	2	a_z
	3	ω_x
	4	ω_y
	5	ω_z
	6	u_{com_r}
	7	u_{com_l}
Output	Class	Nominal or faulty

Table 5.2: Input and output attributes of UAV dataset for fault detection. a_* - Linear acceleration , ω_* - angular velocity , u_* - autopilot command controls.

multi-labeled data. Data have been split into two parts containing 80% training data and 20% test data.

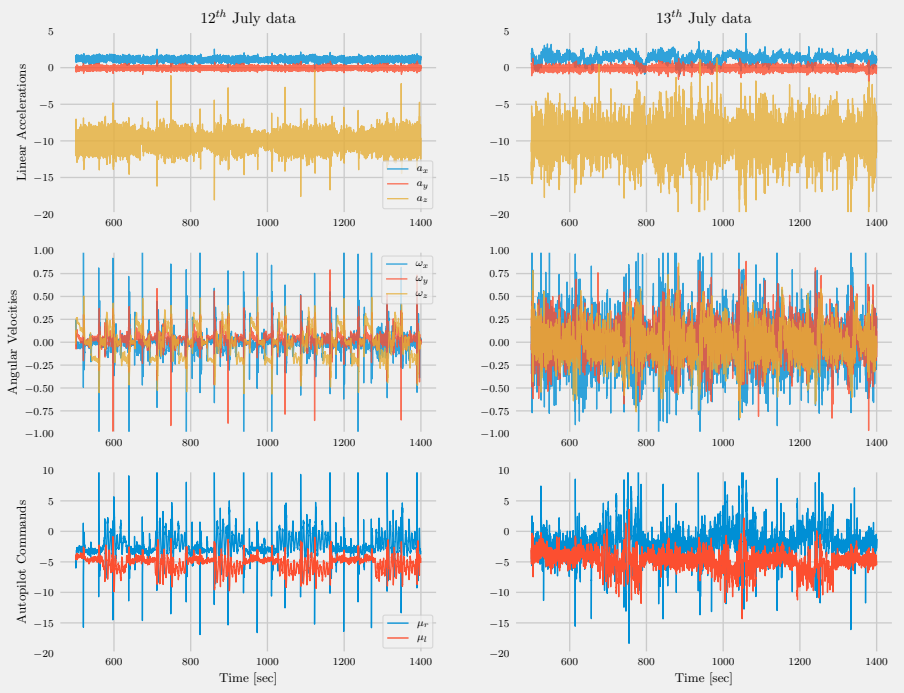

Figure 5.2: Time variations in the features of the tabular dataset for small fixed-wing UAV on 12^{th} and 13^{th} July.

.

We use three performance measures: classification accuracy (%), F1 score, and Matthews Correlation coefficient (MCC). For stability measure we use Lipschitz constant estimate as calculated using LipSDP-Neuron [16]. We recall that, for

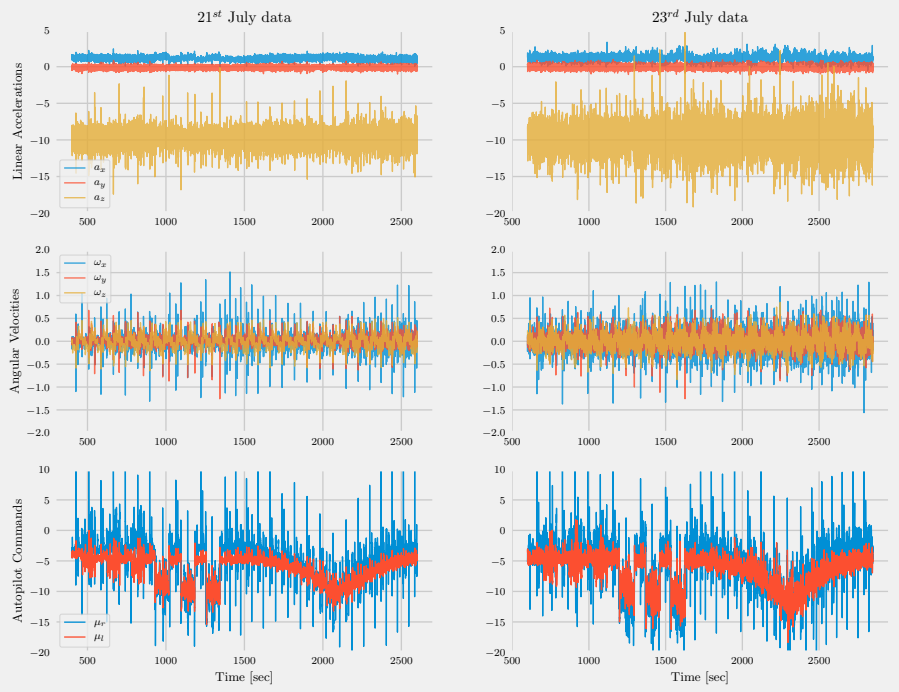

Figure 5.3: Time variations in the features of the tabular dataset for mall fixed-wing UAV on 21^{st} and 23^{rd} July.

.

LipSDP, we do not use the tightest bound LipSDP-Network, since it is erroneous as mentioned in Section 2.2.1.2.

5.4.5 . Binary Classification Results

We train the data on 12^{th} July and test the trained model on 12^{th} july unseen and 13^{th} July data. The 12^{th} and 13^{th} July flights contain approximately 15 minutes of similar fault pattern of reduced efficiency of right elevon ($f_r = 0.3$) making it suitable for binary classification. It is interesting to use 12^{th} July flight data for training and predict fault for the next day, i.e 13^{th} July since 12^{th} and 13^{th} July had different atmospheric conditions at the time of data collection. For further checking the efficacy and generalization capabilities of the methods, we checked the trained models on flight data for 21^{st} July. The faults were injected in the right elevon $f_r = 0.3$. Fault code 0 implies 'Nominal fault' and fault code 1 implies fault in the right elevon.

Implementation details: We propose to use fully connected feed-forward networks (FCN) for solving the problem. We train such neural networks of varying depth and width. We use ReLU as the activation function except for the last layer where we employed sigmoid for binary classification. We use the cross entropy function as the loss function for training the model. The feature set is standardized by removing the mean and scaling to unit variance. While training, the dataset

Model	12^{th} July			13^{th} July			21^{st} July		
	Acc. (%)	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**
SVM	98.7	0.99	0.98	60.5	0.59	0.38	85.3	0.85	0.72
LogReg	89.8	0.90	0.79	58.0	0.59	0.31	56.6	0.45	0.18
Baseline	98.6	0.99	0.97	73.3	0.74	0.52	93.1	0.93	0.86

Table 5.3: Binary Classification : Comparison between SVM, logistic regression, and best NN trained with baseline training on 12^{th} July.

Model	12^{th} July			13^{th} July			L
	Acc. (%)	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**	
(10)	98.3	0.983	0.967	66.4	0.67	0.41	12.26
(10,6)	98.4	0.985	0.969	71.0	0.71	0.47	27.56
(50,10,6)	98.7	0.987	0.974	73.3	0.74	0.52	37.76

Table 5.4: Binary Classification : Results on 12^{th} and 13^{th} July using NN trained with baseline training on 12^{th} July for various configurations of FCN.

is split into 4:1 ratio for train and validation sets. The models are trained using Keras with Tensorflow backend. The initializers are set to Glorot uniform and we use Adam as the optimizer. To capture the temporal dynamics of the system, feature set of length 8 $(a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, u_{\text{com}_\text{r}}, u_{\text{com}_\text{l}})$ is concatenated T times. Therefore, the input feature vector to the model is of length $8T$. We observe a general increasing trend in the testing performance as we increase the value of T, which suggests that concatenating more feature lists from the time history of the flight data improves the performance of the real-time prediction. Hence, we found the value $T = 20$ as optimal for our experiments. Adding time history beyond this level did not contribute to the learning and the neural network starts to overfit.

In the **first** set of experiments for binary classification, we train on 12^{th} July flight data and test on of 12^{th} July, 13^{th}, and 21^{st} July, and directly compare our NN best models to SVM and logistic regression. SVM classifier is trained with a RBF kernel and is optimized using grid-search. The results are given in Table 5.3. The results obtained using SVM classifier are satisfactory, better than for logistic regression, but they fail to generalize on the unseen data from 13^{th} and 21^{st} July. We remark that it is possible to train neural networks with much better generalization performance on different days with varying weather conditions.

In the **second** set of experiments we discuss the stability performance of neural networks which is our primary concern pertaining to stable model design. The comparison of the classification task accuracy as well as the stability performance

Model	Baseline		Spectral	
	Acc. (%)	L	Acc. (%)	L
(10)	66.4	12.26	75.1	3.7
(10,6)	71.0	27.56	75.3	3.8
(50,10,6)	73.3	37.76	76.7	3.7

(a) Binary Classification: Results on 13$^{\text{th}}$ July trained on 12$^{\text{th}}$ July data.

Model	Baseline		Spectral	
	Acc. (%)	L	Acc. (%)	L
(50,10)	69.7	31.44	72.65	3.8
(100,10)	69.54	34.17	71.07	3.6
(100,75,10)	68.9	69.06	71.93	3.7

(b) Multi-class Classification: Results on 23$^{\text{rd}}$ July trained on 21$^{\text{st}}$ July data.

