Robustness Analysis of Deep Neural Networks in the Presence of Adversarial Perturbations and Noisy Labels

Von der Fakultät für Elektrotechnik und Informationstechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines Doktors
der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Master of Science
Emilio Rafael Balda Cañizares

aus Guayaquil, Ecuador

Berichter: Universitätsprofessor Dr. rer. nat. Rudolf Mathar
Universitätsprofessor Dr. sc. techn. Bastian Leibe

Tag der mündlichen Prüfung: 18. November 2019

Diese Dissertation ist auf den Internetseiten
der Hochschulbibliothek online verfügbar.

Acknowledgments

I would like expressing my deep gratitude to Univ.-Prof. Dr. rer. nat. Rudolf Mathar for his guidance during my Ph. D. at the Institute for Theoretical Information Technology at RWTH Aachen University. In particular, for encouraging me to explore different research directions and providing an adequate environment. Additionally, I would like thanking Univ.-Prof. Dr. sc. techn. Bastian Leibe for examining my thesis and delivering insightful comments.

I cannot begin expressing my thanks to my beloved wife Joana Gómez who constantly encouraged me to move forward and demonstrated her unwavering support along this journey. Thanks as well to my colleague Dr. Arash Behboodi for collaborating with me on various research topics, and for all the fruitful discussions. I would like extending my sincere thanks to Vimal Radhakrishnan for sharing an office with me these years. The same goes for all my colleagues at the Institute for Theoretical Information Technology. It was my absolute pleasure to work with everyone and exchange knowledge about various topics.

Ultimately, I am deeply indebted to my parents who always believed in my abilities, for nurturing and advising me through my studies.

Aachen, November 2019 Emilio Rafael Balda Cañizares

Contents

1

Introduction

Deep Neural Networks (DNNs) excelled in recent years in many learning tasks and demonstrated outstanding achievements in speech analysis [1] and visual recognition [2]–[5]. Despite their success, they have been shown to suffer from instability in their classification under adversarial perturbations [6]. Adversarial perturbations are intentionally worst case designed noise that aims at changing the output of a DNN to an incorrect one. Although DNNs might achieve robustness to random noise [7], it was shown that there is a clear distinction between the robustness of a classifier to random noise and its robustness to adversarial perturbations. The existence of adversarial perturbations was known for machine learning algorithms [8], however, they were first noticed in deep learning research in 2014 by Szegedy *et al.* [6]. The peculiarity of adversarial perturbations laid in the fact that they managed to fool state of the art neural networks into making confident and wrong decisions in classification tasks, and they, nevertheless, appeared unperceived to the naked eye. These discoveries gave rise to extensive research on understanding the instability of DNNs, exploring various attacks, and devising multiple defenses (for instance refer to [9]–[11] and references therein).

Another kind of corruption that is present, when training and evaluating DNNs, is label noise. The study of robustness and generalization of properties of classifiers to this type of corruption dates back to 1988, in the well known work of Angluin & Laird [12]. Since then, there have been several advances on understanding the robustness, optimization, and generalization properties of classifiers in the presence of label noise [13], [14]. Up to date, most of the theoretical work in this direction considers classification with binary

labels. However, recent works generalized existing results to the case of multi-class classification, such as the work of Ghosh *et al.* [15].

In this thesis, we study the robustness and generalization properties of DNNs under various noisy regimes, due to corrupted inputs or labels. Such corruptions can be either random or intentionally crafted to disturb the target DNN. Note that inputs corrupted by maliciously designed perturbations (*i.e.*, adversarial examples) have been shown to severely degrade the performance of DNNs. However, due to the non-linearity of DNNs, crafting such perturbations is non-trivial.

In Chapter 3, we first address the problem of designing algorithms for generating adversarial examples, known as adversarial attacks. We start with a general formulation of this problem and, through successive convex relaxations, propose a framework for computing adversarial examples under various desired constraints. Using this approach, we derive novel methods that consistently outperform existing algorithms in tested scenarios. In addition, new algorithms are also formulated for regression problems. We show that adversarial vulnerability is also an issue in various regression tasks, a problem that has so far been overlooked in the literature.

One central question among the community is which characteristics must DNNs have in order to be robust against adversarial examples. In Chapter 4, we address this question in terms of compression, which appears in the form of low-complexity structures present in the weight matrices of robust DNNs. We show that adversarial training seems to induce low complexity structures such as sparsity and low-rankness in the weight matrices of DNNs. We give a theoretical justification for this phenomenon using linear classifiers and the notion of effective sparsity. We show that, if the input data lies in a linear low-dimensional subspace, the robustness of linear classifiers can be improved without loss of accuracy by appropriately decreasing the rank of the weight matrix without increasing its effective sparsity. Motivated by these results, we use different regularization techniques to show empirically that simultaneously inducing effective low-rankness and sparsity leads to significant improvements in robustness. The effect low-rankness on robustness seems to be more pronounced for Convolutional Neural Networks (CNNs). Although adversarial training still produced more robust models than the proposed regularization schemes, these results show that low-rankness has an important role in the robustness of DNNs.

While there has been a vast amount of works on the design and understanding of DNNs resistant to these attacks, their generalization properties are less understood. How well does adversarial robustness generalize from the training set to unseen data? In Chapter 5, we use Statistical Learning Theory (SLT) to bound the so-called adversarial risk of DNNs. Proving SLT bounds for deep learning is on-going research with various existing frameworks. Among these SLT frameworks, we choose a compression-based technique that established state of the art results for DNNs in the non-adversarial regime. Our bound leverages the sparsity structures induced by adversarial training and has no explicit dimension dependence, which is particularly challenging for ℓ_∞ adversarial attacks. In addition, it improves over existing results to ℓ_∞ bounded inputs.

To complete this work, in Chapter 6, we shift our focus from perturbed inputs to noisy labels and analyze how DNNs learn when a portion of the inputs is incorrectly labeled. In this setup, we use information theory to characterize the behavior of classifiers. Under noisy labels, we study the trajectory of DNNs in the information plane, formed by the entropy of estimated labels and the conditional entropy between given and estimated labels. We first generalize the error bound of Menon *et al.* [16] to the multi-class setting and derive fundamental error entropy relations. We use these results to analyze the trajectory of DNNs in the information plane and show their de-noising capabilities. Under simplified scenarios, we are able to analytically characterize these trajectories for linear classifiers and linearly separable data. This result shows a trajectory for properly trained networks that seems to be consistent among DNNs in real image classification tasks. In addition, we show that underfitted, overfitted, and well-trained DNNs exhibit significantly different trajectories in the information plane. Such phenomena are not visible when considering only training and validation error, thus showing the potential of using information-theoretic quantities to provide a richer view of the learning process than standard classification error.

This thesis contains unpublished work that may be used in further publications.

2

Technical Background

2.1. Notation and Norms

Throughout this thesis, the letters $a, b, \ldots$ are used for scalars, $\mathbf{a}, \mathbf{b}, \ldots$ for vectors, $\mathbf{A}, \mathbf{B}, \ldots$ for matrices and $\mathcal{A}, \mathcal{B}, \ldots$ for sets. These scalars, vectors, matrices, and sets, may be random depending on the context. To denote the set $\{1, \ldots, n\}$, we make use of the shorthand notation $[n]$ for $n \in \mathbb{N}$. The operator $(\cdot)^\top$ denotes the matrix transposition.

For any vector $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ and $p > 0$, the ℓ_p-norm of $\mathbf{x}$ is defined by

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} x_i^p \right)^{1/p} .$$

When p tends to zero, the above definition converges to the number of non-zero entries of the vector. This is called, with an abuse of terminology, the ℓ_0-norm. The explicit definition is given as

$$\|\mathbf{x}\|_0 := \sum_{i=1}^{n} \mathbb{1}_{(x_i \neq 0)} ,$$

where $\mathbb{1}_{(\cdot)}$ is the indicator function. The ℓ_0-norm gives the sparsity order of the vector $\mathbf{x}$. The ℓ_∞-norm of a vector $\mathbf{x}$ is obtained when $p \to \infty$. It is defined as

$$\|\mathbf{x}\|_\infty := \max_{i \in [n]} |x_i| .$$

For $p > 0$, the dual norm of the ℓ_p-norm is defined as

$$\|\mathbf{x}\|_p^* := \sup_{\|\mathbf{z}\|_p=1} \langle \mathbf{x}, \mathbf{z} \rangle \ .$$

For the case of matrices, any matrix $\boldsymbol{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^{m \times n}$ has mixed (p, q)-norm given by

$$\|\boldsymbol{X}\|_{p,q} := \left\| \left(\|\mathbf{x}_1\|_p, \ldots, \|\mathbf{x}_n\|_p \right)^\top \right\|_q \ .$$

Similarly, the operator (p, q)-norm is denoted by $\|\boldsymbol{X}\|_{p \to q}$ and defined as

$$\|\boldsymbol{X}\|_{p \to q} := \sup_{\|\mathbf{z}\|_p \leq 1} \|\boldsymbol{X}\mathbf{z}\|_q = \sup_{\|\mathbf{z}\|_p = 1} \|\boldsymbol{X}\mathbf{z}\|_q \ . \tag{2.1}$$

For the case when $p = q$, we make use of the shorthand notation $\|\boldsymbol{X}\|_p$ to denote $\|\boldsymbol{X}\|_{p \to p}$. The Frobenius norm of $\boldsymbol{X}$ is denoted by $\|\boldsymbol{X}\|_{\mathrm{F}}$ and defined as

$$\|\boldsymbol{X}\|_{\mathrm{F}} := \left(\sum_{j=1}^{n} \|\mathbf{x}_j\|^2 \right)^{1/2} \ . \tag{2.2}$$

The Singular Value Decomposition (SVD) of $\boldsymbol{X}$ is given by $\boldsymbol{X} = \sum_{i=1}^{\min\{m,n\}} \sigma_i \, \mathbf{u}_i \mathbf{v}_i^\top$, where $\sigma_1 \geq \cdots \geq \sigma_{\min\{m,n\}}$ are the singular values of $\boldsymbol{X}$ and $\mathbf{u}_i, \mathbf{v}_i$ the singular vectors for $i = 1, \ldots, \min\{m, n\}$. Then, let us define the operator $\sigma : \mathbb{R}^{m \times n} \to \mathbb{R}^{\min\{m,n\}}$ which outputs the column vector containing the singular values of a matrix, in descending order. Formally, that is

$$\sigma(\boldsymbol{X}) := \left(\sigma_1, \ldots, \sigma_{\min\{m,n\}} \right)^\top \ .$$

Using this notion, the nuclear norm of $\boldsymbol{X}$ is defined as the ℓ_1-norm of $\sigma(\boldsymbol{X})$, that is

$$\|\boldsymbol{X}\|_* := \|\sigma(\boldsymbol{X})\|_1 \ .$$

In other words, since singular values are always non-negative, the nuclear norm of $\boldsymbol{X}$ corresponds to the sum of its singular values. The vectorization operation is denoted by $\mathrm{vec}\,(\cdot) : \mathbb{R}^{m \times n} \to \mathbb{R}^{mn}$, which corresponds to

$$\mathrm{vec}\,(\boldsymbol{X}) := \left(\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top \right)^\top \ .$$

Now, let us assume that $h : \mathbb{R}^n \to \mathbb{R}$ is a scalar function that takes the vector $\mathbf{x} = (x_1, \ldots, x_n)^\top$ as its input. Then, the gradient of such function is defined as

$$\nabla h(\mathbf{x}) := \left(\frac{\partial h}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial h}{\partial x_n}(\mathbf{x}) \right)^\top \ .$$

Extending this notion to vectorial functions leads to the definition of Jacobian matrix. To that end, let f be a vectorial function $f : \mathbb{R}^n \to \mathbb{R}^m$ given by $f(\mathbf{x}) =$

DOMAIN \ CO-DOMAIN	ℓ_1	ℓ_2	ℓ_∞
ℓ_1	$\|\boldsymbol{X}\|_{1,\infty}$	$\|\boldsymbol{X}\|_{2,\infty}$	$\|\boldsymbol{X}\|_{\infty,\infty}$
ℓ_2	NP-hard	$\|\sigma(\boldsymbol{X})\|_\infty$	$\|\boldsymbol{X}^\top\|_{2,\infty}$
ℓ_∞	NP-hard	NP-hard	$\|\boldsymbol{X}^\top\|_{1,\infty}$

Table 2.1.: Known solutions of $\|\boldsymbol{X}\|_{p\to q}$, from the work of Tropp and Aaron [17]. By the definition of operator norm in (2.1), p corresponds to the domain, while q to the co-domain.

$(f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, where $f_i : \mathbb{R}^n \to \mathbb{R}$ are scalar functions for $i = 1, \ldots, m$. The Jacobian of f at the point $\mathbf{x}$ is denoted by $\mathbf{J}_f(\mathbf{x})$ and defined as

$$\mathbf{J}_f(\mathbf{x}) := \left[\frac{\partial f_i}{\partial x_j}(\mathbf{x})\right]_{i\in[n],j\in[m]} .$$

One useful property used in this work is the relation between vector norms, that is, for any $\mathbf{x} \in \mathbb{R}^n$ and $p > r > 0$, we have that

$$\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_r \leq n^{1/r-1/p} \|\mathbf{x}\|_p .$$

For $p > 1$, the dual norm of the ℓ_p-norm boils down to $\|\mathbf{x}\|_p^* = \|\mathbf{x}\|_q$ where $q > 1$ is such that $1/p + 1/q = 1$. For example, the ℓ_2-norm is the dual norm of itself. By continuity, the ℓ_1-norm has ℓ_∞ as its dual norm. One main result derived from this fact is Hölders inequality, which states the following result

$$|\langle \mathbf{x}, \mathbf{z} \rangle| \leq \|\mathbf{x}\|_p \|\mathbf{z}\|_q ,$$

where $\langle \cdot, \cdot \rangle$ denotes the canonical inner product[1].

Let us introduce some commonly used relations between matrix norms. First, by the definition of Frobenius norm in (2.2) we can also express it as

$$\|\boldsymbol{X}\|_{\mathrm{F}} = \|\boldsymbol{X}\|_{2,2} .$$

On the other hand, in some cases, we can also relate operator norms with mixed norms [17]. A summary of solutions for computing some commonly used operator norms is shown in Table 2.1. Finally, we introduce further notation to deal with random processes. Let $x \in \mathbb{R}$ be a random variable and $\mathcal{D}_X : \mathbb{R} \to [0, 1]$ some probability measure. We use the notation $\mathbb{P}_{x\sim\mathcal{D}_X}[\cdot]$ to denote the probability of an event when x is distributed according to $\mathcal{D}_X$. Similarly, the operators $\mathbb{E}_{x\sim\mathcal{D}_X}[\cdot]$ and $\mathrm{Var}_{x\sim\mathcal{D}_X}[\cdot]$ denote the expectation and variance of x with the probability measure $\mathcal{D}_X$. When the distribution of x is clear by the context, we drop the subscripts and use the shorthand notations $\mathbb{P}[\cdot], \mathbb{E}[\cdot]$ and $\mathrm{Var}[\cdot]$.

[1] That is $\langle \mathbf{x}, \mathbf{z} \rangle := \mathbf{x}^\top \mathbf{z}$.

2.2. Statistical Learning

2.2.1. System Model

This work is mainly focused on classification. Along this thesis, we keep the following domain-specific notation:

- Let $\mathcal{X}$ be the feature space, that is the set of all possible inputs.
- Similarly, the label space is denoted by $\mathcal{Y}$.
- The dimensionality of the input is denoted by n, that is $\mathcal{X} \subseteq \mathbb{R}^n$.
- The number of possible classes is k and the label space is set to be $\mathcal{Y} = \{1, 2, \ldots, k\}$.
- $\mathbf{x} \in \mathcal{X}$ is the input vector with its corresponding label $y \in \mathcal{Y}$.
- The tuple $(\mathbf{x}, y) \sim \mathcal{D}$ is assumed to be random and distributed according to the data distribution $\mathcal{D}$.
- The training set $\mathcal{S}$ is defined as a set of independent and identically distributed (i.i.d.) realizations of $(\mathbf{x}, y) \sim \mathcal{D}$, which are used to tune the parameters of a classifier.
- The size of the training set is denoted by $m := |\mathcal{S}|$.

Definition 2.2.1 (Classifier and score functions)**.** *A classifier is defined by the mapping $c : \mathcal{X} \to \mathcal{Y}$ that maps an input $\mathbf{x} \in \mathcal{X}$ to its estimated class $c(\mathbf{x}) \in \mathcal{Y}$. The mapping $c(\cdot)$ is itself defined by*

$$c(\mathbf{x}) = \underset{l \in \mathcal{Y}}{\operatorname{argmax}} \{f_l(\mathbf{x})\} , \tag{2.3}$$

where $f_l(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$'s are called score functions representing the probability of class belonging. Then, a classifier is fully defined by its vector of score functions. That is, the function $f : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$ defined as

$$f(\mathbf{x}) = \left(f_1(\mathbf{x}), \ldots, f_{|\mathcal{Y}|}(\mathbf{x})\right)^\top .$$

For example, the score functions of a linear classifier are given by

$$f(\mathbf{x}) = \boldsymbol{W}^\top \mathbf{x} ,$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times k}$ is a matrix of tunable parameters. Note that we neglect the bias term, since it can be easily included by appending a scalar 1 to the input vector $\mathbf{x}$. Similarly, a Fully Connected Neural Network (FCNN) with d-layers has a score function given by

$$f(\mathbf{x}) = \phi^d \left(\boldsymbol{W}^{d^\top} \phi^{d-1} \left(\boldsymbol{W}^{d-1^\top} \cdots \phi^1 \left(\boldsymbol{W}^{1^\top} \mathbf{x}\right) \cdots\right)\right) , \tag{2.4}$$

where ϕ^i is a (possibly non-linear) function applied point-wise to the entries of vectors, for $i = 1, \ldots, d$. Strictly speaking, a linear classifier is also a 1-layer neural network with a linear activation function.

Using these notions we can now define the margin of a classifier, given a sample. To that end, given the instance $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$, the margin of f is defined as

$$\ell(f; \mathbf{x}, y) = f_y(\mathbf{x}) - \max_{j \neq y} f_j(\mathbf{x}) \,. \tag{2.5}$$

In this manner, a positive margin implies correct classification. This allows us to define the expected risk, with margin $\gamma \geq 0$, of a classifier f as

$$L_\gamma(f) = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\ell(f; \mathbf{x}, y) \leq \gamma\right] .$$

Note that this definition implies that $L_0(f)$ is the probability of incorrect classification, also known as test error. In practice, we do not have access to the data distribution $\mathcal{D}$, thus we cannot compute $L_\gamma(f)$. However, we have access to the training set $\mathcal{S}$, which allows us to obtain an empirical estimate of this quantity. This estimate is called the empirical risk and it is defined as

$$\widehat{L}_\gamma(f) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{(\ell(f;\mathbf{x}_i,y_i)\leq\gamma)} \,,$$

where $(\mathbf{x}_i, y_i)$ are the elements of $\mathcal{S}$, that is $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$. Note that, for a fixed f, we have that

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^m} \left[\widehat{L}_\gamma(f)\right] = L_\gamma(f) \,.$$

This property allows bounding the expected risk by using famous concentration inequalities from the literature. Nonetheless, one should take care since f is usually not fixed, instead it is computed using $\mathcal{S}$.

2.2.2. Concentration Inequalities

Let us introduce some well known concentration inequalities that are employed along this work. First, when the expectation and variance of a random variable are known and finite, we may employ Chebyshev's inequality, which is stated in the following lemma.

Lemma 2.2.2 (Chebyshev's Inequality)**.** *Let x be a random variable with finite mean and variance. Then, for any $t > 0$ the following holds*

$$\mathbb{P}(|x - \mathbb{E}[x]| \geq t) \leq \frac{\mathrm{Var}[x]}{t^2} \,.$$

Additionally, for a sequence of i.i.d. random variables, if these variables are bounded, we may use the Hoeffding's inquality. This inequality is introduced in the following lemma.

Lemma 2.2.3 (Hoeffding's Inequality)**.** *Let x_i be i.i.d. random variables bounded between $[a_i, b_i]$ for $a_i < b_i$ and $i = 1, \ldots, m$. Let $\overline{x}$ be the random variable $\overline{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$, then, for any $t > 0$ it holds*

$$\mathbb{P}(|\overline{x} - \mathbb{E}[\overline{x}]| \geq t) \leq 2\exp\left(-\frac{2m^2t^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right).$$

Moreover, this lemma is often reformulated in the following form as well, which may be useful in some situations.

Corollary 2.2.3.1. *In the same setup as Lemma 2.2.3, let $z := \sum_{i=1}^{m} x_i$. Then, for any $t > 0$ it holds*

$$\mathbb{P}(|z - \mathbb{E}[z]| \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right).$$

2.2.3. Elementary Definitions from Information Theory

We shortly review some elementary concepts from information theory, such as entropy and mutual information, that are used in this thesis. The entropy of a discrete random variable $y \in \mathcal{Y}$ is defined as

$$H(y) = -\sum_{i \in \mathcal{Y}} \mathbb{P}(y = i) \log \mathbb{P}(y = i) = -\mathbb{E}\left[\log \mathbb{P}(y)\right].$$

This quantity is bounded as $0 \leq H(y) \leq \log|\mathcal{Y}|$ and measures the amount of uncertainty present in y. Similarly, the conditional entropy of y given $\widehat{y}$ is defined as

$$H(y|\widehat{y}) = -\mathbb{E}\left[\log \mathbb{P}(y|\widehat{y})\right]$$

and quantifies the uncertainty about y given that $\widehat{y}$ is known. Finally, the mutual information $I(y;\widehat{y})$ between y and $\widehat{y}$ measures how much information does one random variable carry about the other. It may be defined in terms of entropies as

$$I(y;\widehat{y}) = H(y) - H(y|\widehat{y}) = H(\widehat{y}) - H(\widehat{y}|y).$$

3

Generation of Adversarial Examples for Classification and Regression

Despite the tremendous success of deep neural networks in various learning problems, it has been observed that adding intentionally designed adversarial perturbations to inputs of these architectures leads to erroneous classification with high confidence in the prediction. In this chapter[1], we show that adversarial examples can be generated using a generic approach that relies on the perturbation analysis of learning algorithms. Formulated as a convex program, the proposed approach retrieves many current adversarial attacks as special cases. It is used to propose novel attacks against learning algorithms for classification and regression tasks under various new constraints with closed form solutions in many instances. In particular, we derive new attacks against classification algorithms which are shown to be top performing on various architectures. Although classification tasks have been the main focus of adversarial attacks, we use the proposed approach to generate adversarial perturbations for various regression tasks. Designed for single pixel and single subset attacks, these attacks are applied to autoencoding, image colorization and real time object detection tasks, showing that adversarial perturbations can degrade equally gravely the output of regression tasks.

[1]This chapter contains the work from our publications [18], [19].

3.1. Related Work

Most adversarial attacks fall generally into two classes, white-box and black-box attacks. White-box attacks use complete or partial knowledge of the machine learning architecture, see for instance [20]. In constrast, black-box attacks do not require any information about the target neural network, see for instance [21]. In this work, we focus only on white-box attacks with full knowledge of the function implemented by the learning algorithm. The attacks, as in [20], [22], [23], act on the system inputs and add perturbations that are not perceived by the system's administrator such that the performance of the system is severely degraded.

Adversarial perturbations were obtained in [6] to maximize the prediction error at the output and were approximated using the box-constrained Limited memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm. The Fast Gradient Sign Method (FGSM), proposed by Goodfellow *et al.* [20], was based on finding the scaled sign of the gradient of the cost function. Note that the FGSM aims at minimizing ℓ_∞-norm of the perturbation while the former algorithm minimizes ℓ_2-norm of the perturbation under box constraint on the perturbed example.

More effective attacks utilize either iterative procedures or randomization of the perturbations and instances as in [22], [24]. The algorithm DeepFool [22] conducts an iterative linearization of the DNN to generate perturbations that are minimal in the ℓ_p-norm for $p > 1$. Kurakin *et al.* [25] proposed an iterative version of FGSM, called Basic Iterative Method (BIM). This method was later extended by Madry *et al.* [24], where randomness was introduced in the computation of adversarial perturbations. This attack is called the Projected Gradient Descent (PGD) method and was employed in [24] to devise a defense against adversarial examples. An iterative algorithm based on PGD combined with randomization was introduced in [26] and has been used to dismantle many defenses so far [27]. Another popular way of generating adversarial examples is by constraining the ℓ_0-norm of the perturbation. These types of attacks are known as single pixel attacks [28] and multiple pixel attacks [29]. Despite their differences, many of the above attacks share a similar underlying principle. At some moment, they rely on a simple characterization of input perturbations on the output of the algorithm. In this work, we build on this idea to design new attacks for classification and regression tasks.

An interesting feature of these perturbations is their generalization over other datasets and DNNs [20], [23]. These perturbations are called universal adversarial perturbations. This is partly explained by the fact that certain underlying properties of the perturbation, such as orientation in case of image perturbation, matters the most and is therefore generalized through different datasets. For example, the attack of Tramèr *et al.* [30] shows that adversarial examples transfer from one random instance of a neural network to another. In that work, the authors showed the effectiveness of these types of attacks

for enhancing the robustness of neural networks, since they provide diverse perturbations during adversarial training. Moreover, Moosavi-Dezfooli *et al.* [31] showed the existence of universal adversarial perturbations that are independent from the system and the target input. We will not go into more details about universal perturbations in this work.

Since the rise of adversarial examples for image classification, novel algorithms have been developed for attacking other types of systems. In the field of computer vision, [32] constructed an attack on image segmentation, while Xie *et al.* [33] designed attacks for object detection. The Houdini attack [34] aims at distorting speech recognition systems. Moreover, Papernot *et al.* [35] taylored an attack for recurrent neural networks, and Lin *et al.* [36] for reinforcement learning. Adversarial examples exist for probabilistic methods as well. For instance, Kos *et al.* [37] showed the existence of adversarial examples for generative models. For regression problems, Tabacof *et al.* [38] designed an attack that specifically targets variational autoencoders. There seems, however, to be a gap in the literature when it comes to adversarial attacks on regression problems. One of the goals of this work is to fill this gap by proposing attacks on various regression problems.

Before going further, we shortly overview some theoretical explanations of adversarial examples as well as common defenses. There are various theories regarding the nature of adversarial examples and the subject is heavily investigated. Initially, the authors in [20] proposed the linearity hypothesis where the existence of adversarial images is attributed to the approximate linearity of classifiers, although this hypothesis has been challenged by Tanay & Griffin [39]. Some other theories focus mostly on decision boundaries of classifiers and their analytic properties [7], [40]. The work of Raghunathan *et al.* [41] provides a framework for determining the robustness of a classifier against adversarial examples with some performance guarantees. For a more recent theoretical approach to this problem refer to [42]. There exist several types of defenses against adversarial examples, as well as subsequent methods for bypassing them. Defensive distillation was proposed in [43] as a defense method. It extracts knowledge about the training points and feeds it back as part of the training to improve the robustness of the network. The method directly controls the amplitude of network gradients as a defense technique. The authors in [44], however, proposed three attacks to bypass defensive distillation. Similarly, the attacks of Athalye *et al.* [26], bypassed 7 out of 9 non-certified defenses presented at ICLR 2018 that claimed to be white-box secure. The term non-certified refers to defenses that do not provide any certificate that guarantees robustness against particular attacks. The most common defense is adding adversarial examples to the training set, also known as adversarial training. For that purpose different adversarial attacks may be employed. Recently, training with the PGD attack is used in [24] to provide the state of the art defense against adversarial examples for various image classification datasets.

3.1.1. Our Contributions

In this work, we provide various new adversarial attacks for classification and regression tasks. We consider only white-box attacks where the full knowledge of the machine learning model is assumed to be available. The common thread of these new attacks is the perturbation analysis of learning algorithms that yields a tractable optimization problem for generating new attacks. This idea, as it will be shown, underlies many existing attacks. We build upon our previous work [19] to introduce a connection between perturbation analysis of learning algorithms and adversarial examples. For classification tasks, we start with a fairly general formulation of the adversarial generation problem (AGP), and show how perturbation analysis can be used to obtain a convex optimization problem for generating adversarial examples. This problem, (AGP.II), is the basis for generating adversarial examples. We address feasibility issues of the preceding problem and propose multiple techniques to get around this issue in general, among which (3.4) is introduced for the first time. Besides, closed form solutions are provided for a few cases. We propose novel adversarial attacks for classification, given in Algorithm 1, which are benchmarked with state of the art attacks. Our proposed attack is an iterative method that constrains the norm of adversarial perturbations and we apply it after randomization of the training instances.

Another contribution of this work is to use a similar technique in context of adversarial perturbations for regression problems, a topic that has not yet been widely explored. Regression loss functions differ from classification loss functions in that it is sufficient to maximize the output perturbation measured by an application-dependent function, for instance the ℓ_2-norm of the output error. In classification tasks, such maximization might not necessarily change the output label particularly because these perturbations might push the instances far away from classification margins. There is, however, no natural margin in regression tasks. We use perturbation analysis to formulate the loss maximization problem in a tractable fashion. Similar to classification problems, adversarial examples can be generated using convex optimization with closed-form solutions for a few special cases. The proposed optimization problems for regression are, to the best of our knowledge, novel. We discuss single pixel and single subset attacks for regression tasks. It is shown that this problem is related to the MaxCut problem and hence difficult to solve. We propose a greedy algorithm to solve this problem.

Finally, the proposed algorithms are experimentally evaluated using state of the art benchmarks for classification and regression tasks. In classification tasks, the performance of our proposed attack is consistently among the top attacks and sometimes the best one. In regression tasks, we demonstrate several attacks for regression tasks such as image colorization, autoencoding and object detection. Although autoencoders show better robustness in general, the other algorithms are severely degraded under adversarial perturbations.

3.2. Fooling Classifiers with First-Order Perturbation Analysis

The perturbation analysis, also called sensitivity analysis, is used in signal processing for analytically quantifying the error at the output of a system that occurs as consequence of a known perturbation at the system's input. Adversarial images can also be considered as a slightly perturbed version of original images that manage to change the output of the classifier. Indeed, the generation of adversarial examples in [20], [22] is implicitly based on maximizing the effect of an input perturbation on a relevant function which is either the classifier function or the cost function used for training. In the FGSM, given in [20], the perturbation at the output of the training cost function is first analyzed using first-order perturbation analysis of the cost function and then maximized to fool the algorithm. The DeepFool method, given in [22], maximizes the output perturbation for the linearized approximation of the underlying classifier which is indeed its first order-perturbation analysis. In this section, we develop further the connection between perturbation analysis and adversarial examples. To generate adversarial examples, we provide a generic approach that starts from a fairly abstract formulation and transforms it into a tractable problem by sequentially using perturbation analysis.

3.2.1. Adversarial Perturbation Design Problem

We start with formulating the problem of adversarial perturbation design. As it was mentioned above, adversarial examples can be considered as a perturbed version of training examples changed by an adversarial perturbation $\boldsymbol{\eta}$. As we will see later, the perturbation analysis is straightforward when the underlying system behaves smoothly and can be modeled by differentiable functions. The classifier function, however, maps inputs to discrete set of labels and therefore it is not differentiable. Instead, the classification problem is slightly modified, by defining the classifier as in Definition 2.2.1, to facilitate the perturbation analysis. Using that definition, a classifier $c : \mathbb{R}^n \to [k]$ can be expressed as

$$c(\mathbf{x}) = \underset{l \in [k]}{\operatorname{argmax}} \{f_l(\mathbf{x})\} , \tag{3.1}$$

where $f_l(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$'s are called score functions representing the probability of class belonging. The function $f(\mathbf{x})$ given by the vector $(f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$ can be assumed to be differentiable almost everywhere for many classifiers.

The problem of adversarial generation consists of finding a perturbation that changes the classifier's output. However, it is desirable for adversarial perturbations to modify training instances only in an insignificant and unnoticeable way. This is controlled by adding a constraint on the adversarial perturbation. For instance, the perturbation generated by the FGSM is bounded in the ℓ_∞-norm and the DeepFool method directly minimizes the norm of the perturbation that changes the classifier's output.

While DeepFool might generate perturbations that are perceptible, the FGSM might not change the classifier's output. It is indeed an intriguing property of adversarial examples that the perturbation does not distort the image significantly so that the naked eye cannot detect any notable change in the images. One way of imposing this property in adversarial design is to constrain the input perturbation to keep the output of the ground truth classifier, also called oracle classifier [10], intact. The oracle classifier represents the naked eye in case of image classification. The score functions of the oracle classifier are denoted by $g_l(\cdot)$. The undetectability constraint for an adversarial perturbation $\boldsymbol{\eta}$ is formulated as

$$\mathcal{T}_g(\mathbf{x}, \boldsymbol{\eta}) = g_{c(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq c(\mathbf{x})} g_l(\mathbf{x} + \boldsymbol{\eta}) > 0\,. \tag{3.2}$$

This inequality means that even after applying the perturbation $\boldsymbol{\eta}$ to an instance $\mathbf{x}$ with the correct label $c(\mathbf{x})$, the score function of the correct class $g_{c(\mathbf{x})}(\mathbf{x}+\boldsymbol{\eta})$ is superior to the other score functions. A fundamental trait of the oracle classifier, therefore, is its robustness to the adversarial perturbation $\boldsymbol{\eta}$. Therefore the problem of adversarial design can be formulated as follows.

Problem 3.2.1 (Adversarial Generation Problem)**.** *For a given $\mathbf{x} \in \mathbb{R}^n$, find a perturbation $\boldsymbol{\eta} \in \mathbb{R}^n$ to fool the classifier $c(\cdot)$ by the adversarial sample $\widehat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\eta}$ such that $c(\mathbf{x}) \neq c(\widehat{\mathbf{x}})$ and the oracle classifier is not changed, i.e.,*

$$\begin{aligned} \text{Find}: \quad & \boldsymbol{\eta} \\ \text{s.t.} \quad & \mathcal{T}_f(\mathbf{x}, \boldsymbol{\eta}) = f_{c(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq c(\mathbf{x})} f_l(\mathbf{x} + \boldsymbol{\eta}) < 0 \\ & \mathcal{T}_g(\mathbf{x}, \boldsymbol{\eta}) = g_{c(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq c(\mathbf{x})} g_l(\mathbf{x} + \boldsymbol{\eta}) > 0\,. \end{aligned} \tag{AGP}$$

We pose the problem (AGP) as a general starting point for the adversarial design. As it is, the problem (AGP) does not have that much utility in practice but serves as the starting point. Next we explore different methods for making this problem tractable, all of them based on perturbation analysis of constraints. Note that the function $\mathcal{T}_f(\mathbf{x}, \cdot)$ is exactly the margin of the classifier f, as defined in (2.5), at the point $\mathbf{x} + \boldsymbol{\eta}$ with label $y = c(\mathbf{x})$. Formaly, that is $\mathcal{T}_f(\mathbf{x}, \boldsymbol{\eta}) = \ell(f; \mathbf{x} + \boldsymbol{\eta}, c(\mathbf{x}))$, thus it is natural for an attacker to aim at minimizing such margin. Since $\mathbf{x}$ and f are fixed for the attacker, we simplify the notation by dropping the subscript f and assuming that gradients are always with respect to $\boldsymbol{\eta}$, that is $\mathcal{T}(\mathbf{x}, \cdot) = \mathcal{T}_f(\mathbf{x}, \cdot)$ and $\nabla\mathcal{T}(\mathbf{x}, \cdot) = \nabla_{\boldsymbol{\eta}}\mathcal{T}_f(\mathbf{x}, \cdot)$. We keep these shorthand notations throughout the chapter.

3.2.2. Perturbation Analysis

The problem (AGP) constitutes the starting point for adversarial design. Two issues prevent us from directly applying it. We use perturbation analysis to propose a solution to these issues. First, the oracle function is not known in general. However, since the

oracle function is assumed to be robust to adversarial perturbations, the constraint can be replaced with constraints on the perturbation itself, for instance by imposing upper bounds on the ℓ_p-norm of the perturbation. Indeed, if the perturbation analysis is applied to the oracle function, its robustness implies that the output perturbation is $\mathcal{O}(\|\boldsymbol{\eta}\|_p)$. Therefore, adding constraints on the ℓ_p-norm translates into constraints on the oracle function as in (3.2). The constraint on the oracle function can be therefore approximated by $\|\boldsymbol{\eta}\|_p \leq \varepsilon$ for sufficiently small $\varepsilon \in \mathbb{R}^+$. This means that the noise is sufficiently small in ℓ_p-norm sense so that the observer does not notice it. Different classes of attacks can be obtained for different choices of p and are well known in the literature such as ℓ_∞-attacks, ℓ_2-attacks and ℓ_1-attacks (see the survey in [9] for details).

The second problem with (AGP) is that the function $\mathcal{T}(\mathbf{x}, \cdot)$ can be non-convex or difficult to optimize directly. Here again, perturbation analysis can be employed to approximate $\mathcal{T}(\mathbf{x}, \cdot)$ with a tractable function like linear functions. The first order perturbation analysis of $\mathcal{T}$ yields

$$\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) = \mathcal{T}(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \mathbf{0}) + \mathcal{O}(\|\boldsymbol{\eta}\|_2^2)\,,$$

where $\mathcal{O}(\|\boldsymbol{\eta}\|_2^2)$ contains higher order terms. Following the above approximations using perturbation analysis, we propose the following relaxed optimization problem.

Problem 3.2.2 (Relaxed Adversarial Generation Problem)**.** *For a given $\mathbf{x} \in \mathbb{R}^n$, a perturbation $\boldsymbol{\eta} \in \mathbb{R}^n$ is found to fool the classifier $c(\cdot)$ using*

$$\begin{aligned} \textit{Find:} \quad & \boldsymbol{\eta} \\ \textit{s.t.} \quad & \mathcal{T}(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \mathbf{0}) < 0, \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \end{aligned} \tag{AGP.II}$$

The above problem is the result of applying perturbation analysis to the problem (AGP). It is a convex optimization problem that can be efficiently solved. One significant advantage of the problem (AGP.II) is that it can incorporate additional constraints to the optimization problem such as sparsity constraint that leads to single-pixel attacks. Furthermore, as we will see later, this formulation of the problem yields some of the well known existing adversarial methods. Unfortunately, the above problem is not always feasible as stated in the following proposition.

Theorem 3.2.3. *The optimization problem* (AGP.II) *is not feasible if for $q = \frac{p}{p-1}$*

$$\varepsilon \|\nabla \mathcal{T}(\mathbf{x}, \mathbf{0})\|_q < \mathcal{T}(\mathbf{x}, \mathbf{0})\,. \tag{3.3}$$

Proof. The proof follows a simple duality argument and is an elementary optimization theory result. A similar result can be inferred from [45]. We repeat the proof for completeness. Note that the dual norm of ℓ_p is defined by

$$\|\mathbf{x}\|_p^* = \sup\{\mathbf{a}^\top \mathbf{x} : \|\mathbf{a}\|_p \leq 1\}.$$

Furthermore $\|\mathbf{x}\|_p^* = \|\mathbf{x}\|_q$ for $q = \frac{p}{p-1}$. Since the ℓ_p-norm of $\boldsymbol{\eta}$ is bounded by ε, the value of $\boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \mathbf{0})$ is always bigger than $-\varepsilon \|\nabla \mathcal{T}(\mathbf{x}, \mathbf{0})\|_p^*$. However if the condition (3.3) holds, then we have

$$\mathcal{T}(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \mathbf{0}) \geq \mathcal{T}(\mathbf{x}, \mathbf{0}) - \varepsilon \|\nabla \mathcal{T}(\mathbf{x}, \mathbf{0})\|_p^* > 0\,.$$

Therefore, the problem is not feasible. □

Theorem 3.2.3 shows that given a vector $\mathbf{x}$, the adversarial perturbation should have at least ℓ_p-norm equal to $\frac{\mathcal{T}(\mathbf{x},\mathbf{0})}{\|\nabla \mathcal{T}(\mathbf{x},\mathbf{0})\|_q}$. In other words if the ratio $\frac{\mathcal{T}(\mathbf{x},\mathbf{0})}{\|\nabla \mathcal{T}(\mathbf{x},\mathbf{0})\|_q}$ is small, then it is easier to fool the network by the ℓ_p-attacks. In that sense, Theorem 3.2.3 provides an insight into the stability of classifiers. In [22], the authors suggest that the robustness of the classifiers can be measured as

$$\widehat{\rho}_1(f) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{\|\widehat{\mathbf{r}}(\mathbf{x})\|_p}{\|\mathbf{x}\|_p}\,,$$

where $\mathcal{D}$ denotes the test set and $\widehat{\mathbf{r}}(\mathbf{x})$ is the minimum perturbation required to change the classifier's output. The above theorem suggests that one can also use the following as the measure of robustness, which was also derived in [45]

$$\widehat{\rho}_2(f) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{\mathcal{T}(\mathbf{x}, \mathbf{0})}{\|\nabla \mathcal{T}(\mathbf{x}, \mathbf{0})\|_q}\,.$$

The lower $\widehat{\rho}_2(f)$, the easier it gets to fool the classifier and therefore it becomes less robust to adversarial examples. One can also look at other statistics related to $\frac{\mathcal{T}(\mathbf{x},\mathbf{0})}{\|\nabla \mathcal{T}(\mathbf{x},\mathbf{0})\|_q}$ in order to evaluate the robustness of classifiers.

To get around the feasibility problem, a first approach is to start with a small enough ε and iteratively solve the problem (AGP.II). Other methods consist of relaxing one of the constraints while keeping the other constraint intact. We will explicate these methods in the next section.

3.2.3. Feasible Adversarial Perturbation Designs

Theorem 3.2.3 shows that the optimization problem (AGP.II) might not be feasible. We propose to get around this issue by solving an optimization problem which keeps only one of the constraints, depending on the scenario, and selects an appropriate objective function to preserve the other constraint as much as possible. The objective function in this sense models the deviation from the constraint and is minimized in the optimization problem. We consider two optimization problems for this purpose.

First, the norm-constraint on the perturbation is preserved. The following optimization problem, called gradient-based Norm-constrained Method (GNM), aims at minimizing $\mathcal{T}(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \mathbf{0})$ by solving the following problem:

$$\min_{\boldsymbol{\eta}} \left\{ \mathcal{T}(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \mathbf{0}) \right\} \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{3.4}$$

This method finds the best perturbation under the norm-constraint and is a novel formulation to the best of our knowledge. The constraint aims at guaranteeing that the adversarial images are still imperceptible by an ordinary observer. Note that (3.4) is fundamentally different from [22], [45], where the norm of the noise does not appear as a constraint. Using a similar duality argument, the problem (3.4) has a closed form solution given below.

Theorem 3.2.4. *If* $\nabla\mathcal{T}(\mathbf{x},\boldsymbol{\eta}) = (\frac{\partial\mathcal{T}(\mathbf{x},\boldsymbol{\eta})}{\partial\eta_1},\dots,\frac{\partial\mathcal{T}(\mathbf{x},\boldsymbol{\eta})}{\partial\eta_n})$*, the closed form solution to the minimizer of the problem* (3.4) *is given by*

$$\boldsymbol{\eta}^* = -\varepsilon\frac{1}{\|\nabla\mathcal{T}(\mathbf{x},\mathbf{0})\|_q^{q-1}}\operatorname{sign}(\nabla\mathcal{T}(\mathbf{x},\mathbf{0}))\odot|\nabla\mathcal{T}(\mathbf{x},\mathbf{0})|^{q-1} \tag{3.5}$$

for $q = \frac{p}{p-1}$*, where* $\operatorname{sign}(\cdot)$ *and* $|\cdot|^{q-1}$ *are applied element-wise, and* $\odot$ *denotes the element-wise (Hadamard) product. Particularly for* $p=\infty$*, we have* $q=1$ *and the solution is given by the following*

$$\boldsymbol{\eta}^* = -\varepsilon\operatorname{sign}(\nabla\mathcal{T}(\mathbf{x},\mathbf{0}))\,. \tag{3.6}$$

Proof. Based on the duality argument from convex analysis, it is known that

$$\sup_{\|\boldsymbol{\eta}\|_p\leq 1}\boldsymbol{\eta}^\top\nabla\mathcal{T}(\mathbf{x},\mathbf{0}) = \|\nabla\mathcal{T}(\mathbf{x},\mathbf{0})\|_p^*\,,$$

where $\|\cdot\|^*$ is the dual norm. This implies that the objective function is lower bounded by $\mathcal{T}(\mathbf{x},\mathbf{0}) - \varepsilon\|\nabla\mathcal{T}(\mathbf{x},\mathbf{0})\|_p^*$. It is easy to verify that the minimum is attained by (3.5). □

The advantage of (3.4), apart from being convex and enjoying computationally efficient solutions, is that one can incorporate other convex constraints into the optimization problem to guarantee additional required properties of the perturbation. Note that the introduced method in (3.4) can also be used for other target functions or learning problems. If the training cost function is maximized under a norm constraint, as in [20], the solution of (3.4) with $p=\infty$ recovers the adversarial perturbations obtained via the FGSM. The problem (3.4) guarantees that the perturbation is small, however, it might not change the classifier's output.

The second optimization problem, on the other hand, preserves the constraint for changing the classifier's output and minimizes the perturbation norm instead. The feasibility problem of (AGP.II) can therefore be simplified to

$$\min_{\boldsymbol{\eta}}\|\boldsymbol{\eta}\|_p \quad \text{s.t.} \quad \mathcal{T}(\mathbf{x},\mathbf{0}) + \boldsymbol{\eta}^\top\nabla\mathcal{T}(\mathbf{x},\mathbf{0}) \leq 0\,, \tag{3.7}$$

which recovers the result in [45] and in [22] although without the iterative procedure. This problem has a similar closed form solution.

Proposition 3.2.5. *If* $\nabla\mathcal{T}(\mathbf{x},\boldsymbol{\eta}) = (\frac{\partial\mathcal{T}(\mathbf{x},\boldsymbol{\eta})}{\partial\eta_1},\ldots,\frac{\partial\mathcal{T}(\mathbf{x},\boldsymbol{\eta})}{\partial\eta_n})$*, the closed form solution to the problem* (3.7) *is given by*

$$\boldsymbol{\eta}^* = -\frac{\mathcal{T}(\mathbf{x},\mathbf{0})}{\|\nabla\mathcal{T}(\mathbf{x},\mathbf{0})\|_q^{q-1}}\text{sign}(\nabla\mathcal{T}(\mathbf{x},\mathbf{0}))\odot|\nabla\mathcal{T}(\mathbf{x},\mathbf{0})|^{q-1} \tag{3.8}$$

for $q = \frac{p}{p-1}$.

Note that the perturbation found in Proposition 3.2.5, like the solution to GNM, aligns with the gradient of the classifier function and they only differ in their norm. Although the perturbation in (3.8), unlike the solution to GNM, is able to fool the classifier, the perturbation in (3.8) might be perceptible by the oracle classifier.

The formulations (3.4) and (3.7) do not exhaust all possibilities for solving the feasibility problem above. There are other variants of adversarial generation methods that rely on an implicit perturbation analysis of a relevant function. These methods can be easily obtained by small modification of the methods above.

Iterative procedures can be easily adapted to the current formulation by repeating the optimization problem until the classifier output changes while keeping the perturbation small enough at each step such that the problem (AGP.II) remains feasible. Later we provide an iterative version of the GNM and compare it with DeepFool [22], as well as other methods.

Another class of methods relies on introducing randomness in the generation process. We call this technique dithering. A notable example is the PGD attack introduced in [24] which is one of the state of the art attacks. The first-order approximation is then taken around another point $\widetilde{\boldsymbol{\eta}}$ with $\widetilde{\varepsilon} \triangleq \|\widetilde{\boldsymbol{\eta}}\|_p \leq \varepsilon$. In other words we approximate $\mathcal{T}(\mathbf{x},\cdot)$ by a linear function around the point $\widetilde{\boldsymbol{\eta}}$ within an $\widetilde{\varepsilon}$-radius from $\boldsymbol{\eta} = \mathbf{0}$. This new point $\widetilde{\boldsymbol{\eta}}$ can be computed at random using arbitrary distributions with ℓ_p-norm bounded by ε. Changing the center of the first order approximation from $\mathbf{0}$ to $\widetilde{\boldsymbol{\eta}}$ does not change the nature of the problem since $\mathcal{T}(\mathbf{x},\boldsymbol{\eta}) \approx \mathcal{T}(\mathbf{x},\widetilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta}-\widetilde{\boldsymbol{\eta}})^\top\nabla\mathcal{T}(\mathbf{x},\widetilde{\boldsymbol{\eta}})$ leads to the following problem

$$\min_{\boldsymbol{\eta}} \mathcal{T}(\mathbf{x},\widetilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta}-\widetilde{\boldsymbol{\eta}})^\top\nabla\mathcal{T}(\mathbf{x},\widetilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,,$$

which is equivalent to:

$$\min_{\boldsymbol{\eta}} \boldsymbol{\eta}^\top\nabla\mathcal{T}(\mathbf{x},\widetilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{3.9}$$

From this result one can add randomness to the computation of adversarial examples by selecting $\widetilde{\boldsymbol{\eta}}$ in a random fashion. This is desirable when training models with adversarial examples since it increases the diversity of the adversarial perturbations during training (See [30] where the authors introduce the Randomized Fast Gradient Sign Method (R-FGSM) attack).

The summary of our proposed approach is as follows. We start with the problem (AGP) by determining an appropriate loss function $\mathcal{T}(\mathbf{x}, \cdot)$. Next the problem is simplified to a tractable problem like (AGP.II) using perturbation analysis. Additional constraints on the perturbation are added at will. If the problem is feasible, the final solution yields the adversarial perturbation. Otherwise, one can use iterative and randomization techniques explained above or solve problems (3.4) or (3.7). We follow similar steps to generate adversarial examples for regression problems in the next section.

3.3. From Classification to Regression

In classical statistical learning theory, regression problems are defined in the following manner. Given $m \in \mathbb{N}$ samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ drawn according to some unknown distribution $P_{X,Y}$, a regression model computes a function $f : \mathbb{R}^n \to \mathbb{R}^k$ that aims to minimize the expected loss $\mathbb{E}_P(\mathcal{L}(f(\mathbf{x}), \mathbf{y}))$, where $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ is a function that measures the deviation between $f(\mathbf{x})$ and $\mathbf{y}$. While logarithmic losses are popular in classification problems, the squared loss $\mathcal{L}(f(\mathbf{x}), \mathbf{y}) = \|f(\mathbf{x}) - \mathbf{y}\|_2^2$ is mostly used for the general regression setting. There is, however, no general natural loss function for the regression problems, although each problem disposes of specific appropriate choices. Squared loss is certainly suitable for function approximation tasks and in particular autoencoders. Peak Signal to Noise Ratio (PSNR) is suitable for measuring the quality of image outputs. Throughout this work, it is assumed that the loss function is properly chosen for the adversarial attack on the underlying regression problem. For the sake of notation, given $\mathbf{y}$ and f, let us redefine $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta})$ as $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) = \mathcal{L}(f(\mathbf{x} + \boldsymbol{\eta}), \mathbf{y})$.

For a given f, $\mathbf{x}$ and $\mathbf{y}$, an adversarial attacker finds an additive perturbation vector $\boldsymbol{\eta}$ that is imperceptible to the administrator of the target system, while maximizing the loss of the perturbed input $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta})$ as

$$\max_{\boldsymbol{\eta}} \mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) \quad \text{s.t.} \quad \boldsymbol{\eta} \text{ is imperceptible}\,.$$

In contrast with classification problems where maximum perturbations at the output might not change the class, adversarial instances maximize the output perturbation in regression problems.

As in (3.4), a constraint on the ℓ_p-norm of $\boldsymbol{\eta}$ models imperceptibility leading to the following formulation of the problem

$$\max_{\boldsymbol{\eta}} \|\mathbf{y} - f(\mathbf{x} + \boldsymbol{\eta})\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{3.10}$$

Consider the image colorization problem where the goal is to add proper coloring on top of gray scale images. In this problem, $f(\cdot)$ is the regression algorithm and assumed to be known however the ground truth colorization $\mathbf{y}$ is generally unknown. Without knowing $\mathbf{y}$, the optimization problem (3.10) is ill posed and cannot be solved in general.

There are some cases where the output $\mathbf{y}$ is known by the nature of the problem, for instance, when $f(\cdot)$ is an encoder-decoder pair as in autoencoders for which $\mathbf{y} = \mathbf{x}$.

Since the goal is to perturb the acting regression algorithm, we can assume that $\mathbf{y} \approx f(\mathbf{x})$ which means that the algorithm provides a good although not perfect approximation of the ground truth function. We use the formulation in (3.10) and discuss the implications of applying the approximation $\mathbf{y} \approx f(\mathbf{x})$ in later sections.

3.3.1. A Quadratic Programming Problem

In general $f(\mathbf{x})$ is a non-linear and non-convex function, so we have that $\mathcal{T}(\mathbf{x}, \cdot)$ is non-convex. Here again the perturbation analysis of $f(\cdot)$ can be used to relax (3.10) and to obtain a convex formulation of the adversarial problem. The first order perturbation analysis of $f(\mathbf{x})$ yields the approximation $f(\mathbf{x} + \boldsymbol{\eta}) \approx f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}$, where $\mathbf{J}_f(\cdot)$ is the Jacobian matrix of $f(\cdot)$. This approximation leads to the following convex approximation of $\mathcal{T}(\mathbf{x}, \cdot)$:

$$\begin{aligned}
&\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) \\
&\quad \approx \|\mathbf{y}\|_2^2 - 2\mathbf{y}^\top (f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}) + \|f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \\
&\quad = \|\mathbf{y}\|_2^2 - 2\mathbf{y}^\top f(\mathbf{x}) + \|f(\mathbf{x})\|_2^2 \\
&\qquad + 2\left(f(\mathbf{x}) - \mathbf{y}\right)^T \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta} + \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 .
\end{aligned}$$

Since the first three terms of this expression do not depend on $\boldsymbol{\eta}$, the optimization problem from (3.10) reduces to

$$\max_{\boldsymbol{\eta}} 2\left(f(\mathbf{x}) - \mathbf{y}\right)^T \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta} + \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon . \tag{3.11}$$

The above convex maximization problem is, in general, challenging and NP-hard. Nevertheless, since $\mathbf{y}$ is usually not known, we may use the assumption that $\mathbf{y} \approx f(\mathbf{x})$, which simplifies the problem to

$$\max_{\boldsymbol{\eta}} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon . \tag{3.12}$$

Although this problem is a convex quadratic maximization under an ℓ_p-norm constraint and in general challenging, it can be solved efficiently in some cases. For general p, the maximum value is indeed related to the operator norm of $\mathbf{J}_f(\mathbf{x})$ [46]. This norm is central in stability analysis of many signal processing algorithms (for instance see [47]). The operator norm of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ between ℓ_p and ℓ_q is defined as

$$\|\mathbf{A}\|_{p \to q} \triangleq \sup_{\|\mathbf{x}\|_p \leq 1} \|\mathbf{A}\mathbf{x}\|_q .$$

Using this notion, we can see that $\|\frac{\boldsymbol{\eta}}{\varepsilon}\|_p \leq 1$ leads to $\|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2 = \varepsilon \|\mathbf{J}_f(\mathbf{x})\frac{\boldsymbol{\eta}}{\varepsilon}\|_2 \leq \varepsilon \|\mathbf{J}_f(\mathbf{x})\|_{p \to 2}$. Therefore, the problem of finding a solution to (3.12) amounts to finding

the operator norm $\|\mathbf{J}_f(\mathbf{x})\|_{p\to 2}$. First observe that the maximum value is achieved on the border namely for $\|\boldsymbol{\eta}\|_p = \varepsilon$. In the case where $p = 2$, this problem has a closed-form solution. If $\mathbf{v}_{\max}$ is the unit ℓ_2-norm eigenvector corresponding to the maximum eigenvalue of $\mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x})$, then

$$\boldsymbol{\eta}^* = \pm\varepsilon\, \mathbf{v}_{\max} \tag{3.13}$$

solves the optimization problem. The maximum eigenvalue of $\mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x})$ corresponds to the square of the spectral norm $\|\mathbf{J}_f(\mathbf{x})\|_{2\to 2}$.

Another interesting case is $p = 1$. In general, the ℓ_1-norm is usually used as a regularization technique to promote sparsity. When the solution of a problem should satisfy a sparsity constraint, the direct introduction of this constraint into the optimization leads to NP-hardness of the problem. Instead the constraint is relaxed by adding ℓ_1-norm regularization. The adversarial perturbation designed in this way tends to have only a few non-zero entries. This corresponds to scenarios like single pixel attacks where only a few pixels are supposed to change. For this choice, we have

$$\|\mathbf{A}\|_{1\to 2} = \max_{k\in[n]} \|\mathbf{a}_k\|_2 \,,$$

where $\mathbf{a}_k$'s are the columns of $\mathbf{A}$. Therefore, if the columns of the Jacobian matrix are given by $\mathbf{J}_f(\mathbf{x}) = [\mathbf{j}_1 \dots \mathbf{j}_n]$, then

$$\|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2 \leq \varepsilon \max_{k\in[n]} \|\mathbf{j}_k\|_2 \,,$$

and the maximum is attained with

$$\boldsymbol{\eta}^* = \pm\varepsilon \mathbf{e}_{k^*} \quad \text{for} \quad k^* = \operatorname*{argmax}_{k\in[n]} \|\mathbf{j}_k\|_2 \,, \tag{3.14}$$

where the vector $\mathbf{e}_i$ is the i-th canonical vector. For the case of gray-scale images, where each pixel is represented by a single entry of $\mathbf{x}$, this constitutes a single pixel attack. Some additional constraints must be added in the case of RGB images, where each pixel is represented by a set of three values.

Finally, the case where the adversarial perturbation is bounded with the ℓ_∞-norm is also of particular interest. This bound guarantees that the noise entries have bounded values. The problem of designing adversarial noise corresponds to finding $\|\mathbf{J}_f(\mathbf{x})\|_{\infty\to 2}$. Unfortunately, this problem turns out to be NP-hard [48]. However, it is possible to approximate this norm using semi-definite programming as proposed in [49]. Semi-definite programming scales badly with input dimension in terms of computational complexity, namely $O(n^6)$ with n the underlying dimension, and therefore might not be suitable for fast generation of adversarial examples when the input dimension is very high. We address these problems later in Section 3.4, where we obtain fast approximate solutions for $\|\mathbf{J}_f(\mathbf{x})\|_{\infty\to 2}$ and single pixel attacks.

3.3.2. A Linear Programming Problem

The methods derived in Section 3.3.1 suffer from one main drawback, they require storing $\mathbf{J}_f(\mathbf{x}) \in \mathbb{R}^{k \times n}$ into memory. While this may be doable for some applications, it is not feasible for others. For example, if the target system is an autoencoder for RGB images with size 680×480, that is $n = k = 680 \cdot 480 \cdot 3 \approx 9 \cdot 10^5$, storing $\mathbf{J}_f(\mathbf{x}) \in \mathbb{R}^{9 \cdot 10^5 \times 9 \cdot 10^5}$ requires loading around $8 \cdot 10^{11}$ values into memory, which is in most cases not tractable. Note that, in order to solve (3.12) for $p = 2$, we would require computing the eigenvalue decomposition of $\mathbf{J}_f(\mathbf{x})^\top \mathbf{J}_f(\mathbf{x})$ as well. This motivates us to relax the problem into a linear programming problem as in Section 3.2, where $\mathbf{J}_f(\mathbf{x})$ is computed implicitly and we do not require to store it. To that end, we relax (3.10) by directly applying a first order approximation of $\mathcal{T}$, that is $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) \approx \mathcal{T}(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \mathbf{0})$. Using this approximation the problem from (3.10) is now simplified to

$$\max_{\boldsymbol{\eta}} \nabla \mathcal{T}(\mathbf{x}, \mathbf{0})^\top \boldsymbol{\eta} \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon \,, \tag{3.15}$$

where $\nabla \mathcal{T}(\mathbf{x}, \mathbf{0}) = -2\mathbf{J}_f(\mathbf{x})^T (\mathbf{y} - f(\mathbf{x}))$. Note that the attacks discussed in Section 3.2 for classification follow the same formulation with another choice of $\mathcal{T}(\mathbf{x}, \cdot)$. Therefore, the closed-form solution of (3.15) can be obtained from (3.5).

Unfortunately using $\mathbf{y} \approx f(\mathbf{x})$ yields zero gradient in (3.15), thus leaving this approximation useless for obtaining adversarial perturbations. This problem is tackled by taking the approximation around another random point $\widetilde{\boldsymbol{\eta}}$ within and $\widetilde{\varepsilon}$-ball radius from $\boldsymbol{\eta} = \mathbf{0}$ as in (3.9), with $\widetilde{\varepsilon} \leq \varepsilon$. As it was mentioned above, this dithering mechanism is also used in classification problems for instance in [24].

3.4. Single Subset Attacks

Another popular way of modeling undetectability in the field of image recognition, is by constraining the number of pixels that can be modified by the attacker. This gave birth to single and multiple pixel attacks. Note that, for the case of gray-scale images, the solutions obtained in (3.14) and (3.5) provide already single pixels attacks. This is not true for RGB images where each pixel is represented by a subset of three values. Since our analysis is not limited to image based systems, we refer to these type of attacks which target only a subset of entries as single subset attacks.

Since perturbations belong to $\mathbb{R}^n$, let us partition $[n] = \{1, \ldots, n\}$ into S possible subsets $\mathcal{S}_1, \ldots, \mathcal{S}_S$. The sets can in general have different cardinalities. However, we assume here that all of them have the same cardinality of $Z = n/S$, where $\mathcal{S}_s = \{i_s^1, \ldots, i_s^Z\} \subseteq [n]$. We define the mixed zero-$\mathcal{S}$ norm $\|\cdot\|_{0,\mathcal{S}}$ of a vector, for the partition $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_S\}$, as the number of subsets containing at least one index associated to a non-zero entry

of $\mathbf{x}$[2], that is

$$\|\mathbf{x}\|_{0,\mathcal{S}} = \sum_{i=1}^{S} \mathbb{1}_{\left(\|\mathbf{x}_{\mathcal{S}_i}\| \neq 0\right)} ,$$

where $\mathbf{x}_{\mathcal{S}_i}$ denotes the vector containing the entries of $\mathbf{x}$ with index belonging to $\mathcal{S}_i$. Therefore, $\|\boldsymbol{\eta}\|_{0,\mathcal{S}}$ counts the number of subsets modified by an attacker. To guarantee that only one subset is active, an additional constraint can be added to the optimization problem. This leads to the following formulation of the single subset attack for the regression problem

$$\max_{\boldsymbol{\eta}} \|\mathbf{y} - f(\mathbf{x}+\boldsymbol{\eta})\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon \, , \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1 \, . \tag{3.16}$$

A similar formulation holds as well for classification problems. The mixed norm $\|.\|_{0,\mathcal{S}}$ in widely used in signal processing and compressed sensing to promoting group sparsity [23].

3.4.1. Single Subset Attack for the Quadratic Problem

As in Section 3.3.1, the approximations $f(\mathbf{x}+\boldsymbol{\eta}) \approx f(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}$ and $\mathbf{y} \approx f(\mathbf{x})$ simplify the problem (3.16) to

$$\max_{\boldsymbol{\eta}} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon \, , \, \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1 \, . \tag{3.17}$$

As it was mentioned above, the problem is NP-hard without the mixed-norm constraint. We try to find an approximate solution to a simpler problem where only the set $\mathcal{S}_s = \{i_s^1, \ldots, i_s^Z\}$ is to be modified by the attacker for $s \in [S]$. Finding the perturbation on this set amounts to solving the following problem:

$$\boldsymbol{\eta}_s = \operatorname*{argmax}_{\boldsymbol{\eta}} \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2^2 \text{ s.t. } \|\boldsymbol{\eta}\|_\infty \leq \varepsilon \, , (\boldsymbol{\eta})_i = 0 \; \forall i \notin \mathcal{S}_s \, , \tag{3.18}$$

where $(\boldsymbol{\eta})_i$ denotes the i-th entry of $\boldsymbol{\eta}$. As discussed in Section 3.3.1, this problem is NP-hard. Since the maximization of a quadratic bowl over a box constraint lies in the corner points of the feasible set, we have

$$\boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \rho_{i_s^z}^* \mathbf{e}_{i_s^z}$$

with $\boldsymbol{\rho}_s^* \triangleq (\rho_{i_s^1}^*, \ldots, \rho_{i_s^Z}^*)^\top \in \{-1,+1\}^Z$. The optimization problem can be equivalently formulated as follows:

$$\begin{aligned} \boldsymbol{\rho}_s^* &= \operatorname*{argmax}_{\boldsymbol{\rho}_s \in \{-1,+1\}^Z} \left\| \mathbf{J}_f(\mathbf{x}) (\varepsilon \sum_{z=1}^{Z} \rho_{i_s^z} \mathbf{e}_{i_s^z}) \right\|_2^2 \\ &= \operatorname*{argmax}_{\boldsymbol{\rho}_s \in \{-1,+1\}^Z} \sum_{z=1}^{Z} \sum_{w=1}^{Z} \rho_{i_s^z} \rho_{i_s^w} \mathbf{j}_{i_s^z}^\top \mathbf{j}_{i_s^w} \, , \end{aligned}$$

[2]Similar to the so-called ℓ_0-norm, this is not a proper norm.

for $\boldsymbol{\rho}_s \triangleq (\rho_{i_s^1}, \ldots, \rho_{i_s^Z})^\top \in \{-1, +1\}^Z$ and $\mathbf{j}_k$ the k-th column of $\mathbf{J}_f(\mathbf{x})$. This problem is indeed related to the well known MaxCut problem introduced by [50]. The literature abounds with works on the MaxCut problem, the efficient solutions and their recovery guarantees. A common solution to this problem is a relaxation by a semi-definite programming problem. However, as we discussed semi-definite programming solvers scales badly with the input dimension. Therefore, in the spirit of obtaining fast and scalable approximate solutions, that can later be used to design adversarial perturbations through iterative approximations, we propose to obtain approximate solutions using a greedy approach. To that end, and without loss of generality, let us assume that for a given $\mathcal{S}_s$, the indices $i_s^1, \ldots, i_s^Z \in \mathcal{S}_s$ are sorted such that $\|\mathbf{j}_{i_s^1}\|_2 \geq \cdots \geq \|\mathbf{j}_{i_s^Z}\|_2$. An approximate solution for $\rho^*_{i_s^z}$ is calculated in a greedy manner by setting $\rho^*_{i_s^1} = 1$ and recursively calculating

$$\rho^*_{i_s^z} = \operatorname{sign}\left(\left(\sum_{j=1}^{z-1} \rho^*_{i_s^j} \mathbf{j}_{i_s^j}\right)^\top \mathbf{j}_{i_s^z}\right) \quad \forall\, z = 2, \ldots, Z\,. \tag{3.19}$$

As for greedy algorithms, this solution is fast, however, there is no optimality guarantee for it. For the case where $S = 1$ and $S = n$, the expression (3.19) is an approximate solution for (3.12) under the ℓ_∞-norm constraint on the perturbation (*i.e.*, $p = \infty$).

This method provides an approximate solution to the problem for a given choice of $\mathcal{S}_s$. The solution to (3.17) can then be obtained by solving the following problem:

$$\begin{aligned} &\boldsymbol{\eta}^* = \boldsymbol{\eta}_{s*}\,, \\ &\text{with } s^* = \operatorname*{argmax}_s \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}_s\|_2^2 \text{ and } \boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \rho^*_{i_s^z} \mathbf{e}_{i_s^z}\,. \end{aligned} \tag{3.20}$$

This is based on naive exhaustive search over the subsets which is tractable only when the number of subsets is small enough.

3.4.2. Single Subset Attack for the Linear Problem

Following the steps from Section 3.3.2, we make use of the approximation $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) \approx \mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta} - \widetilde{\boldsymbol{\eta}})^\top \nabla\mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}})$ which leads to the formulation of (3.16) as a linear programming problem

$$\max_{\boldsymbol{\eta}} \boldsymbol{\eta}^\top \nabla\mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,,\ \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,. \tag{3.21}$$

In the same manner as Section 3.4.1, for a given subset $\mathcal{S}_s$ we define $\boldsymbol{\eta}_s$ as in (3.18). For this linear problem that results in

$$\boldsymbol{\eta}_s = \operatorname*{argmax}_{\boldsymbol{\eta}} \nabla\mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}})^\top \boldsymbol{\eta} \text{ s.t. } \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,, (\boldsymbol{\eta})_{i_s^z} = 0\ \forall i_s^z \notin \mathcal{S}_s\,.$$

Type of Attack	Design Algorithm	Adversarial Perturbation	Exact Solution	Application
ℓ_2-constrained	(3.15)	(3.13)	Yes	Regression
ℓ_2-constrained	(3.12)	(3.5)	Yes	Classification/Regression
ℓ_∞-constrained	(3.15)	(3.20)	No	Regression
ℓ_∞-constrained	(3.12)	(3.6)	Yes	Classification/Regression
Single-subset	(3.17)	(3.20)	No	Regression
Single-subset	(3.21)	(3.22)	Yes	Classification/Regression

Table 3.1.: Summary of the obtained closed-form solutions for adversarial perturbations

Attack	Design Algorithm	Iterative	Dithering
FGSM [20]	(3.4) with cross-entropy loss	×	×
R-FGSM [30]	(3.4) with cross-entropy loss	×	✓
BIM [25]	(3.4) with cross-entropy loss	✓	×
PGD [24]	(3.4) with cross-entropy loss	✓	✓
DeepFool [22]	(3.7) with $p = 2$	✓	×
Ensemble [30]	(3.4) with combined cross-entropies	×	✓
Targeted	(3.4) with (3.23) loss	✓	✓
Algorithm 1	(3.4)	✓	✓

Table 3.2.: Summary of existing attacks in classification obtained from presented algorithms AGP.II

In contrast to the definition of $\boldsymbol{\eta}_s$ from (3.18), in this case we have a closed form solution for $\boldsymbol{\eta}_s$ as

$$\boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \operatorname{sign}((\nabla \mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}}))_{i_s^z}) \mathbf{e}_{i_s^z} ,$$

which implies that $\nabla \mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}})^\top \boldsymbol{\eta}_s = \sum_{z=1}^{Z} \left| (\nabla \mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}}))_{i_s^z} \right|$. Therefore, the linear problem for the single subset attack (3.21) has the closed form solution

$$\boldsymbol{\eta}^* = \boldsymbol{\eta}_{s*} , \text{ with } s^* = \operatorname*{argmax}_s \sum_{z=1}^{Z} \left| (\nabla \mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}}))_{i_s^z} \right| \tag{3.22}$$

and $\boldsymbol{\eta}_s = \varepsilon \sum_{z=1}^{Z} \operatorname{sign}((\nabla \mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}}))_{i_s^z}) \mathbf{e}_{i_s^z}$. This results are valid for classification as well when replacing $\mathcal{T}$ with $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) = -(f_{c(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq c(\mathbf{x})} f_l(\mathbf{x} + \boldsymbol{\eta}))$.

3.5. Iterative Versions of the Linear Problem

In the previous sections we have formulated several variations of the problem of generating adversarial perturbations. In the same spirit as DeepFool, we make use of the

obtained closed form solutions to design adversarial perturbations using iterative approximations. In Algorithm 1 an iterative method based on the linear problem (3.15) is introduced. This corresponds to a gradient ascent method for maximizing $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta})$ with a fixed number of iterations and steps of equal ℓ_p-norm. While generalizing the

Algorithm 1 Iterative extension for ℓ_p constrained methods.

input: $\mathbf{x}$, f, T, ε, $\widetilde{\varepsilon}_1, \ldots, \varepsilon_T$.
output: $\boldsymbol{\eta}^*$.
Initialize $\boldsymbol{\eta}_1 \leftarrow \mathbf{0}$.
for $t = 1, \ldots, T$ **do**
  $\widetilde{\boldsymbol{\eta}}_t \leftarrow \boldsymbol{\eta}_t + \text{random}(\widetilde{\varepsilon}_t)$
  $\boldsymbol{\eta}_t^* \leftarrow \text{argmax}_{\boldsymbol{\eta}} \boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}}_t)$ s.t. $\|\boldsymbol{\eta}\|_p \leq \varepsilon / T$ (Table 3.1)
  $\boldsymbol{\eta}_{t+1} \leftarrow \boldsymbol{\eta}_t + \boldsymbol{\eta}_t^*$
end for
return: $\boldsymbol{\eta}^* \leftarrow \boldsymbol{\eta}_T$

results for (3.15) into a gradient ascent method is trivial, the same is not true for the quadratic problem (3.12). The main reason for this is that, using the approximation $\mathbf{y} \approx f(\mathbf{x})$, we were able to simplify (3.11) into (3.12) since $\mathbf{y} - f(\mathbf{x}) \approx \mathbf{0}$. For an iterative version of this solution we must successively approximate $f(\cdot)$ around different points $\widetilde{\mathbf{x}}$, which leads to $\mathbf{y} - f(\widetilde{\mathbf{x}}) \neq \mathbf{0}$ even if $\mathbf{y} = f(\mathbf{x})$. We leave the task of investigating alternatives for designing iterative methods with the results for (3.12) for future works, and in Section 3.6 show that the non-iterative solutions for this method are still top performing.