Table 5.5: Comparison between models trained with baseline and spectral normalization constraint ($\theta_{\text{target}} = 4$).

Model	Training	No attack		FGSM ($\epsilon = 0.2$)		PGD($\epsilon = 0.2$)	
		Acc. (%)	MCC	Acc. (%)	MCC	Acc. (%)	MCC
(50,10,6)	**Baseline**	98.6	0.97	77.6	0.62	76.0	0.59
	Spectral	98.7	0.97	81.0	0.66	79.2	0.62

Table 5.6: Binary Classification: Results on 12$^{\text{th}}$ July clean data, FGSM and PGD attacks using model trained on 12$^{\text{th}}$ July.

in terms of Lipschitz constant for a varying number of hidden layers and hidden neurons is shown in Table 5.4. Notation (10,6) in Table 5.4 implies that the FCN has two hidden layers with 10 and 6 hidden neurons and so on. In the baseline training, no constraints are employed while training the NN. The Lipschitz constant (L) is computed using LipSDP-neuron [16] on the trained network.

Third, we show the performance and stability effect of constraining the training of the neural network by spectral normalization, as introduced in Section 5.3 with Algorithm 1. We chose the same configuration for the neural networks as the baseline and introduced constraints by setting $\theta_{\text{target}} = 4$ and $\text{acc}_{\text{target}} = 75\%$. Table 5.5a shows a comparison of the classification accuracy and Lipschitz constant value for baseline training and the new training procedure described in Algorithm 1. We remark that the proposed approach improves both accuracy and stability of neural network models.

We also test and compare the results of the trained neural networks when they

are attacked with widely known adversarial attacks such as FGSM [1] and PGD [102]. The attacks are generated using ART toolbox.[1] It is useful to test the robustness of the neural network since the adversarial noise is crafted in a manner to generate misclassification errors. We show the results on 12^{th} July test dataset in Table 5.6.[2] We remark that spectral normalized training is better in handling the adversarial attacks, leading to more stable models.

5.4.6 . Multi-class Classification Results

We use the flight data captured on the days of 21^{st} and 23^{rd} July for detecting the faults in right and left elevons. For injecting the faults, efficiency of the right elevon is reduced (f_{r}) and similarly for the left control elevon (f_{l})Fault codes for different classes are : '0' - nominal , '1' - $f_{\text{r}} = 0.3$, '2' - $f_{\text{l}} = 0.9$ and '3' - $f_{\text{l}} = 0.3$, defining a 4-class classification problem. The emphasis has been put on detecting faults on the left elevon which has the effect of a geometric twist. For training the neural network, flight data are selected between 400 and 2600 seconds. Identical faults are injected in the flight data for both days. For capturing the temporal dynamics in the prediction performance, the optimal value of T is taken to be 20 as in the previous section. It is to be noted here again that the flights made on different days will be affected by the environmental factors, which will impact the real-time prediction of the faults.

Model	21^{st} July			23^{rd} July		
	Acc. (%)	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**
SVM	94.1	0.94	0.91	60.3	0.60	0.51
LogReg	89.0	0.88	0.84	57.8	0.55	0.43
Baseline	95.2	0.95	0.93	69.7	0.68	0.58

Table 5.7: Multi-class Classification: Comparison between SVM, logistic regression and best NN trained with baseline training on 21^{st} July and tested on 21^{st} and 23^{rd} July .

For the experiments, we train a FCN with ReLU activation function for all layers, except the last one for which softmax is used. We train the neural network on the flight data of 21^{st} July and test on data of 21^{st} July not seen by the model at the training time, and also on the flight data of 23^{rd} July. We show the prediction and generalization capabilities of the neural networks over state-of-the-art SVM and logistic regression. We tabulate the classification metrics

[1] https://github.com/Trusted-AI/adversarial-robustness-toolbox

[2] In case of SVM model, classification metric MCC had negative values for FGSM and PGD attacks implying model performs very poorly against adversarial attacks, hence we discard them from comparison.

Model	21^{st} July			23^{rd} July			L
	Acc. (%)	**F1**	**MCC**	**Acc. (%)**	**F1**	**MCC**	
(50,10)	95.2	0.95	0.93	69.7	0.68	0.58	31.44
(100,10)	95.3	0.95	0.93	69.5	0.68	0.57	34.17
(100,75,10)	95.4	0.95	0.93	68.9	0.67	0.57	69.06

Table 5.8: Multi-class Classification : Results on 21^{st} July and 23^{rd} July using NN trained with baseline training on 21^{st} July for various configurations of FCN.

in Table 5.7. Varying configurations of model layers and neurons classification metrics along with the stability metric (L) have been considered in Table 5.8. We consider $\text{acc}_{\text{target}} = 70\%$ and $\theta_{\text{target}} = 4$ as the model performance and stability requirements. The results are shown in Table 5.5b.

5.5 . Summary

The contributions of this work are summarised as follows:

- We demonstrate the efficiency of formal guarantees based on the Lipschitz constant for checking the stability of DNNs. We present a stability control loop grounded on spectral normalization constraints to achieve an accuracy-stability trade-off.
- We first show the efficacy of our proposed algorithm on an open source dataset. Our algorithm achieved an optimal solution with a trade-off between performance and stability.
- Next, we analyse real flight data application for fault detection function embedded in an UAV. Through the evaluation of classification performance, we have shown that our approach based on DNN compares favorably with SVM and logistic regression techniques. Unlike many works related to the use of neural networks in industrial applications, we address the stability of our trained neural networks using a formal verification method based on the quantification of the Lipschitz constant of the model. We show the effectiveness of spectral normalization techniques in controlling the Lipschitz constant of the network to reach a Lipschitz safety target which can be defined at system level.
- We also demonstrate the effectiveness of spectral normalized models in handling adversarial tasks.

Chapter 6
Conclusion and Future Work

6.1 . Conclusion

Although they may appear at the top of their advancements, neural networks are hindered by security, privacy and safety issues due to their sensitivity to various attacks and perturbations they might encounter during their operation. It thus becomes essential to understand the causes of neural network instability, identify the concern areas, and provide solutions aiming to improve their stability, and ensure that AI-based systems work as desired during their whole operational phase.

In this thesis, we have discussed various limitations and challenging problems in the emerging field of robust AI, especially related to safety critical applications. We have emphasized that the Lipschitz regularity of the network is a useful concept to address stability issues from a mathematically rigorous perspective. In a nutshell, we have studied various concepts and latest developments towards formalizing the robustness and Lipschitz stability of neural network models. As shown in Chapter 2, the concepts of robustness is quite broad and varied. For example, according to experts in safety at Thales, we have to distinguish as:

- ML Model Robustness: The capacity of an ML model to preserve its expected/intended performance under well-characterized abnormalities (e.g. lack of generalization) or deviations to its inputs and operating conditions **outside** its Operational Design Domain (ODD).

- ML Model Stability: The capacity of an ML model to preserve its expected/intended output(s) under well-characterized and bounded perturbations to its inputs and operating conditions **within** its operational design domain .

The perturbations encountered can be natural (e.g. sensor noise, bias,...), due to failures (e.g. invalid data from degraded sensors) or maliciously inserted (e.g. pixels modified in an image) to fool the model predictions. Perturbations can also be simply defined as true data locally different from the original data used for

the model training and that might lead to a wrong prediction and an incorrect behaviour of the system.

Below are the summarised contributions of this thesis before discussing potential future work to promote the ideas presented in this thesis.

- We discussed a major gap in the field of adversarial machine learning and regression tasks. Regression tasks cover a wide range of safety- and mission-critical applications, and significantly fewer efforts have been put into understanding the vulnerabilities in regression models. To overcome this limitation, we have proposed a generic white-box attacker for regression neural networks. The Jacobian of the trained neural network is used to drive the attacks. Analogous to classification scenarios, useful metrics are proposed to quantify the effectiveness of these attacks.

- Our proposed attacker has several unique features such as attacking few or group of inputs and can address different norm perturbations on input and output. Our proposed attacker is shown to be more powerful than random attackers.

- Moving towards the notion of Lipschitz certificates which is the backbone of the stability analysis carried out in this thesis, we propose a new multivariate analysis of the Lipschitz regularity of a neural network by defining partial Lipschitz constants. The theoretical foundations of this approach are presented in section 4.3. Most current works utilizing Lipschitz certificates offer a single parameter analysis and limits the information that can be extracted from existing theoretical guarantees of stability.

- Our sensitivity analysis allows us to capture the influence of an individual input or group of inputs on the output of the neural network. The results of this analysis are displayed by a novel graphical representation termed as a Lipschitz star. This representation is helpful in displaying how each input or group of inputs contributes to the global Lipschitz behaviour of a network.

- For controlling the Lipschitz bound of the neural network, we present a stability control loop based on spectral normalization to train optimal neural models achieving performance and stability targets. Model trained with tight Lipschitz certificates are more robust to adversarial attacks and also maintain good accuracy.

- All of the algorithms and methods presented in this thesis are tested on various open-source tabular datasets, an air traffic management case provided by Thales LAS France and a real dataset of UAV safety application which is publicly available.

- Following the insights presented in the work, we have shown that it is possible to build neural networks which are "Stable-by-Design" and with guaranteed

stability through Lipschitz certificates. These results regarding robustness are strong developments towards understanding the vulnerabilities in ML models and provide defenses against attacks.

6.2 . Future work

The contributions of the thesis being summarised, a few possible future research directions based on the contributions and new concepts related to the work are mentioned in this section.

6.2.1 . Stronger Adversarial Attacks for Regression

The adversarial attacker presented in the thesis is effective, however stronger attacks can be achieved aligning to the classification scenarios. For generating adversarial attacks in chapter 3, we use fooling error to make the perturbed sample deviate from the ground truth sample. We can use stronger or multiple objectives to create stronger attacks. More sophisticated optimization algorithms such as augmented Lagrangian principles [57] and proximal splitting techniques [103] could be explored for this purpose in the context of regression tasks. Our tests concentrated on fully connected networks; it is worth pointing out that the proposed adversarial attacker can be applied to any network architecture, but it needs to be validated on other architectures.
Also, there is a growing body of work on intermediate-level attacks (ILAP) [104, 105] which try to eliminate or reduce the need for soft or hard labels. An initial direction - in the latent space of the hidden, intermediate layers - is determined via some suitable attack. Gradient steps are then used to maximally perturb the intermediate hidden representation to find a suitable adversarial attack. Interestingly, such a method can often outperform the method used to set the initial direction itself, while being more transferable across different models. These work eliminating the use of labels can help to design adequate adversarial attackers for regression tasks. Overall, the literature studying adversarial attacks and defenses for regression tasks is limited and needs more attention in future works.

6.2.2 . MIMO and Volterra Networks

One obvious extension of our analysis presented in chapter 4, where we have concentrated mostly on MISO (Multi-Input Single-Output) systems, is the sensitivity analysis of MIMO ((Multi-Input Multi-Output) systems. In chapter 4, we provide partial Lipschitz constants w.r.t to input when all the outputs are active—ignoring how the individual or group inputs can affect individual or a group of outputs. To get further analysis, we can easily calculate these partial Lipschitz constants corresponding to only specific outputs. Such analysis is important for understanding the sensitivity of inputs, but it can also help to control unintended couplings between variables that can be introduced by the neural network model.
We also observe a cross-link between the weighting matrices allowing us to obtain

partial Lipschitz constants and explainability masks used in images, highlighting the regions of the image which contribute most to the decision of CNNs to predict a certain label. Similarly, the partial Lipschitz constant highlights the importance of certain inputs w.r.t to output.

Another possible direction is to design new neural network architectures for tabular data learning using the concept of Volterra neural networks [106] with reduced or no use of activation functions. A neural network with a lower number of activation functions should be intuitively more robust with less regions where the approximation function exhibits steep slopes (i.e. higher local Lipschitz bounds), hence it is more stable to perturbations. We can design a neural network using second-order Volterra kernels to learn all relationships as in Eq. (4.36).

6.2.3 . Spectral Normalization and its variants

In chapter 5, we use the same level of Lipschitz normalization at each layer in the network. This might not be best way to normalize the Lipschitz constant of the layer because different layers in the neural network architecture learn more or less informative features, which implies that Lipschitz normalization should depend on the importance of the layer or its width. To mitigate this, we can add a learnable parameter, associated with each individual layer, to the spectral normalization algorithm which can adjust itself while training to attain an optimal robust model. Also, we have used fully connected layers in our experiments in chapter 5 but Spectral normalization is well suited and tested for convolutional neural networks as well. We are limited by the use of LipSDP module or any other module to estimate Lipschitz bounds that work well only for dense networks with non-expansive operators. Obtaining tight and numerically tractable Lipschitz bounds for convolutional networks thus remains a main challenge. In [77], author proposes a relaxed approach to normalize weights of different layers which is not as restrictive as Spectral normalization. They do so for improving the performance of the neural networks. Such a relaxation of normalization can also be explored to have better control on the stability. Also, another means of attaining an accuracy-stability trade-off can be achieved by utilizing two objective functions in order to maintain accuracy (i.e. first achieve the best possible accuracy and keep the accuracy in the tolerance levels) and gradually changing the weights to achieve tighter Lipschitz bounds.

6.2.4 . Other architectures and application domains

Our work concentrated on using simple fully connected layers, understanding them, and controlling their stability. More prevalent architectures such as ResNet, EfficinetNet, graph networks, transformers, and generative models such as GANs need to be explored. Even in case of tabular data, though researchers may rely on methods such as XgBoost [107], deeper learning models, especially attention-based mechanisms have emerged, such as TabTransformer [108], FT-Transformer [109], TabNet transformer [110], and most recently Hopfield networks [111], successful

in learning such data.
Also, a point to be noticed for these architectures is the use of activation functions such as GELU in architectures, which are not firmly non-expansive, while the architecture in itself consists of simple fully connected layers. In addition, transformers using standard DP product attention mechanism were shown to be non Lipschitz continuous [112], again questioning the use of such networks in real-world applications. A Lipschitz continuous attention mechanism was proposed, but the robustness of this block is still unknown. Another area which needs to be studied is the use of batch and layer normalization layers and how they contribute to the overall robustness of the neural networks. Similarly, [113] proposes Lipschitz normalization for improving the performance of attention based graph neural networks without discussion of their adversarial and certified robustness. These architectures must be studied for robustness and vulnerabilities to deliver more trustworthy models. Overall, there have been plethora of works designing and defending such networks against adversarial attacks, but handful that certify theoretical guarantees against perturbations.

6.2.5 . Other notions of robustness

A very close notion to Lipschitz bounds is related to curvature, which is a mathematical quantity that encodes the deviation of a curve from 'flatness'. The high degree of non-linearity present in deep neural networks, or equivalently their large curvature, is critical to achieving good performance in difficult tasks. Networks learnt by penalizing the curvature keep the trained model as linear as possible. This implies that networks are learnt to obtain certain upper bounds on the curvature similar to Lipschitz bounds. When a function is twice differentiable, the maximum of the norm of the Hessian is the local Lipschitz constant of the Jacobian of the network. In deep learning, the curvature of a function at a point is quantified as the norm of the Hessian at that point. So, it is equivalent to look at the Lipschitz properties of the Jacobian. Low curvature bounds is yet another way to verify that the linearity is indeed the source of increased robustness. In literature, a low curvature is imposed to induce robustness in the networks, and this is done by penalizing the norms of the Hessian. Moosavi et al. [114] penalizes the Frobenius norm of the Hessian, and [115] introduces a local linearity regularizer, which also implicitly penalizes the Hessian. Relationships and analysis can be explored between Lipschitz bounds and curvature since the two concepts are complementary and closely related. It would thus be interesting to explore training strategies utilizing curvature as the objective.

Appendix A
Lipschitz Constant Analysis of Tabular Data

This appendix describes one of our preliminary studies related to Lipschitz analysis on tabular data published in [21].

Deep learning on tabular data has received much less attention than deep learning for standard signal/image processing problems particularly seen in the area of computer vision and natural language processing. Tabular data allows to take advantage of heterogeneous sources of information coming from different sensors or registered variables (like altitude of an aircraft, departure and destination airport, duration of the flight, company type). Tabular data analysis covers a wide variety of applications, e.g. fraud detection,product failure prediction, anti-money laundering, recommendation systems [116], click-through rate prediction [117] etc. A generalised NN framework for tabular data is presented in [87]. Most of the mentioned tasks may be hampered with safety concerns and require reliability in the performance of the NN used for predicting or classifying data. In [80], authors presented a method for generation of imperceptible adversarial attacks for tabular data.

In our stability analysis, we study three widely used tabular datasets from UCI Repository: 1) Combined Cycle Power Plant Data Set has 4 attributes with 9568 instances , 2) Auto MPG Data Set consists of 398 instances with 7 attributes, 3) Adult Data Set consists of 48842 instances in total with 14 attributes. Dataset 2 and 3 include both continuous and categorical attributes, whereas dataset 1 contains only continuous attributes. The datasets are divided with a ratio of 4:1 between training and testing data. The categorical attributes are dealt with by using one hot encoding based on the number of categories. The input attributes are normalised by removing their mean and scaling to unit variance. All the architectures are made up of two hidden fully connected layers with the following characteristics:

- Combined Cycle Power Plant Data set - $(4, 10, 6, 1)$
- Auto MPG Data set - $(9, 16, 8, 1)$
- Adult Data set - $(88, 6, 6, 1)$

			ℓ_1					ℓ_2				
	α	0	0.00001	0.0001	0.001	0.01	0.1	0.00001	0.0001	0.001	0.01	0.1
	MAE	0.0069	0.0071	0.0069	0.0094	0.0300	0.0300	0.0073	0.007	0.0077	0.0299	0.030
$L_{2,2}$	LipSDP [16]	0.657	0.705	0.26	0.024	≈ 0	≈ 0	1.01	0.741	0.027	≈ 0	≈ 0
	CPLip[9]	0.638	0.681	0.25	0.024	5.33e-11	3.99e-11	0.96	0.73	0.027	1.26e-09	1.09e-17
$L_{+\infty,+\infty}$	Lipopt-k [43]	1.41	1.39	0.47	0.028	≈ 0	≈ 0	1.89	1.26	0.040	1.07e-09	≈ 0
	CPLip[9]	1.23	1.17	0.46	0.028	9.16e-11	6.75e-11	1.486	1.26	0.040	2.35e-09	1.95e-17