Finally, replacing line 5 of Algorithm 1 with

$$\boldsymbol{\eta}_t^* \leftarrow \underset{\boldsymbol{\eta}}{\text{argmax}}\, \boldsymbol{\eta}^\top \nabla \mathcal{T}(\mathbf{x}, \widetilde{\boldsymbol{\eta}}_t) \text{ s.t. } \|\boldsymbol{\eta}\|_p \leq \varepsilon\,, \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1$$

leads to a multiple subset attack, since we modify the values of one subset at every iteration. At every iteration, we may exclude the previously modified subsets from $\mathcal{S}$ in order to ensure that a new subset is modified.

3.6. Experiments

In this section, the proposed methods are used to fool neural networks in classification and regression problems. But first, we summarize the presented algorithms and the relation with other existing attacks. The proposed attacks are summarized in Table 3.1. These attacks rely on perturbation analysis of learning algorithms and can be used for classification and regression tasks. The approach we chose to derive these algorithms is capable of rendering other existing methods by adjusting the choice of $\mathcal{T}(\mathbf{x}, \cdot)$ and other design parameters. This point has been discussed in Table 3.2. In that table we

also include the black-box ensemble attack of [30] and targeted attacks [21], [29], [34], [44], [52]. Consider first the ensemble attack of [30]. A black-box attack in nature, this attack does not have access to the target neural network and uses another function f, potentially similar to the neural network's function, hoping that the obtained adversarial example transfers to the unknown network. After choosing multiple similar neural networks, the attack adopts the loss function as the averaged sum of cross entropies of these networks. Once the loss function is fixed, the algorithm, similar to above methods, uses a combination of perturbation analysis with an optimization problem of type (3.4) or (3.7) to solve the problem. Targeted attacks are used when the goal is to generate adversarial examples that are classified by target system as belonging to some given target class $l \in [k]$. That corresponds to fixing the loss function to

$$\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) = f_{c(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - f_l(\mathbf{x} + \boldsymbol{\eta})\,. \tag{3.23}$$

After that, the process of finding adversarial examples leverages the same techniques, *i.e.*, perturbation analysis and convex optimization, discussed in this work. This is, however, only a side feature of our approach. Our proposed attacks are still of independent interest.

The goal of this section is twofold. First, we would like to examine the performance of the newly proposed attack in classification tasks, thereby showing the strength of our proposed method. Secondly we generate adversarial perturbations for various regression tasks which only received small attention in the literature. For this purpose we use the MNIST [53], CIFAR-10 [54], STL-10 and PASCAL VOC 2012 datasets.

3.6.1. Classification

As discussed in Section 3.2, the appropriate loss function $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta})$ for image classification tasks that should be used in (3.4) is given by (AGP). For this problem, $\|\boldsymbol{\eta}\|_\infty \leq \varepsilon$ is a common constraint that models the undetectability, for sufficiently small ε, of adversarial noise by an observer. However solving (3.4) involves finding the function $\mathcal{T}(\mathbf{x}, \mathbf{0})$ which is defined as the minimum of $k-1$ functions with k being the number of different classes. In large problems, this may significantly increase the computations required to fool one image. Therefore, we include a simplified version of this algorithm in our simulations. The non-iterative methods might not guarantee the fooling of the underlying network but on the other hand, the iterative methods might suffer from convergence problems.

To benchmark the proposed adversarial algorithms, we consider the following methods tested on the aforementioned datasets:

- **Algorithm 1**: This algorithm solves (3.4) with $\mathcal{T}(\mathbf{x}, \cdot)$ given by (AGP). Note that, for evaluating $\mathcal{T}$ at a given $\mathbf{x}$ one must search over all $l \neq c(\mathbf{x})$. This can be computationally expensive when the number of possible classes (*i.e.*, the number of possible values for l) is large. The ℓ_∞-norm is chosen for the constraint. Moreover,

an example of adversarial images obtained using this algorithm is shown in Figure 3.3.

- **Algorithm 1**-T: This is the iterative version of Algorithm 1 with T iterations. The adversarial perturbation is the sum of T perturbation vectors with ℓ_∞-norm of ε/T computed through T successive approximations.

- **Algorithm 2**: This algorithm approximates (AGP) with $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) \approx f_{c(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta})$, thus reducing the computation of $\mathcal{T}(\mathbf{x})$ when the number of classes is large. Note that we cannot use $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) < 0$ to guarantee that we have fooled the network. Nevertheless, the lower the value of $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta})$ the most likely it is that the network has been fooled. The same reasoning is valid for the FGSM algorithm.

- **FGSM**: This well-known method was proposed by [20] where $\mathcal{T}(\mathbf{x}, \boldsymbol{\eta})$ is replaced by the negative training loss for the input $\mathbf{x} + \boldsymbol{\eta}$. Usually the cross-entropy loss is used for this purpose. With the newly replaced function, (3.4) is solved for $p = \infty$.

- **PGD**: This method, given in [24], is the iterative version of FGSM ($T > 1$) with $\widetilde{\varepsilon}_1 = \varepsilon$ and $\widetilde{\varepsilon}_t = 0$ for all $t > 1$. It constitutes one of the state of the art attacks in the literature.

- **DeepFool**: This method was proposed in [22] and makes use of iterative approximations. Every iteration of DeepFool can be written within our approach by replacing $\mathcal{T}$ by

$$\mathcal{T}(\mathbf{x}, \boldsymbol{\eta}) = f_{c(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - f_{\widehat{l}}(\mathbf{x} + \boldsymbol{\eta}), \quad \text{where}$$
$$\widehat{l} = \underset{l \neq c(\mathbf{x})}{\operatorname{argmin}} \left\{ \frac{|f_{c(\mathbf{x})}(\mathbf{x}) - f_l(\mathbf{x})|}{\|\nabla f_{c(\mathbf{x})}(\mathbf{x}) - \nabla f_l(\mathbf{x})\|_q} \right\}.$$

  The adversarial perturbations are computed using $p = \infty$, thus $q = 1$, with a maximum of 50 iterations. These parameters were taken from [22]. Note that $\widehat{l}$ is chosen to minimize the robustness $\widehat{\rho}_1(f)$ for $\mathcal{S} = \{\mathbf{x}\}$.

- **Random**: For benchmarking purposes, we also consider random perturbations with independent Bernoulli distributed entries with $\mathbb{P}(\varepsilon) = \mathbb{P}(-\varepsilon) = \frac{1}{2}$. This helps to demarcate the essential difference of adversarial and random perturbations.

The above methods are tested on the following deep neural network architectures:

- **MNIST** : A fully connected network with two hidden layers of size 150 and 100 respectively, as well as the LeNet-5 architecture [55].

- **CIFAR-10** : The Network In Network (NIN) architecture [56], and a 40 layer DenseNet [57].

	Test error	$\widehat{\rho}_1(f)$ [22]	$\widehat{\rho}_2(f)$ (ours)	fooled >99%
FCNN (MNIST)	1.7%	0.036	0.034	ε =0.076
LeNet-5 (MNIST)	0.9%	**0.077**	**0.061**	ε =**0.164**
NIN (CIFAR-10)	13.8%	**0.012**	**0.004**	ε =**0.018**
DenseNet (CIFAR-10)	5.2%	0.006	0.002	ε =0.010

Table 3.3.: Robustness measures for different classifiers

As a performance measure, we use the *fooling ratio* defined in [22] as the percentage of correctly classified images that are missclassified when adversarial perturbations are applied. Of course, the fooling ratio depends on the constraint on the norm of adversarial examples. Therefore, in Figure 3.1 we observe the fooling ratio for different values of ε on the aforementioned neural networks. As expected, the increased computational complexity of iterative methods such as DeepFool and Algorithm 1-T translates into increased performance with respect to non-iterative methods. Nevertheless, as shown in Figures 3.1(a) and (c), the performance gap between iterative and non-iterative algorithms is not always significant. For the case of iterative algorithms, the proposed Algorithm 1-T outperforms DeepFool and PGD. The same holds true for Algorithm 1 with respect to other non-iterative methods such as FGSM, while Algorithm 2 obtains top performance with respect to FGSM. However, note that adversarial training using PGD is the state of the art defense against adversarial examples, thus PGD may still be a better choice than Algorithm 1-T for adversarial training. Some adversarial examples obtained using Algorithm 1-T are shown in Figure 3.2. Finally, we measure the robustness of different networks using $\widehat{\rho}_1(f)$ and $\widehat{\rho}_2(f)$, with $p = \infty$. We also include the minimum ε, such that DeepFool obtains a fooling ratio greater than 99%, as a performance measure as well. These results are summarized in Table 3.3, where we obtain coherent results between the 3 measures.

3.6.2. Regression

For the sake of clarity we use the following notation to distinguish between the different regression methods used in this section:

- **quadratic-ℓ_p**: This algorithm computes adversarial perturbations by solving the quadratic problem (3.12) under the ℓ_p-norm constraint.
- **linear-ℓ_p-T**: Similarly, this attack refers to Algorithm 1 with T iterations and the ℓ_p-norm constraint.
- **linear-pixel-T**: Since the experiments carried out in this section are exclusively image based, we use this notation to refer to the multiple subset attack with $\|\boldsymbol{\eta}\|_{0,\mathcal{S}} = T$.

- **random-ℓ_p**: Similarly to [22], we show the validity of our attacks by comparing their performance against appropriate types of random noise. For $p = 2$, the random perturbation is computed as $\boldsymbol{\eta} = \varepsilon\, \mathbf{w}/\|\mathbf{w}\|_2$, where the entries of $\mathbf{w}$ are independently drawn from a Gaussian distribution. For $p = \infty$ the random perturbation $\boldsymbol{\eta}$ has independent Bernoulli distributed entries with parameter 1/2.
- **random-pixel-T**: Random attacks are considered as well for multiple subset attacks with $p = \infty$. The random perturbations are added to only T randomly chosen pixels, while the other pixels are untouched.

The random attacks are used for benchmarking purposes. The algorithms behind our proposed attacks, *i.e.*, the first three attacks, are summarized in Table 3.1. Since the aim of the proposed attacks is to maximize the MSE of the target system, we use the Peak Signal to Noise Ratio, which is a common measure for image quality and is defined as PSNR = (maximum pixel value)^2/MSE, as the performance metric.

For our experiments we use the MNIST, CIFAR-10 and STL-10 and PASCAL VOC 2012 datasets. A different neural network is trained for each of these datasets. As in [38], we also consider autoencoders. For MNIST and CIFAR-10 we have trained fully connected autoencoders with 96% and 50% compression rates respectively. Next, we train the image colorization architecture of [58] for the STL-10 dataset. Finally, we use the YOLO architecture for object detection of [51]. The convolutional layers of the proposed network are trained on the ImageNet dataset with 1000 classes. The experiments are run on the PASCAL VOC 2012 dataset [59], which is a common dataset for object detection tasks.

Different example images obtained from applying the proposed methods on these networks are shown in Figure 3.3. For instance, in Figure 3.3(a) we observe that the autoencoder trained on MNIST is able to denoise random perturbation correctly but fails to do so with adversarial perturbations obtained using the quadratic-ℓ_∞ method. Similarly, in Figure 3.3(b), the random-pixel-100 algorithm distorts the output significantly more than its random counterpart. These two experiments align with the observation of [38] that autoencoders tend to be more robust to adversarial attacks than deep neural networks used for classification. The deep neural network trained for colorization is highly sensitive to adversarial perturbations as illustrated in Figure 3.3(c), where the original and adversarial images are nearly identical.

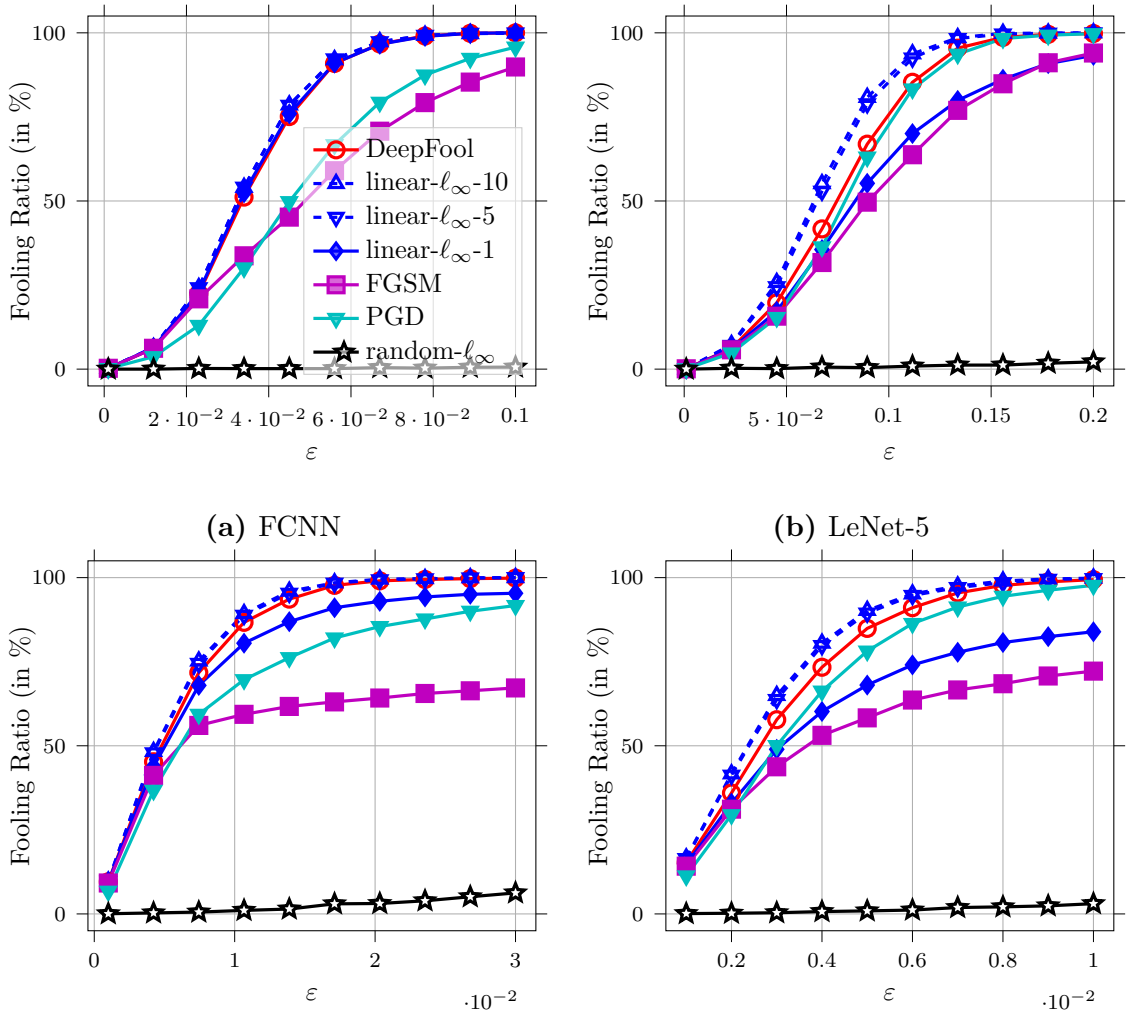
While the results shown in Figure 3.3 are for some particular images, in Figure 3.4 we measure the performance of different adversarial attacks using the average output PSNR over 20 randomly selected images from the corresponding datasets. In Figures 3.4(a) and 3.4(b) we observe how computing adversarial perturbations through successive linearizations improves the performance. This behavior is more pronounced in Figure 3.4(d), where iterative linearizations are responsible for more than 10 dB of output PSNR reduction. Note that, in Figures 3.4(a) and 3.4(b) the non-iterative quadratic-ℓ_p algorithm performs competitively, even when compared to iterative methods. In Figure 3.4(b) we observe that the autoencoder trained on CIFAR-10 is robust

to single pixel attacks. However, an important degradation of the systems performance, with respect to random noise, can be obtained through adversarial perturbations in the 100 pixels attack ($\approx 9.7\%$ of the total number of pixels). Moreover, in Figure 3.4(d), we can clearly observe the instability of the image colorization network to adversarial attacks. Finally, object detection tasks can be treated as regression problems, since the predicted location and size of the detected object can be approximated as continuous variables. Therefore, we can observe in Figure 3.5(a) how the performance of YOLO at detecting objects is severely degraded when adding adversarial perturbations. The loss function used in this experiment is the same used in [51], which accounts for the correct labeling of the detected objects as well as the correct placement of the boxes surrounding them. Some concrete example images from the PASCAL VOC 2012 dataset are shown in Figure 3.5(b). These experiments show that, even though autoencoders are somehow robust to adversarial noise, this may not be true for deep neural networks in other regression problems.

3.7. Outlook

The perturbation analysis of different learning algorithms underlies many attacks. In this work, the generation of adversarial examples is pursued by applying perturbation analysis to a fairly general problem and is formulated as a convex program. The other techniques like iterative methods and randomization of instances can be additionally considered. We used this generic approach to propose new attacks for classification and regression problems under various desirable constraints. This includes in particular single-pixel and single-subset attacks. Through multiple examples, regression problems are shown to be equally vulnerable to adversarial perturbations. Our new attacks on classification functions are benchmarked through empirical simulations of the fooling ratio against the well-known FGSM, DeepFool, and PGD methods. For regression tasks, three use cases are considered, namely, autoencoders, images colorization and object detection algorithms.

For classification tasks, the adversarial vulnerability is directly related to the properties of classification margin. Regression algorithms, however, do not dispose such a notion. It is an interesting research question to investigate the vulnerabilities of these algorithms.



(a) FCNN **(b)** LeNet-5

(c) NIN **(d)** DenseNet

Figure 3.1.: (a) and (b): Fooling ratio of the adversarial samples for different values of ε on the MNIST test dataset. (c) and (d): Fooling ratio of the adversarial samples for different values of ε on the CIFAR-10 test datasets.

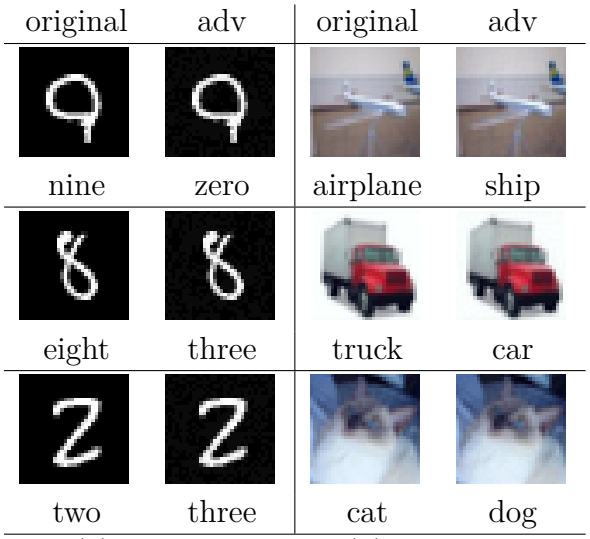
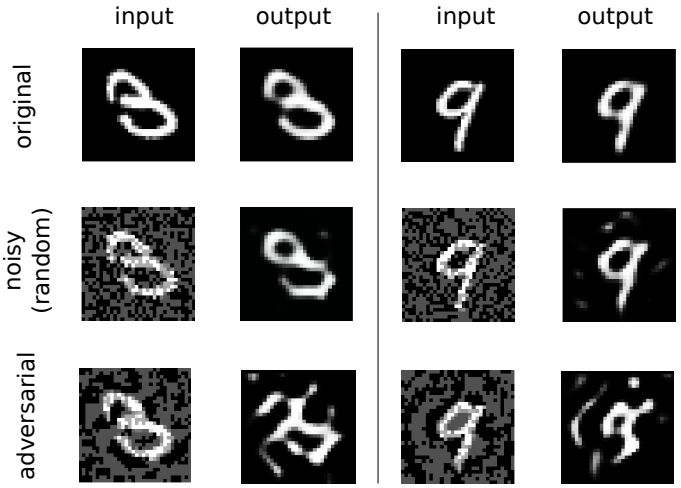
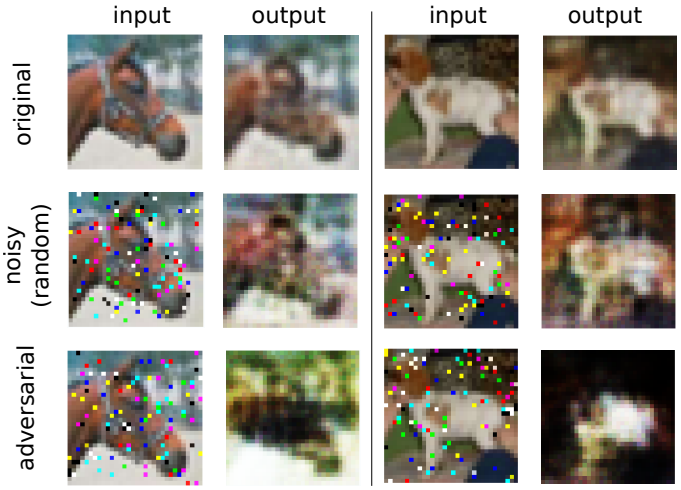


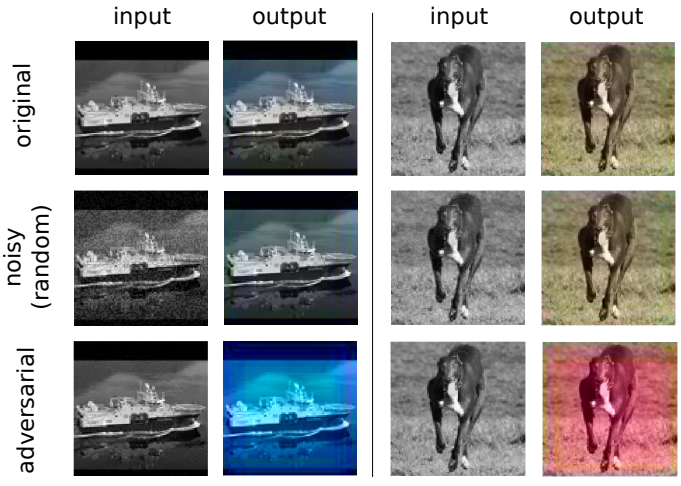
Figure 3.2.: Examples of correctly classified images that are misclassified when adversarial noise is added using Algorithm 1. The LeNet-5 architecture is used in the MNIST examples, while DenseNet is used in the CIFAR-10 example.



(a) Autoencoder (96% compression)



(b) Autoencoder (50% compression)



(c) Image Colorization

Figure 3.3.: Adversarial examples for (a): MNIST autoencoder obtained using quadratic-ℓ_∞, (b): CIFAR-10 autoencoder obtained using linear-pixel-100, (c): STL-10 colorization network obtained using linear-ℓ_∞-20.

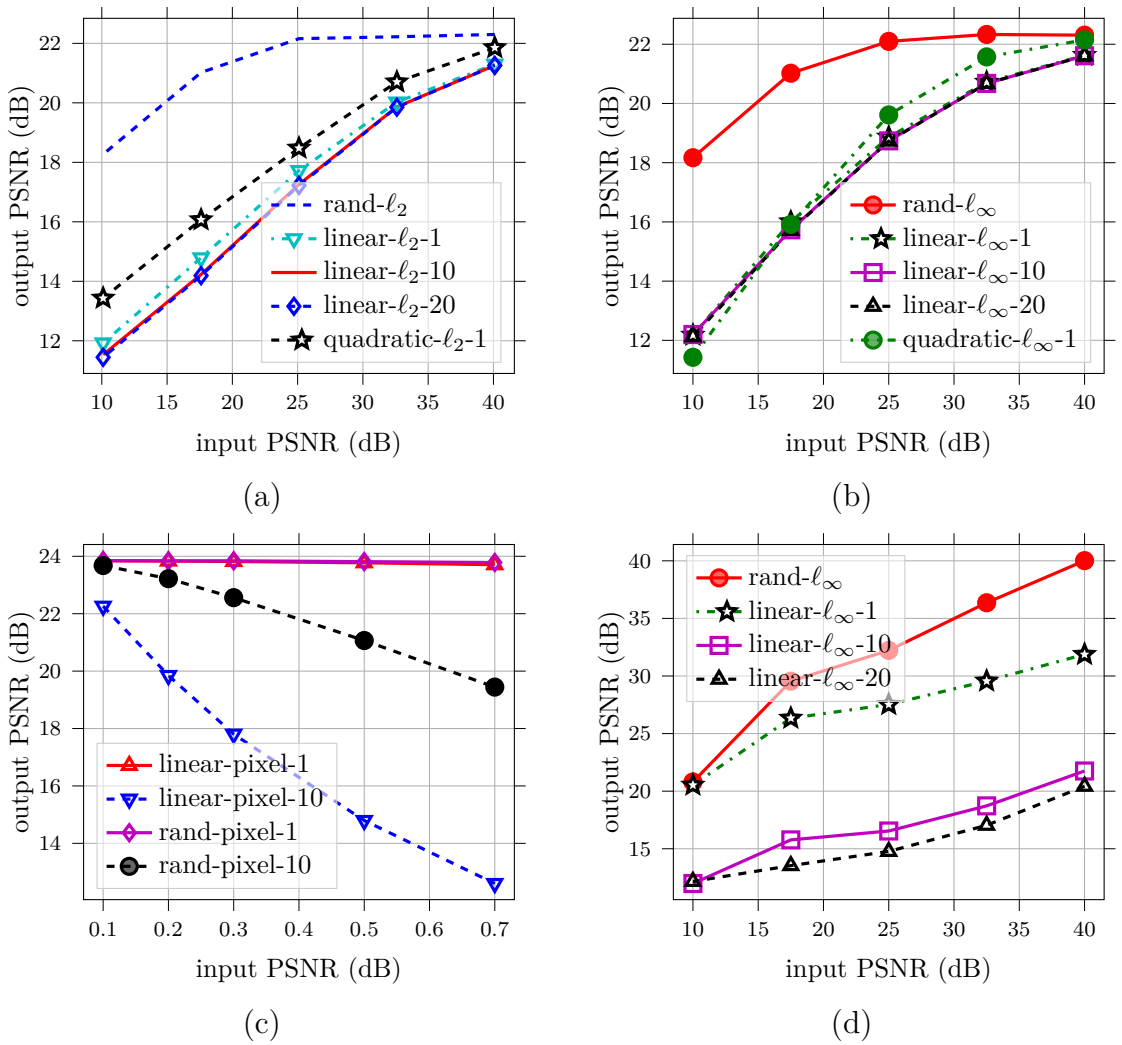
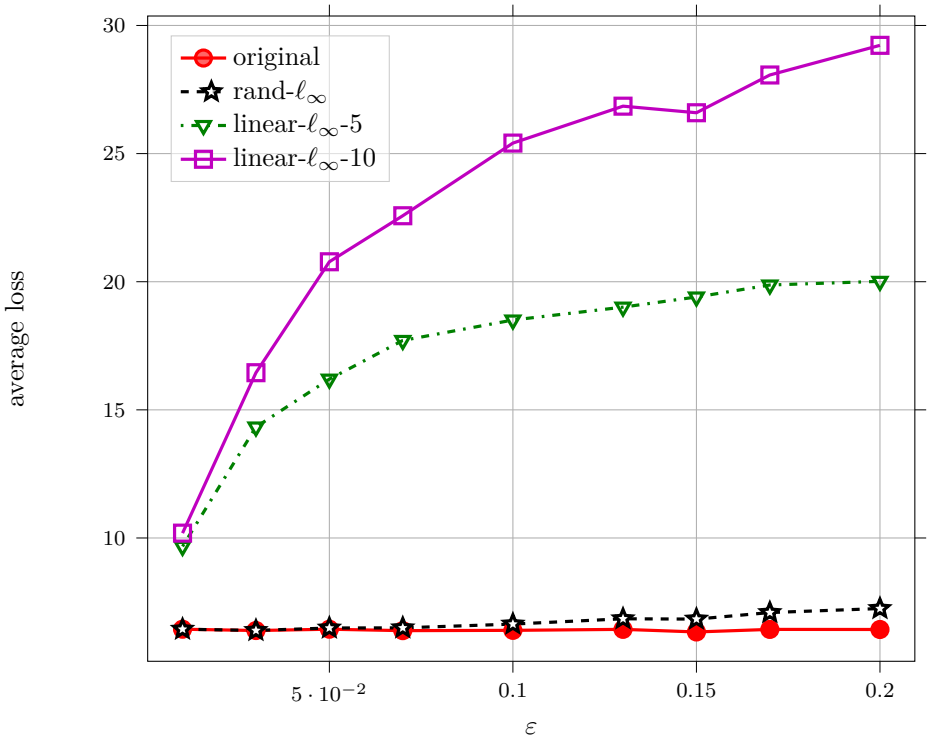
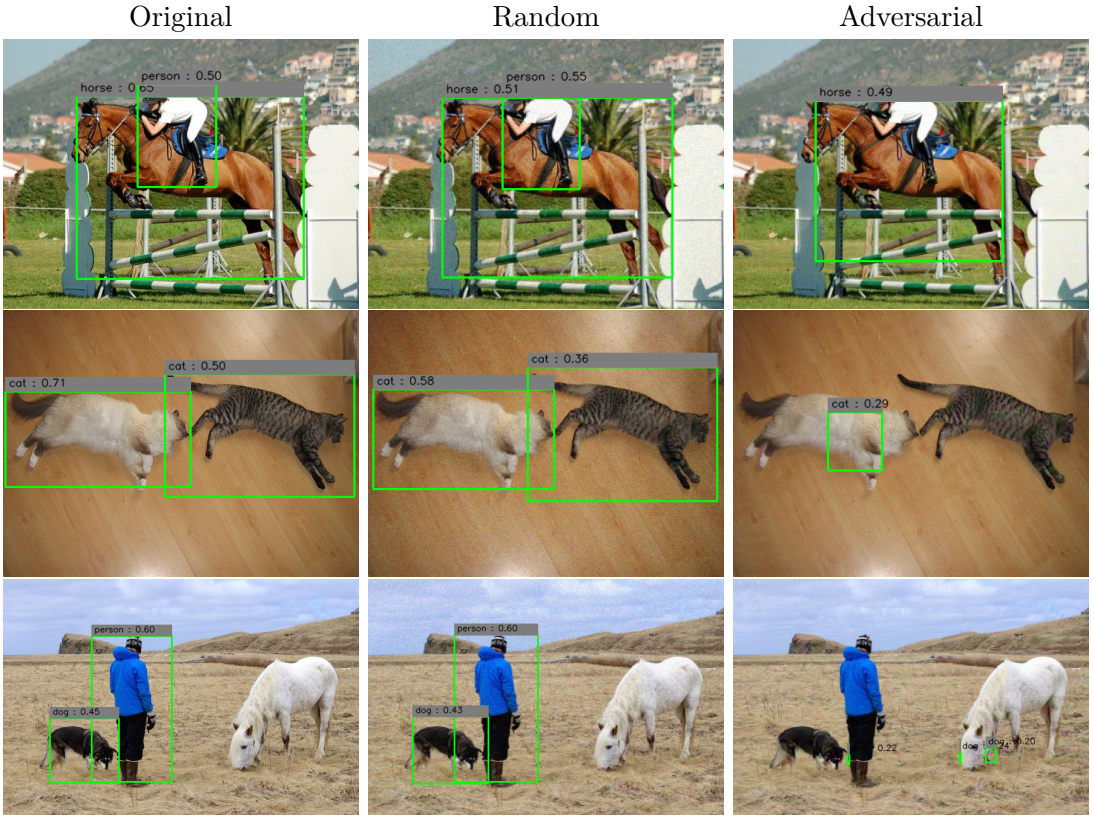


Figure 3.4.: Output PSNR for (a): MNIST autoencoder under ℓ_2-norm constraint, (b): MNIST autoencoder under ℓ_∞-norm constraint, (c): CIFAR-10 autoencoder under multiple pixel attacks, (d): STL-10 colorization network under ℓ_∞-norm constraint.



(a)



(b)

Figure 3.5.: The YOLO architecture under ℓ_∞-norm constraint in the PASCAL VOC 2012 dataset. (a) Output loss, defined as in [51]. (b) Adversarial examples obtained using linear-ℓ_∞-10.

4

On the Effect of Low-Rank Weights on Adversarial Robustness

Recently, there has been an abundance of works on designing DNNs that are robust to adversarial examples. In particular, a central question is which features of DNNs influence adversarial robustness and, therefore, can be used to design robust DNNs. In this chapter, this problem is studied through the lens of compression which is captured by the low-rank structure of weight matrices. It is first shown that adversarial training tends to promote simultaneously low-rank and sparse structure in the weight matrices of neural networks. This is measured through the notions of effective rank and effective sparsity. In the reverse direction, when the low rank structure is promoted by nuclear norm regularization and combined with sparsity inducing regularizations, neural networks show significantly improved adversarial robustness. The effect of nuclear norm regularization on adversarial robustness is paramount when it is applied to convolutional neural networks. Although still not competing with adversarial training, this result contributes to understanding the key properties of robust classifiers.

4.1. Related Work

Different hypotheses have been made about the existence of adversarial examples. In [20], it is hypothesized that DNNs are particularly vulnerable to adversarial examples

because of their too linear decision boundary in the vicinity around the data points together with the assumption of sufficiently large dimensionality of the problem. Tanay & Griffin [39] showed that it is possible to train linear classifiers that are resistant to adversarial attacks which stands in contrast to the linearity hypothesis. Moreover, it is exemplified that high dimensional problems are not necessarily more sensitive to adversarial examples. Further, Sabour *et al.* [60] manipulated deep representations instead of the input and argued that the linearity hypothesis is not sufficient to explain this type of attack. Regarding the observed transferability of adversarial examples across different DNN architectures, Szegedy *et al.* [6] and Tràmer *et al.* [61] proposed a method for estimating the dimensionality of the space of adversarial examples. The authors showed that adversarial examples span a contiguous subspace of large dimensionality. In addition, such subspaces intersect for different DNNs that have common adversarial examples. Fawzi *et al.* [62] proposed the low flexibility of DNNs, compared to the difficulty of the classification task, as a reason for the existence of adversarial examples. However, on subsequent work [7] the authors argued that the flatness of the decision boundary is a reason for the existence of adversarial examples. Another perspective was proposed by Tanay *et al.* [39] with the boundary tilting mechanism. It was argued that adversarial examples exist when the decision boundary lies close to the sub-manifold of sampled data. The authors introduced the notion of adversarial strength which refers to the deviation angle between the target classifier and the nearest centroid classifier. It was shown that the adversarial strength can be arbitrarily increased, independently of the classifier's accuracy, by tilting the boundary. Rozsa *et al.* [63] give another explanation, arguing that over the course of training the correctly classified samples do not have a significant impact on shaping the decision boundary and eventually remain close to it. This phenomenon is called evolutionary stalling. Further work of Rozsa *et al.* [64] performed an empirical study of the correlation between robustness and accuracy by attacking different state-of-the-art DNNs. Their results suggest that higher accuracy DNNs are more sensitive to adversarial attacks than lower accuracy ones. Regarding universal adversarial examples, Moosavi-Dezfooli *et al.* [31] showed empirically and formally [65] that their existence is partly caused by the correlation between the normals to the decision boundary in the vicinity of natural images, *i.e.*, these normals span a low dimensional space. It was observed that, in such subspace, the decision boundary is positively curved in the vicinity of natural images.

Kurakin *et al.* [66] and Madry *et al.* [24] provided evidence that increasing the model capacity alone can help to induce DNNs to be more robust against adversarial attacks. Additionally, it was observed that robust models (obtained through adversarial training) exhibit rather sparse weights compared to unrobust ones. Madry *et al.* [24] compared the weights of a CNN that is trained naturally, with one trained adversarially, on the MNIST dataset [53]. After training, it is observed that the adversarially trained CNN has more sparse weights in the first two convolutional filters than the naturally trained one. Moreover, it has been proposed that producing sparse input representations improves the robustness of linear and deeper binary classifiers [67], [68]. Guo *et al.* [69] conducted a more detailed study of the relation between sparsity and robustness of

DNNs. It was shown formally and empirically that more sparse DNNs are more robust against adversarial attacks, up to a certain limit where robustness starts decreasing again. In contrast to that, Guo *et al.* [70] conducted an empirical study and came to the conclusion that there is a trade-off between weight pruning and robustness. Alemi *et al.* [71] proposed the variational information bottleneck (VIB) method that aims at learning an encoding that is maximally expressive about the input but maximally compressive about the representation. The authors showed that models trained with the VIB objective have improved robustness. Recently, Sanyal *et al.* [72] provided evidence that adversarial robustness can be improved by encouraging the learned representations to lie in a low-rank subspace.