Table A.1: Results on Combined Cycle Power Plant Data Set for ℓ_1 and ℓ_2 regularization

			ℓ_1						ℓ_2					
	α	0	0.00001	0.0001	0.001	0.01	0.1	0.2	0.00001	0.0001	0.001	0.01	0.1	0.2
	MAE	0.0418	0.0444	0.0405	0.0443	0.0505	0.1490	0.1490	0.0515	0.0404	0.0424	0.0454	0.1489	0.1489
$L_{2,2}$	LipSDP [16]	2.75	1.74	0.38	0.16	0.11	≈ 0	≈ 0	2.13	0.78	0.201	0.089	≈ 0	≈ 0
	CPLip[9]	2.747	1.705	0.373	0.16	0.110	8.75e-09	7.44e-09	2.11	0.721	0.201	0.089	9.96e-09	9.72e-09
$L_{+\infty,+\infty}$	Lipopt-k [43]	9.47	5.66	1.04	0.286	0.1642	1.66e-08	1.83e-08	7.048	3.088	0.4365	0.2099	1.41e-08	1.45e-08
	CPLip[9]	6.98	4.36	1.03	0.29	0.16	2.30e-08	2.06e-08	4.97	1.93	0.44	0.21	2.9e-08	2.8e-08

Table A.2: Results on Auto MPG Data Set for ℓ_1 and ℓ_2 regularization

			ℓ_1						ℓ_2					
	α	0	0.00001	0.0001	0.001	0.01	0.1	0.2	0.00001	0.0001	0.001	0.01	0.1	0.2
	Acc	84.94	85.16	85.57	85.54	76.30	76.30	76.30	85.32	85.26	85.46	84.73	76.32	76.32
$L_{2,2}$	LipSDP[16]	8.21	6.29	5.15	3.45	≈ 0	≈ 0	≈ 0	9.15	5.32	4.19	1.91	≈ 0	≈ 0
	CPLip[9]	8.208	6.29	5.15	3.45	5.81e-10	1.84e-10	3.02e-10	9.15	5.32	4.19	1.91	1.839e-10	6.04e-11
$L_{+\infty,+\infty}$	Lipopt-k [43]	56.22	31.77	20.53	12.36	≈ 0	≈ 0	≈ 0	42.74	22.48	17.90	9.06	2.43e-11	≈ 0
	CPLip[9]	56.22	31.77	20.53	12.36	4.33e-09	1.34e-09	2.21e-09	42.74	22.48	17.90	9.06	9.25e-10	3.85e-10

Table A.3: Results on Adult Data Set for ℓ_1 and ℓ_2 regularization

A.1 . Effect of Regularization Techniques

We present the results for three regularisation techniques: ℓ_1, ℓ_2, and Dropout [118]. We study the relationship between the parameters associated with each regularisation and the NN stability quantified by its Lipschitz constant. Attention also to be paid to the resulting accuracy. We apply these regularization techniques while training our NNs, then compute a Lipschitz constant associated with the obtained weights. We use the three state-of-the-art estimation methods which have been described in Section 2.2.1.2. The first one is LipSDP [16] which uses Euclidean norms for both the input and output spaces ($L_{2,2}$ spectral norm). The second one is the polynomial based approach LipOpt [44] where the input and output spaces are equipped with the sup norm while estimating the Lipschitz constant. This estimation approach is thus linked to the $L_{+\infty,+\infty}$ subordinate matrix norm. The third estimation method is CPLip [9] which can work for any norm on the input and output spaces.

Each experiment was run 10 times and we chose the model with best performance (least MAE or highest classification accuracy) and computed a Lipschitz constant for this model. The results for the datasets using ℓ_1 and ℓ_2 regularization are reported in Tables A.1, A.2, and A.3 for varying values of the regularization parameter α which controls the strength of the ℓ_1 and ℓ_2 penalty on the weights. Similar results with varying drop-rates are presented in Tables A.4, A.5, and A.6. Droprate corresponds to the proportion of the neurons which will be shut-off while training a neural network.

	Drop-rate	0	0.05	0.1	0.2	0.3	0.4	0.5
	MAE	0.0069	**0.0069**	0.0074	0.0086	0.0115	0.0111	0.016
$L_{2,2}$	LipSDP [16]	0.66	**0.16**	0.23	0.56	0.45	0.27	0.40
	CPLip[9]	0.64	**0.16**	0.23	0.54	0.42	0.27	0.39
$L_{+\infty,+\infty}$	Lipopt-k [43]	1.41	**0.26**	0.32	1.01	0.79	0.42	0.69
	CPLip[9]	1.23	**0.26**	0.32	1.01	0.69	0.42	0.69

Table A.4: Results on Combined Cycle Power Plant Data set with Dropout

	Drop-rate	0	0.05	0.1	0.2	0.3	0.4	0.5
	MAE	0.042	0.039	0.0364	0.043	0.044	0.045	0.050
$L_{2,2}$	LipSDP [16]	2.75	1.86	2.1	1.89	2.28	1.88	1.41
	CPLip[9]	2.75	1.73	2.1	1.87	2.28	1.88	1.42
$L_{+\infty,+\infty}$	Lipopt-k [43]	9.47	6.41	5.49	4.89	5.2	3.98	3.07
	CPLip [9]	6.98	4.45	4.631	4.89	5.19	3.98	3.07

Table A.5: Results on Auto MPG Data set with Dropout

	Drop-rate	0	0.05	0.1	0.2	0.3	0.4	0.5
	Acc	84.94	85.13	85.14	85.08	85.03	85.09	84.81
$L_{2,2}$	LipSDP [16]	8.21	7.17	7.39	6.82	6.53	6.68	6.63
	CPLip[9]	8.21	7.17	7.39	6.82	6.53	6.68	6.63
$L_{+\infty,+\infty}$	Lipopt-k [43]	56.22	46.07	49.32	43.58	40.21	39.87	41.26
	CPLip[9]	56.22	46.07	49.32	43.58	40.21	39.87	41.26

Table A.6: Results on Adult Data Set with Dropout

A.2 . Positive Weighted Networks

Next we analyse the stability of NNs when the weights are constrained to be non-negative. The comparison between arbitrary signed network and positively signed network for all the three datasets is shown in Table A.7.

A.3 . Addition of Noise to the Dataset

We also perform a stability analysis when the original dataset is corrupted with random noise which is a standard practice while training NNs having continuous input variables. From the original dataset, we generated a dataset 20 times larger by including noisy samples.

More precisely, to all the normalised input features, we added a noise value drawn from a random i.i.d. zero-mean Gaussian distribution with a small standard deviation. We combined the original training set with the generated noisy samples and trained our model on the augmented dataset. The results on Combined Cycle Power plant with variation of standard deviation are given in Table A.8.

		Dataset 1		Dataset 2		Dataset 3	
		Arbitrary	Positive	Arbitrary	Positive	Arbitrary	Positive
	MAE/ACC	0.0069	0.021	0.042	0.08	84.94	83.54
$L_{2,2}$	LipSDP [16]	0.66	0.03	2.75	0.57	8.21	3.65
	CPLip [9]	0.64	0.03	2.75	0.57	8.21	3.65
$L_{+\infty,+\infty}$	Lipopt-k [43]	1.41	0.06	9.47	3.29	56.22	18.81
	CPLip[9]	1.23	0.06	6.98	1.24	56.22	18.81

Table A.7: Results with positive constraint on the weights

	std	No Noise	0.01	0.05	0.1	0.2
	MAE	0.0069	0.0064	0.0061	0.0065	0.0071
$L_{2,2}$	LipSDP [16]	0.66	0.61	0.34	0.37	0.10
	CPLip[9]	0.64	0.61	0.33	0.36	0.09
$L_{+\infty,+\infty}$	Lipopt-k [43]	1.411	1.12	0.58	0.61	0.24
	CPLip[9]	1.234	1.12	0.57	0.59	0.16

Table A.8: Results on Combined Cycle Power Plant Data set with added noise

A.4 . Comments on the results

A first observation is that CPLip provides slightly tighter bounds than LipSDP and LipOpt. The two latter approaches may however be more scalable when applied to deeper networks. Another remark is that similar behaviours can be seen when using different norms. While designing a network for deployment, it also appears that there is a trade-off between stability and performance.

- ℓ_1 and ℓ_2 regularization increase the stability of the network consistently, but the performance is maintained up to a certain value of α, from where the accuracy drops. ℓ_1 usually yields better results, confirming the robustness of this norm as training measure.
- We globally observe an increasing stability of the neural networks as we increase the value of droprate. The results are however less consistent than with regularization.
- The positive constraint leads to a significant loss of performance in terms of MAE and classification accuracy, but the stability of the networks is improved by a significant margin.
- As expected, adding noise leads to a drop in the value of the Lipschitz constant as the noise level increases, without having a negative impact on the accuracy.

Appendix B

Robustness of a New Type of Neural Networks using Positive Weights

This appendix presents the work done in collaboration with another PhD student at CVN. The work is published in [22].

B.1 . Making the bridge between CNNs and Fully Connected Networks (FCN)

We use the fact that, if all the weights of a feed-forward neural network are non-negative, an optimal Lipschitz constant reduces to $\|W_m \cdots W_1\|_S$ [9]. In other words, it has the same Lipschitz constant as a linear network where the identity function is substituted for all the activation operators. For a network with weights having arbitrary signs, $\vartheta_m = \|W_m \cdots W_1\|_S$ only constitutes a lower bound on the Lipschitz constant [9].

In this work, we aim at filling the gap between FCNs and CNNs. In signal processing context, a convolutive layer is a Multiple-Input Multiple-Output (MIMO) filter. For one-dimensional signals, each of these filters can be viewed as a Tœplitz matrix generated by the impulse response of the filter, which is applied to the vector of signal samples. If the filter length is short, large upper and lower triangular parts of this matrix are null. In our proposed approach, we will keep this band structure for the weight matrix, which is equivalent to performing local processing at each time within a sliding window. However, in order to add more flexibility in this architecture, we will allow all the nonzero coefficients of this matrix to be fully optimized.

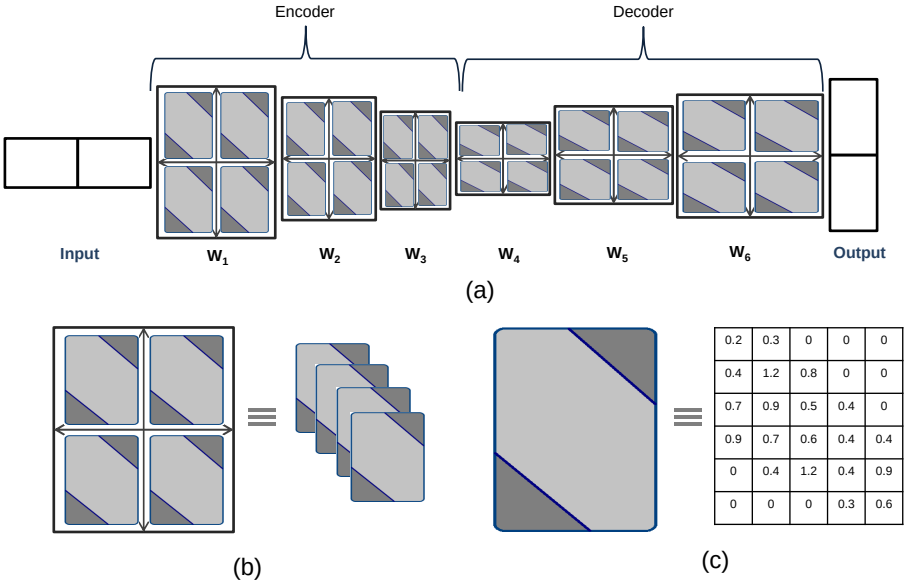

Figure B.1: Proposed architecture of Adaptive Convolutional Neural Network (ACNN). a) An encoder-decoder architecture composed of a 6-layer FCN followed by ReLU activation function. b) Relation between proposed FCNs and CNNs; the weights are split into sub-matrices simulating convolutive filters in CNNs c) Each of the sub-matrices is constrained to have a band structure as shown in this example. The dark gray area marks the zero-entries, while the light-gray colour corresponds to the ones that are allowed to be non-zero.

The proposed architecture is depicted in Figure B.1. Figure B.1.b is the graphical representation of the presented concept. As it can be observed, the weights are split to emulate a MIMO system. Each kernel is associated with a specific shape of the matrix, which is depicted in Figure 1.c. The lower and upper triangular null parts are displayed in dark gray, while the light gray central part contains overlap and may use different weight values.

B.1.1 . Learning algorithm

For training the proposed ACNN (Adaptive Convolutional Neural Network), we use a stochastic gradient-like optimization based on Adam. Consider the vector of parameters of the network, $\eta = (\eta_i)_{1 \leqslant i \leqslant m}$, such that, for each layer $i \in \{1, \ldots, m\}$, η_i represents a vector of dimension $N_i(N_{i-1} + 1)$, composed of the elements of weight matrix W_i and the components of the bias vector b_i.

To secure the conditions of robustness while imposing the desired structure for our network, the parameter vector η is projected onto a closed set $\mathcal{S}$ that expresses all these constraints. The parameter update at epoch $n > 0$ is performed for mini-batches $(\mathbb{M}_{q,n})_{1 \leqslant q \leqslant Q}$. If the training data are denoted by $(z_k)_{1 \leqslant k \leqslant K}$, where z_k is the k-th pair of inputs and their associated outputs, the operations performed

during the n-th epoch reads:

$$\text{Partition } \{1,\dots,K\} \text{ into mini-batches } (\mathbb{M}_{q,n})_{1\leqslant q\leqslant Q} \tag{B.1}$$

$$\text{For every } q \in \{1,\dots,Q\} \tag{B.2}$$

$$\left\lfloor \begin{array}{l} t = (n-1)Q + q \\ \text{For every } i \in \{1,\dots,m\} \\ \left\lfloor \begin{array}{l} g_{i,t} = \sum_{k\in\mathbb{M}_{q,n}} \nabla_i \ell\big(z_k, (\eta_{i,t})_{1\leqslant i\leqslant m}\big) \\ \mu_{i,t} = \beta_1 \mu_{i,t-1} + (1-\beta_1) g_{i,t} \\ \nu_{i,t} = \beta_2 \nu_{i,t-1} + (1-\beta_2) g_{i,t}^2 \\ \gamma_t = \gamma \sqrt{1-\beta_2^t}/(1-\beta_1^t) \\ \eta_{i,t+1} = \mathsf{P}_{\mathcal{S}_{i,t}}\Big(\eta_{i,t} - \gamma_t \mu_{i,t}/(\sqrt{\nu_{i,t}} + \epsilon)\Big), \end{array} \end{array} \right. \tag{B.3}$$

where the square, the square root, and the division are performed component-wise, and

$$\mathcal{S}_{i,t} = \big\{\eta_i \mid [(\eta_{j,t+1}^\top)_{j<i} \;\; \eta_i^\top \;\; (\eta_{j,t}^\top)_{j>i}]^\top \in \mathcal{S}\big\}. \tag{B.4}$$

Here-above, ℓ denotes the loss function, ∇_i represents the gradients with respect to η_i; $\mu_{i,t}$ and $\nu_{i,t}$ represent the first and second momentum estimates at the iteration t, initialized with $\mu_{i,0} = \nu_{i,0} = 0$. $\mathsf{P}_{\mathcal{S}_{i,t}}$ designates the projection onto the constraint set $\mathcal{S}_{i,t}$. Although the set $\mathcal{S}$ is non-convex, the sets $\mathcal{S}_{i,t}$ will be defined as the intersection of three closed and convex constraint sets, as detailed next. The parameters used for learning are set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\gamma = 0.001$, and $\epsilon = 10^{-12}$.

To ensure the computation of a tight robustness bound we impose non-negative weights for every $i \in \{1,\dots,m\}$ by considering the constraint set:

$$\mathcal{D}_i = \{W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid W_i \geqslant 0\} \tag{B.5}$$

Let R_i (resp. Q_i) be the number of output (resp. input) channels used in layer $i \in \{1,\dots,m\}$. The proposed algebraic structure of each weight operator is guaranteed by splitting the corresponding matrix W_i into $R_i \times Q_i$ sub-matrices of dimension $N_i' \times M_i'$ (with $N_i' = N_i/R_i$ and $M_i' = N_{i-1}/Q_i$), denoted by $(W_i^{(r,q)})_{1\leqslant r\leqslant R_i, 1\leqslant q\leqslant Q_i}$. The desired band structure of the sub-matrix $W_i^{(r,q)}$ is ensured by imposing that it belongs to the following vector space:

$$\begin{aligned} \mathcal{E}_i = \{(V_{u,v})_{1\leqslant u\leqslant N_i', 1\leqslant v\leqslant M_i'} \in \mathbb{R}^{N_i'\times M_i'} \mid \forall (u,v) \text{ s.t.} \\ |(M_i'-1)(u-1) - (N_i'-1)(v-1)| \geqslant d_i, V_{u,v} = 0\}. \end{aligned}$$

Herein, d_i is an integer and, if $N_i' = M_i' > 1$, $2\lfloor d_i/(N_i'-1)\rfloor - 1$ plays a role similar to a kernel length in standard CNNs.