4.1.1. Our Contributions

We have discussed many different hypotheses about the nature of adversarial examples which, in fact, do not perfectly match. For this reason, research on the nature of adversarial examples is an active area. Inspired by recent works [24], [69], [72], in this chapter we suggest that one should design robust DNNs considering the effective sparsity and the effective rank of the weights. We observe that adversarial training induces low-rankness and sparsity on the weights of DNNs. So far, these two properties have not been pointed out together in the context of adversarial robustness of DNNs. We are able to substantially improve adversarial robustness by simultaneously promoting low-rank and sparse weights using different regularization techniques. In addition, we raise an open question whether there is an optimal combination of sparsity and low-rankness of the weights that can further improve robustness. Our experiments suggest that simultaneous sparsity and low-rankness of the weight matrices play a significant role in robustness, but do not fully explain the success of adversarial training.

4.2. Preliminaries

Recall Definition 2.2.1, where a classifier is defined through its score function f and classifier function c. In that chapter (*i.e.*, Chapter 2.2.1), the score function of a neural network was introduced in (2.4). Such expression can be written in a recursive manner through the intermediate variables $\mathbf{x}^0, \dots, \mathbf{x}^d$, that is

$$\mathbf{x}^i := \phi(\boldsymbol{W}^{i^\top} \mathbf{x}^{i-1}), \quad \text{for all} \quad i = 1, \dots, d, \tag{4.1}$$

where $\mathbf{x}^0 := \mathbf{x}$ and $f(\mathbf{x}) := \mathbf{x}^d$. Using this notation, the vectors $\mathbf{x}^1, \dots, \mathbf{x}^d$ are known as hidden representations. Moreover, each layer is represented by its own weight matrix $\boldsymbol{W}^i$, with appropriate dimensions, that defines the mapping from $\mathbf{x}^{i-1}$ to $\mathbf{x}^i$. Note that, the dimensionality of the hidden representations are parameters that can be chosen arbitrarily. In this chapter, we assume these parameters to be chosen beforehand and to remain unchanged during the course of training.

The weight matrices resulting from standard training with gradient-based methods contains dense entries, which are never exactly zero. The same can be said about the singular values of these matrices. For these reasons, the standard notions of sparsity and low-rankness are not useful to study such matrices. However, we may use relaxed versions of these notions, which allow for measuring approximate (or effective) sparsity and low-rankness. We start by defining the effective sparsity of a matrix $\boldsymbol{W}$ as

$$\overline{s}(\boldsymbol{W}) = \frac{\|\mathrm{vec}\,(\boldsymbol{W})\|_1}{\|\mathrm{vec}\,(\boldsymbol{W})\|_2} = \frac{\|\boldsymbol{W}\|_{1,1}}{\|\boldsymbol{W}\|_{\mathrm{F}}}\,. \tag{4.2}$$

The definition applies to vectors as well with $\overline{s}(\mathbf{x}) = \|\mathbf{x}\|_1 / \|\mathbf{x}\|_2$. Note that if a vector $\mathbf{x}$ is s-sparse, *i.e.*, has only s non-zero values, we have $\|\mathbf{x}\|_1 \leq \sqrt{s}\,\|\mathbf{x}\|_2$. In that sense, this notion is commonly used in compressed sensing as an extension of standard sparsity [73]. Similarly, the effective sparsity of the vector of singular values of a matrix $\boldsymbol{W}$ is called the effective rank of such matrix, that is

$$\overline{r}(\boldsymbol{W}) = \frac{\|\sigma(\boldsymbol{W})\|_1}{\|\sigma(\boldsymbol{W})\|_2} = \frac{\|\boldsymbol{W}\|_*}{\|\boldsymbol{W}\|_{\mathrm{F}}}\,. \tag{4.3}$$

which is a continuous relaxation of the notion of rank.

These notions can be seen as measures of complexity for weight matrices, since they quantify low-complexity structures such as sparsity and low-rankness. A weight matrix with these low-complexity characteristics can be seen as a linear transformation that maps a hidden representation into a low-complexity space. Note that some information about the original hidden representation may be lost during such transformation. In other words, we can see every layer of a neural network as a lossy compression encoding from one hidden representation to another. Therefore, information theoretic notions, such as mutual information, provide yet another measure of complexity, since they measure how much information is compressed at every stage of a neural network. A well known study into this direction was initiated by Tishby *et al.* [74], where the authors estimated information theoretic quantities during their training phase of neural networks. In that work, the authors proposed that the hidden representations $\mathbf{x}^i$ of a DNN form a Markov Chain, that is

$$(\mathbf{x}, y) \to \mathbf{x}^1 \to \ldots \to \mathbf{x}^{d-1} \to f(\mathbf{x})\,. \tag{4.4}$$

Such successive representations are studied with the notion of mutual information, which quantifies how much information about one random variable can be obtained if the other random variable is observed. We use the notation $I(\mathbf{a};\mathbf{b})$ for the mutual information between two random variables $\mathbf{a}, \mathbf{b}$, as introduced in Chapter 2.2.3. Using this quantity, the authors studied the behavior of the so called information plane which is the 2-dimensional plane of values between successive representations and the input $I(\mathbf{x}^i;\mathbf{x})$ and the output $I(\mathbf{x}^i;y)$ of DNNs during training. In this context, the hidden representation $\mathbf{x}^i$ is seen as a compressed version of $\mathbf{x}$, where information is lost during compression. Then, $I(\mathbf{x}^i;\mathbf{x})$ quantifies the amount of information about $\mathbf{x}$ that is

contained in $\mathbf{x}^i$. Even though this chapter focuses on the low-rankness and sparsity of the weights to quantify compression, we show experimentally that our findings align with the information theoretic notion of compression, which is measured using mutual information.

4.3. Enhancing the Robustness of Sparse Linear Classifiers through Compression

Consider the case of linear binary classifiers as an example. For convenience, in this section we deviate from the system model in Chapter 2.2.1. Instead let us redefine the score function f and classifier function c, from Definition 2.2.1, as

$$f(\mathbf{x}) := \mathbf{w}^\top \mathbf{x} \qquad \text{and} \qquad c(\mathbf{x}) := \operatorname{sign}\left(f(\mathbf{x})\right) . \tag{4.5}$$

In this manner, the set of possible labels is now $\mathcal{Y} = \{-1, +1\}$ instead of $\{1, 2\}$, while the set of possible inputs remains being $\mathcal{X} \subseteq \mathbb{R}^n$ as in Chapter 2.2.1. Using this formulation, the expected error (also known risk) of this linear classifier is given by

$$L_0(f_\mathbf{w}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathbb{1}_{\left(\mathbf{w}^\top\mathbf{x}\cdot y<0\right)}\right] = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbf{w}^\top\mathbf{x}\cdot y < 0] ,$$

which is equivalent to $L_0(f_\mathbf{w}) = \mathbb{P}[c(\mathbf{x}) \neq y]$. As introduced in Chapter 3, an ℓ_∞ adversarial attack against this classifier is obtained by applying input perturbations $\boldsymbol{\eta}$ with ℓ_∞-norm bounded by some $\varepsilon > 0$. Then, let us define the adversarial risk as

$$L_0^\varepsilon(f_\mathbf{w}) = \mathbb{P}\left[\min_{\|\boldsymbol{\eta}\|_\infty \leq \varepsilon} \mathbf{w}^\top(\mathbf{x}+\boldsymbol{\eta})\cdot y < 0\right] .$$

In other words, $L_0^\varepsilon(f_\mathbf{w})$ is the error probability of this classifier under worst case input perturbations, with ℓ_∞-norm bounded by ε. The goal of a robust classifier is to minimize this error by an appropriate choice of $\mathbf{w}$. Formally, the adversarial accuracy $1 - L_0^\varepsilon(f_\mathbf{w})$ measures the robustness $f_\mathbf{w}$.

A typical choice of $\boldsymbol{\eta}$ is given by the FGSM, described in Table 3.2, for which $\boldsymbol{\eta} = -y \cdot \varepsilon \operatorname{sign}(\mathbf{w})$. Note that, in this linear setup, this perturbation corresponds to the best possible ℓ_∞ attack. It was previously discussed in [69] that the robustness of linear classifiers is upper bounded by $1/\left\|\mathbf{w}\right\|_1$. Since the ℓ_1-norm is widely used as a sparsity promoting regularization, the authors concluded that the sparsity of $\mathbf{w}$ contributes to the robustness of classifiers against ℓ_∞ attacks. As a matter of fact, it can be seen that the robustness to attacks with a bounded ℓ_p-norm is related to having a small corresponding dual norm of $\mathbf{w}$. In this chapter, we show that the robustness can be improved if the sparsity is complemented with an adequate dimensionality reduction. Although only the example of binary linear classifiers is considered, the result can be

also extended to the multi-class setup similar to [69], as discussed in the following section.

It is often observed in machine learning problems that high dimensional inputs lie in low dimensional manifolds. In this section, we mimic that situation by assuming that any input $\mathbf{x} \in \mathcal{X}$ lies in an r-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$ spanned by r orthonormal basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_r$ with $r < n$. Nevertheless, the data distribution $\mathcal{D}$ remains unknown. Now, let us apply an intermediate linear transformation $\mathbf{Q}$ on the data, in order to reduce its dimensionality, based on the available training samples. The same classifier c is now applied to $\mathbf{Qx}$, instead of $\mathbf{x}$. Note that, since our linear classifier is given by (4.5), this compression step is equivalent to replacing the weight vector $\mathbf{w}$ with $\mathbf{Qw}$. The main question is whether this compression step affects adversarial robustness. Intuitively, an ℓ_∞ attack allows for ε perturbation of each entry which can translate each data point by an Euclidean distance of $\varepsilon\sqrt{n}$ in the space. From this point of view, these ℓ_∞-norm attacks can severely perturb high-dimensional inputs $\mathbf{x} \in \mathbb{R}^n$. Therefore, reducing their effective dimension seems to be favorable for increasing adversarial robustness. A natural choice of $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is the orthogonal projection matrix onto the data subspace $\mathcal{V}$, given by

$$\mathbf{Q} = \sum_{i=1}^{r} \mathbf{v}_i \mathbf{v}_i^\top ,$$

since it compresses the inputs without loosing relevant information, *i.e.*, denoising it. The following theorem shows that, if the transformation $\mathbf{w} \to \mathbf{Qw}$ manages to preserve the effective sparsity of the weight vector $\mathbf{w}$, this projection can indeed improve the robustness of the classifier without compromising the overall accuracy.

Theorem 4.3.1. *For a binary classification task, suppose that the data samples belong to r-dimensional subspace $\mathcal{V}$ and $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is the orthogonal projection onto this subspace. Then, for any given binary linear classifiers with parameters $\mathbf{w}$, if the effective sparsity of $\mathbf{w}$ after projection satisfies $\overline{s}(\mathbf{Qw}) \leq \overline{s}(\mathbf{w})$, then*

- *The accuracy remains unchanged, that is*

$$L_0(f_{\mathbf{w}}) = L_0(f_{\mathbf{Qw}}) .$$

- *The adversarial robustness against ℓ_∞ attacks is either improved or unchanged, that is*

$$L_0^\varepsilon(f_{\mathbf{Qw}}) \leq L_0^\varepsilon(f_{\mathbf{w}}) .$$

Remark 4.3.2. *Before stating the proof, some remarks are in order. First of all, note that the new classifier applies to the projected vectors $\mathbf{Qx}$. Since $\mathbf{Q}$ is symmetric, this amounts to a new binary classifier with parameters $\mathbf{Qw}$. As it will be seen from the proof, if the vector $\mathbf{w}$ already belongs to the data subspace $\mathcal{V}$, that is, the discriminating hyperplane is orthogonal to $\mathcal{V}$, then the adversarial robustness remains unchanged. However, this is highly unlikely for most datasets that are noisy and therefore difficult to find exactly this hyperplane among many choices.*

Proof (Theorem 4.3.1). The optimal ℓ_∞ attack for a vector $\mathbf{x}$ with label y is given by $\boldsymbol{\eta} = -y \cdot \varepsilon \operatorname{sign}(\mathbf{w})$. Therefore, the adversarial risk is simplified to

$$\begin{aligned} L_0^\varepsilon(f_\mathbf{w}) &= \mathbb{P}(\mathbf{w}^\top(\mathbf{x}+\boldsymbol{\eta}) \cdot y < 0) \\ &= \mathbb{P}(y \cdot \mathbf{w}^\top \mathbf{x} < \varepsilon \left\|\mathbf{w}\right\|_1) \\ &= L_0(f_\mathbf{w}) \mathbb{P}(|\mathbf{w}^\top \mathbf{x}| < \varepsilon \left\|\mathbf{w}\right\|_1 \mid y \cdot \mathbf{w}^\top \mathbf{x} < 0)\,. \end{aligned}$$

First suppose that the same classifier is applied after the orthogonal projection. Since the inputs belongs to $\mathcal{V}$, we have that $\mathbf{Q}\mathbf{x} = \mathbf{x}$, thus $L_0(f_\mathbf{w}) = L_0(f_{\mathbf{Q}\mathbf{w}})$. Similarly, the adversarial risk of the classifier with the compression step yields

$$\begin{aligned} L_0^\varepsilon(f_{\mathbf{Q}\mathbf{w}}) &= \mathbb{P}(\mathbf{w}^\top \mathbf{Q}(\mathbf{x}+\boldsymbol{\eta}) \cdot y < 0) \\ &= \mathbb{P}(y \cdot \mathbf{w}^\top \mathbf{Q}\mathbf{x} < \varepsilon \left\|\mathbf{Q}\mathbf{w}\right\|_1) \\ &= L_0(f_\mathbf{w}) \mathbb{P}(|\mathbf{w}^\top \mathbf{x}| < \varepsilon \left\|\mathbf{Q}\mathbf{w}\right\|_1 \mid y \cdot \mathbf{w}^\top \mathbf{x} < 0)\,. \end{aligned}$$

If $\left\|\mathbf{Q}\mathbf{w}\right\|_1 \leq \left\|\mathbf{w}\right\|_1$, the theorem follows. Since $\mathbf{Q}$ is a projection matrix, for any $\mathbf{w}$ we have that $\left\|\mathbf{Q}\mathbf{w}\right\|_2 \leq \left\|\mathbf{w}\right\|_2$. This inequality and the definition of effective sparsity yield

$$\frac{\left\|\mathbf{Q}\mathbf{w}\right\|_1}{\overline{s}(\mathbf{Q}\mathbf{w})} \leq \frac{\left\|\mathbf{w}\right\|_1}{\overline{s}(\mathbf{w})}\,.$$

Given the initial assumption $\overline{s}(\mathbf{Q}\mathbf{w}) \geq \overline{s}(\mathbf{w})$ we get

$$\left\|\mathbf{Q}\mathbf{w}\right\|_1 \leq \frac{\overline{s}(\mathbf{Q}\mathbf{w})}{\overline{s}(\mathbf{w})} \left\|\mathbf{w}\right\|_1 \leq \left\|\mathbf{w}\right\|_1\,.$$

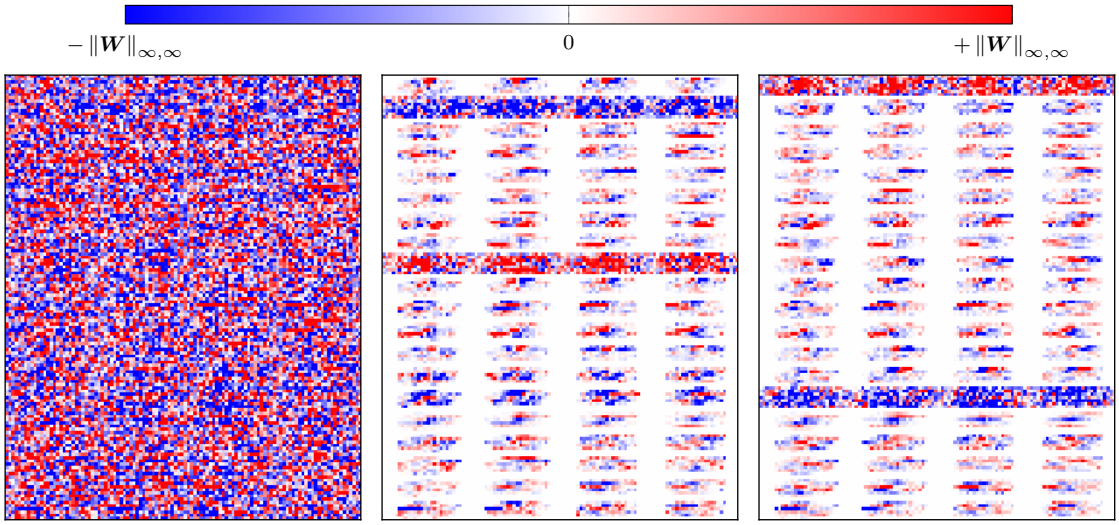
Hence, $\varepsilon \left\|\mathbf{w}\right\|_1 \geq \varepsilon \left\|\mathbf{Q}\mathbf{w}\right\|_1$ thus $L_0^\varepsilon(f_\mathbf{w}) \geq L_0^\varepsilon(f_{\mathbf{Q}\mathbf{w}})$. □

The condition on effective sparsity is not very demanding. Indeed, assume that $\mathcal{V} = \operatorname{span}(\mathbf{e}_1, \ldots, \mathbf{e}_r)$ for $r < n$. Then, we have $\left\|\mathbf{Q}\mathbf{w}\right\|_1 = \sum_{i=1}^r |w_i| \leq \left\|\mathbf{w}\right\|_1$. The equality holds only if $\mathbf{w} \in \mathcal{V}$. In other words, if the discriminating hyperplane is not orthogonal to the data subspace, there is always a gain in low dimensional projection. Note that for an arbitrary classifier, the accuracy remains unchanged after the projection. In this case, if the corresponding weight vector is projected such that its ℓ_1-norm is not increased, then the robustness will be enhanced. Finally, if we overparametrize the classifier $\operatorname{sign}((\mathbf{Q}\mathbf{w})^\top \mathbf{x})$ as $\operatorname{sign}\left((\mathbf{Q}\boldsymbol{W}^1 \cdots \boldsymbol{W}^{d-1}\mathbf{w})^\top \mathbf{x}\right)$ we obtain a d-layered neural network with linear activations whose initial weight matrix given by $\mathbf{Q}\boldsymbol{W}^1$ has low rank. This remark points our attention specially into the low-rankness the first weight matrices of neural networks.

4.4. Inducing Compression through Regularization

Motivated by the result in Theorem 4.3.1, for linear classifiers, we investigate the effect that simultaneous low-rankness and sparsity of the weight matrices of DNNs has on

their adversarial robustness. More precisely, we investigate if promoting sparsity and low-rankness leads to robustness and vice-versa. Let us first illustrate this relation with a simple experiment, shown in Figure 4.1. In that figure, we observe how adversarial training, using the FGSM and PGD attacks, seems to induce effective sparsity as well as effective low-rankness to the weight matrices of a FCNN (details about the simulation setup are provided in Section 4.5). Motivated by this result, as well as Theorem 4.3.1, we



(a) Natural training **(b)** FGSM training **(c)** PGD training

Figure 4.1.: Reshaped input weight matrix $\boldsymbol{W}^1 \in \mathbb{R}^{20\times 784}$ of FCNN after natural vs. adversarial training with $\varepsilon = 0.05$.

proceed to the question of whether is possible to improve the adversarial robustness by promoting sparsity and low-rankness without adversarial training. We introduce three regularization techniques designed to promote sparsity and low-rankness to the weight matrices of DNNs. In order to promote sparsity on the weight matrices, a common choice is to apply the ℓ_1 regularization. To that end, let us define the ℓ_1 regularization loss as

$$\mathcal{L}_{\ell_1}(\mathbf{w}, \boldsymbol{\lambda}_1) = \sum_{i=1}^{d} \lambda_{1,i} \left\| \operatorname{vec}\left(\boldsymbol{W}^i\right) \right\|_1 = \sum_{i=1}^{d} \lambda_{1,i} \left\| \boldsymbol{W}^i \right\|_{1,1} , \tag{4.6}$$

where $\boldsymbol{\lambda}_1 = (\lambda_{1,1}, \ldots, \lambda_{1,d}) \in \mathbb{R}_+^d$ is a vector of hyper-parameters. Just like the ℓ_1 regularization is a relaxation of sparsity measure, the nuclear norm is the convex relaxation of rank of a matrix, thus the nuclear norm regularization is used to promote low-rankness. Similarly, the nuclear regularization loss is given by

$$\mathcal{L}_{*}(\mathbf{w}, \boldsymbol{\lambda}_*) = \sum_{i=1}^{d} \lambda_{*,i} \left\| \sigma(\boldsymbol{W}^i) \right\|_1 = \sum_{i=1}^{d} \lambda_{*,i} \left\| \boldsymbol{W}^i \right\|_{*} , \tag{4.7}$$

with the corresponding hyper-parameters $\boldsymbol{\lambda}_* = (\lambda_{*,1}, \ldots, \lambda_{*,d}) \in \mathbb{R}_+^d$.

As discussed in Section 4.3 and in [69], the robustness to ℓ_∞ attacks of multilayer neural networks with linear activations is controlled by the ℓ_1-norm of the product of its weight matrices, that is $\left\|\boldsymbol{W}^1 \cdots \boldsymbol{W}^d\right\|_1$. Motivated by this result, we include into our study the following regularization

$$\mathcal{L}_{\text{joint}}(\mathbf{w}, \lambda_{\text{joint}}) = \lambda_{\text{joint}} \left\|\boldsymbol{W}^1 \cdots \boldsymbol{W}^d\right\|_{1,1} , \tag{4.8}$$

which is known to promote robustness in linear neural networks. Such approach may lead to robustness for non-linear networks if they operate in approximately linear regimes. Note that $\mathcal{L}_{\text{joint}}(\mathbf{w}, \lambda_{\text{joint}})$ is cheaper to compute than $\mathcal{L}_*(\mathbf{w}, \boldsymbol{\lambda}_*)$, which makes it a good alternative to avoid computing SVDs during training. Finally, we incorporate these regularization losses into one global loss function yielding

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, y) = \mathcal{L}_{\text{CE}}(\mathbf{w}; \mathbf{x}, y) + \mathcal{L}_{\ell_1}(\mathbf{w}, \boldsymbol{\lambda}_1) + \mathcal{L}_*(\mathbf{w}, \boldsymbol{\lambda}_*) + \mathcal{L}_{\text{joint}}(\mathbf{w}, \lambda_{\text{joint}}) ,$$

where $\mathcal{L}_{\text{CE}}$ denotes the cross-entropy loss function.

4.5. Experiments

In this chapter, we consider two architectures and datasets:

1. **FCNN**: is a fully-connected neural network with 5 hidden layers used on the MNIST handwritten digits dataset [53]. Each hidden layer contains 20 neurons with hyperbolic tangent activations, while the output layer uses the softmax function. We use a learning rate of 0.001 and a batch size of 128 for both training from scratch and fine-tuning.

2. **CNN**: is a convolutional neural network applied on the Fashion-MNIST (F-MNIST) clothing articles dataset [75]. It is composed by two convolution/max-pooling blocks followed by two fully connected layers. Both convolution/max-pooling blocks use 32 convolution filters of window size 3×3 and pooling window of size 2×2. The first fully connected layer outputs 128 outputs with hyperbolic tangent activations, while the output layer outputs 10 values. We use a batch size of 64 with a learning rate of 0.01 for training from scratch, and 0.001 for fine-tuning.

In both cases, stochastic gradient descent is used for minimizing the loss function.

We consider the FGSM [20], PGD [24] and the GNM [19] as adversarial attacks and study their success on DNNs that are trained naturally, adversarially, or with the aforementioned regularization techniques (see Section 4.4). As in Chapter 3.6, the fooling ratio is defined as the percentage of inputs, among the correctly classified ones, whose classification outcome changes after adding adversarial perturbations. This ratio is calculated on the test set as an empirical measure of sensitivity of the classifier against

adversarial attacks for a given value of ε. We focus on the low ε regime, which corresponds to adversarial examples that are hard to detect by an observer. In such regime, the FGSM often leads to similar attacks as PGD since the decision boundary is linear enough in a vicinity around the clean sample. We found that $\varepsilon = 0.05$ was a proper choice that leads to a fooling ratio slightly below 100% in the case of natural training for both considered architectures. We generate the attack using the GNM with $T = 5$ iterations, while PGD attacks are generated with $T = 20$ iterations and a step size of $\alpha = 0.005$.

In the same spirit as Madry *et al.* [24], we raise the question to what happens to the weight matrices $\boldsymbol{W}^i$ during adversarial training. For both settings, we fine-tune naturally trained models through adversarial training with the FGSM and PGD attack, using a fixed $\varepsilon = 0.05$. As expected, adversarial training substantially decreases the fooling ratios. We illustrate the fooling ratios over the whole course of training, in Figure 4.2 and Figure 4.3, for FCNN and CNN respectively. Moreover, we compute the fooling ratios for different types of attacks using the same fixed $\varepsilon = 0.05$. For the sake of completeness, we report the test accuracies achieved on clean data (see the Appendix A.1). For CNN model, we observe slightly lower test accuracies for the adversarially trained models when compared to the naturally trained one.

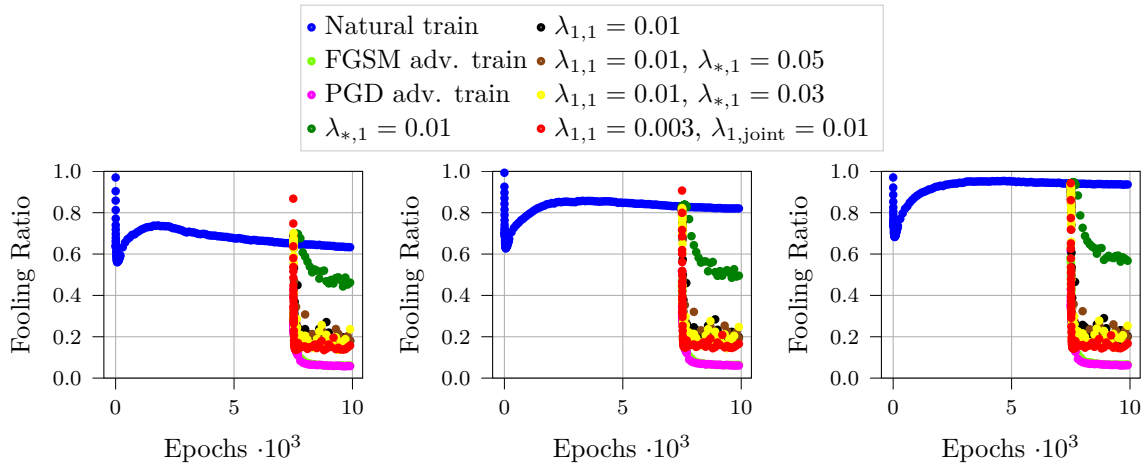
In Figure 4.4 and Figure 4.5, we present the effective rank and the effective sparsity over the whole course of training for FCNN and CNN respectively. In case of FCNN, we also estimate the mutual information between the input $\mathbf{x}$ and the hidden representations $\mathbf{x}^i$ using the Kernel-based Density Estimator (KDE) method [76], [77] with a noise variance of 0.1. Note that the KDE method provides upper and lower bounds for mutual information, instead of a single estimate. However, for the sake of clarity, in these figures we plot the arithmetic average between those upper and lower bounds.

We observe how the proposed regularization techniques successfully induce low-rankness and sparsity to the weight matrices. At the same time, the mutual information between the input and hidden representations is reduced. Interestingly, the effective low-rankness of the weight matrices coincides with lower mutual information values, between the input and hidden representations, which can be seen as a form of compression in the information theoretic sense. This effect is most visible for the input weight matrix $\boldsymbol{W}^1 \in \mathbb{R}^{20\times 784}$. This weight matrix is visualized for natural and adversarial training in Figure 4.1. While $\boldsymbol{W}^1$ looks like noise after natural training, it clearly looks like lower-rank and more sparse after adversarial training, with strongly correlated patterns. We can also observe this effect in an attenuated form for CNN experiments, for instance see Figure 4.6.

Further, we can see in Figure 4.5 that adversarial training substantially decreases the effective sparsity[1] of the first two convolutional filters (*i.e.*, $\boldsymbol{W}^1 \in \mathbb{R}^{3\times 3\times 1\times 32}$ and $\boldsymbol{W}^2 \in \mathbb{R}^{3\times 3\times 32\times 32}$). Moreover, adversarial training slightly decreases the effective rank of $\boldsymbol{W}^2$ as well. Note that using PGD adversarial training reduces the effective rank of $\boldsymbol{W}^2$ more

[1]Note that highly sparse matrices have low effective sparsity, *i.e.*, low value of $\overline{s}(\cdot)$.

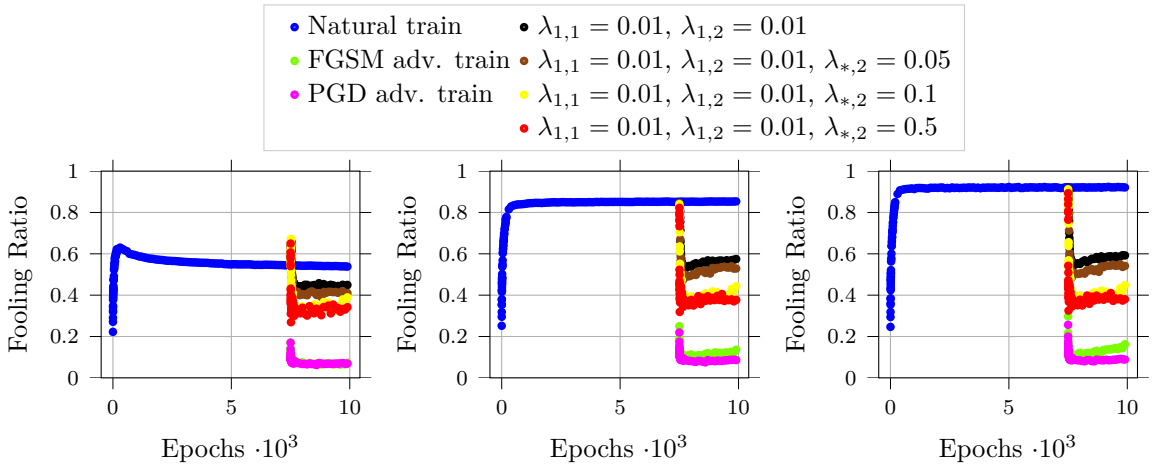
than FGSM adversarial training, as shown in Figure 4.5. This is also visible in Figure 4.7 where we visualize $\boldsymbol{W}^2$ for natural and adversarial training. The weights after both FGSM and PGD adversarial training look more sparse and slightly lower-rank than after natural training. In addition, the weights after PGD adversarial training look slightly lower-rank than after FGSM adversarial training. From these observations we first conclude that adversarial training leads to compression in the information theoretic sense. Besides, it seems that adversarial training leads to low-rank and sparse weights.



(a) FGSM **(b)** GNM **(c)** PGD

Figure 4.2.: Fooling ratios of FCNN with respective attack and defense using a fixed $\varepsilon = 0.05$. Dataset: MNIST.

We now study the reverse direction, that is whether low-rank and sparse weight matrices lead to enhanced adversarial robustness. To that end, we mimic adversarial training by promoting low-rank and sparse weight matrices. This is carried out through regularization of the loss function as explained in Section 4.4. In case of FCNN, we first regularize the input weight matrix $\boldsymbol{W}^1$ with different choices of $\lambda_{1,1}$ and $\lambda_{*,1}$ as this weight matrix substantially changes its effective rank and effective sparsity during adversarial training. We also provide results for performing only l_1-regularization on $\boldsymbol{W}^1$ with $\lambda_{1,1} = 0.01$. Both kinds of regularization substantially decrease the fooling ratios, as shown in Figure 4.2. Moreover, in Figure 4.4 we observe that both kinds of regularization substantially decrease the effective sparsity and effective rank of $\boldsymbol{W}^1$, as well as $\boldsymbol{W}^2$. Surprisingly, performing only l_1-regularization on $\boldsymbol{W}^1$ also decreases its effective rank. In Figure 4.8, we visualize $\boldsymbol{W}^1$ after training with both kinds of regularization, where we can clearly observe approximately sparse and low-rank structures. As shown in Figure 4.4, we are able to simultaneously decrease the effective rank of $\boldsymbol{W}^1$, $\boldsymbol{W}^2$ and $\boldsymbol{W}^3$ by using the joint regularization technique (4.8). Together with explicitly promoting sparsity on $\boldsymbol{W}^1$, we are able to improve the robustness again compared to regularizing only $\boldsymbol{W}^1$.



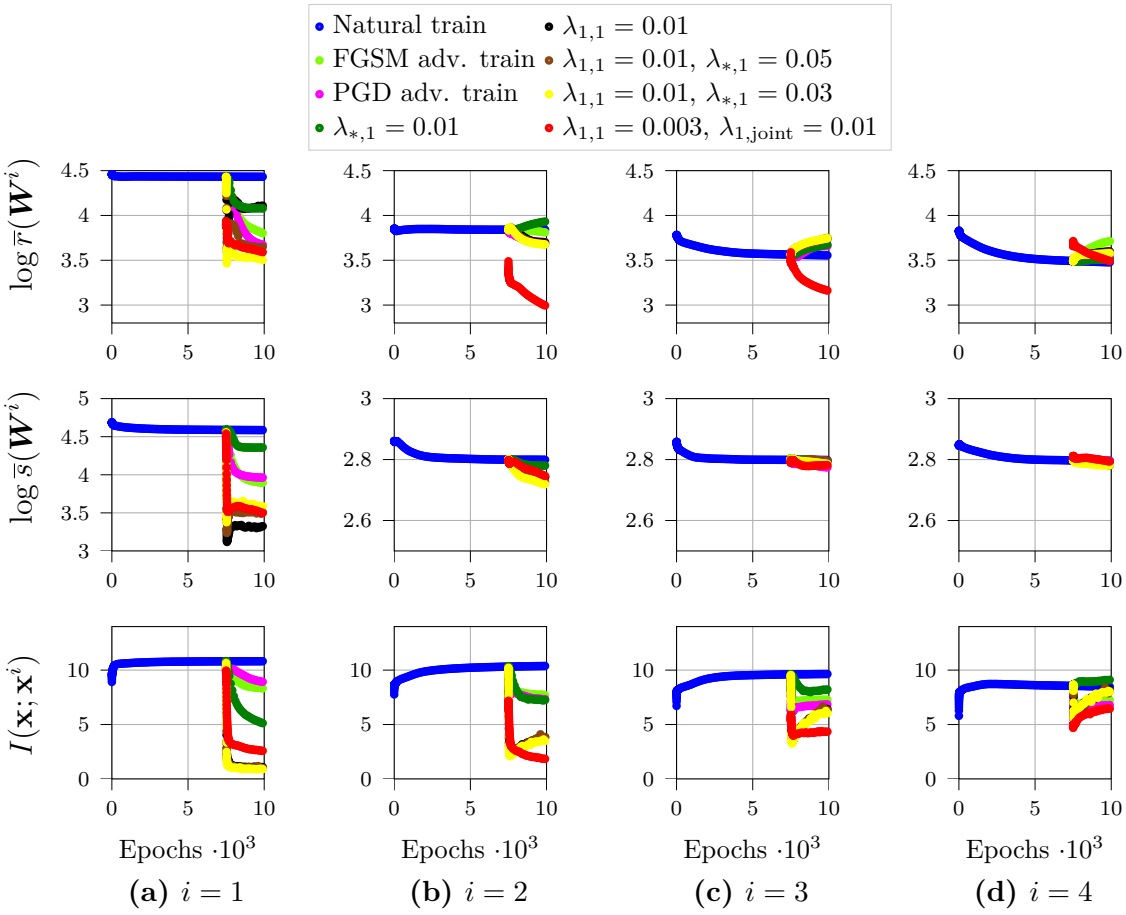
(a) FGSM **(b)** GNM **(c)** PGD

Figure 4.3.: Fooling ratios of CNN with respective attack and defense using a fixed $\varepsilon = 0.05$. Dataset: F-MNIST.

In case of CNN, we regularize the reshaped versions of the first two convolutional filters $\boldsymbol{W}^1$, $\boldsymbol{W}^2$ (see Appendix A.2 for details about reshaping) with different choices of $\lambda_{1,1}$, $\lambda_{1,2}$ and $\lambda_{*,2}$. We observe a significant improvement in terms of robustness (see Figure 4.3) when increasing the explicit low-rank promotion on $\boldsymbol{W}^2$ by nuclear-norm regularization. In Figure 4.5, we see that a considerably lower effective rank of $\boldsymbol{W}^2$ can be obtained by adding nuclear-norm regularization. This result supports the idea that simultaneous sparsity and low-rankness of the weights should be favored on the way towards adversarial robustness. In Figure 4.9 and Figure 4.10, we visualize the first and second convolutional filter $\boldsymbol{W}^1$, $\boldsymbol{W}^2$ after training with the different kinds of regularization. These empirical findings suggest that simultaneously low-rank and sparse weight matrices indeed promote robustness against adversarial examples.