Finally, to control the robustness, we need to limit the Lipschitz constant of

				PSNR	MSE	CC
Noisy Signal				18.25	1.18×10^{-2}	0.76
Denoised Signal	Baseline - Wavelet-based denoiser			20.66	$1.00 \times 10{-}3$	0.80
	ACNN denoiser	Scenario (i)	$\overline{\vartheta} = 1$	24.27	3.73×10^{-3}	0.96
			$\overline{\vartheta} = 5$	29.03	1.25×10^{-3}	0.97
			$\overline{\vartheta} = 10$	33.76	6.53×10^{-4}	0.98
		Scenario (ii)	$\overline{\vartheta} = 1$	25.87	3.12×10^{-3}	0.96
			$\overline{\vartheta} = 5$	30.63	8.63×10^{-4}	0.98
			$\overline{\vartheta} = 10$	36.02	2.23×10^{-4}	0.99
	Standard FCN denoiser		$\overline{\vartheta} = 1$	23.38	4.59×10^{-3}	0.90

Table B.1: Comparison of different variants of the proposed method with baselines.

the network to a given value $\overline{\vartheta} > 0$. The related constraint can be expressed as:

$$\mathcal{C}_{i,t} = \{W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid \|A_{i,t} W_i B_{i,t}\|_{\mathrm{S}} \leqslant \overline{\vartheta}\} \tag{B.6}$$

$$A_{i,t} = W_{m,t} \cdots W_{i+1,t}, \tag{B.7}$$

$$B_{i,t} = W_{i-1,t+1} \cdots W_{1,t+1} \tag{B.8}$$

with the convention that for the first layer $(i = 1)$, $B_{i,t}$ is the $N_i \times N_i$ identity matrix Id_{N_i} and for the last layer $(i = m)$, $A_{i,m} = \mathrm{Id}_{N_i}$. Above, $(W_{j,t})_{1 \leqslant j \leqslant m}$ designates the estimates of the weight matrices at iteration t of the projected ADAM optimizer.

The projection onto the intersection of the above three closed convex sets has no closed-form expression. The intersection $\mathcal{D}_i \cap \mathcal{E}_i^{R_i \times Q_i}$ is however quite simple to handle since the projection onto this set reduces to $\mathrm{P}_{\mathcal{D}_i} \circ \mathrm{P}_{\mathcal{E}_i^{R_i \times Q_i}}$. To compute the final projection onto $\mathcal{C}_{i,t} \cap (\mathcal{D}_i \cap \mathcal{E}_i^{R_i \times Q_i})$ of a weight matrix W_i, we use the dual forward-backward algorithm, as presented in [119].

B.2 . Experimental Evaluation

The proposed network has been evaluated for denoising music signals.

B.2.1 . Dataset Description

We train our proposed ACNN on a dataset consisting of musical exercises and songs performed on a Ronald organ. The organ covers 5 octaves (range `C2–C7`), each octave having 12 semitones, generating a total of 61 different possible notes. For the recordings, the whole range of notes is used. The songs are recorded in MIDI format using MidiEditor, in the following manner: the recording mode from MidiEditor is activated before each song being played and is stopped after the song is finished (so there is silence at the beginning and end of each recording). In total, the dataset contains 100 MIDI recordings, with a sampling frequency $F_s = 44100$ Hz, constituting 1 h and 17 min of audio. The data set is available online[1] .

[1] https://speed.pub.ro/downloads/

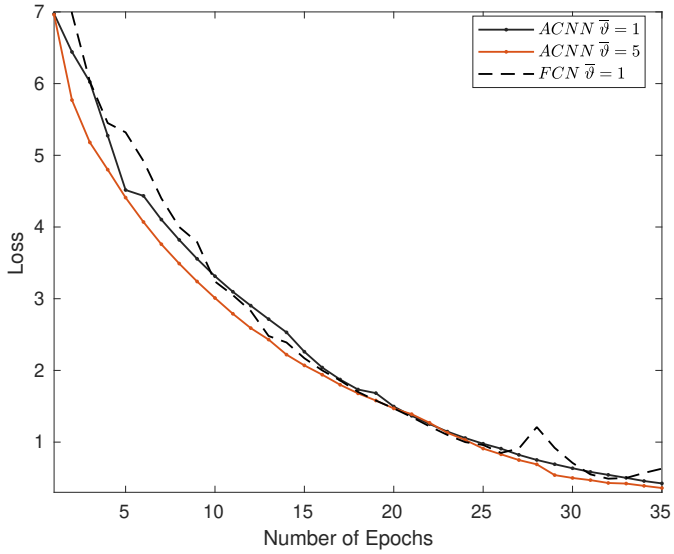

Figure B.2: Convergence profile of the proposed method.

The dataset is divided into training, validation, and test sets. The training set contains 90 clips with variable length, ranging from 6 s up to 150 s, having in total over an hour (67 min) of audio recordings. Some songs are repeated on different octaves to obtain a minimum number of occurrences for all notes. The validation dataset contains 9 songs with length between 12 and 120 s, forming around 10 minutes of audio signals. The test set has one 2-minute long clip.

B.2.2 . Experimental setup

The noisy data for training, validating, and testing is generated by adding white Gaussian noise to the original samples. The noise has zero mean and its standard deviation is randomly chosen so that the resulting signal-to-noise ratio (SNR) varies between 5 and 30dB. The dataset samples are normalized between 0 and 1. We extract the frequency features from the audio signal using a STFT. The network estimates the STFT coefficients of the samples and an ISTFT is performed as the post-processing step. We consider a `Hanning` sliding analysis window of length $T = 23$ ms, with an overlap between two consecutive windows of 50%. The STFT is performed on 1024 points. In total, from each audio segment, a vector of length $L = 513$ frequency coefficients is obtained, constituting the input of our ACNN.

The denoising is performed using a 6-layer ACNN architecture, as presented in Figure B.1. The network has an encoder-decoder structure. Each layer employs *ReLU* as activation function followed by a *batch normalization* step that acts as a regularizer and prevents the model from overfitting.

B.2.3 . Simulations and results

In order to measure the performance of our proposed ACNN architecture, we perform two sets of experiments. In the first set, we control the Lipschitz constant

of the architecture for three values $\overline{\vartheta}$ equal to 1, 5, and 10. In the second experiment, we test our architecture by varying the number of channels, i.e. the way we split each weight matrix. Note that for both scenarios we consider a network with $m = 6$ layers and $\forall i \in \{1, \ldots, 6\}$, $d_i = d'_i(\max\{N'_i, M'_i\} - 1)$, having the following characteristics:

(i) $R_1 = 2 = Q_6$, $Q_1 = 1 = R_6$, $\forall i \in \{2, \ldots, 5\}$ $R_i = Q_i = 2$, $(N_i)_{1 \leqslant i \leqslant 6} = (400, 200, 100, 200, 400, 513)$, $(d'_i)_{1 \leqslant i \leqslant 6} = (237, 80, 30, 30, 80, 237)$;

(ii) $R_1 = 3 = Q_6$, $Q_1 = 1 = R_6$, $\forall i \in \{2, \ldots, 5\}$ $R_i = 3$, $Q_i = 3$, $(N_i)_{1 \leqslant i \leqslant 6} = (630, 510, 420, 510, 630, 513)$, $(d'_i)_{1 \leqslant i \leqslant 6} = (237, 85, 65, 65, 85, 237)$.

We evaluate the performance on 3 standard metrics: *Peak to Signal Noise Ratio (PSNR)*, *Mean squared error (MSE)*, and *Cross-correlation (CC)*, as shown in the Table B.1. We compare our method with a standard denoising technique, based on a *Wavelet* decomposition. We employ a 5-level decomposition using $Symlet8$ filters combined with *SureShrink* thresholding. We also compare ACNN with a FCN implementation, for which we ensured the Lipschitz bound $\overline{\vartheta} = 1$. Here, the FCN has no structural constraints but, in addition to the imposed Lipschitz property, it has positive weights.

Table B.1 reports the quantitative results obtained by all three approaches, evaluated over the test set. Our method outperforms the baseline on all measures by a significant margin. ACNN also outperforms classical FCN, inferring that structure imposed on the weight matrix for ACNN leads to a model with better generalization power, whereas FCN implementation may be prone to overfitting. We observed that, without any Lipschitz constraint, FCNs tend to have a very high Lipschitz constant of the order $10^5 - 10^6$, which emphasizes the importance of controlling the Lipschitz behaviour of the system. The Lipschitz constant of the network is also closely related to the expressiveness of the trained model. Architectures with tight robustness constraint have fewer degrees of freedom and are thus expected to be less accurate, and lead to a slower convergence. The convergence profile of our proposed method is shown in Figure B.2.

Bibliography

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv:1412.6572, 2014. (Cited on 13, 18, 26, 28, 39, 41, 46, 58, 76, 96)

[2] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," in International Conference on Learning Representations, 2018. (Cited on 13, 19, 36, 40)

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in International Conference on Learning Representations, 2014. (Cited on 17, 18, 39, 42)

[4] S. Thiebes, S. Lins, and A. Sunyaev, "Trustworthy artificial intelligence," Electronic Markets, vol. 31, pp. 447–464, 2021. (Cited on 17)

[5] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," ACM Computing Surveys, vol. 54, pp. 1–35, 2021. (Cited on 17)

[6] R. Singh, M. Vatsa, and N. Ratha, "Trustworthy AI," in ACM IKDD CODS and COMAD, pp. 449–453. 2021. (Cited on 17)

[7] L. Cheng, K. R. Varshney, and H. Liu, "Socially responsible AI algorithms: Issues, purposes, and challenges," Journal of Artificial Intelligence Research, vol. 71, pp. 1137–1181, 2021. (Cited on 17)

[8] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," Advances in Neural Information Processing Systems, vol. 31, 2018. (Cited on 18, 28, 35, 37)

[9] P. L. Combettes and J.-C. Pesquet, "Lipschitz certificates for neural network structures driven by averaged activation operators," SIAM Journal on Mathematics of Data Science, vol. 2, pp. 529–557, 2020. (Cited on 18, 19, 29, 30, 31, 58, 106, 107, 108, 109)

[10] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in Artificial intelligence safety and security, pp. 99–112. Chapman and Hall/CRC, 2018. (Cited on 19, 39, 46)

[11] F. Farnia, J. M. Zhang, and D. Tse, "Generalizable adversarial training via spectral normalization," arXiv:1811.07457, 2018. (Cited on 19, 43)

[12] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in International Conference on Machine Learning, 2018, pp. 5286–5295. (Cited on 19, 26)

[13] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in International Conference on Learning Representations, 2018. (Cited on 19, 26)

[14] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," Advances in neural information processing systems, vol. 31, 2018. (Cited on 19, 26, 36)

[15] K. Leino, Z. Wang, and M. Fredrikson, "Globally-robust neural networks," in International Conference on Machine Learning, 2021, pp. 6212–6222. (Cited on 19, 42)

[16] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of Lipschitz constants for deep neural networks," Advances in Neural Information Processing Systems, vol. 32, 2019. (Cited on 19, 31, 33, 34, 58, 73, 86, 92, 95, 106, 107, 108)

[17] S. Verma and K. Gupta, "Robustness of Neural Networks used in Electrical Motor Time-Series," in Workshop on Robustness in Sequence Modeling, Advances in Neural Information Processing Systems, 2022. (Cited on 22)

[18] K. Gupta, F. Kaakai, B. Pesquet-Popescu, and J.-C. Pesquet, "Safe design of stable neural networks for fault detection in small UAVs," in International Conference on Computer Safety, Reliability, and Security. Springer, 2022, pp. 263–275. (Cited on 22)

[19] K. Gupta, F. Kaakai, B. Pesquet-Popescu, J.-C. Pesquet, and F. D. Malliaros, "Multivariate Lipschitz analysis of the stability of neural networks," Frontiers in Signal Processing, 2022. (Cited on 22)

[20] K. Gupta, J.-C. Pesquet, B. Pesquet-Popescu, F. Malliaros, and F. Kaakai, "An adversarial attacker for neural networks in regression problems," in IJCAI Workshop on Artificial Intelligence Safety, 2021. (Cited on 22, 76)

[21] K. Gupta, B. Pesquet-Popescu, F. Kaakai, and J.-C. Pesquet, "A quantitative analysis of the robustness of neural networks for tabular data," in IEEE International Conference on Acoustics, Speech and Signal Processing, 2021, pp. 8057–8061. (Cited on 23, 105)

[22] A. Neacsu, K. Gupta, J.-C. Pesquet, and C. Burileanu, "Signal denoising using a new class of robust neural networks," in European Signal Processing Conference, 2021, pp. 1492–1496. (Cited on 23, 109)

[23] L. Pulina and A. Tacchella, "Challenging SMT solvers to verify neural networks," AI Communications, vol. 25, pp. 117–135, 2012. (Cited on 25)

[24] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," Advances in Neural Information Processing Systems, vol. 29, 2016. (Cited on 25)

[25] L. Pulina and A. Tacchella, "An abstraction-refinement approach to verification of artificial neural networks," in International Conference on Computer Aided Verification. Springer, 2010, pp. 243–257. (Cited on 25)

[26] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in International Conference on Computer Aided Verification. Springer, 2017, pp. 97–117. (Cited on 25)

[27] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," in International Conference on Learning Representations, 2018. (Cited on 25)

[28] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for relu networks," in International Conference on Machine Learning, 2018, pp. 5276–5285. (Cited on 26, 36)

[29] M. Mirman, T. Gehr, and M. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in International Conference on Machine Learning, 2018, pp. 3578–3586. (Cited on 26)

[30] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," arXiv:1810.12715, 2018. (Cited on 26)

[31] H. Zhang, H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D. Boning, and C.-J. Hsieh, "Towards stable and efficient training of verifiably robust neural networks," arXiv:1906.06316, 2019. (Cited on 26, 36)

[32] Z. Shi, Y. Wang, H. Zhang, J. Yi, and C.-J. Hsieh, "Fast certified robust training with short warmup," Advances in Neural Information Processing Systems, vol. 34, 2021. (Cited on 26)

[33] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in International Conference on Machine Learning, 2019, pp. 1310–1320. (Cited on 26)

[34] B. Li, C. Chen, W. Wang, and L. Carin, "Certified adversarial robustness with additive noise," Advances in Neural Information Processing Systems, vol. 32, 2019. (Cited on 26)

[35] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in IEEE Symposium on Security and Privacy, 2019, pp. 656–672. (Cited on 26)

[36] H. Salman, J. Li, I. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang, "Provably robust deep learning via adversarially trained smoothed classifiers," Advances in Neural Information Processing Systems, vol. 32, 2019. (Cited on 26)

[37] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "AI2: Safety and robustness certification of neural networks with abstract interpretation," in IEEE Symposium on Security and Privacy, 2018, pp. 3–18. (Cited on 28)

[38] Y.-Y. Yang, C. Rashtchian, H. Zhang, R. Salakhutdinov, and K. Chaudhuri, "Adversarial robustness through local Lipschitzness," arXiv:2003.02460, 2020. (Cited on 28)

[39] P. L. Combettes and J.-C. Pesquet, "Deep neural network structures solving variational inequalities," Set-Valued and Variational Analysis, vol. 28, pp. 1–28, 2020. (Cited on 29, 31)

[40] A. D. Lewis, "A top nine list: Most popular induced matrix norms," 2020. (Cited on 31)

[41] P. L. Combettes and J.-C. Pesquet, "Proximal thresholding algorithm for minimization over orthonormal bases," SIAM Journal on Optimization, vol. 18, pp. 1351–1376, 2008. (Cited on 31)

[42] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using Lipschitz bounds," IEEE Control Systems Letters, vol. 6, pp. 121–126, 2021. (Cited on 34)

[43] F. Latorre, P. Rolland, and V. Cevher, "Lipschitz constant estimation of neural networks via sparse polynomial optimization," in International Conference on Learning Representations, 2019. (Cited on 34, 35, 58, 106, 107, 108)

[44] T. Chen, J.-B. Lasserre, V. Magron, and E. Pauwels, "Polynomial optimization for bounding Lipschitz constants of deep networks," arXiv:2002.03657, 2020. (Cited on 35, 106)

[45] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," Advances in Neural Information Processing Systems, vol. 31, 2018. (Cited on 35)

[46] S. Lee, J. Lee, and S. Park, "Lipschitz-certifiable training with a tight outer bound," Advances in Neural Information Processing Systems, vol. 33, 2020. (Cited on 36, 42)

[47] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," IEEE Transactions on Automatic Control, 2020. (Cited on 36)

[48] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," Advances in Neural Information Processing Systems, vol. 30, 2017. (Cited on 36)

[49] H. Zhang, P. Zhang, and C.-J. Hsieh, "Recurjac: An efficient recursive algorithm for bounding jacobian matrix of neural networks and its applications," in AAAI Conference on Artificial Intelligence, 2019, vol. 33, pp. 5757–5764. (Cited on 36)

[50] M. Jordan and A. G. Dimakis, "Exactly computing the local Lipschitz constant of relu networks," Advances in Neural Information Processing Systems, vol. 33, 2020. (Cited on 36)

[51] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 9185–9193. (Cited on 39)

[52] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in IEEE European Symposium on Security and Privacy, 2016, pp. 372–387. (Cited on 40, 41)

[53] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582. (Cited on 40, 76)

[54] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2017, pp. 1765–1773. (Cited on 40)

[55] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in IEEE Symposium on Security and Privacy, 2017, pp. 39–57. (Cited on 40, 42)

[56] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," arXiv:1611.02770, 2016. (Cited on 40)

[57] J. Rony, E. Granger, M. Pedersoli, and I. B. Ayed, "Augmented Lagrangian adversarial attacks," in IEEE/CVF International Conference on Computer Vision, 2021, pp. 7718–7727. (Cited on 40, 101)

[58] E. R. Balda, A. Behboodi, and R. Mathar, "Perturbation analysis of learning algorithms: A unifying perspective on generation of adversarial examples," arXiv:1812.07385, 2018. (Cited on 40)

[59] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," IEEE Transactions on Evolutionary Computation, vol. 23, pp. 828–841, 2019. (Cited on 40)

[60] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 1625–1634. (Cited on 40, 76)

[61] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in International Conference on Machine Learning, 2018, pp. 284–293. (Cited on 40)

[62] L. Tong, S. Yu, S. Alfeld, et al., "Adversarial regression with multiple learners," in International Conference on Machine Learning, 2018, pp. 4946–4954. (Cited on 41)

[63] A. Ghafouri, Y. Vorobeychik, and X. Koutsoukos, "Adversarial regression for detecting attacks in cyber-physical systems," in International Joint Conference on Artificial Intelligence, 2018, pp. 3769–3775. (Cited on 41)

[64] Y. Deng, X. Zheng, T. Zhang, C. Chen, G. Lou, and M. Kim, "An analysis of adversarial attacks and defenses on autonomous driving models," in IEEE International Conference on Pervasive Computing and Communications, 2020, pp. 1–10. (Cited on 41)