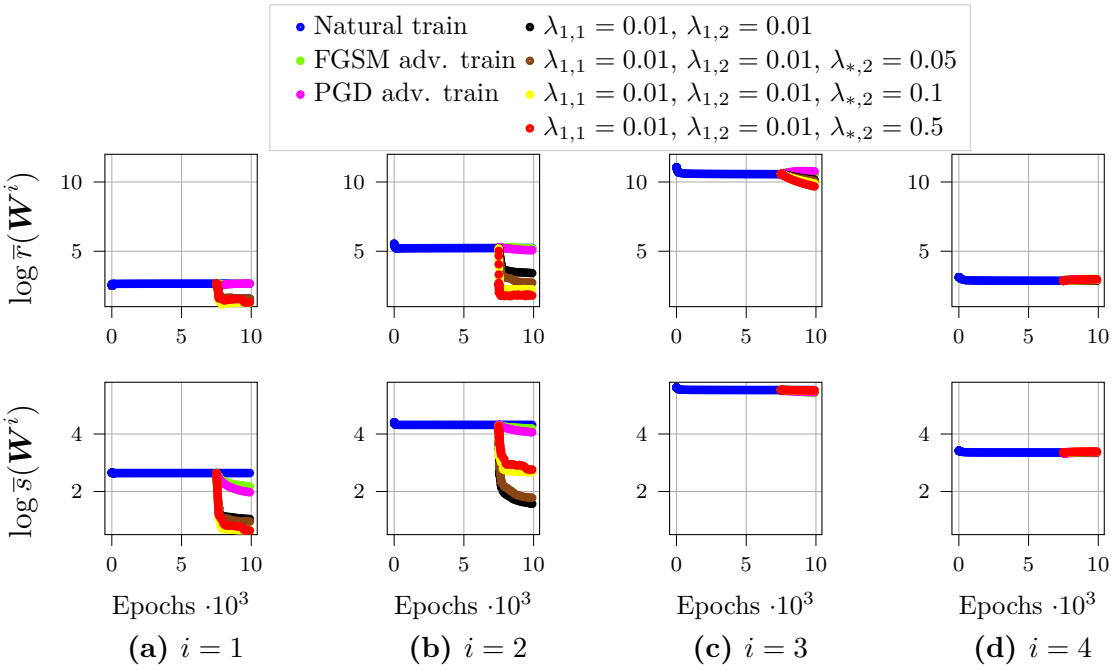
4.6. Discussion

We are able to obtain more sparse and lower-rank weights through regularizations, which led to substantial improvement in adversarial robustness. However, this approach did not match the robustness of adversarial training. This result suggests that, besides sparsity and low-rankness of the weights, further attributes should be considered. Moreover, it raises an open question whether there is an optimal combination of sparsity and low-rankness of the weights that can further improve robustness. The experiments suggest that, despite their important role, sparsity and low-rankness of the weights do not fully explain the success of adversarial training.



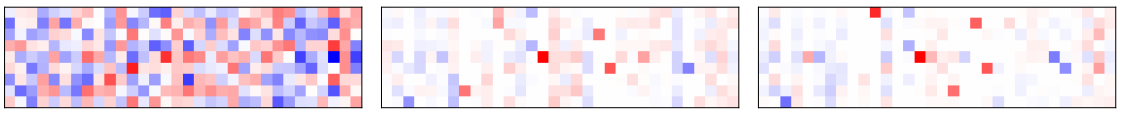
(a) $i = 1$ **(b)** $i = 2$ **(c)** $i = 3$ **(d)** $i = 4$

Figure 4.4.: Effective rank, effective sparsity and mutual information for different layers of FCNN. Layers 5 and 6 are shown in Appendix A.1.



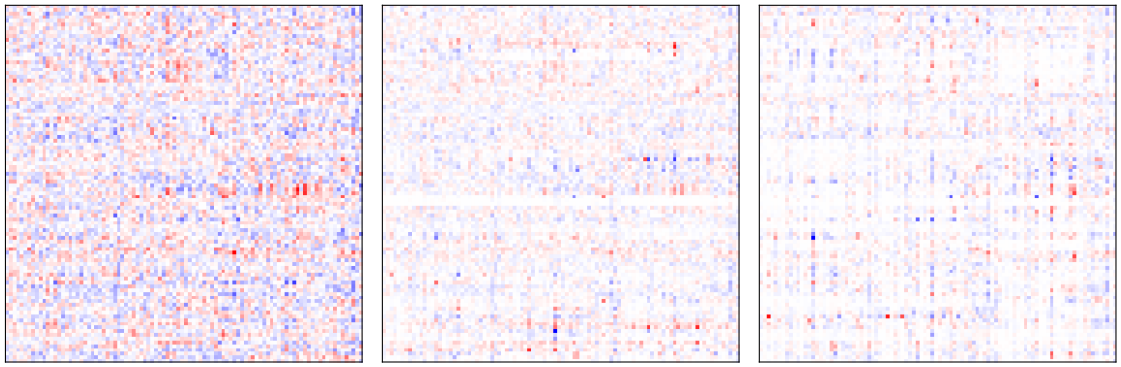
(a) $i = 1$ **(b)** $i = 2$ **(c)** $i = 3$ **(d)** $i = 4$

Figure 4.5.: Effective rank and effective sparsity for different layers of CNN.



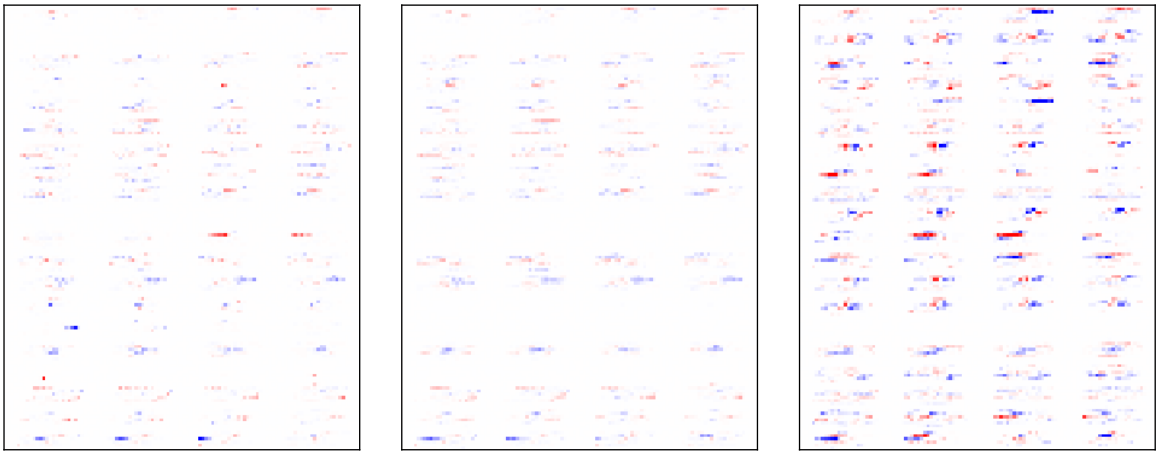
(a) Natural training **(b)** FGSM training **(c)** PGD training

Figure 4.6.: Reshaped input convolutional filter $\boldsymbol{W}^1 \in \mathbb{R}^{3\times3\times1\times32}$ of CNN after natural vs. adversarial training with $\varepsilon = 0.05$. The color coding is done as in Figure 4.1.



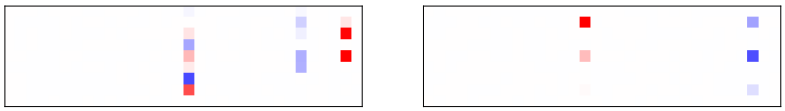
(a) Natural training **(b)** FGSM training **(c)** PGD training

Figure 4.7.: Reshaped second convolutional filter $\boldsymbol{W}^2 \in \mathbb{R}^{3\times3\times32\times32}$ of CNN after natural vs. adversarial training with $\varepsilon = 0.05$. The color coding is done as in Figure 4.1.



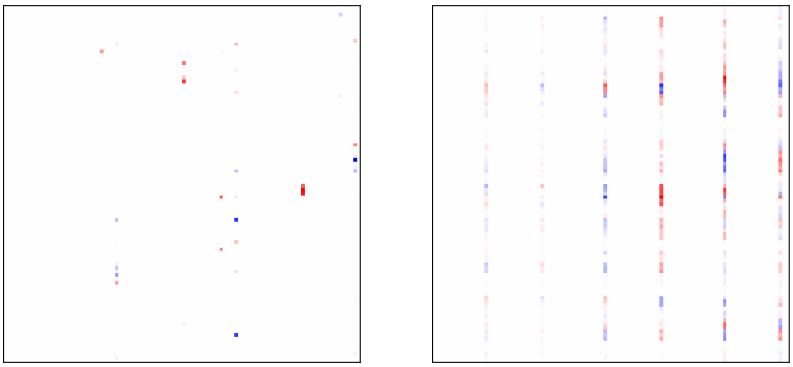
(a) $\lambda_{1,1} = 0.01$ **(b)** $\lambda_{1,1} = 0.01$, $\lambda_{*,1} = 0.01$ **(c)** $\lambda_{1,1} = 0.003$, $\lambda_{\text{joint}} = 0.04$

Figure 4.8.: Reshaped input weight matrix $\boldsymbol{W}^1 \in \mathbb{R}^{20\times784}$ of FCNN after training with respective regularization. The color coding is done as in Figure 4.1.



(a) $\lambda_{1,1} = 0.01$, $\lambda_{1,2} = 0.01$ **(b)** $\lambda_{1,1} = 0.01$, $\lambda_{1,2} = 0.01$, $\lambda_{*,2} = 0.5$

Figure 4.9.: Reshaped input convolutional filter $\boldsymbol{W}^1 \in \mathbb{R}^{3\times3\times1\times32}$ of CNN after training with respective regularization. The color coding is done as in Figure 4.1.



(a) $\lambda_{1,1} = 0.01$, $\lambda_{1,2} = 0.01$ **(b)** $\lambda_{1,1} = 0.01$, $\lambda_{1,2} = 0.01$, $\lambda_{*,2} = 0.5$

Figure 4.10.: Reshaped second convolutional filter $\boldsymbol{W}^2 \in \mathbb{R}^{3\times3\times32\times32}$ of CNN after training with respective regularization. The color coding is done as in Figure 4.1.

5

Adversarial Risk Bounds through Sparsity based Compression

As discussed in Chapters 3 and 4, neural networks are known to be vulnerable against minor adversarial perturbations of their inputs, especially for high dimensional data under ℓ_∞ attacks. To combat this problem, techniques like adversarial training have been employed to obtain models which are robust on the training set. However, the robustness of such models against adversarial perturbations may not generalize to unseen data. To study how robustness generalizes, recent works assume that the inputs have bounded ℓ_2-norm in order to bound the adversarial risk for ℓ_∞ attacks with no explicit dimension dependence. In this chapter[1], we focus on ℓ_∞ attacks on ℓ_∞ bounded inputs and prove margin-based bounds. Specifically, we use a compression based approach that relies on efficiently compressing the set of tunable parameters without distorting the adversarial risk. To achieve this, we propose a notion of effective sparsity and effective joint sparsity on the weight matrices of neural networks. This leads to bounds with no explicit dependence on the input dimension, neither on the number of classes. Our results show that neural networks with approximately sparse weight matrices not only possess enhanced robustness, but also better generalization.

[1]This chapter contains the work in [78].

5.1. Related Work

In Chapter 3, we have discussed the vulnerability of neural networks to maliciously designed perturbations of their inputs, known as adversarial examples. We showed examples where such perturbed inputs are only slightly distorted versions of the original inputs, and yet they manage to fool neural networks into incorrect classifications. For example, in image classification, adversarial examples have been shown to be indistinguishable from the original image to the human eye (see Figure 3.2). This phenomena motivated several works aiming at understanding the nature of classifiers, and in particular neural networks, in the presence of adversarial examples. Initial works focused on the linearity (and non-linearity) of classifiers and its implications on the robustness of DNNs against adversarial examples [20], [39], [60]. Subsequent works shed some light on the nature of adversarial examples by studying the properties of decision boundaries [7], [39], [63]–[65], while others focused on the model capacity of neural networks in relation to the problem difficulty [24], [62], [66]. While these approaches contributed to understanding the nature of adversarial examples, they do not consider whether the robustness of classifiers against adversarial perturbations generalizes to unseen data.

If a classifier is robust to perturbations of the training set, can we guarantee that it will also be robust to perturbations of the test set? This question is not particularly new. The optimization community has studied this problem for quite some time. The work of Xu, *et al.* [79], studied robust regression in Lasso, while later work [80] obtained results for support vector machines. Other works considered the generalization properties of robust optimization in a distributional sense [81], that is when adversarial examples are assumed to be samples from the worst possible distribution within a Wasserstein ball around the original one. As discussed, these works provide algorithms for training various types of classifiers with robustness guarantees. Regarding neural networks, for the case where no adversarial perturbations are present, there exists an extensive literature on their generalization properties. Many of these works are based on bounding the Rademacher complexity of the function class [82]–[85], while others make use of the PAC-Bayes framework [86]–[88]. There are other works which rely on different techniques, for instance, Arora *et al.* [89] rely on compressing the weights of neural networks. Despite this knowledge, proving robustness guarantees for neural networks remained unstudied till recently. Initial works going into this direction studied neural networks in artificial scenarios. For instance, Attias *et al.* [90] proved generalization bounds for the case when the adversary can modify a finite number of entries per input. Following this approach, Diochnos *et al.* [91] showed that the number of flipped bits required to fool almost all inputs is less than $\mathcal{O}(\sqrt{n})$, for the case when the input is binary and uniformly distributed. As similar subsequent result [92] for binary inputs, proved the existence of polynomial-time attacks that find adversarial examples of Hamming distance $\mathcal{O}(\sqrt{n})$. Concurrently, the work of Schmidt *et al.* [93] showed that the amount of data necessary to classify n-dimensional Gaussian data grows by a factor of $\sqrt{n}$ in the presence of an adversary. However, Cullina *et al.* [94] showed

that the Vapnik-Chervonenkis (VC)-dimension of linear classifiers does not increase in the adversarial setting. Additionally, they derived generalization guarantees for binary linear classifiers. Moreover, Montasser *et al.* [95] showed that VC-classes are learnable in the adversarial setting, but only if we refrain from using standard empirical risk minimization approaches. Later works considered more general scenarios. Using a PAC-Bayes approach, Farnia *et al.* [96] proved a generalization bound for neural networks under ℓ_2 attacks. However, deriving bounds for attacks with bounded ℓ_∞-norm (instead of ℓ_2-norm) is of particular interest, since most successful attacks in computer vision are of this type. In addition, such attacks tend to be more effective for scenarios where the input dimension is large, thus deriving generalization bounds without explicit dimension dependence is promising.

Now, let us overview recent works addressing the problem of proving generalization bounds for neural networks in the adversarial setting, where the attacker has bounded ℓ_∞ perturbations. Since these works are closely related to this chapter, we discuss them in more detail in the following list.

- Yin *et al.* [97] bounded the Rademacher complexity for linear classifiers and neural networks in the adversarial setting. This lead to explicit bounds on the notion of adversarial risk for the linear classifier as well as neural networks. Nevertheless, such bound applied only to neural networks with one hidden layer and ReLU activations.
- Concurrent work of Khim *et al.* [98] proved bounds on a surrogate of the adversarial test error. In that work, the authors use the so-called tree transform on the function class to derive their results. Under the assumption that the original inputs have ℓ_2 bounded norm, the authors proved generalization bounds with no explicit dimension dependence in the binary classification setting. Yet, the authors extend this to k-class classification by incurring an additional factor k on their bound.
- Later work of Tu *et al.* [99] formulated generalization in the adversarial setting as a minimax problem. Their proposed framework is more general than previous ones in the sense that it can be applied to support vector machines and principal component analysis, as well as neural networks. Nonetheless, for neural networks this approach yielded a generalization bound with explicit dimension dependence.

One common assumption shared by these works is that the inputs come from a distribution with bounded ℓ_2-norm, which is a weaker notion than assuming ℓ_∞ bounded inputs.

5.1.1. Our Contributions

In this chapter, we study the problem of bounding the generalization error of multilayer neural networks under ℓ_∞ attacks, where we assume that the original inputs have

ℓ_∞ bounded norm. Using a compression approach, we obtain bounds with no explicit dependence on the input dimension or the number of classes. We summarize our contributions as follows.

- We prove generalization bounds in the presence of adversarial perturbations of bounded ℓ_∞-norm under the assumption that the input distribution has bounded ℓ_∞-norm as well. This is an improvement with respect to recent works where the input is assumed to be ℓ_2 bounded.

- We extend the compression approach of [89] by incorporating the notion of effective sparsity. Using this technique we prove that the sample complexity of neural networks, under adversarial perturbations, is bounded by the effective sparsity and effective joint sparsity of its weight matrices. This result has no explicit dimension dependence, neither it depends on the number of classes. We show that approximately sparse weights not only improve robustness against ℓ_∞ bounded adversarial perturbations, but also provide better generalization as well.

- We corroborate our result with experiments on the MNIST and CIFAR-10 datasets, where the bound correlates with adversarial risk. We observe that adversarial training significantly decreases the bound, while standard training does not. Similarly, adversarial training seems to decrease both, effective sparsity and effective joint sparsity, as predicted by our result. Moreover, in these experiments, effective joint sparsity appears to be the dominant quantity in our bound. This shows the importance of effective joint sparsity for achieving generalization in the adversarial setting, a relation that was not discovered so far.

5.1.2. Notation

The notation $\mathcal{B}^n_{p,\varepsilon}$ is used to refer to an n-dimensional ℓ_p ball of radius ε, that is the set $\mathcal{B}^n_{p,\varepsilon} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_p \leq \varepsilon\}$. We use the compact notation $\widetilde{\mathcal{O}}(n) := \mathcal{O}(n \log n)$ to ignore logarithmic factors.

5.2. Problem Setup

We start with the standard margin-based statistical learning framework, introduced in Section 2.2. Since the notation introduced in that section is consistent among chapters, $\mathcal{X} \subseteq \mathbb{R}^n$ denotes the feature space, $\mathcal{Y} = \{1, 2, \ldots, k\}$ the label space, and $\mathcal{D} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ data distribution. In this chapter, it is assumed that all instances $\mathbf{x} \in \mathcal{X}$ have ℓ_∞-norm bounded by 1, that is $\mathcal{X} \subseteq \mathcal{B}^n_{\infty,1} \subset \mathbb{R}^n$. Using this notation, a classifier is defined in Definition 2.2.1 through its so called score function $f : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{Y}|}$ such that

the predicted label is $\operatorname{argmax}_{j\in\mathcal{Y}} f_j(\cdot)$, where $f_j(\cdot)$ is the j-th entry of $f(\cdot)$. Moreover, given an instance $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$, the classification margin is defined in (2.5) as

$$\ell(f; \mathbf{x}, y) = f_y(\mathbf{x}) - \max_{j\neq y} f_j(\mathbf{x})\,.$$

In this manner, a positive margin implies correct classification. Then, for any distribution $\mathcal{D}$ the expected margin loss with margin $\gamma \geq 0$ is defined as

$$L_\gamma(f) = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\ell(f; \mathbf{x}, y) \leq \gamma\right]\,.$$

We study the case where an adversary is present. This adversary has access to the input $\mathbf{x}$ and is allowed to add a perturbation $\boldsymbol{\eta}$ with ℓ_∞-norm bounded by some $\varepsilon \geq 0$ (*i.e.*, $\boldsymbol{\eta} \in \mathcal{B}^n_{p,\varepsilon}$) such that the classification margin is as small as possible. This perturbed input $\mathbf{x} + \boldsymbol{\eta}$ is usually known as an adversarial example. Furthermore, let us define the margin under adversarial perturbations as

$$\ell_\varepsilon(f; \mathbf{x}, y) = \inf_{\boldsymbol{\eta}\in\mathcal{B}^n_{\infty,\varepsilon}} \ell(f; \mathbf{x} + \boldsymbol{\eta}, y)\,.$$

This leads to the definition of adversarial margin loss, that is

$$L^\varepsilon_\gamma(f) = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\ell_\varepsilon(f; \mathbf{x}, y) \leq \gamma\right]\,.$$

Let $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be the training set composed of m instances drawn independently from $\mathcal{D}$. Using these instances we define $\widehat{L}^\varepsilon_\gamma(f) = \frac{1}{m}\sum_{i=1}^m \mathbb{1}_{(\ell_\varepsilon(f;\mathbf{x}_i,y_i)\leq\gamma)}$ as the empirical estimate of $L^\varepsilon_\gamma(f)$, where $\mathbb{1}_{(\cdot)}$ denotes the indicator function. Note that $L^\varepsilon_0(f)$ and $\widehat{L}^\varepsilon_0(f)$ are the expected and training error under adversarial perturbations, respectively.

For many classifiers, such as deep neural networks, the score function f belongs to a complicated function class $\mathcal{F}$, which usually has more sample complexity than the size of the training set. Even without the presence of an adversary, it is challenging to bound the generalization error, given by the difference $L_0(f) - \widehat{L}_\gamma(f)$, of such function classes. The key idea behind the compression framework presented in [89] is to show that there exists a finite function class $\mathcal{G}$ with low sample complexity and a mapping that assigns a function $g \in \mathcal{G}$ to every $f \in \mathcal{F}$ such that the empirical loss is not severely degraded. This trick allows us to bound the generalization error using the sample complexity of $\mathcal{G}$ instead of $\mathcal{F}$. A drawback of this method is that we are only able to bound $L_0(g) - \widehat{L}_\gamma(f)$ instead of the original generalization error. Nevertheless, as the authors mentioned in [89], a similar issue is present as well in standard PAC-Bayes bounds, where the bound is on a noisy version of f. Moreover, the authors discuss some possible ways to solve this issue, but these approaches were left for future work. In this chapter we leverage such a compression framework by extending it to the case when an adversary is present. Our goal is to bound the generalization error under the presence of an adversary. We start by introducing some formal definitions and theorems, similar to the ones in [89].

Definition 5.2.1 ($(\gamma, \varepsilon, \mathcal{S})$-compressible)**.** *Given a set of parameter configurations $\mathcal{A}$, let $\mathcal{G}_{\mathcal{A}} = \{g_A | A \in \mathcal{A}\}$ be a set of parametrized functions g_A. We say that the score function $f \in \mathcal{F}$ is $(\gamma, \varepsilon, \mathcal{S})$-compressible through $\mathcal{G}_{\mathcal{A}}$ if*

$$\forall \mathbf{x} \in \mathcal{S}, y \in \mathcal{Y} : |\ell_\varepsilon(f; \mathbf{x}, y) - \ell_\varepsilon(g_A; \mathbf{x}, y)| \leq \gamma \,.$$

Theorem 5.2.2. *Given the finite sets $\mathcal{A}$ and $\mathcal{G}_{\mathcal{A}} = \{g_A | A \in \mathcal{A}\}$, if f is $(\gamma, \varepsilon, \mathcal{S})$-compressible via $\mathcal{G}_{\mathcal{A}}$ then there exists $A \in \mathcal{A}$ such that with high probability*

$$L_0^\varepsilon(g_A) \leq \widehat{L}_\gamma^\varepsilon(f) + \mathcal{O}\left(\sqrt{\frac{\log|\mathcal{A}|}{m}}\right) .$$

Proof (Theorem 5.2.2). Since $\widehat{L}_0^\varepsilon(g_A)$ is an average of m i.i.d random variables with expectation equal to $L_0^\varepsilon(g_A)$ we may use Hoeffdingen's inequality, yielding

$$\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\widehat{L}_0^\varepsilon(g_A) - L_0^\varepsilon(g_A) \geq \tau\right] \leq \exp\left(-2m\tau^2\right) .$$

Note that $|\mathcal{A}| = \exp(\log|\mathcal{A}|)$. Then, let us choose $\tau = \sqrt{\frac{\log|\mathcal{A}|}{m}}$ and take a union bound over all $A \in \mathcal{A}$, leading to

$$\mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\widehat{L}_0^\varepsilon(g_A) - L_0^\varepsilon(g_A) \geq \sqrt{\frac{\log|\mathcal{A}|}{m}}\right] \leq \exp(-\log|\mathcal{A}|) \,.$$

Since f is $(\gamma, \varepsilon, \mathcal{S})$-compressible via g, then

$$\forall \mathbf{x} \in \mathcal{S} : \quad |\ell_\varepsilon(f; \mathbf{x}, y) - \ell_\varepsilon(g_A; \mathbf{x}, y)| \leq \gamma \,,$$

which implies that

$$\widehat{L}_0^\varepsilon(g_A) \leq \widehat{L}_\gamma^\varepsilon(f) \,.$$

Combining these results we get that

$$L_0^\varepsilon(g_A) \leq \widehat{L}_\gamma^\varepsilon(f) + \mathcal{O}\left(\sqrt{\frac{\log|\mathcal{A}|}{m}}\right)$$

with probability at least $1 - \exp(-\log|\mathcal{A}|) = 1 - 1/|\mathcal{A}|$, which we consider as high probability. □

Corollary 5.2.2.1. *In the same setting of Theorem 5.2.2, if f is compressible only for a fraction $1 - \delta$ of the training sample, then with high probability*

$$L_0^\varepsilon(g_A) \leq \widehat{L}_\gamma^\varepsilon(f) + \mathcal{O}\left(\sqrt{\frac{\log|\mathcal{A}|}{m}}\right) + \delta \,.$$

This main definition and following theorems are trivial extensions of the ones used in [89] to the adversarial setting. However, even for the linear classifier, the main technique used in that work for compressing f cannot be applied to the setup of this chapter without incurring into explicit dimensionality dependencies in the resulting bounds. This will be explained in detail in the next section.

5.3. Main Results

In this section we introduce our main results. We start with linear classifiers on binary classification and move forward to neural networks and multi-class classification.

5.3.1. Linear Classifier

We start with a linear classifier for binary labels. Assume that $\mathbf{x} \in \mathcal{B}^n_{\infty,1}$, $y \in \{1, 2\}$ and let $\mathbf{w} = (w_1, \ldots, w_n)^\top$ be a vector of weights of a linear classifier. Then, the score function of the linear classifier is given by

$$f_{\mathbf{w}}(\mathbf{x}) = \begin{pmatrix} 0 \\ \langle \mathbf{w}, \mathbf{x} \rangle \end{pmatrix} .$$

This simplifies the margin to $\ell(f; \mathbf{x}, y) = (2y - 3) \langle \mathbf{w}, \mathbf{x} \rangle$, which leads to

$$\ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y) = (2y - 3)(\langle \mathbf{w}, \mathbf{x} \rangle - \varepsilon \|\mathbf{w}\|_1) .$$

Note that $(2y - 3) \in \{-1, +1\}$. The weight vector $\mathbf{w} \in \mathbb{R}^n$ of this classifier, with margin γ, can be compressed into another $\widehat{\mathbf{w}}$ such that both classifiers make the same predictions with reasonable probability (as we will see in Lemma 5.3.2). Given $\delta \in (0, 1]$, the compressed classifier $\widehat{\mathbf{w}}$ is constructed entry-wise as $\widehat{w}_i = z_i w_i / p_i$ where $p_i = (1 + \varepsilon)^2 |w_i| / \delta\gamma^2$ and $z_i \sim \text{Bern}(p_i)$.

Definition 5.3.1 (CompressVector$(\gamma, \mathbf{w})$)**.** *Given* $\mathbf{w} \in \mathcal{B}^n_{1,1}, \delta \in (0, 1], \gamma > 0$ *and* $\varepsilon > 0$*, let us define the random mapping* CompressVector$(\gamma, \cdot)$ *which outputs* $\widehat{\mathbf{w}} = (\widehat{w}_1, \ldots, \widehat{w}_n)^\top =$ CompressVector$(\gamma, \mathbf{w})$ *as follows*

$$\widehat{w}_i = z_i w_i / p_i , \quad \textit{with} \quad z_i \sim \text{Bern}(p_i) \ \textit{and} \ p_i = \frac{|w_i|}{\delta\gamma^2}(1 + \varepsilon)^2 ,$$

where Bern(p_i) *denotes the Bernoulli distribution with probability* p_i *and outcomes* $\{0, 1\}$.

Such classifier $\widehat{\mathbf{w}}$ outputs the same prediction as $\mathbf{w}$ with probability $1 - \delta$ and has only $\mathcal{O}\left((\log n)(1 + \varepsilon^2)/\delta\gamma^2\right)$ non-zero entries with high probability.

Lemma 5.3.2. *Given* $\mathbf{w} \in \mathcal{B}^n_{1,1}, \delta \in (0, 1], \gamma > 0$ *and* $\varepsilon > 0$*. If* $\widehat{\mathbf{w}} =$ CompressVector$(\gamma, \mathbf{w})$ *then*

$$\forall \mathbf{x} \in \mathcal{B}^n_{\infty,1},\ y \in \mathcal{Y} : \quad \mathbb{P}_{\widehat{\mathbf{w}}} \left[|\ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)| \geq \gamma \right] \leq \delta ,$$

and the number of non-zero entries in $\widehat{\mathbf{w}}$ *is less than* $\mathcal{O}((\log n)(1 + \varepsilon)^2/\delta\gamma^2)$ *with high probability.*

Proof (Lemma 5.3.2). Note that $\mathbb{E}[\widehat{w}_i] = \frac{w_i}{p_i}\mathbb{E}[z_i] = w_i$ thus $\mathbb{E}[\widehat{\mathbf{w}}] = \mathbf{w}$. Similarly, $\mathbb{E}[|\widehat{w}_i|] = \left|\frac{w_i}{p_i}\right|\mathbb{E}[z_i] = |w_i|$ and since $\widehat{w}_i$'s are independent, we get $\mathbb{E}[\|\widehat{\mathbf{w}}\|_1] = \|\mathbf{w}\|_1$. This implies that

$$\mathbb{E}\left[\ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)\right] = \mathbb{E}\left[\langle \widehat{\mathbf{w}}, \mathbf{x}\rangle - \varepsilon \|\widehat{\mathbf{w}}\|_1\right] = \langle \mathbf{w}, \mathbf{x}\rangle - \varepsilon \|\mathbf{w}\|_1 = \ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y)\,.$$

Now let us compute the variance of $\widehat{w}_i$ as

$$\mathrm{Var}\left[\widehat{w}_i\right] = \mathbb{E}\left[\widehat{w}_i^2\right] - \mathbb{E}\left[\widehat{w}_i\right]^2 = (w_i/p_i)^2 p_i - w_i^2 = \frac{1-p_i}{p_i} w_i^2\,.$$

The same calculation yields

$$\mathrm{Var}\left[|\widehat{w}_i|\right] = \frac{1-p_i}{p_i} w_i^2\,.$$

The covariance between $|\widehat{w}_i|$ and $\widehat{w}_i$ is

$$\mathrm{Cov}\left(|\widehat{w}_i|\,, \widehat{w}_i\right) = \mathbb{E}\left[|\widehat{w}_i|\widehat{w}_i\right] - \mathbb{E}[|\widehat{w}_i|]\mathbb{E}[\widehat{w}_i] = \frac{1-p_i}{p_i}|w_i|w_i\,.$$

Now putting all together we get

$$\begin{aligned}
\mathrm{Var}\left[\widehat{w}_i x_i - \varepsilon|\widehat{w}_i|\right] &= x_i^2 \mathrm{Var}[\widehat{w}_i] - 2\varepsilon x_i \mathrm{Cov}(\widehat{w}_i, |\widehat{w}_i|) + \varepsilon^2 \mathrm{Var}[|w_i|^2] \\
&= \frac{1-p_i}{p_i}\left(x_i^2 w_i^2 - 2\varepsilon x_i |w_i| w_i + \varepsilon^2 w_i^2\right) \\
&\leq \frac{w_i^2}{p_i}\left(x_i^2 + 2\varepsilon|x_i| + \varepsilon^2\right) \\
&= \frac{\delta\gamma^2}{(1+\varepsilon)^2}|w_i|\left(x_i^2 + 2\varepsilon|x_i| + \varepsilon^2\right)\,.
\end{aligned}$$

Since $\widehat{w}_i$'s are independent, we get

$$\begin{aligned}
&\mathrm{Var}\left[\langle \widehat{\mathbf{w}}, \mathbf{x}\rangle - \varepsilon\|\widehat{\mathbf{w}}\|_1\right] \\
&\quad= \mathrm{Var}\left[\sum_{i=1}^n \widehat{w}_i x_i - \varepsilon|\widehat{w}_i|\right] \\
&\quad= \sum_{i=1}^n \mathrm{Var}\left[\widehat{w}_i x_i - \varepsilon|\widehat{w}_i|\right] \\
&\quad\leq \frac{\delta\gamma^2}{(1+\varepsilon)^2}\sum_{i=1}^n |w_i|\left(x_i^2 + 2\varepsilon|x_i| + \varepsilon^2\right) \\
&\quad= \frac{\delta\gamma^2}{(1+\varepsilon)^2}\left(\left\langle |\mathbf{w}|, \mathbf{x}^2\right\rangle + 2\varepsilon\left\langle |u|, |c|\right\rangle + \varepsilon^2\|\mathbf{w}\|_1\right) \qquad (\mathbf{x}^2 \text{ is entry-wise}) \\
&\quad\leq \frac{\delta\gamma^2}{(1+\varepsilon)^2}\left(\|\mathbf{w}\|_1\left\|\mathbf{x}^2\right\|_\infty + 2\varepsilon\|\mathbf{w}\|_1\|\mathbf{x}\|_\infty + \varepsilon^2\|\mathbf{w}\|_1\right) \\
&\quad\leq \frac{\delta\gamma^2}{(1+\varepsilon)^2}(1 + 2\varepsilon + \varepsilon^2) = \delta\gamma^2\,.
\end{aligned}$$

By Chebyshev's inequality we get

$$\mathbb{P}\left[|(\langle \widehat{\mathbf{w}}, \mathbf{x}\rangle - \varepsilon \left\|\widehat{\mathbf{w}}\right\|_1) - \langle \mathbf{w}, \mathbf{x}\rangle - \varepsilon \left\|\mathbf{w}\right\|_1| > \gamma\right] \leq \delta\,.$$

On the other hand, the expected number of non-zero entries in $\widehat{\mathbf{w}}$ is given by

$$\mathbb{E}\left[\left\|\widehat{\mathbf{w}}\right\|_0\right] = \sum_{i=1}^{n} p_i = \sum_{i=1}^{n} \frac{|w_i|}{\delta\gamma^2}(1+\varepsilon^2) = \frac{(1+\varepsilon)^2}{\delta\gamma^2}\,.$$

Then, by Hoefdingen's inequality the number of non-zero entries in $\widehat{\mathbf{w}}$ is less than $\mathcal{O}((\log n)(1+\varepsilon)^2/\delta\gamma^2)$ with high probability. □

By discretizing $\widehat{\mathbf{w}}$, we obtain a compression setup that maps $\mathbf{w}$ into a discrete set but fails with probability δ. To that end, we handle discretization by clipping and then rounding in the following lemma.

Lemma 5.3.3. *Let us define*

- $\mathbf{w}'$ *component-wise as* $w'_i = w_i \mathbb{1}_{\left(|w_i| \geq \frac{\gamma}{4n(1+\varepsilon)}\right)}$,
- $\widetilde{\mathbf{w}} = \mathrm{CompressVector}(\gamma/2, \mathbf{w}')$,
- $\widehat{\mathbf{w}}$ *is obtained by rounding each entry of* $\widetilde{\mathbf{w}}$ *to the nearest multiple of* $\frac{\gamma}{2n(1+\varepsilon)}$.