[65] A. T. Nguyen and E. Raff, "Adversarial attacks, regression, and numerical stability regularization," arXiv:1812.02885, 2018. (Cited on 41)

[66] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in ACM on Asia conference on computer and communications security, 2017, pp. 506–519. (Cited on 41)

[67] F. Tramèr, D. Boneh, A. Kurakin, I. Goodfellow, N. Papernot, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in International Conference on Learning Representations, 2018. (Cited on 41)

[68] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in International Conference on Machine Learning, 2018, pp. 274–283. (Cited on 42)

[69] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in International Conference on Machine Learning, 2017, pp. 854–863. (Cited on 42)

[70] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," arXiv:1802.05957, 2018. (Cited on 42, 43, 86)

[71] M. Serrurier, F. Mamalet, A. González-Sanz, T. Boissin, J.-M. Loubes, and E. del Barrio, "Achieving robustness in classification using optimal transport with hinge regularization," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 505–514. (Cited on 42, 43)

[72] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," Advances in neural information processing systems, vol. 31, 2018. (Cited on 42)

[73] Y. Huang, H. Zhang, Y. Shi, J. Z. Kolter, and A. Anandkumar, "Training certifiably robust neural networks with efficient local Lipschitz bounds," Advances in Neural Information Processing Systems, vol. 34, 2021. (Cited on 43)

[74] L. Béthune, A. González-Sanz, F. Mamalet, and M. Serrurier, "The many faces of 1-Lipschitz neural networks," arXiv:2104.05097, 2021. (Cited on 43)

[75] W. Piat, J. Fadili, F. Jurie, and S. da Veiga, "Towards an evaluation of Lipschitz constant estimation algorithms by building models with a known Lipschitz constant," in Workshop on Trustworthy Artificial Intelligence ECML/PKDD, 2022. (Cited on 43)

[76] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," Advances in Neural Information Processing Systems, vol. 30, 2017. (Cited on 43)

[77] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing Lipschitz continuity," Machine Learning, vol. 110, pp. 393–416, 2021. (Cited on 43, 102)

[78] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in IEEE International Conference on Robotics and Automation, 2019, pp. 9784–9790. (Cited on 43)

[79] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural-fly enables rapid learning for agile flight in strong winds," Science Robotics, vol. 7, pp. eabm6597, 2022. (Cited on 43)

[80] V. Ballet, J. Aigrain, T. Laugel, P. Frossard, M. Detyniecki, et al., "Imperceptible adversarial attacks on tabular data," in NeurIPS Workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness and Privacy, 2019. (Cited on 46, 54, 76, 105)

[81] J. Bolte and E. Pauwels, "Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning," Mathematical Programming, vol. 188, pp. 19–51, 2021. (Cited on 47)

[82] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," International Journal of Electrical Power and Energy Systems, vol. 60, pp. 126 – 140, 2014. (Cited on 50)

[83] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," Decision support systems, vol. 47, pp. 547–553, 2009. (Cited on 50)

[84] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," Journal of Big Data, vol. 7, pp. 1–41, 2020. (Cited on 58)

[85] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in International Conference on Learning Representations, 2019. (Cited on 73)

[86] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in Artificial intelligence safety and security, pp. 99–112. 2018. (Cited on 76)

[87] G. Ke, J. Zhang, Z. Xu, J. Bian, and T.-Y. Liu, "TabNN: A universal neural network solution for tabular data," 2018. (Cited on 76, 105)

[88] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in European Conference on Information Retrieval. Springer, 2016, pp. 45–57. (Cited on 76)

[89] Z. Tu, J. Zhang, and D. Tao, "Theoretical analysis of adversarial learning: A minimax approach," Advances in Neural Information Processing Systems, vol. 32, 2019. (Cited on 76)

[90] Z. Pan and P. Mishra, "Fast approximate spectral normalization for robust deep neural networks," arXiv:2103.13815, 2021. (Cited on 86)

[91] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in International Conference on Machine Learning, 2019, pp. 7472–7482. (Cited on 86)

[92] J.-C. Laprie, J. Arlat, C. Beounes, and K. Kanoun, "Definition and analysis of hardware-and software-fault-tolerant architectures," Computer, vol. 23, pp. 39–51, 1990. (Cited on 90)

[93] P. Freeman, R. Pandita, N. Srivastava, and G. J. Balas, "Model-based and data-driven fault detection performance for a small UAV," IEEE/ASME Transactions on mechatronics, vol. 18, pp. 1300–1309, 2013. (Cited on 91)

[94] D. Guo, M. Zhong, H. Ji, Y. Liu, and R. Yang, "A hybrid feature model and deep learning based fault diagnosis for unmanned aerial vehicle sensors," Neurocomputing, vol. 319, pp. 155–163, 2018. (Cited on 91)

[95] X.-h. Xie, L. Xu, L. Zhou, and Y. Tan, "GRNN model for fault diagnosis of unmanned helicopter rotor's unbalance," in International Conference on Electrical Engineering And Control Applications. Springer, 2016, pp. 539–547. (Cited on 91)

[96] B. Eroglu, M. C. Sahin, and N. K. Ure, "Autolanding control system design with deep learning based fault estimation," Aerospace Science and Technology, vol. 102, pp. 105855, 2020. (Cited on 91)

[97] V. Sadhu, S. Zonouz, and D. Pompili, "On-board deep-learning-based unmanned aerial vehicle fault cause detection and identification," in IEEE International Conference on Robotics and Automation, 2020, pp. 5255–5261. (Cited on 91)

[98] G. Iannace, G. Ciaburro, and A. Trematerra, "Fault diagnosis for UAV blades using artificial neural network," Robotics, vol. 8, pp. 59, 2019. (Cited on 91)

[99] M. Bronz, E. Baskaya, D. Delahaye, and S. Puechmore, "Real-time fault detection on small fixed-wing UAVs using machine learning," in IEEE Digital Avionics Systems Conference, 2020, pp. 1–10. (Cited on 91)

[100] G. Hattenberger, M. Bronz, and M. Gorraz, "Using the Paparazzi UAV System for Scientific Research," in International Micro Air Vehicle Conference and Competition, 2014, pp. pp 247–252. (Cited on 91)

[101] M. D. Tandale and J. Valasek, "Fault-tolerant structured adaptive model inversion control," Journal of Guidance, Control, and Dynamics, vol. 29, pp. 635–642, 2006. (Cited on 91)

[102] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in International Conference on Learning Representations, 2018. (Cited on 96)

[103] J. Rony, J.-C. Pesquet, and I. B. Ayed, "Proximal splitting adversarial attacks for semantic segmentation," arXiv:2206.07179, 2022. (Cited on 101)

[104] Q. Huang, I. Katsman, H. He, Z. Gu, S. Belongie, and S.-N. Lim, "Enhancing adversarial example transferability with an intermediate level attack," in IEEE/CVF international conference on computer vision, 2019, pp. 4733–4742. (Cited on 101)

[105] Q. Li, Y. Guo, and H. Chen, "Yet another intermediate-level attack," in European Conference on Computer Vision. Springer, 2020, pp. 241–257. (Cited on 101)

[106] S. Ghanem, S. Roheda, and H. Krim, "Latent code-based fusion: A volterra neural network approach," arXiv preprint arXiv:2104.04829, 2021. (Cited on 102)

[107] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 785–794. (Cited on 102)

[108] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, "Tabtransformer: Tabular data modeling using contextual embeddings," arXiv preprint arXiv:2012.06678, 2020. (Cited on 102)

[109] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," Advances in Neural Information Processing Systems, vol. 34, 2021. (Cited on 102)

[110] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in AAAI Conference on Artificial Intelligence, 2021, vol. 35, pp. 6679–6687. (Cited on 102)

[111] B. Schäfl, L. Gruber, A. Bitto-Nemling, and S. Hochreiter, "Hopular: Modern Hopfield networks for tabular data," arXiv preprint arXiv:2206.00664, 2022. (Cited on 102)

[112] H. Kim, G. Papamakarios, and A. Mnih, "The Lipschitz constant of self-attention," in International Conference on Machine Learning, 2021, pp. 5562–5571. (Cited on 103)

[113] G. Dasoulas, K. Scaman, and A. Virmaux, "Lipschitz normalization for self-attention layers with application to graph neural networks," in International Conference on Machine Learning, 2021, pp. 2456–2466. (Cited on 103)

[114] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard, "Robustness via curvature regularization, and vice versa," in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9078–9086. (Cited on 103)

[115] C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli, "Adversarial robustness through local linearization," Advances in Neural Information Processing Systems, vol. 32, 2019. (Cited on 103)

[116] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in ACM conference on recommender systems, 2016, pp. 191–198. (Cited on 105)

[117] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," in International Joint Conference on Artificial Intelligence, 2017, pp. 1725–1731. (Cited on 105)

[118] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014. (Cited on 106)

[119] F. Abboud, E. Chouzenoux, J.-C. Pesquet, J.-H. Chenot, and L. Laborelli, "Dual block-coordinate forward-backward algorithm with application to deconvolution and deinterlacing of video sequences," Journal of Mathematical Imaging and Vision, vol. 59, pp. 415–431, 2017. (Cited on 112)