Then, we have that

$$\forall \mathbf{x} \in \mathcal{B}^n_{\infty,1},\, y \in \mathcal{Y}: \quad \mathbb{P}_{\widehat{\mathbf{w}}}\left[|\ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)| \geq \gamma\right] \leq \delta\,.$$

Proof (Lemma 5.3.3). We start by bounding the error incurred by clipping, that is

$$\begin{aligned}
|\ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\mathbf{w}'}; \mathbf{x}, y)| &\leq |\langle \mathbf{w}, \mathbf{x}\rangle - \langle \mathbf{w}', \mathbf{x}\rangle| + \varepsilon \left|\left\|\mathbf{w}\right\|_1 - \left\|\mathbf{w}'\right\|_1\right| \\
&\leq |\langle \mathbf{w} - \mathbf{w}', \mathbf{x}\rangle| + \varepsilon \left\|\mathbf{w} - \mathbf{w}'\right\|_1 \\
&\leq \left\|\mathbf{w} - \mathbf{w}'\right\|_1 \left\|\mathbf{x}\right\|_\infty + \varepsilon \left\|\mathbf{w} - \mathbf{w}'\right\|_1 \\
&\leq \left\|\mathbf{w} - \mathbf{w}'\right\|_1 (1+\varepsilon) \\
&\leq \frac{\gamma}{4n(1+\varepsilon)} n(1+\varepsilon) = \gamma/4\,.
\end{aligned}$$

Similarly, the error incurred by discretizing $\widetilde{\mathbf{w}}$ is bounded by

$$\begin{aligned}
|\ell_\varepsilon(f_{\widetilde{\mathbf{w}}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)| &\leq \left\|\widetilde{\mathbf{w}} - \widehat{\mathbf{w}}\right\|_1 (1+\varepsilon) \\
&\leq \frac{\gamma}{2n(1+\varepsilon)} \frac{n}{2} (1+\varepsilon) = \gamma/4\,.
\end{aligned}$$

By Lemma 5.3.2, we know that with probability at least $1-\delta$ we have that

$$|\ell_\varepsilon(f_{\widetilde{\mathbf{w}}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)| \leq \gamma/2\,.$$

Combining these three results yields

$$\begin{aligned}|\ell_\varepsilon(f_{\mathbf{w}};\mathbf{x},y)-\ell_\varepsilon(f_{\widehat{\mathbf{w}}};\mathbf{x},y)| \leq \quad & |\ell_\varepsilon(f_{\mathbf{w}};\mathbf{x},y)-\ell_\varepsilon(f_{\mathbf{w}'};\mathbf{x},y)| \\ & +|\ell_\varepsilon(f_{\mathbf{w}'};\mathbf{x},y)-\ell_\varepsilon(f_{\widetilde{\mathbf{w}}};\mathbf{x},y)| \\ & +|\ell_\varepsilon(f_{\widetilde{\mathbf{w}}};\mathbf{x},y)-\ell_\varepsilon(f_{\widehat{\mathbf{w}}};\mathbf{x},y)| \\ \leq \quad & \gamma/4+\gamma/2+\gamma/4 \leq \gamma\end{aligned}$$

with probability at least $1-\delta$. □

Therefore, we can apply Corollary 5.2.2.1 and choose $\delta = ((1+\varepsilon)^2/\gamma^2 m)^{1/3}$, which yields a generalization bound of order $\widetilde{\mathcal{O}}\left(((1+\varepsilon)^2/\gamma^2 m)^{1/3}\right)$ as shown in the following theorem.

Theorem 5.3.4. *With high probability*

$$L_0^\varepsilon(f_{\widehat{\mathbf{w}}}) \leq \widehat{L}_\gamma^\varepsilon(f_{\mathbf{w}}) + \widetilde{\mathcal{O}}\left(\left(\frac{(1+\varepsilon)^2}{\gamma^2 m}\right)^{1/3}\right),$$

where $\widetilde{\mathcal{O}}(\cdot)$ ignores logarithmic factors.

Proof (Theorem 5.3.4). Let $\mathcal{A}$ be the set of vectors with at most $\mathcal{O}((\log n)(1+\varepsilon)^2/\delta\gamma^2)$ non-zero entries, where each entry is a multiple of $2\gamma/2n(1+\varepsilon)$ between $-\delta\gamma^2/(1+\varepsilon)^2$ and $\delta\gamma^2/(1+\varepsilon)^2$. Then, $|\mathcal{A}| = r^q$ with

$$r = 2\frac{\delta\gamma^2/(1+\varepsilon)^2}{2\gamma/2n(1+\varepsilon)} = \frac{4n\delta\gamma}{(1+\varepsilon)}, \qquad q = \frac{(1+\varepsilon)^2}{\delta\gamma^2}.$$

Let $\widehat{\mathbf{w}}$ be defined as in Lemma 5.3.3. Then, by Lemma 5.3.2, we have that $\mathbb{P}_{\widehat{\mathbf{w}}}[\widehat{\mathbf{w}} \in \mathcal{A}] \leq 1-\delta$. We define $\mathcal{G} = \{f_{\widehat{\mathbf{w}}} : \widehat{\mathbf{w}} \in \mathcal{A}\}$. Note that the mapping from $f_{\mathbf{w}}$ to $f_{\widehat{\mathbf{w}}}$ fails (*i.e.*, $\widehat{\mathbf{w}} \notin \mathcal{A}$) with probability at most δ, thus corollary 5.2.2.1 yields

$$L_0^\varepsilon(f_{\widehat{\mathbf{w}}}) \leq \widehat{L}_\gamma^\varepsilon(f_{\mathbf{w}}) + \mathcal{O}\left(\sqrt{\frac{(1+\varepsilon)^2\log(n)\log\left(\frac{4n\delta\gamma}{(1+\varepsilon)}\right)}{\delta\gamma^2 m}}\right) + \delta = \widehat{L}_\gamma^\varepsilon(f_{\mathbf{w}}) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{(1+\varepsilon)^2}{\delta\gamma^2 m}}\right) + \delta$$

with high probability. Then, we choose $\delta = ((1+\varepsilon)^2/\gamma^2 m)^{1/3}$ which leads to

$$L_0^\varepsilon(f_{\widehat{\mathbf{w}}}) \leq \widehat{L}_\gamma^\varepsilon(f_{\mathbf{w}}) + \widetilde{\mathcal{O}}\left(\left(\frac{(1+\varepsilon)^2}{\gamma^2 m}\right)^{1/3}\right)$$

with high probability. □

This approach is fairly similar to the original one in the work of Arora *et al.* [89], but the p_i values are chosen differently in order to deal with the new term $\varepsilon \|\mathbf{w}\|_1$ that appears in the margin's expression.

This result provides a dimension-free bound[2]. However, that bound scales with $m^{1/3}$ instead of $\sqrt{m}$, since the compression approach fails with probability δ. To tackle this issue, Arora *et al.* [89] proposed a compression algorithm based on random projections. In their setup, this technique works due to a famous corollary of Johnson-Lindenstrauss lemma that shows that we can construct random projections which preserve the inner product $\langle \mathbf{w}, \mathbf{x} \rangle$. In addition, since the Euclidean inner product can be induced by the ℓ_2-norm, the ℓ_2-norm of $\mathbf{w}$ is preserved as well. However, in this setup we would need a random projection that preserves $\|\mathbf{w}\|_1$ and $\langle \mathbf{w}, \mathbf{x} \rangle$ at the same time, which seems unattainable unless additional assumptions are made. Therefore, we propose to assume an effective sparsity bound on $\mathbf{w}$, which is defined as follows.

Definition 5.3.5 (Effective $\overline{s}$-sparsity)**.** *A vector $\mathbf{w} \in \mathbb{R}^n$ is effectively $\overline{s}$-sparse, with $\overline{s} \in [1, n]$, if*

$$\|\mathbf{w}\|_{1/2} \leq \overline{s} \, \|\mathbf{w}\|_1 \ .$$

Note that for all $s = 1, 2, \ldots$, it holds that any s-sparse vector (*i.e.*, a vector with at most s non-zero entries) is effectively s-sparse, but not vice-versa. Assuming that $\mathbf{w}$ is effectively sparse allows us to compress it by simply setting its lowest entries to zero. The following lemma provides a tight bound on the error, in the ℓ_1 sense, that is caused by this process.

Lemma 5.3.6 ([47]: Theorem 2.5)**.** *For any $\mathbf{w} \in \mathbb{R}^n$ the following inequalities hold:*

$$\inf \{\|\mathbf{w} - \mathbf{z}\|_1 : \mathbf{z} \text{ is } s\text{-sparse}\} \leq \frac{1}{4s} \|\mathbf{w}\|_{1/2} \ ,$$
$$\inf \{\|\mathbf{w} - \mathbf{z}\|_\infty : \mathbf{z} \text{ is } s\text{-sparse}\} \leq \frac{1}{s} \|\mathbf{w}\|_1 \ .$$

In both cases, the infimum is attained when $\mathbf{z}$ is an s-sparse vector whose non-zero entries are the s-largest absolute entries of $\mathbf{w}$.

For any effectively $\overline{s}$-sparse classifier $\mathbf{w}$ with margin γ, this lemma allows us to compress it into a vector $\widehat{\mathbf{w}}$, with only $\mathcal{O}(\overline{s}(1+\varepsilon)/\gamma)$ non-zero entries, such that both classifiers assign the same label to any input.

Lemma 5.3.7. *Given an effectively $\overline{s}$-sparse vector $\mathbf{w} \in \mathcal{B}^n_{1,1}$, let us define $\mathbf{w}' \in \mathcal{B}^n_{1,1}$ as the s-sparse vector whose non-zero entries are the s-largest absolute entries of $\mathbf{w}$. In addition, the vector $\widehat{\mathbf{w}}$ is obtained by rounding each entry of $\mathbf{w}'$ to the nearest multiple of $\gamma/s(1+\varepsilon)$. If we choose $s = \overline{s}(1+\varepsilon)/2\gamma$ then*

$$\forall \mathbf{x} \in \mathcal{B}^n_{\infty,1}, y \in \mathcal{Y} : \quad |\ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)| \leq \gamma \, .$$

[2]Except for logarithmic terms.

Proof (Lemma 5.3.7). Let us first bound how much does sparsifying $\mathbf{w}$ affects inner products, that is

$$|\langle \mathbf{w}, \mathbf{x} \rangle - \langle \mathbf{w}', \mathbf{x} \rangle| \leq \|\mathbf{w} - \mathbf{w}'\|_1 \|\mathbf{x}\|_\infty \leq \|\mathbf{w} - \mathbf{w}'\|_1 .$$

This distorts the adversarial margin as follows:

$$\begin{aligned}
&|\ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\mathbf{w}'}; \mathbf{x}, y)| \\
&\leq |\langle \mathbf{w}, \mathbf{x} \rangle - \langle \mathbf{w}', \mathbf{x} \rangle| + \varepsilon \, |\|\mathbf{w}\|_1 - \|\mathbf{w}'\|_1| \\
&\leq \|\mathbf{w} - \mathbf{w}'\|_1 + \varepsilon \|\mathbf{w} - \mathbf{w}'\|_1 && \text{(triangle inequality)} \\
&= (1 + \varepsilon) \|\mathbf{w} - \mathbf{w}'\|_1 \\
&\leq (1 + \varepsilon) \frac{1}{4s} \|\mathbf{w}\|_{1/2} && \text{(Lemma 5.3.6)} \\
&\leq (1 + \varepsilon) \frac{\overline{s}}{4s} \|\mathbf{w}\|_1 && \text{(Definition of effective sparsity)} \\
&= \gamma/2 \, . && \text{(Choice of } s\text{)}
\end{aligned}$$

Similarly,

$$\begin{aligned}
|\ell_\varepsilon(f_{\mathbf{w}'}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)| &\leq (1 + \varepsilon) \|\mathbf{w}' - \widehat{\mathbf{w}}\|_1 \\
&\leq (1 + \varepsilon) s \frac{1}{2} \left(\frac{\gamma}{s(1 + \varepsilon)} \right) \\
&= \gamma/2 \, .
\end{aligned}$$

Putting all together, we get

$$\begin{aligned}
|\ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)| &\leq |\ell_\varepsilon(f_{\mathbf{w}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\mathbf{w}'}; \mathbf{x}, y)| + |\ell_\varepsilon(f_{\mathbf{w}'}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\mathbf{w}}}; \mathbf{x}, y)| \\
&\leq \gamma/2 + \gamma/2 \, = \gamma \, ,
\end{aligned}$$

which completes the proof. □

Since this compression approach does not fail, we can discretize $\widehat{\mathbf{w}}$ and apply Theorem 5.2.2. This allows to prove the following generalization bound for the linear classifier in the presence of an adversary.

Theorem 5.3.8. *Let* $\mathbf{w}$ *be any linear classifier with* $\|\mathbf{w}\|_{1/2} / \|\mathbf{w}\|_1 \leq \overline{s}$*, and margin* $\gamma > 0$ *on the training set* $\mathcal{S}$*. Then, if* $|\mathcal{S}| = m$*, with high probability the adversarial risk is bounded by*

$$L_0^\varepsilon(f_{\widehat{\mathbf{w}}}) \leq \widehat{L}_\gamma^\varepsilon(f_{\mathbf{w}}) + \widetilde{\mathcal{O}} \left(\sqrt{\frac{(1 + \varepsilon)\overline{s}}{\gamma m}} \right) ,$$

where $\widetilde{\mathcal{O}}(\cdot)$ *ignores logarithmic factors.*

Proof (Theorem 5.3.8). Let $\mathcal{A}$ be the set of vectors with at most $\overline{s}(1+\varepsilon)/2\gamma$ non-zero entries, where each entry is a multiple of $\gamma/s(1+\varepsilon)$ between -1 and 1. Then, $|\mathcal{A}| = r^q$ with

$$r = \frac{2}{\gamma/s(1+\varepsilon)} = \frac{2}{\gamma^2/2\overline{s}(1+\varepsilon)^2} = \frac{4\overline{s}(1+\varepsilon)^2}{\gamma^2}\,, \qquad q = s = \overline{s}(1+\varepsilon)/2\gamma\,.$$

Let $\mathcal{G} = \{f_{\widehat{\mathbf{w}}} : \widehat{\mathbf{w}} \text{ is defined as in Lemma 5.3.7 with } \mathbf{w} \in \mathcal{B}^n_{1,1}\}$. Then, by Lemma 5.3.7 we know that $f_{\mathbf{w}}$ is $(\gamma, \varepsilon, \mathcal{S})$-compressible via $\mathcal{G}$, thus Theorem 5.2.2 yields

$$L^{\varepsilon}_0(f_{\widehat{\mathbf{w}}}) \leq \widehat{L}^{\varepsilon}_{\gamma}(f_{\mathbf{w}}) + \mathcal{O}\left(\sqrt{\frac{2\overline{s}(1+\varepsilon)\log\left(\frac{4\overline{s}(1+\varepsilon)^2}{\gamma^2}\right)}{\gamma m}}\right) = \widehat{L}^{\varepsilon}_{\gamma}(f_{\mathbf{w}}) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{(1+\varepsilon)\overline{s}}{\gamma m}}\right)$$

with high probability. □

This result provides a bound with no explicit dimension dependence. Moreover, we observe that the presence of an adversary only increases the sample complexity by a factor $(1+\varepsilon)$.

5.3.2. Neural Networks

Due to the ℓ_∞-norm bound on the perturbation $\boldsymbol{\eta}$, in this chapter the mixed $(1,\infty)$-norm of the weight matrices plays a central role. As an example, let us consider a linear classifier in multi-class classification, that is $f(\mathbf{x}) = \boldsymbol{W}^\top \mathbf{x}$. Then, a perturbation $\boldsymbol{\eta}$ can perturb any entry of the vector of score functions (*i.e.*, $f(\mathbf{x})$) at most

$$\sup_{\|\boldsymbol{\eta}\|_\infty \leq 1} \|\boldsymbol{W}^\top \boldsymbol{\eta}\|_\infty = \|\boldsymbol{W}^\top\|_\infty = \|\boldsymbol{W}\|_{1,\infty}\,.$$

The last equality comes from the properties of operator norms, see [17] for more details. Similar statements can be made for the layers of a neural network with 1-Lipschitz activation functions. Let us start by defining a d-layered fully connected neural network as

$$\mathbf{x}^i := \phi(\boldsymbol{W}^{i^\top} \mathbf{x}^{i-1})\,, \quad \forall i = 1, 2, \ldots, d\,, \tag{5.1}$$

where ϕ is a 1-Lipschitz activation function applied entry-wise, $\mathbf{x}^0 := \mathbf{x}$ and $f(\mathbf{x}) := \mathbf{x}^d$. Then, the following lemma allows us to quantify how much error is incurred by perturbing the input of a layer, or by switching the matrix $\boldsymbol{W}$ to a different one.

Lemma 5.3.9. *If ϕ is a 1-Lipschitz activation function, then for any $\boldsymbol{W}, \widehat{\boldsymbol{W}}$ the following inequalities hold*

$$\left\|\phi(\boldsymbol{W}^\top \mathbf{x}) - \phi(\boldsymbol{W}^\top(\mathbf{x}+\boldsymbol{\eta}))\right\|_\infty \leq \|\boldsymbol{W}\|_{1,\infty} \|\boldsymbol{\eta}\|_\infty\,,$$
$$\left\|\phi(\boldsymbol{W}^\top \mathbf{x}) - \phi(\widehat{\boldsymbol{W}}^\top \mathbf{x})\right\|_\infty \leq \left\|\boldsymbol{W} - \widehat{\boldsymbol{W}}\right\|_{1,\infty} \|\mathbf{x}\|_\infty\,.$$

Proof (Lemma 5.3.9). Since ϕ is 1-Lipschitz, we have that for any vector $\mathbf{w}$ of the same size as $\boldsymbol{\eta}$, it holds

$$|\phi(\langle \mathbf{w}, \mathbf{x} \rangle) - \phi(\langle \mathbf{w}, \mathbf{x} + \boldsymbol{\eta} \rangle)| \leq |\langle \mathbf{w}, \boldsymbol{\eta} \rangle| \leq \|\mathbf{w}\|_1 \|\boldsymbol{\eta}\|_\infty \,.$$

This proves the first inequality of the lemma. Similarly, for any $\mathbf{w}$ and $\widehat{\mathbf{w}}$ it follows

$$|\phi(\langle \mathbf{w}, \mathbf{x} \rangle) - \phi(\langle \widehat{\mathbf{w}}, \mathbf{x} \rangle)| \leq |\langle \mathbf{w} - \widehat{\mathbf{w}}, \mathbf{x} \rangle| \leq \|\mathbf{w} - \widehat{\mathbf{w}}\|_1 \|\mathbf{x}\|_\infty \,,$$

thus implying the second inequality. □

Following the steps of Section 5.3.1, we now impose some conditions on $\boldsymbol{W}$ that allow us to efficiently compress it into another matrix $\widehat{\boldsymbol{W}}$ which belongs to a potentially small set. To that end, let us start by introducing the notion of effective joint sparsity.

Definition 5.3.10 (Effective joint sparsity)**.** *A matrix $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n_2}$ is effectively joint $\overline{s}$-sparse, with $\overline{s} \in [1, n_2]$, if*

$$\|\boldsymbol{W}\|_{1,1} \leq \overline{s} \, \|\boldsymbol{W}\|_{1,\infty} \,.$$

Any matrix with s non-zero columns is effectively joint s-sparse as well, but not vice-versa. Note that, given a matrix $\boldsymbol{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$, its effectively joint sparsity can be written as the effective sparsity of the vector $(\|\mathbf{w}_1\|_1, \ldots, \|\mathbf{w}_n\|_1)^\top$. A consequence of Lemma 5.3.6 is that we can compress effectively joint-sparse matrices by setting to zero their columns with lowest ℓ_1-norm. For example, assume that $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n_2}$ is an effective joint $\overline{s}$-sparse matrix and that $\widehat{\boldsymbol{W}}$ is constructed by setting to zero all columns of $\boldsymbol{W}$ except for its s largest in the ℓ_1 sense. Then, by Lemma 5.3.6, we can bound the $\|\cdot\|_{1,\infty}$ error as

$$\left\|\boldsymbol{W} - \widehat{\boldsymbol{W}}\right\|_{1,\infty} \leq \frac{1}{s} \|\boldsymbol{W}\|_{1,1} \leq \frac{\overline{s}}{s} \|\boldsymbol{W}\|_{1,\infty} \,.$$

The resulting compressed matrix $\widehat{\boldsymbol{W}}$ would have only s non-zero columns instead of the original n_2. However, every column has potentially n_1 non-zero values. Therefore, in order to compress $\boldsymbol{W}$ further, we assume that each one of its columns has bounded effective sparsity as well. In summary, effective joint sparsity allows us to reduce the number of non-zero columns in a matrix, while effective sparsity of the columns allows us to reduce the number of non-zero elements that each of those columns may have. Finally, discretization is handled using a standard covering number argument. Putting all together into the following compression algorithm (Algorithm 2) allows us to map $\boldsymbol{W}$ into a discrete set while keeping the $\|\cdot\|_{1,\infty}$ error bounded.

By construction, using this algorithm guarantees that the error is bounded, as stated in the following lemma.

Lemma 5.3.11. *Let $\boldsymbol{W}$ be an effectively joint $\overline{s}_2$-sparse matrix with effectively $\overline{s}_1$-sparse columns, such that $\|\boldsymbol{W}\|_{1,\infty} \leq 1$. If $\widehat{\boldsymbol{W}} = \text{MatrixCompress}\,(\boldsymbol{W}, \gamma)$, then*

$$\left\|\boldsymbol{W} - \widehat{\boldsymbol{W}}\right\|_{1,\infty} \leq \gamma \,,$$

where $\widehat{\boldsymbol{W}}$ belongs to a discrete set $\mathcal{C}$ such that $\log|\mathcal{C}| \leq \widetilde{\mathcal{O}}\left(\|\boldsymbol{W}\|_{1,\infty}^2 \, \overline{s}_1 \overline{s}_2 / \gamma^2\right)$.

Algorithm 2 MatrixCompress $(\cdot, \gamma)$

Require: $\gamma > 0$ and $\boldsymbol{W} \in \mathbb{R}^{n_1 \times n_2}$ with $\|\boldsymbol{W}\|_{1,\infty} = 1$, effectively $\overline{s}_1$-sparse columns and is effectively joint $\overline{s}_2$-sparse

Ensure:

$$\left\|\boldsymbol{W} - \widehat{\boldsymbol{W}}\right\|_{1,\infty} \leq \gamma\,,$$

where $\widehat{\boldsymbol{W}}$ belongs to a discrete set $\mathcal{C}$ such that $\log|\mathcal{C}| \leq \widetilde{\mathcal{O}}\left(\|\boldsymbol{W}\|_{1,\infty}^2\, \overline{s}_1 \overline{s}_2/\gamma^2\right)$

Choose $s_1 = 3\,\|\boldsymbol{W}\|_{1,\infty}\, \overline{s}_1/4\gamma$ and $s_2 = 3\,\|\boldsymbol{W}\|_{1,\infty}\, \overline{s}_2/\gamma$

Let $\overline{\boldsymbol{W}} \in \mathbb{R}^{n_1 \times n_2}$ be obtained by setting to zero the columns of $\boldsymbol{W}$ except for the s_2 columns with largest ℓ_1 norm

Let $\widetilde{\boldsymbol{W}} \in \mathbb{R}^{n_1 \times n_2}$ be constructed by keeping the s_1 largest values of every column of $\overline{\boldsymbol{W}}$ and setting to zero the other entries

Let $\mathcal{W}$ be the set all possible $\widetilde{\boldsymbol{W}}$

Let $\mathcal{C}$ be the covering set of $\mathcal{W}$ such that $\forall \widetilde{\boldsymbol{W}} \in \mathcal{W}, \exists \widehat{\boldsymbol{W}} \in \mathcal{C} : \left\|\widetilde{\boldsymbol{W}} - \widehat{\boldsymbol{W}}\right\|_{1,\infty} \leq \gamma/3$

Let $\widehat{\boldsymbol{W}} \in \mathcal{C}$ be the closest matrix in the $\|\cdot\|_{1,\infty}$ sense to $\widetilde{\boldsymbol{W}}$

Return: $\widehat{\boldsymbol{W}}$

Proof (Lemma 5.3.11). Since $\boldsymbol{W}$ is effectively joint sparse, we can bound $\left\|\boldsymbol{W} - \overline{\boldsymbol{W}}\right\|_{1,\infty}$ as follows

$$\begin{aligned}
\left\|\boldsymbol{W} - \overline{\boldsymbol{W}}\right\|_{1,\infty} &\leq \frac{1}{s_2}\,\|\boldsymbol{W}\|_{1,1} && \text{(Lemma 5.3.6)}\\
&\leq \frac{\overline{s}_2}{s_2}\,\|\boldsymbol{W}\|_{1,\infty}\,. && \text{(Definition of effective joint sparsity)}
\end{aligned}$$

Similarly, since the remaining non-zero columns $\overline{\boldsymbol{W}}$ are effectively sparse, we get

$$\begin{aligned}
\left\|\overline{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}\right\|_{1,\infty} &= \inf_{\boldsymbol{X}:\|\boldsymbol{X}\|_{0,\infty}=s_1} \left\|\overline{\boldsymbol{W}} - \boldsymbol{X}\right\|_{1,\infty}\\
&\leq \frac{1}{4s_1}\left\|\overline{\boldsymbol{W}}\right\|_{1/2,\infty} && \text{(Lemma 5.3.6)}\\
&\leq \frac{\overline{s}_1}{4s_1}\left\|\overline{\boldsymbol{W}}\right\|_{1,\infty}\,. && \text{(Definition of effective sparsity)}
\end{aligned}$$

By the definition of $\widehat{\boldsymbol{W}}$, we have that $\left\|\widetilde{\boldsymbol{W}} - \widehat{\boldsymbol{W}}\right\|_{1,\infty} \leq \gamma/3$. Combining all these statements, the choice of s_1 and s_2 (see Algorithm 2) yields

$$\begin{aligned}
\left\|\boldsymbol{W} - \widehat{\boldsymbol{W}}\right\|_{1,\infty} &\leq \left\|\boldsymbol{W} - \overline{\boldsymbol{W}}\right\|_{1,\infty} + \left\|\overline{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}\right\|_{1,\infty} + \left\|\widetilde{\boldsymbol{W}} - \widehat{\boldsymbol{W}}\right\|_{1,\infty}\\
&\leq \frac{\overline{s}_1}{4s_1}\,\|\boldsymbol{W}\|_{1,\infty} + \frac{\overline{s}_2}{s_2}\,\|\boldsymbol{W}\|_{1,\infty} + \frac{\gamma}{3}\\
&\leq \frac{\gamma}{3} + \frac{\gamma}{3} + \frac{\gamma}{3} = \gamma\,.
\end{aligned}$$

It remains to bound the covering number of $\mathcal{W}$ with the mixed $(1,\infty)$-norm, denoted by $\mathcal{N}(\mathcal{W}, \|\cdot\|_{1,\infty}, \gamma/3)$. By definition, the set $\mathcal{W}$ is composed of all matrices $\widetilde{\boldsymbol{W}}$ with

at most s_2 non-zero columns, where each column has at most s_1 non-zero entries and ℓ_1-norm not greater than one. Since any $\widehat{\boldsymbol{W}} \in \mathcal{W}$ has at most s_2 non-zero columns, we get

$$\begin{aligned}
\mathcal{N}(\mathcal{W}, \|\cdot\|_{1,\infty}, \gamma/3) &\leq \binom{n_2}{s_2} \mathcal{N}(\gamma/3, \mathcal{B}_{1,1}^{n_1} \cap \mathcal{B}_{0,s_1}^{n_1}, \|\cdot\|_1)^{s_2} \\
&\leq \binom{n_2}{s_2} \left[\binom{n_1}{s_1} \mathcal{N}(\gamma/3, \mathcal{B}_{1,1}^{s_1}, \|\cdot\|_1) \right]^{s_2} \\
&\leq \left(\frac{en_2}{s_2}\right)^{s_2} \left[\left(\frac{en_1}{s_1}\right)^{s_1} \mathcal{N}(\gamma/3, \mathcal{B}_{1,1}^{s_1}, \|\cdot\|_1) \right]^{s_2} \\
&\leq \left(\frac{en_2}{s_2}\right)^{s_2} \left(\frac{en_1}{s_1}\right)^{s_1 s_2} \left(1 + \frac{6}{\gamma}\right)^{s_1 s_2} .
\end{aligned}$$

This leads to

$$\log \mathcal{N}(\mathcal{W}, \|\cdot\|_{1,\infty}, \gamma/3) \leq \widetilde{\mathcal{O}}\left(s_1 s_2\right) = \widetilde{\mathcal{O}}\left(\|\boldsymbol{W}\|_{1,\infty}^2 \, \overline{s}_1 \overline{s}_2 / \gamma^2 \right) .$$

choosing $\mathcal{C}$ to be the covering set of $\mathcal{W}$ completes the proof. □

From this lemma, we can see that the set of possible compressed matrices has reasonable size. Moreover, approximately sparse matrices can be compressed efficiently. This result leads us to the main contribution of this chapter, which is stated in the following theorem.

Theorem 5.3.12. *Assume $\mathbf{x} \in \mathcal{B}_{\infty,1}^n$. Let $f_{\boldsymbol{W}}$ be a d-layer neural network with ReLU activations, and effectively joint $\overline{s}_2^j$-sparse weight matrices with effectively $\overline{s}_1^j$-sparse columns for $j = 1, \ldots, d$. Assume that the network is rebalanced so that $\left\|\boldsymbol{W}^1\right\|_{1,\infty} = \cdots = \left\|\boldsymbol{W}^d\right\|_{1,\infty} = 1$. Then, given $\gamma > 0$ and $\varepsilon < \gamma/4$, there exists a finite function set $\mathcal{G}$ composed of the functions $f_{\widehat{\boldsymbol{W}}}$ such that for any $f_{\boldsymbol{W}}$ the adversarial risk is bounded as*

$$L_0^{\varepsilon}(f_{\widehat{\boldsymbol{W}}}) \leq \widehat{L}_{\gamma}^{\varepsilon}(f_{\boldsymbol{W}}) + \widetilde{\mathcal{O}}\left(\sqrt{ \frac{d}{m} \left(\frac{1 + \gamma/2 - \varepsilon}{\gamma/2 - 2\varepsilon} \right)^2 \left(\sum_{j=1}^{d} \sqrt{\overline{s}_1^j \overline{s}_2^j} \right)^2 } \right)$$

with high probability.

Proof (Theorem 5.3.12). By assumption, the activation functions are all set to be the ReLU activation ϕ. Then, due to its positive homogeneity property, we re-balance the network by setting $\left\|\boldsymbol{W}^i\right\|_{1,\infty} = 1$ for all $i = 1, \ldots, d$ without altering the classification function. For any given adversarial noise $\boldsymbol{\eta}_1$ with ℓ_∞-norm bounded by ε, let us re-define $\mathbf{x}^i$ as in (5.1) but with $\mathbf{x}^0 = \mathbf{x} + \boldsymbol{\eta}_1$. Similarly, for another adversarial noise $\boldsymbol{\eta}_2$ with ℓ_∞-norm bounded by ε and compressed matrices $\widehat{\boldsymbol{W}}^i$, let us define the error vector of the i-th layer $\boldsymbol{\eta}^i$ in a recursive fashion, that is $\boldsymbol{\eta}^i := \phi(\boldsymbol{W}^{i^\top} \mathbf{x}^{i-1}) - \phi(\widehat{\boldsymbol{W}}^{i^\top} (\mathbf{x}^{i-1} + \boldsymbol{\eta}^{i-1}))$

for $i = 1, \dots, d$ with $\boldsymbol{\eta}^0 := \boldsymbol{\eta}_2 - \boldsymbol{\eta}_1$. Note that $\|\boldsymbol{\eta}^0\|_\infty \leq 2\varepsilon$. With this definition of $\mathbf{x}^i$, since

$$\left\|\phi(\boldsymbol{W}^{i^\top}\mathbf{x}^{i-1})\right\|_\infty \leq \left\|\boldsymbol{W}^{i^\top}\mathbf{x}^{i-1}\right\|_\infty \leq \left\|\boldsymbol{W}^{i^\top}\right\|_\infty \left\|\mathbf{x}^{i-1}\right\|_\infty = \left\|\boldsymbol{W}^{i}\right\|_{1,\infty} \left\|\mathbf{x}^{i-1}\right\|_\infty$$

we have that $\|\mathbf{x}^i\|_\infty \leq \|\mathbf{x}^0\|_\infty \prod_{j=1}^{i} \left\|\boldsymbol{W}^j\right\|_{1,\infty} \leq 1 + \varepsilon$.

Our first goal is to bound $\|\boldsymbol{\eta}^i\|_\infty$ for $i = 1, 2, \dots, d$, which we do by induction. For any $i > 0$, let us assume that $\|\boldsymbol{\eta}^{i-1}\| \leq \varepsilon^{i-1}$ where ε^{i-1} is some positive value. Given some $\varepsilon^i > \varepsilon^{i-1}$, we compress $\boldsymbol{W}^i$ as $\widehat{\boldsymbol{W}}^i = \text{MatrixCompress}((\varepsilon^i - \varepsilon^{i-1})/(1 + \varepsilon + \varepsilon^{i-1}), \boldsymbol{W}^i)$. Then, using Lemma 5.3.9, we get

$$\begin{aligned}
\left\|\boldsymbol{\eta}^i\right\|_\infty &= \left\|\phi(\boldsymbol{W}^{i^\top}\mathbf{x}^{i-1}) - \phi(\widehat{\boldsymbol{W}}^{i^\top}(\mathbf{x}^{i-1} + \boldsymbol{\eta}^{i-1}))\right\|_\infty \\
&= \quad \left\|\phi(\boldsymbol{W}^{i^\top}\mathbf{x}^{i-1}) - \phi(\boldsymbol{W}^{i^\top}(\mathbf{x}^{i-1} + \boldsymbol{\eta}^{i-1}))\right\|_\infty \\
&\quad + \left\|\phi(\boldsymbol{W}^{i^\top}(\mathbf{x}^{i-1} + \boldsymbol{\eta}^{i-1})) - \phi(\widehat{\boldsymbol{W}}^{i^\top}(\mathbf{x}^{i-1} + \boldsymbol{\eta}^{i-1}))\right\|_\infty \\
&\leq \left\|\boldsymbol{W}^i\right\|_{1,\infty} \left\|\boldsymbol{\eta}^{i-1}\right\|_\infty + \left\|\boldsymbol{W}^i - \widehat{\boldsymbol{W}}^i\right\|_{1,\infty} \left\|\mathbf{x}^{i-1} + \boldsymbol{\eta}^{i-1}\right\|_\infty \qquad \text{(Lemma 5.3.9)} \\
&\leq \varepsilon^{i-1} + \left\|\boldsymbol{W}^i - \widehat{\boldsymbol{W}}^i\right\|_{1,\infty} (1 + \varepsilon + \varepsilon^{i-1}) \\
&\leq \varepsilon^i\,. \qquad \text{(Definition of } \widehat{\boldsymbol{W}}^i\text{)}
\end{aligned}$$

Given y and $f_{\boldsymbol{W}}$, let us define $\widetilde{f}_{\boldsymbol{W}}(\mathbf{x}) := [f_{\boldsymbol{W}}(\mathbf{x})]_{j \neq y}$. By setting $\varepsilon^0 := 2\varepsilon$ and $\varepsilon^d := \gamma/2$, we get

$$\begin{aligned}
&\left|\ell_\varepsilon(f_{\boldsymbol{W}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\boldsymbol{W}}}; \mathbf{x}, y)\right| \\
&= \left|[f_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1)]_y - \max_{j \neq y}[f_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1)]_j - [f_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)]_y + \max_{j \neq y}[f_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)]_j\right| \\
&= \left|[f_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1)]_y - \left\|\widetilde{f}_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1)\right\|_\infty - [f_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)]_y + \left\|\widetilde{f}_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)\right\|_\infty\right| \\
&\leq \left|[f_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1)]_y - [f_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)]_y\right| + \left|\left\|\widetilde{f}_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1)\right\|_\infty - \left\|\widetilde{f}_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)\right\|_\infty\right| \\
&\leq \left\|f_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1) - f_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)\right\|_\infty + \left\|\widetilde{f}_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1) - \widetilde{f}_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)\right\|_\infty \\
&\leq 2\left\|f_{\boldsymbol{W}}(\mathbf{x} + \boldsymbol{\eta}_1) - f_{\widehat{\boldsymbol{W}}}(\mathbf{x} + \boldsymbol{\eta}_2)\right\|_\infty \\
&= 2\left\|\boldsymbol{\eta}^d\right\|_\infty \leq \gamma\,.
\end{aligned}$$

We are free to choose $\varepsilon^1, \dots, \varepsilon^{d-1}$ without loosing this bound on $\left|\ell_\varepsilon(f_{\boldsymbol{W}}; \mathbf{x}, y) - \ell_\varepsilon(f_{\widehat{\boldsymbol{W}}}; \mathbf{x}, y)\right|$, as long as $\varepsilon^i > \varepsilon^{i-1}$. However, the choice of these values will determine the sample complexity of the compressed function class. A naive way of choosing ε^i, like $\varepsilon^i := i(\gamma/2 - 2\varepsilon)/d + 2\varepsilon$, will lead to a sample complexity of $\mathcal{O}(d^2)$ instead of $\mathcal{O}(d)$. Therefore, we choose these parameters in a smarter way, that is

$$\varepsilon^0 := 2\varepsilon\,, \qquad \varepsilon^i := \varepsilon^{i-1} + \frac{\sqrt{\overline{s}_1^i \overline{s}_2^i}}{\sum_{j=1}^{d} \sqrt{\overline{s}_1^j \overline{s}_2^j}}(\gamma/2 - 2\varepsilon)\,,$$

so that more error is allocated to the layers with more effective parameters. Note that this selection implies $\varepsilon^d = \gamma/2$ and $\varepsilon^i > \varepsilon^{i-1}$, so $f_{\boldsymbol{W}}$ is $(\gamma, \varepsilon, \mathcal{S})$-compressible via $\mathcal{G} = \{f_{\widehat{\boldsymbol{W}}} : \widehat{\boldsymbol{W}} = \text{MatrixCompress}((\varepsilon^i - \varepsilon^{i-1})/(1+\varepsilon+\varepsilon^{i-1}), \boldsymbol{W})\}$. In the same manner as in Lemma 5.3.9, for all $i = 1, \dots, d$ let us define $\mathcal{C}^i$ to be the set of all possible $\widehat{\boldsymbol{W}}^i$. With this choice, the logarithm of the cardinality of the compressed function class is

$$\begin{aligned}
\log|\mathcal{G}| &= \log \prod_{i=1}^{d} \left|\mathcal{C}^i\right| = \sum_{i=1}^{d} \log \left|\mathcal{C}^i\right| \\
&\leq \widetilde{\mathcal{O}}\left(\sum_{i=1}^{d} \overline{s}_1^i \overline{s}_2^i (1+\varepsilon+\varepsilon^{i-1})^2/(\varepsilon^i - \varepsilon^{i-1})^2\right) \\
&\leq \widetilde{\mathcal{O}}\left(\sum_{i=1}^{d} \frac{\overline{s}_1^i \overline{s}_2^i (1+\varepsilon+\gamma/2-2\varepsilon)^2 \left(\sum_{j=1}^{d} \sqrt{\overline{s}_1^j \overline{s}_2^j}\right)^2}{\left((\gamma/2-2\varepsilon)\sqrt{\overline{s}_1^i \overline{s}_2^i}\right)^2}\right) \\
&= \widetilde{\mathcal{O}}\left(\sum_{i=1}^{d} \frac{(1+\gamma/2-\varepsilon)^2 \left(\sum_{j=1}^{d} \sqrt{\overline{s}_1^j \overline{s}_2^j}\right)^2}{(\gamma/2-2\varepsilon)^2}\right) \\
&= \widetilde{\mathcal{O}}\left(d \left(\frac{1+\gamma/2-\varepsilon}{\gamma/2-2\varepsilon}\right)^2 \left(\sum_{j=1}^{d} \sqrt{\overline{s}_1^j \overline{s}_2^j}\right)^2\right) .
\end{aligned}$$

Finally, we apply Theorem 5.2.2, yielding

$$L_0^{\varepsilon}(f_{\widehat{\boldsymbol{W}}}) \leq \widehat{L}_{\gamma}^{\varepsilon}(f_{\boldsymbol{W}}) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{d}{m}\left(\frac{1+\gamma/2-\varepsilon}{\gamma/2-2\varepsilon}\right)^2 \left(\sum_{j=1}^{d} \sqrt{\overline{s}_1^j \overline{s}_2^j}\right)^2}\right)$$

which proves the theorem. □

This result proves a bound with no explicit dimension dependence, which is also independent from the number of classes. On the other hand, there seems to be an unavoidable dependence with $\sqrt{d}$. Yet, this dependence is also present in the bounds for multi-layer neural networks, derived in related works [98], [99]. The rebalancing in Theorem 5.3.12 simplifies the proof by getting rid of the term $\prod_{j=1}^{d} \|\boldsymbol{W}^j\|_{1,\infty}$, which appears in other works such as [98]. Note that, for ReLU networks, rebalancing does not affect the labels that $f_{\boldsymbol{W}}$ assigns to the inputs. However, by the definition of ℓ_ε and L_γ^ε, in practice, γ cannot be larger than $2\prod_{j=1}^{d} \|\boldsymbol{W}^j\|_{1,\infty}$. Then, the requirement $\varepsilon < \gamma/4$, in the setup of Theorem 5.3.12, limits the use of this result to $\varepsilon < 0.5$, or less for neural networks with smaller classification margins. Nonetheless, considering that $\mathbf{x} \in \mathcal{B}_{\infty,1}^n$, this requirement may not be extremely restrictive, since $\varepsilon = 0.5$ is a rather high value. Despite this shortcoming, Theorem 5.3.12 improves existing bounds in other aspects, as shown in Table 5.1. For instance, we observe that Theorem 5.3.12 improves existing works

		Rebalancing	Generalization Bound
[98]	$\|\mathbf{x}\|_2 \leq R$	$\left\|\boldsymbol{W}^j\right\|_{1,\infty} = 1$	$\widetilde{\mathcal{O}}\left(\sqrt{\frac{d}{m}k^2(R\max_j \left\|\boldsymbol{W}^j\right\|_{\mathrm{F}} + \varepsilon)^2}\right)$
[99]	$\|\mathbf{x}\|_2 \leq R$	$\left\|\boldsymbol{W}^j\right\|_2 = 1$	$\widetilde{\mathcal{O}}\left(\lambda_f^+\varepsilon + \sqrt{\frac{R^2}{m}\left(n^2\left(\sum_{j=1}^d \sqrt{\left\|\boldsymbol{W}^j\right\|_{\mathrm{F}}}\right)^2 + \Lambda_\varepsilon\right)^2}\right)$
ours	$\|\mathbf{x}\|_\infty \leq 1$	$\left\|\boldsymbol{W}^j\right\|_{1,\infty} = 1$	$\widetilde{\mathcal{O}}\left(\sqrt{\frac{d}{m}\left(\frac{1+\gamma-\varepsilon}{\gamma-2\varepsilon}\right)^2\left(\sum_{j=1}^d \sqrt{\left\|\boldsymbol{W}^j\right\|_{1/2,\infty}\left\|\boldsymbol{W}^j\right\|_{1,1}}\right)^2}\right)$

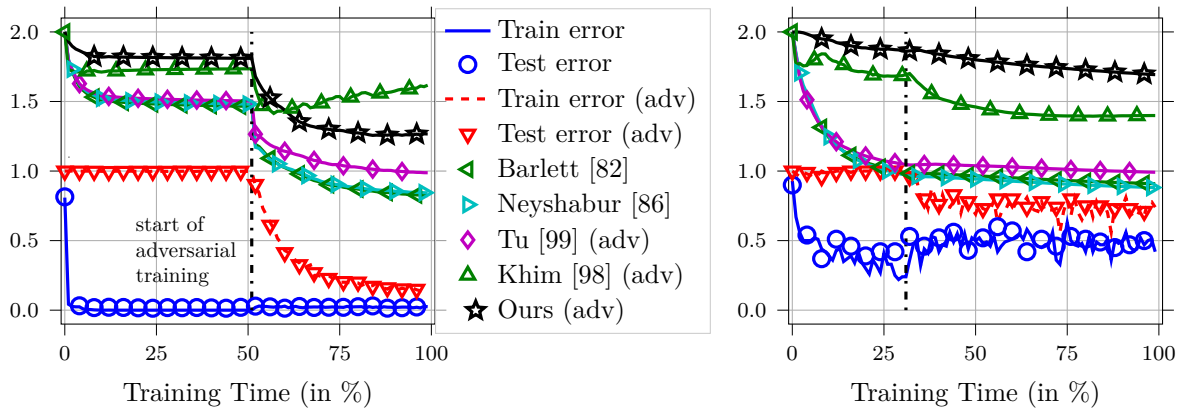
Table 5.1.: Comparison of the result in Theorem 5.3.12 with existing bounds for d-layered neural networks and $\|\boldsymbol{\eta}\|_\infty \leq \varepsilon$. We assume ReLU activations and rebalance the networks such that the bounds are simplified. The input $\mathbf{x} \in \mathbb{R}^n$ is assumed to belong to one of k classes and γ is the classification margin. The term $\lambda_f^+ > 0$ depends on m but may not vanish as m increases, while the term Λ_ε vanishes if $\varepsilon = 0$. For the precise definition of these two terms refer to [99].

from requiring $\|\mathbf{x}\|_2 \leq R$ to $\|\mathbf{x}\|_\infty \leq 1$. Note that, in general, knowing that $\|\mathbf{x}\|_1 \leq 1$ only allows to bound R by $\sqrt{n}$, which would add an explicit dimension dependence on existing results. Moreover, our result does not depend on the number of classes as the work of Khim *et al.* [98], nor it contains terms that do not vanish with m or an explicit dimension dependence (as Tu *et al.* [99]).

5.4. Experiments

We conduct a experiment to corroborate our findings. To that end, we train a fully connected neural network of 3 layers with ReLU activations on the MNIST and CIFAR-10 datasets. After preprocessing, the inputs are 1024-dimensional vectors with ℓ_∞-norm bounded by one. The weight matrices are of size 1024×500, 500×150, and 150×10. To estimate the adversarial risk, we use the projected gradient descent (PGD) attack [24] with ℓ_∞-norm bounded by 0.2 and perturbations computed through 10 iterations of the PGD algorithm. This PGD method is the state of the art algorithm for adversarial training.

In Figure 5.1(a), the network is first trained, on the MNIST dataset, without using adversarial examples. Then, after 50% of the training time, we start introducing adversarial examples to the training set. The same procedure is done in Figure 5.1(b) for the CIFAR-10 dataset, but adversarial examples are introduced after 33% of the training time. These experiments are carried out using the PGD method as described above, except for 0.2 bound on the perturbation's ℓ_∞-norm. Instead, we start with a 0.05 norm bound and slowly increase it until reaching 0.2. We observe that the adversarial error remains unchanged until adversarial training starts, this behavior correlates well with



(a) MNIST **(b)** CIFAR-10

Figure 5.1.: Test and training error (*i.e.*, 1 − accuracy), and (rescaled) generalization bounds during standard and adversarial training. Note that the magnitude of these bounds may differ in several orders of magnitude. Then, for aesthetic reasons we normalized these curves (the ones containing generalization bounds) to be between 0 and 2, while the error curves are between 0 and 1 as usual. We observe that adversarial training improves our bound significantly, while standard training does not.

our result. Interestingly, classic (not adversarial) risk bounds [82], [86] decrease significantly with adversarial training. This agrees with the intuition, from Theorem 5.3.12, that the effective sparsity induced by adversarial training improves generalization. Additionally, we compute the effective sparsity and effective joint sparsity of the weight matrices. In Figure 5.2, we see how these quantities correlate well with the adversarial risk as well. Interestingly, the effective joint sparsity of the weight matrices dominates our generalization bound, a property that was overlooked so far in this context. Overall, these findings show that inducing sparsity structures on the weight matrices does not only provide robustness, but also improves generalization of neural networks.

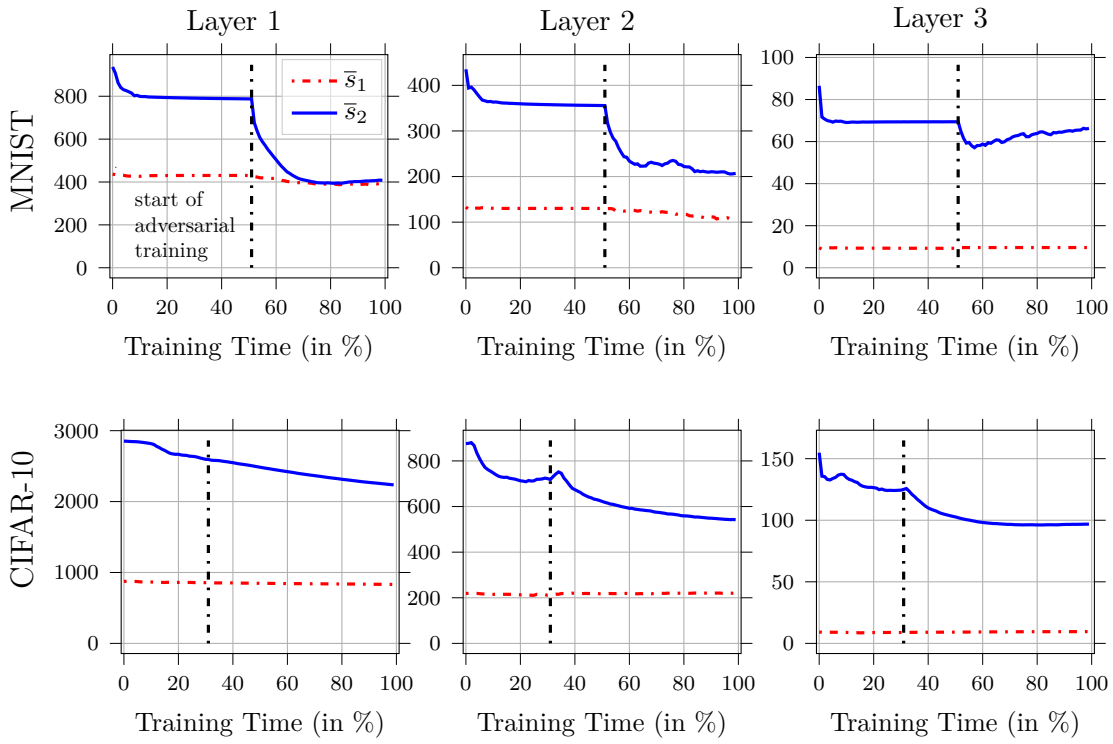


Figure 5.2.: Experiment on the MNIST dataset. Effective sparsity $\overline{s}_1$ and effective joint sparsity $\overline{s}_2$ of the weight matrices, at every layer, of a vanilla neural network. These quantities tend to improve with adversarial training, with $\overline{s}_2$ dominating this tendency.

6

An Information Theoretic View on Learning with Noisy Labels

One key of the recent success of deep learning is the availability of large datasets with high quality labels, such as ImageNet. However, in many applications, obtaining labeled data is costly and the experts may incur into classification error when labeling. In this chapter[1], we study the behavior of information theoretic quantities when learning under label noise. We construct a novel notion of information plane and prove various relations between the quantities involved. In a simplified scenario, where the data is assumed to be linearly separable, we are able to fully characterize the trajectory that a linear classifier takes in the information plane when moving towards the optimal solution. Then, we conduct an empirical study on the trajectories taken by neural networks and observe a behavior that is consistent with our simplified scenario. This allows us to predict the trajectories of correctly trained classifiers in the information plane. Finally, we observe that several undesired learning behaviors can be detected using these notions, since the trajectories of such classifiers deviate significantly from the predicted ones.

[1]This chapter contains the work in [100] with additional results.

6.1. Related Work

The problem of learning from noisy labeled data has been widely studied in the literature [13], [14]. The theoretical study of Random Class Noise (RCN) dates back to 1988 with the well-known work of Angluin & Laird [12]. In that work, the authors showed that a Probably Approximately Correct (PAC) learnable model in the presence of class noise requires more samples to be PAC identifiable. Following results, like the work of Aslam & Decatur [101], upper bounded such number of required samples and showed that the "0-1" loss is PAC-learnable if the VC-dimension is finite. Later, Gentile *et al.* [102] improved these upper bounds.

Many existing works established robustness guarantees (to label noise) for various surrogate loss functions, which are commonly used instead of the 0-1 loss. For instance, B. van Rooyen *et al.* [103] showed that classification-calibrated losses are asymptotically robust. Manwani & Sastry [104] showed, for binary classification, that the function that minimizes the expected error in the presence of RCN is also the minimizer of the error for noiseless labels. Gosh *et al.* [105] extended that work by providing sufficient conditions, on any loss function, to be noise tolerant under uniform and class-conditional noise. This result was recently generalized to the case of multi-class classification [15]. Moreover, Natarajan *et al.* [106] provided guarantees for Empirical Risk Minimization (ERM) with convex surrogate losses in binary classification. The authors showed that biased Support Vector Machines (SVMs) and weighted logistic regression are provably noise tolerant. Natarajan *et al.* [107] extended that work to more general utility measures such as the AM measure[2], a loss function which is used in problems with large class imbalance.

For the case of noiseless labels, Menon *et al.* [108] proposed learning algorithms with proven consistency on the AM measure. Narasimhan *et al.* [109] extended those results to any utility measure that can be expressed as a continuous function of the true positive rate (TPR) and the true negative rate (TNR), such as the F-score. Liu & Tao [110] showed that any surrogate loss function can be used for classification with noisy labels by using importance reweighting[3]. This result led the authors to derive a method for estimating the noise rate as well, a problem that was not addressed previously aside from the work of Scott *et al.* [112]. Wang *et al.* [113] generalized the reweighting strategy from Liu & Tao [110] to multiple classes. Menon *et al.* [16], [114] showed that the balanced error and AUC losses can be optimized even if label noise is present. Patrini *et al.* [115] introduced the concept of linear odd-losses and established robustness guarantees to Class-Conditional random label Noise (CCN) for these losses. Northcutt *et al.* [116] used a rank prunning algorithm for estimating noise rates, with guaranteed consistency under ideal scenarios. Zhang & Sabuncu [117] extended the cross-entropy loss to deal with noisy labels. That approach was based on a theoretical construction,

[2]The AM measure is the arithmetic mean of the True Positive Rate and the True Negative Rate.
[3]Importance reweighting is a technique commonly used for domain adaptation [111].

but with no proven robustness guarantees. Scott [118] estimated noise rates for binary classification using mixture proportion estimation.

On the other hand, studying the behavior of information theoretic quantities in the presence of label noise has not received much attention in the literature. However, for the noiseless regime, this problem was addressed by Shwartz-Ziv & Tishby [119] in an attempt to explain the learning through the lens of the information bottleneck method [120]. In that work, the layers of neural networks are considered random variables forming a Markov chain. The authors constructed a 2D information plane by estimating the mutual information values between hidden layers, inputs, and outputs of neural networks. Using this approach, it was observed that the information bottleneck method provides an approximate explanation for the nature of learning with Stochastic Gradient Descent (SGD). In addition, their experiments showed the role of compression in learning. That initial paper motivated further work on this line of research [121], [122]. The main practical limitation of that type of experiments is that it requires estimating mutual information between high dimensional continuous random variables. This becomes prohibitive as soon as we move to moderately large problems, such as the CIFAR-100 dataset, where the large neural networks are employed. Other works dealing with information theoretic quantities tend to have these experimental limitations. For instance, [123]–[125] used generic chaining techniques to show that generalization error can be upper bounded by the mutual information between the training dataset and the output of the learning algorithm. Nevertheless, estimating that mutual information to verify those results experimentally becomes intractable.

6.1.1. Our Contributions

In this chapter, we define a novel 2D information plane that only requires to estimate information theoretic quantities between the correct and estimated labels. Since these random variables are discrete and one-dimensional, this framework can be used to study learning in large recognition problems as well. We first extend the error bound of Menon *et al.* [16] to the multi-class scenario. Then, we establish fundamental relations between various types of error and information theoretic quantities. In simplified scenarios, we analytically characterize the trajectory in the information plane of linear classifiers when their corresponding weights move towards the optimal solution. Finally, we provide an empirical study on the behavior of those information theoretic quantities during learning, which is consistent with our previous theoretical results. These results show the potential of using information theoretic quantities for detecting undesired learning phenomena, since in such cases the trajectories deviate significantly from the theoretical ones.

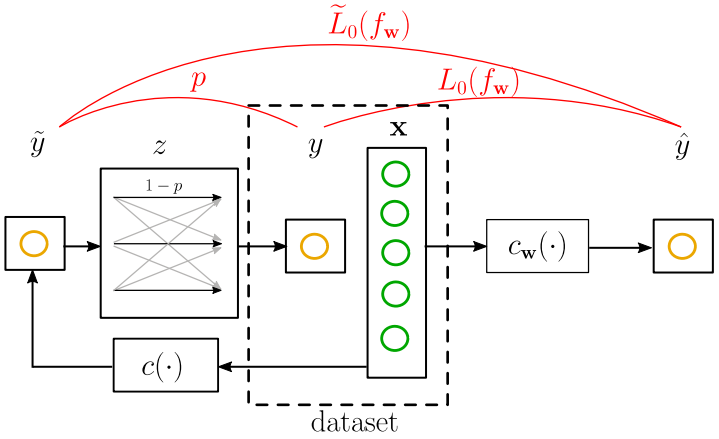


Figure 6.1.: System model.

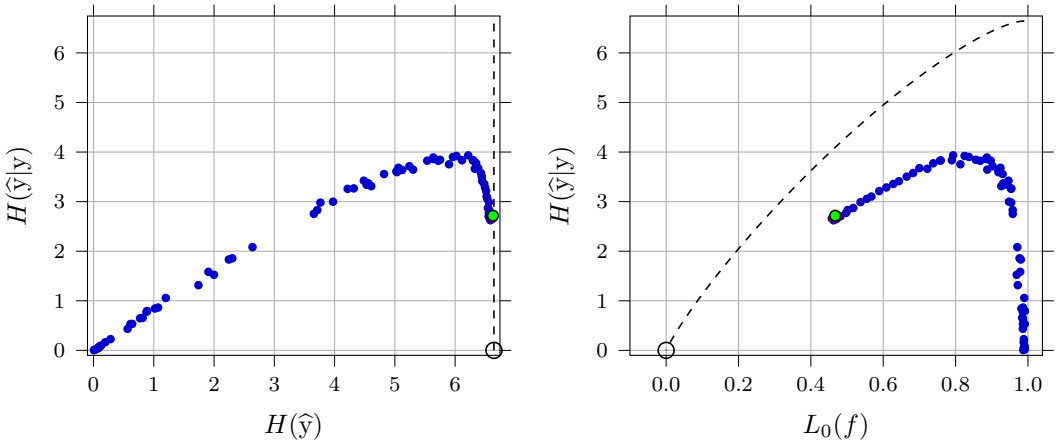
6.2. System Model

So far in this thesis, we have assumed that the random vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ and its associated label $y \in \mathcal{Y}$ follow a completely unknown distribution $\mathcal{D}$, that is $(\mathbf{x}, y) \sim \mathcal{D}$. However, in this chapter we model the relation between $\mathbf{x}$ and y through some hidden random variables $\widetilde{y}$ and z. This relation implicitly defines $\mathcal{D}$.

Before defining $\widetilde{y}$ and z, let us assume that there exists a ground truth classifier, known as "oracle", that maps $\mathbf{x}$ to one of $k \in \mathbb{N}$ classes. Formally, the oracle classifier is a deterministic mapping $c : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{Y} = \{1, \dots, k\}$ is the set of possible classes. Then, $\widetilde{y} \in \mathcal{Y}$ denotes the random variable $\widetilde{y} = c(\mathbf{x})$ which corresponds to the ground truth label associated to $\mathbf{x}$. One common assumption, that is present in popular datasets such as MNIST, CIFAR-10, CIFAR-100, and Imagenet, is that $\widetilde{y}$ is uniformly distributed. We assume this to be true throughout this chapter. Note that the designer of the dataset has control over the marginal distribution of $\widetilde{y}$. Now, we model the situation where there are experts labeling the inputs that are prone to errors. Then, y is defined as a noisy version of $\widetilde{y}$ by introducing a discrete and independent random noise $z \in \{0, \dots, k-1\}$. Formally, $\widetilde{y}$ and y are related via the modulo addition[4] of z, that is $y = \widetilde{y} \oplus z \in \mathcal{Y}$.

For the sake of notation, let $\mathbf{w}$ be the vector containing all tunable parameters of a classifier. The set of all possible vectors $\mathbf{w}$ is denoted by $\mathcal{W}$. Then, a classifier is a deterministic function $c_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$, parametrized by $\mathbf{w}$, that aims to approximate c. Further, $\widehat{y} = c_{\mathbf{w}}(\mathbf{x})$ is defined to be the random variable of the label predicted by the classifier $c_{\mathbf{w}}(\cdot)$. Using this notation, we define three types of error: the dataset error[5] $p = \mathbb{P}(y \neq \widetilde{y})$, the test error $L_0(f) = \mathbb{P}(\widehat{y} \neq y)$, and the true error $\widetilde{L}_0(f) = \mathbb{P}(\widehat{y} \neq \widetilde{y})$. A summary of this system model is provided in Figure 6.1.

[4] We use $\oplus$ to denote the modulo k addition.
[5] This error is also known as noise rate or error rate.



(a) Information Plane **(b)** Entropy-Error plane

Figure 6.2.: Learning trajectory of DenseNet on the CIFAR-100 dataset. The markers in the black dashed lines represent the ideal values of $H(y)$, $H(\widehat{y}|y)$ and $L_0(f)$ when $\widetilde{L}_0(f) = 0$. **(a)**: The dashed lines correspond to the maximum entropy value. **(b)**: The dashed lines are the upper bound given by Fano's inequality.

In the works of [126], it has been shown that $I(y;\widehat{y})$ gives an upper and lower bound on the minimal error[6] between y and $\widehat{y}$. In addition, the minimal error is minimized when $I(y;\widehat{y})$ reaches its maximum. Thanks to this relation, learning can be modeled as finding $\mathbf{w}$ such that $I(y;\widehat{y})$ is maximized. This can be written in terms of entropies as

$$\max_{\mathbf{w}\in\mathcal{W}} I(y;\widehat{y}) = \max_{\mathbf{w}\in\mathcal{W}} H(\widehat{y}) - H(\widehat{y}|y)\,. \tag{6.1}$$

As a result from the above formulation of the learning problem, we define the information plane as the 2-dimensional space composed by $H(\widehat{y})$ and $H(\widehat{y}|y)$. In addition, we also take into account the error $L_0(f)$ by defining the entropy-error plane, which is formed by $L_0(f)$ and $H(\widehat{y}|y)$. We investigate what can the trajectory, in these planes, tell us about the status of the learning process of neural networks. We start addressing this question with a preliminary experiment, shown in Figure 6.2. In that figure, we observe the learning trajectory of the DenseNet architecture with 100 layers as it learns to classify data from the CIFAR-100 dataset using SGD, for $p = 0$. An intuitive interpretation, when solving (6.1), is that maximizing $H(\widehat{y})$ is more related with the unsupervised component of learning since it does not depend on y. On the other hand, keeping $H(\widehat{y}|y)$ low while $H(\widehat{y})$ increases can be seen as the supervised component of (6.1). From this point of view, it would be interesting to characterize the inflection point from which $H(\widehat{y}|y)$ starts decreasing, since it allows us to quantify at which point SGD starts paying more attention to assigning labels correctly than to learn about the distribution of the input. One also may wonder if this increasing-decreasing trajectory is an accidental result that occurs only on this particular experimental setup, or if it is a

[6]The minimal error is the error obtained by a maximum likelihood classifier that predicts y from $\widehat{y}$.

fundamental property of gradient based optimization. In order to analyze these trajectories during training, we must first prove fundamental relations between the quantities involved. This is carried out in the following section.

6.3. Bounds Relating Entropy and Error

In this section, we investigate the fundamental relations that exist between $\widetilde{L}_0(f)$, $L_0(f)$, p, $H(\widehat{y})$ and $H(y|\widehat{y})$. To this end, in the following lemma we introduce Fano's inequality, which is a well-known result from information theory.

Lemma 6.3.1 (Fano's Inequality [127], Lemma 3.8)**.** *The value of $H(y|\widehat{y})$ and $H(\widehat{y}|y)$ is upper bounded by a function of the expected error as*

$$\max\{H(y|\widehat{y}), H(\widehat{y}|y)\} \leq \Psi(L_0(f))\,,$$

where the function $\Psi : [0,1] \to: [0, \log k]$ is defined as

$$\Psi(x) = x\log(k-1) + h_b(x)\,, \quad x \in [0,1]$$

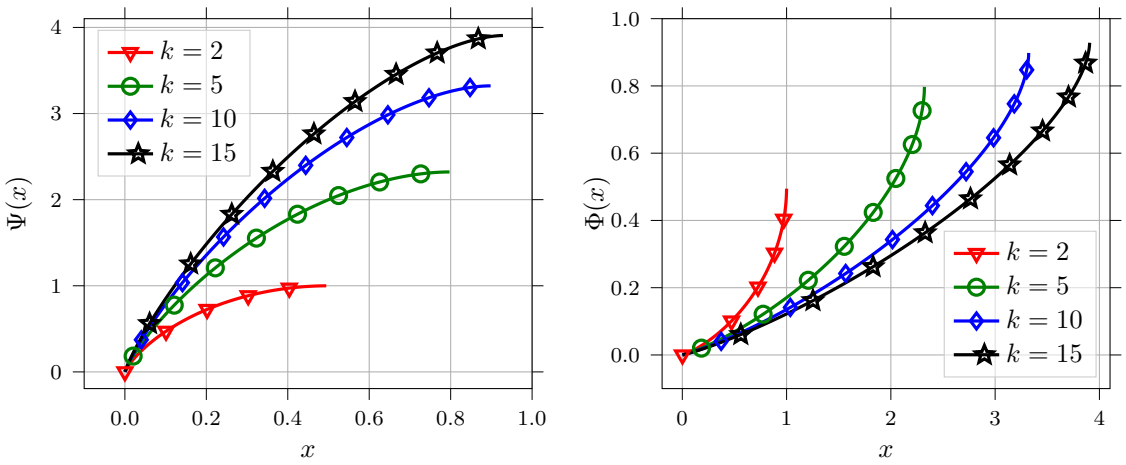
and $h_b(x) = -x\log(x) - (1-x)\log(1-x)$ is the binary entropy function.

This result provides an upper bound on the conditional entropy in terms of $L_0(f)$ and is central in many works from information theory. We make use of this lemma heavily in this chapter as well. We start by using Fano's inequality to establish a fundamental relation between $H(z)$ and p in the following lemma.

Lemma 6.3.2. *Let $z \in \mathcal{Y}$ be a random variable with $\mathbb{P}(z=0) = 1-p$, then*

$$H(z) \leq \Psi(p)\,.$$

Proof (Lemma 6.3.2). Let us define $\beta_l := \mathbb{P}(z=l)$ and the auxiliary random variable



(a) Ψ function **(b)** Φ function

Figure 6.3.: Example of the Ψ between the interval $[0, 1 - 1/k]$, where k is the number of classes, and its inverse function Φ for that interval.

$\tilde{z} \in \{1, \dots, k-1\}$ with $\mathbb{P}(\tilde{z} = l) = \beta_l / p$ for $l = 1, \dots, k-1$. This leads to

$$\begin{aligned}
H(z) &= -(1-p)\log(1-p) - \sum_{l=1}^{k-1} \beta_l \log \beta_l \\
&= -(1-p)\log(1-p) - p \sum_{l=1}^{k-1} \frac{\beta_l}{p} \log \frac{\beta_l}{p} p \\
&= -(1-p)\log(1-p) - p \sum_{l=1}^{k-1} \frac{\beta_l}{p} \log p - p \sum_{l=1}^{k-1} \frac{\beta_l}{p} \log \frac{\beta_l}{p} \\
&= -(1-p)\log(1-p) - \sum_{l=1}^{k-1} \beta_l \log p + pH(\tilde{z}) \\
&= -(1-p)\log(1-p) - p \log p + pH(\tilde{z}) \\
&= h_b(p) + pH(\tilde{z}) \\
&\leq h_b(p) + p \log(k-1) \\
&= \Psi(p)\,,
\end{aligned}$$

thus proving the lemma. □

One property of the function Ψ, which was introduced in Lemma 6.3.2, is that it is strictly increasing on the interval $[0, 1/k]$. Therefore, on that interval, this function has an inverse which we denote as $\Phi : [0, \log k] \to [0, 1 - \frac{1}{k}]$, that is

$$\Phi(\Psi(x)) = x \quad \text{for all} \quad x \in [0, 1/k]\,.$$

These functions are depicted in Figure 6.3 for different values of k. Moreover, using the Φ function, for $p < 1/k$, we can rewrite Lemma 6.3.2 as $\Phi(H(z)) \leq p$, which gives an

upper bound on $H(z)$. These relations lead to the following result, which shows that, under some mild conditions, the lower bound $p \leq L_0(f)$ is in fact sharp. This result was recently established in the following theorem of Ghosh *et al.* [15], which we rediscovered in this chapter through a different proof technique.

Theorem 6.3.3 (Ghosh *et al.* [15])**.** *For $p < 1 - 1/k$, let $\mathbb{P}(z = 0) = 1 - p$ and $\mathbb{P}(z = 0) > \mathbb{P}(z = l)$ for $l = 1, \ldots, k-1$, then*

$$p \leq L_0(f)\,,$$

and equality is attained if and only if $\widetilde{L}_0(f) = 0$.

Proof (Theorem 6.3.3). For the sake of notation let us define

$$\begin{aligned} \delta_l &:= \mathbb{P}(\widehat{y} = \widetilde{y} \oplus l)\,, \\ \beta_l &:= \mathbb{P}(z = l) \text{ for } l \in \{0, \ldots, k-1\}\,, \\ \beta_{\max} &:= \max_{l=1,\ldots,k-1} \beta_l\,. \end{aligned}$$

Since $p \in [0, 1 - 1/k]$, we have that $\Phi : [0, \log k] \to [0, 1 - \frac{1}{k}]$ is the inverse function of Ψ for any p in the allowed interval. Therefore, for the independent noise z, we get

$$H(z) = H(z|\widehat{y}, \widetilde{y}) = H(y|\widehat{y}, \widetilde{y}) \leq H(y|\widehat{y})\,.$$

Since Ψ is an increasing function in the interval $[0, 1 - \frac{1}{k}]$ and Φ is its inverse in that interval, we have that Φ is an increasing function as well. Therefore, applying Lemma 6.3.1 leads to $H(z) \leq \Psi(L_0(f))$ which implies

$$\Phi(H(z)) \leq L_0(f)\,. \tag{6.2}$$

Similarly, from Lemma 6.3.2 we know that $H(z) \leq \Psi(p)$, thus

$$\Phi(H(z)) \leq p\,.$$

Note that this bound is only sharp if $\beta_l = 1/(k-1)$ for all $l = 1, 2, \ldots, k-1$. Then, if the bound from (6.2) were to be sharp, $L_0(f)$ could reach values strictly lower than p. We will show that this is not possible. To that end, let us assume that $H(z) < \Psi(p)$

and $L_0(f) < p$. Then it follows,

$$\begin{aligned}
1 - L_0(f) &= \mathbb{P}(\widehat{y} = y) \\
&= \sum_{l=0}^{k-1} \mathbb{P}(\widehat{y} = y | z = l)\mathbb{P}(z = l) \\
&= \sum_{l=0}^{k-1} \mathbb{P}(\widehat{y} = y \oplus l)\mathbb{P}(z = l) \\
&= \sum_{l=0}^{k-1} \delta_l \beta_l \\
&= (1-p)\delta_0 + \sum_{l=1}^{k-1} \beta_l \delta_l \\
&\leq (1-p)\delta_0 + \sum_{l=1}^{k-1} \beta_{\max} \delta_l \\
&= (1-p)\delta_0 + \beta_{\max}(1 - \delta_0) \qquad (6.3) \\
&\leq (1-p)\delta_0 + (1-p)(1-\delta_0) \\
&= (1-p)\,.
\end{aligned}$$

If $L_0(f) < p$, we obtain $(1-p) < 1 - L_0(f) \leq (1-p)$ which is a contradiction, hence, it must hold that $L_0(f) \geq p$.

Finally, if $L_0(f) = p$ then (6.3) yields

$$1 - p \leq (1-p)\delta_0 + \beta_{\max}(1 - \delta_0)\,.$$

Since $\beta_{\max} < (1-p)$, this inequality holds if and only if $\delta_0 = 1$, that is $\widetilde{L}_0(f) = 0$. □

This theorem shows that the minimum expected error $L_0(f)$ can only be attained if $c_{\mathbf{w}}(\cdot)$ manages to denoise the labels, hence $\widetilde{L}_0(f) = 0$. In other words, the true error $\widetilde{L}_0(f)$ is optimal when the lowest value of $L_0(f)$ is achieved. However, Theorem 6.3.3 does not tell us what happens to $\widetilde{L}_0(f)$ when $L_0(f)$ is not at its lowest possible value. Furthermore, we extend that result by deriving bounds for $\widetilde{L}_0(f)$, given $L_0(f)$ and p, in the following theorem.

Theorem 6.3.4. *Given $L_0(f) < 1 - \frac{1}{k}$ and $p < \frac{1}{2}$, if $\mathbb{P}(z = 0) = 1 - p$, and $\mathbb{P}(z = l) < \frac{1}{2}$ for $l = 1, \ldots, k-1$, then $\widetilde{L}_0(f)$ is bounded by*

$$\frac{L_0(f) - p}{1 - p} \leq \widetilde{L}_0(f) \leq \frac{L_0(f) - p}{1 - 2p}\,.$$

Proof (Theorem 6.3.4). We start by expressing the expected accuracy $\mathbb{P}(\widehat{y} = y)$ in terms

of the true accuracy $\mathbb{P}(\widehat{y} = \widetilde{y})$, that is

$$\begin{aligned} 1 - L_0(f) &= \mathbb{P}(\widehat{y} = y) \\ &= \sum_{l=0}^{k-1} \underbrace{P(\widetilde{y} \oplus l = \widehat{y})}_{\delta_l} \underbrace{P(z = l)}_{\beta_l} \\ &= (1-p)(1 - \widetilde{L}_0(f)) + \sum_{l=1}^{k-1} \delta_l \beta_l \qquad (6.4) \\ &\leq (1-p)(1 - \widetilde{L}_0(f)) + \beta_{\max} \sum_{l=1}^{k-1} \delta_l \\ &= (1-p)(1 - \widetilde{L}_0(f)) + \beta_{\max} \widetilde{L}_0(f) \\ &= (1-p) - \widetilde{L}_0(f)((1-p) - \beta_{\max}) \,. \end{aligned}$$

Therefore, $\widetilde{L}_0(f)$ is upper-bounded as

$$\widetilde{L}_0(f) \leq \frac{(1-p) - (1 - L_0(f))}{(1-p) - \beta_{\max}} \leq \frac{(1-p) - (1 - L_0(f))}{(1-p) - p} = \frac{L_0(f) - p}{1 - 2p} \,.$$

Finally, applying $\sum_{l=1}^{k-1} \delta_l \beta_l \geq 0$ in (6.4) leads to $1 - L_0(f) \geq (1-p)(1 - \widetilde{L}_0(f))$, thus

$$\frac{(1-p) - (1 - L_0(f))}{(1-p)} \leq \widetilde{L}_0(f) \quad \Rightarrow \quad \frac{L_0(f) - p}{1-p} \leq \widetilde{L}_0(f) \,,$$

which completes the proof. □

This theorem shows that, by minimizing $L_0(f)$, a classifier minimizes $\widetilde{L}_0(f)$ as well. Nevertheless, the amount of samples that a classifier requires to successfully minimize $L_0(f)$ is not studied in this work. Such study corresponds to works from the literature such as [101], [102]. Let us use the notation $\widetilde{f}^* := \operatorname{argmin}_f \widetilde{L}_0(f)$ and $f^* := \operatorname{argmin}_f L_0(f)$. Then, we can see that this theorem constitutes an improvement from the result by Menon *et al.* [16], which established

$$\widetilde{L}_0(f) \leq \frac{L_0(f) - L_0(f^*)}{1 - 2L_0(f^*)}$$

for binary classification. Note that, since we established in Theorem 6.3.3 that $L_0(f^*) = p$, we can write Theorem 6.3.4 as

$$\frac{L_0(f) - L_0(f^*)}{1 - L_0(f^*)} \leq \widetilde{L}_0(f) \leq \frac{L_0(f) - L_0(f^*)}{1 - 2L_0(f^*)} \,.$$

Therefore, this result generalizes the result of Menon *et al.* [16] to the case of multi-class classification and provides an additional lower bound.

Finally, applying Lemma 6.3.1 to this result yields the following corollary that directly relates conditional entropies with p and $L_0(f)$.

Corollary 6.3.4.1. *If* $p < \frac{1}{2}$ *and* $L_0(f) - p < (1 - \frac{1}{k})(1 - 2p)$ *then*

$$\max\{H(\widetilde{y}|\widehat{y}), H(\widehat{y}|\widetilde{y})\} \leq \Psi\left(\frac{L_0(f) - p}{1 - 2p}\right).$$

Proof. Since Ψ is an increasing function in the interval $[0, 1 - \frac{1}{k}]$, the proof follows from applying Theorem 6.3.4 on Lemma 6.3.1. □

6.4. Analysis of Learning Trajectories for Linear Classifiers

In Figure 6.2, we showed the trajectory of a classifier in the information plane, as well as in the entropy-error plane. Such trajectory seems to appear regardless of the activation function employed on the spirals and MNIST dataset, as we will show later in Section 6.5. Moreover, in this section we provide a justification for this type of trajectory through linear classifiers. More precisely, in the linearly separable data paradigm with binary labels, we are able to analytically characterize the trajectories of linear classifiers as their weights approach optimal values.

6.4.1. Binary Classification of Linearly Separable Data

We start by proposing a simplified linear model for the relation between $\mathbf{x}$ and $\widetilde{y}$. Such relation, along with the distribution z, implicitly defines $\mathcal{D}$. We model the case where the input data lies in a low-dimensional manifold within a high-dimensional space. This situation is common in many classification tasks. However, we assume this manifold to be linear as well, thus the input data belongs to some hyperplane. Formally, we assume the input vector to be given by $(\mathbf{x}^\top, \boldsymbol{\mu}^\top)^\top \in \mathbb{R}^{n+n'}$, where $\mathbf{x} \in \mathbb{R}^n$ is a random vector, while $\boldsymbol{\mu} \in \mathbb{R}^{n'}$ is a constant vector. In this manner, the input data vectors lie in a n-dimensional hyperplane contained in a $(n + n')$-dimensional space. Note that, since $\boldsymbol{\mu}$ is always constant (regardless of the associated label), only $\mathbf{x}$ is relevant for classification. Let us assume that the true associated label, denoted by $\widetilde{y}$, and given by the relation

$$\widetilde{y} = \operatorname{sign}\left(\left\langle \begin{pmatrix} \mathbf{w}^* \\ \mathbf{u}^* \end{pmatrix}, \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\mu} \end{pmatrix} \right\rangle\right) = \operatorname{sign}\left(\langle \mathbf{w}^*, \mathbf{x}\rangle + \underbrace{\langle \mathbf{u}^*, \boldsymbol{\mu}\rangle}_{=:b^*}\right) = \operatorname{sign}\left(\langle \mathbf{w}^*, \mathbf{x}\rangle + b^*\right),$$

where $\mathbf{w}^* \in \mathbb{R}^n$ and $\boldsymbol{\mu}^* \in \mathbb{R}^{n'}$, which implicitly defines $b^* \in \mathbb{R}$, are unknown but fixed parameters. This formulation corresponds to the paradigm of linearly separable classes.

Let $t \in [0, 1]$ denote the fraction of total training time. In other words, the value $t = 0$ refers to the beginning of training, while $t = 1$ to the end. Using this notation, we

define the random variable $\widehat{y}^{(t)}$ as the parametrized version of $\widehat{y}$ at a fraction t of the total training time. Formally, this corresponds to

$$\widehat{y}^{(t)} = \operatorname{sign}\left(\left\langle \begin{pmatrix}\mathbf{w}^{(t)}\\ \mathbf{u}^{(t)}\end{pmatrix}, \begin{pmatrix}\mathbf{x}\\ \boldsymbol{\mu}\end{pmatrix}\right\rangle\right) = \operatorname{sign}\left(\left\langle \mathbf{w}^{(t)}, \mathbf{x}\right\rangle + \underbrace{\left\langle \mathbf{u}^{(t)}, \boldsymbol{\mu}\right\rangle}_{=:b^{(t)}}\right) = \operatorname{sign}\left(\left\langle \mathbf{w}^{(t)}, \mathbf{x}\right\rangle + b^{(t)}\right),$$

where $\mathbf{w}^{(t)} \in \mathbb{R}^n$ and $\mathbf{u}^{(t)} \in \mathbb{R}^{n'}$, which implicitly defines $b^{(t)} \in \mathbb{R}$, are the tunable parameters of the model at a fraction t of the total training time. This setup corresponds to the classical learning from linearly separable data. Note that, in this setup, our classifier is capable of achieving zero true error (*i.e.*, $\mathbb{P}[\widetilde{y} = \widehat{y}^{(t)}] = 1$) by learning $\mathbf{w}^{(t)} = \mathbf{w}^*$ and $b^{(t)} = b^*$. It is known for various optimizers that, regardless of the initialization, $(\mathbf{w}^{(t)}, b^{(t)})$ converges to $(\mathbf{w}^*, b^*)$. We now need to specify the initialization point, that is the value of $\mathbf{w}^{(t)}$ and $b^{(t)}$ at $t = 0$. Let us assume a staring point $\mathbf{w}^{(0)} = 0$ and $b^{(0)} > 0$, which results in $\mathbb{P}[\widehat{y}^{(0)} = +1] = 1$ (*i.e.*, $H(\widehat{y}^{(0)}) = 0$) regardless of the distribution of $\mathbf{x}$ and the hidden parameters. Note that, in practical problems, random initialization with values close to zero is often used. As we will see in Section 6.5, such initialization often leads to a starting point close to $H(\widehat{y}^{(0)}) = 0$, the same as in our linearly separable model. The question we address is what trajectory do optimizers take to go from this initial point towards the optimal set of parameters. To visualize such trajectories, since the parameter space is usually large, we must first aggregate the information from that space into a smaller space. In this binary classification problem, we may use the space of joint probability measures between y and $\widehat{y}$ since we only require 3 values to fully define the joint probabilities between these variables. To that end, let $\mathcal{D}^{(t)}_{\widetilde{Y},\widehat{Y}}$ denote the distribution of $(\widetilde{y}, \widehat{y}^{(t)})$ and $\mathcal{D}^{*}_{\widetilde{Y},\widehat{Y}}$ the optimal distribution (*i.e.*, when $(\mathbf{w}^{(t)}, b^{(t)}) = (\mathbf{w}^*, b^*)$). Note that $\mathcal{D}^{(0)}_{\widetilde{Y},\widehat{Y}}$ corresponds to the distribution where $\mathbb{P}[\widehat{y}^{(0)} = +1] = 1$, while $\mathcal{D}^{*}_{\widetilde{Y},\widehat{Y}}$ to $\mathbb{P}[\widetilde{y} = \widehat{y}^{(t)}] = 1$. Now, we need to know how is $\mathcal{D}^{(t)}_{\widetilde{Y},\widehat{Y}}$ changing, given that $\mathbf{w}^{(t)}$ is moving towards $\mathbf{w}^*$. This is addressed in the following theorem.

Theorem 6.4.1. *Given $\mathbf{w}^{(0)} = \mathbf{0}, b^{(0)} \neq 0$, assume that $\mathbf{w}^{(t)} = (1-t)\mathbf{w}^{(0)} + t\mathbf{w}^*$ and $b^{(t)} = (1-t)b^{(0)} + tb^*$. Then, there exists a decreasing function $\beta : [0,1] \to [0,1]$, such that $\beta(t)$ tends to one as $t \to 1$ and for any $t \in [0,1]$ we have that*

$$\mathcal{D}^{(t)}_{\widetilde{Y},\widehat{Y}} = (1-\beta(t))\mathcal{D}^{(0)}_{\widetilde{Y},\widehat{Y}} + \beta(t)\mathcal{D}^{*}_{\widetilde{Y},\widehat{Y}}\,. \tag{6.5}$$

Proof (Theorem 6.4.1). Let us define the function $\beta(t)$ as

$$\beta(t) := \frac{1 - \mathbb{P}\left[\widehat{y}^{(t)} = +1\right]}{1 - \mathbb{P}\left[\widetilde{y} = +1\right]}\,.$$

Then, we start by analyzing the marginal distribution of $\widehat{y}^{(t)}$. Since we assume $\widehat{y}^{(t)}$ to be a binary random variable, its distribution is fully determined by $\mathbb{P}[\widehat{y}^{(t)} = +1]$.

Furthermore, through the assumptions of this theorem, this probability simplifies to

$$\begin{aligned}\mathbb{P}\left[\widehat{y}^{(t)}=+1\right] &= \mathbb{P}\left[\left\langle \mathbf{w}^{(t)},\mathbf{x}\right\rangle + b^{(t)} \geq 0\right] \\ &= \mathbb{P}\left[t\left(\langle \mathbf{w}^*,\mathbf{x}\rangle + b^*\right) + (1-t)b^{(0)} \geq 0\right] \\ &= \mathbb{P}\left[\langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \geq -\underbrace{(1-t)b^{(0)}/t}_{=:g(t)}\right] . \end{aligned} \tag{6.6}$$

Without loss of generality, let us assume $b^{(0)} > 0$ instead of only being non-zero[7]. This assumption implies that $g(t)$ is a non-negative and increasing function, thus $\mathbb{P}[\widehat{y}^{(t)} = +1]$ decreases with t. Note that this leads to $\beta(t)$ being an increasing function in t. Moreover, since $g(t)$ tends to zero as $t \to 1$, we have that $\mathbb{P}[\widehat{y}^{(t)} = +1]$ tends to $\mathbb{P}[\langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \geq 0] = \mathbb{P}[y = +1]$. Therefore, the function $\beta(t)$ tends to one as $t \to 1$.

Now, since $g(t)$ is non-negative, $\mathbb{P}[y = +1, \widehat{y}^{(t)} = +1]$ simplifies to

$$\begin{aligned}\mathbb{P}\left[\widetilde{y}=+1,\widehat{y}^{(t)}=+1\right] &= \mathbb{P}\left[\langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \geq 0, \langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \geq -g(t)\right] \\ &= \mathbb{P}\left[\langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \geq 0\right] \\ &= \mathbb{P}\left[\widetilde{y}=+1\right] . \end{aligned}$$

In the same manner, we show that $\mathbb{P}[y = +1, \widehat{y}^{(t)} = -1]$ remains being zero for any t, that is

$$\begin{aligned}\mathbb{P}\left[\widetilde{y}=+1,\widehat{y}^{(t)}=-1\right] &= \mathbb{P}\left[\langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \geq 0, \langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \leq -g(t)\right] \\ &= 0 . \end{aligned}$$

Now, we make use of (6.6) to express $\mathbb{P}[y = -1, \widehat{y}^{(t)} = +1]$ as

$$\begin{aligned}\mathbb{P}\left[\widetilde{y}=-1,\widehat{y}^{(t)}=+1\right] &= \mathbb{P}\left[\widehat{y}^{(t)}=+1\right] - \mathbb{P}\left[\widetilde{y}=+1\right] \\ &= \left(1-\mathbb{P}\left[\widetilde{y}=+1\right]\right)\left(1-\beta(t)\right) . \end{aligned}$$

Likewise, $\mathbb{P}[y = -1, \widehat{y}^{(t)} = -1]$ yields

$$\begin{aligned}\mathbb{P}\left[\widetilde{y}=-1,\widehat{y}^{(t)}=-1\right] &= \mathbb{P}\left[\langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \leq 0, \langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \leq -g(t)\right] \\ &= 1-\mathbb{P}\left[\langle \mathbf{w}^*,\mathbf{x}\rangle + b^* \geq -g(t)\right] = 1-\mathbb{P}\left[\widehat{y}^{(t)}=+1\right] \\ &= \left(1-\mathbb{P}\left[\widetilde{y}=+1\right]\right)\beta(t) . \end{aligned}$$

Putting these results together, we get

$$\begin{pmatrix}\mathbb{P}\left[\widetilde{y}=+1,\widehat{y}^{(t)}=+1\right]\\ \mathbb{P}\left[\widetilde{y}=-1,\widehat{y}^{(t)}=+1\right]\\ \mathbb{P}\left[\widetilde{y}=+1,\widehat{y}^{(t)}=-1\right]\\ \mathbb{P}\left[\widetilde{y}=-1,\widehat{y}^{(t)}=-1\right]\end{pmatrix} = (1-\beta(t))\begin{pmatrix}\mathbb{P}\left[\widetilde{y}=+1\right]\\ 1-\mathbb{P}\left[\widetilde{y}=+1\right]\\ 0\\ 0\end{pmatrix} + \beta(t)\begin{pmatrix}\mathbb{P}\left[\widetilde{y}=+1\right]\\ 0\\ 0\\ 1-\mathbb{P}\left[\widetilde{y}=+1\right]\end{pmatrix} ,$$

which proves the theorem. □

[7] An equivalent version of this proof for $b^{(0)} < 0$ is trivial.

This theorem shows that if an algorithm moves the parameters in a straight line towards the optimal parameter choice, we expect the joint probability measure of $(y, \widehat{y})$ to move in a straight line as well.

6.4.2. Multi-Class Classification

While we may use the result in Theorem 6.4.1 to visualize the trajectories taken by classifiers during training, this approach does not scale well for multi-class scenarios, as one would need to visualize $k^2 - 1$ values. For this reason, we propose using the information and entropy-error planes to visualize the progress of training, which requires characterizing the learning trajectories in terms of $H(\widehat{y}^{(t)}|y), H(\widehat{y}^{(t)})$ and $L_0(f)$. To that end, we consider a multi-class setup with $|\mathcal{Y}| = k$. In this multi-class scenario, the distribution of z is not fully determined by p. For the sake of simplicity, instead of considering arbitrary distributions for z, let us assume that z flips the label $\widetilde{y}$, with probability p, to another label chosen uniformly at random from the remaining ones. Formally, the distribution of z is given by

$$\mathbb{P}(z = i) = \begin{cases} 1 - p, & i = 0 \\ \frac{p}{k-1}, & i \in \{1, \dots, k-1\} \end{cases} . \tag{6.7}$$

In this section, we aim to characterize the trajectory of classifiers in this scenario as they move towards the correct solution. Note that the definition of $\mathcal{D}^{(t)}_{\widetilde{Y},\widehat{Y}}$ and $\mathcal{D}^{*}_{\widetilde{Y},\widehat{Y}}$ from the previous section are still valid for this multi-class scenario. The result from Theorem 6.4.1 established that, for the binary case, these distributions move withing the curve shown in (6.5) when a classifier tunes its weights towards the correct solution. From the proof of this theorem, generalizing this result for a multi-class linearly separable scenario seems to be tedious, yet not necessarily complicated. Therefore, and for the sake of clarity, we assume that $\mathcal{D}^{(t)}_{\widetilde{Y},\widehat{Y}}$ moves as in (6.5) for the multi-class setup as well. Using this assumption, along with the distribution of z in (6.7), we provide a simplified expression for the derivative $\mathcal{D}^{*}_{\widetilde{Y},\widehat{Y}}$ with respect to $\beta(t)$ in the following lemma.

Lemma 6.4.2. *Let us assume a multi-class scenario where $\widetilde{y} \in \mathcal{Y}$, with $|\mathcal{Y}| = k$. Additionally, let z follow the distribution in* (6.7)*. If $\mathcal{D}^{(t)}_{\widetilde{Y},\widehat{Y}} = (1-\beta(t))\mathcal{D}^{(0)}_{\widetilde{Y},\widehat{Y}} + \beta(t)\mathcal{D}^{*}_{\widetilde{Y},\widehat{Y}}$, then it holds*

$$\begin{aligned} \frac{\partial H(\widehat{y}^{(t)}|y)}{\partial \beta(t)} = \frac{1}{k}\Big[& p \log(1 - \beta(t)p) + (k - 1 - p) \log\left(1 - \beta(t) + \beta(t)p/(k-1)\right) \\ & - (k-1)(1-p)\log(\beta(t)(1-p)) - (k-1)p\log\left(\beta(t)p/(k-1)\right)\Big] . \end{aligned} \tag{6.8}$$

Proof (Lemma 6.4.2). The proof of this theorem follows from calculating the conditional distribution of $\widehat{y}^{(t)}$ given y, and then differentiating with respect to $\beta(t)$. This leads to a lengthy calculation, thus it has been deferred to the Appendix A.3. □

This result is useful since it allows us to calculate the inflection point (*i.e.*, when $H(\widehat{y}^{(t)}|y)$ reaches its maximum value) by setting this derivative to zero. For example, this is fairly simple for p tending to zero, as shown in the following corollary of Lemma 6.4.2.

Corollary 6.4.2.1. *In the setting of Lemma 6.4.2, if* $p \to 0$ *then* $H(\widehat{y}^{(t)}|y)$ *has one maximum at* $\beta(t) = \frac{1}{2}$, $H(\widehat{y}^{(t)}|y) = \frac{k-1}{k} \log 2$.

This result allows us to characterize the shape of the 2D curves $(H(\widehat{y}^{(t)}), H(\widehat{y}^{(t)}|y))$ and $(L_0(f), H(\widehat{y}^{(t)}|y))$ for $p = 0$. To do so, we just need to set $\beta(t)$ to $1/2$, then calculate the corresponding quantities using $\mathcal{D}^{(t)}_{\widetilde{Y},\widehat{Y}}$. However, for arbitrary values of p, we require to find the value of $\beta(t)$ where (6.8) vanishes using numerical computations. Using this result, in Figure 6.4 we show the obtained trajectories for different values of k, with their corresponding inflection points. We can observe how the point where $H(\widehat{y}|y)$ starts decreasing corresponds to a value of $H(\widehat{y})$ that increases with p. This supports the intuition that, for scenarios with increased label noise, a classifier needs to learn more about the input distribution before successfully assigning labels to those inputs. A similar conclusion can be drawn using the entropy-error plane, where more label noise implies inflection points closer to Fano's bound.

6.5. Experimental Study for Neural Networks

In this section, we empirically study the behavior of information theoretic quantities $H(\widehat{y})$ and $H(\widehat{y}|y)$ during the training for different datasets and neural networks. The first issue is to properly estimate these quantities. Since these quantities depend only on y and $\widehat{y}$, it is sufficient to obtain a good estimation of the joint distribution of $(\widehat{y}, y) \in \{0, \dots, k-1\}^2$ in order to approximate the mutual information and conditional entropies. A naive estimator would calculate the empirical distribution of $(\widehat{y}, y)$, using m independent observations, and then directly compute conditional entropies. It is shown in [128] that the approximation error incurred by this method is of order k^2/m. Hence, this approach yields a good approximation if $m \gg k^2$. This holds particularly for our experiments where the number of classes does not exceed 10 while the number of test examples are much larger than 10^2. Therefore, we use this method to estimate these information theoretic quantities. When the number of classes k is large, one can consider more sophisticated methods such as [129] and [130].

For our experiments, we assume to have datasets composed of independent realizations of $(\mathbf{x}, \widetilde{y})$. In particular, we use the following three datasets:

- **Spirals:** The spirals dataset consists of two-dimensional points belonging to one of three spirals. These points correspond to $(\mathbf{x}, \widetilde{y})$ generated by

$$\mathbf{x} = \begin{pmatrix} (\sqrt{a} + b) \cos\left(2\pi a + \frac{2\pi}{3}\widetilde{y}\right) \\ (\sqrt{a} + b) \sin\left(2\pi a + \frac{2\pi}{3}\widetilde{y}\right) \end{pmatrix},$$

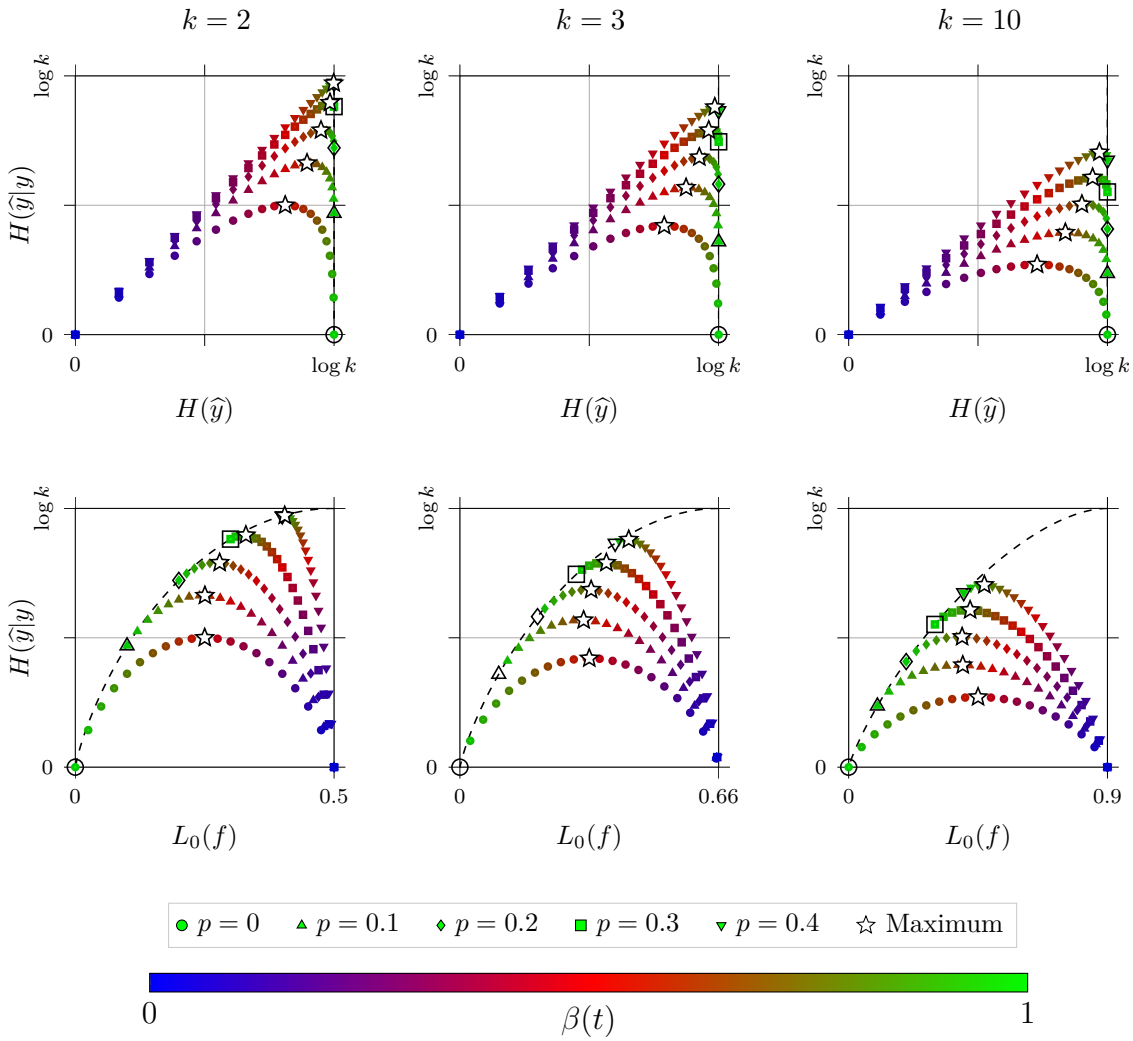


Figure 6.4.: Trajectory of (6.5) in the information and entropy-error planes for different values of k and p. The point of maximum conditional entropy is calculated using Lemma 6.4.2. The dashed lines correspond to the bound given by Fano's inequality, and the black markers to the point of minimum attainable error (see Theorem 6.3.3).

where $a \in [0, 1]$, $b \in [0, 0.1]$ and $\widetilde{y} \in \{1, 2, 3\}$ are independent uniformly distributed random variables. This leads to the following spirals, color coded according to their corresponding class.

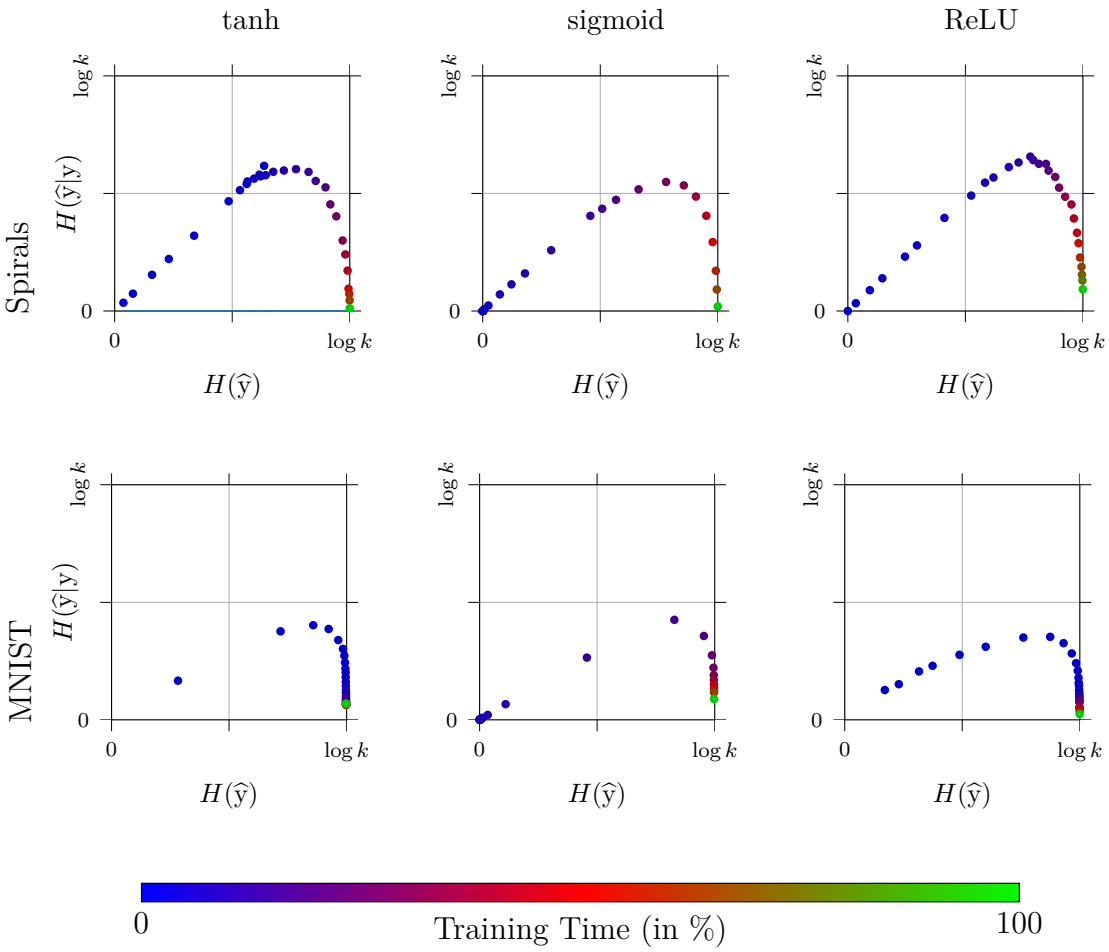
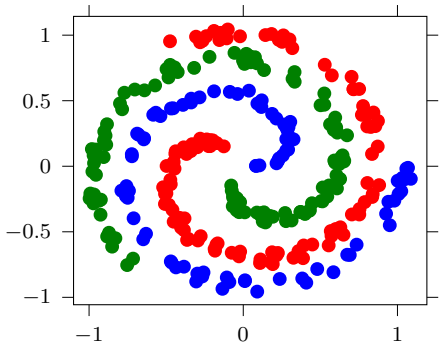


Figure 6.5.: Information plane trajectory during the learning process for $p = 0$.



Moreover, this dataset is divided into a training set of 50 000 samples and a test set of 2 000.

- **MNIST:** This is a well known dataset for handwritten digit recognition [53]. It consists of 55 000 training images and 10 000 test images.
- **CIFAR-10:** This dataset consists of tiny RGB images belonging to 10 categories

Dataset	Activation	Batch Size	$\alpha_{\max}$	$\alpha_{\min}$	Test Accuracy
Spirals	tanh	128	10^{-1}	10^{-2}	99.7%
	sigmoid	128	10^{-1}	10^{-5}	99.6%
	ReLU	700	10^{-1}	10^{-5}	97.8%
MNIST	tanh	128	10^{-2}	10^{-2}	97.1%
	sigmoid	128	10^{-2}	10^{-4}	96.3%
	ReLU	128	10^{-2}	10^{-4}	99.1%
CIFAR-10	ReLU	64	10^{-1}	10^{-1}	80.2%

Table 6.1.: Simulation Parameters

[54]. It contains 50 000 images for training and 10 000 for testing.

A fully connected neural network with four hidden layers of five neurons each, referred to as FCNN, is trained on the spirals dataset. For the MNIST dataset, the popular convolutional network called LeNet-5 [55] is used. To train these networks, we let the learning rate $\alpha \in \mathbb{R}$ start at a given $\alpha_{\max} \in \mathbb{R}$ and then decay by 40% per epoch until reaching some given minimum learning rate $\alpha_{\min} < \alpha_{\max}$, that is $\alpha = \max\{\alpha_{\max} 0.6^{\lfloor \text{epoch} \rfloor}, \alpha_{\min}\}$. For the CIFAR-10 dataset, we train a 40 layer DenseNet architecture as done in [57], but we stop the training after 10 epochs instead of the original 300 used by the authors. We train FCNN and LeNet-5 using various hidden layer activation functions, learning rates, and mini-batch sizes. The different configurations used for these experiments are summarized in Table 6.1.

Based on the above setup, these neural networks are trained to learn the noisy labels from the data. The true labels are corrupted by an independent additive noise according to (6.7). We are interested in studying the behavior of neural networks as they learn classification tasks to perfection. More precisely, we are first interested in examining only those neural networks that managed to achieve the best performance after training. For FCNN and LeNet-5, we only take into account neural networks that achieved less than $0.05 + p$ error on their corresponding test set and assume that they are correctly trained. For CIFAR-10, where DenseNet is only trained for 10 epochs instead of 300, we assume that neural networks with less than $0.2 + p$ test error are well trained. In Figure 6.5, the average behavior over 100 independent realizations of correctly trained neural networks, during their learning process, is depicted. In these figures, a similar trend can be observed regarding the evolution of the information theoretic quantities during training. Regardless of the non-linearity and the dataset used, we observe that the learning process consists of two phases. The first phase occurs at the beginning of learning, where $H(\widehat{y})$ increases even at the expense of increasing $H(\widehat{y}|y)$. A possible explanation of this phenomenon is that, at the beginning of the training process, it is more important to facilitate the information flow between $\mathbf{x}$ and $\widehat{y}$ than learning about the labels. In other words, learning about the input distribution is more important at early stages of the learning process. Note that learning about the distribution of $\mathbf{x}$ may be done in an unsupervised manner since the labels y are not needed for this task. This

behavior continues until a certain value of $H(\widehat{y})$ is reached, from that point onward the second phase starts. In the spirals dataset the first learning phase ends around $H(\widehat{y}) \approx 0.75 \log k$ for all used activation functions. The same behavior holds true for the MNIST and CIFAR-10 (see Figure 6.5 and Figure 6.6) datasets, where the second phase of learning starts at $H(\widehat{y}) \approx 0.8 \log k$. From this result, we may conjecture the existence of a fundamental relation between the prediction task and a typical value of $H(\widehat{y})$, where the second learning phase starts, which seems to be nearly independent of the neural networks architecture, as in the ideal scenario from Figure 6.4. As said, the second phase of learning starts when $H(\widehat{y})$ is high enough. Then, minimizing $H(\widehat{y}|y)$ plays a more significant role than maximizing $H(\widehat{y})$. This phase can be intuitively seen as the supervised phase of training, where neural networks learn to master the prediction task.

The behavior of $H(\widehat{y})$ and $H(\widehat{y}|y)$ during the learning process appears to be independent of the particular activation function. Hence, in the information planes of Figure 6.6, we focus on tanh for the spirals dataset, ReLU for the MNIST dataset, and assume that similar results hold for other activation functions. In that figure, the trajectory of information theoretic quantities is observed when the noise rate p is changed. We first see that $H(\widehat{y})$, $H(\widehat{y}|y)$, and $L_0(f)$ approach their corresponding bounds as the training goes on. This supports the assumption that the models considered have in fact learned correctly, since the expected error achieves the lower bound given by Fano's inequality. Note that regardless of p, the entropy $H(\widehat{y})$ approaches its maximum value, *i.e.*, $H(\widehat{y}) = \log k$ in all experiments as expected. The first phase of learning ends at the value of $H(\widehat{y})$ where $H(\widehat{y}|y)$ reaches its inflection point. This value of $H(\widehat{y})$, where the first phase ends, seems to increase with p in all simulations, as predicted in the ideal setting of Figure 6.4. This supports the idea that the channel between $\mathbf{x}$ and $\widehat{y}$ should be good enough, that is $H(\widehat{y})$ should be above some threshold value before taking full advantage of the labels in the dataset. As the labels get noisier, we need a better channel between $\mathbf{x}$ and $\widehat{y}$ in order to learn correctly.

Recall Fano's inequality (*i.e.*, Lemma 6.3.1), which gives an upper bound on $H(\widehat{y}|y)$ in terms of the expected error $L_0(f)$. Then, in the entropy-error planes of Figure 6.6, we assume that the test error is a good estimate of $L_0(f)$ and use it to compute this bound. This figure shows that the pair $(L_0(f), H(\widehat{y}|y))$ approaches the dashed curve $(x, \Psi(x))$ for correctly trained neural networks given various values of p. As expected, if neural networks are properly trained and the expected error $L_0(f)$ tends to p, the value of $H(\widehat{y}|y)$ approaches Fano's bound. When DenseNet is used for classification in the CIFAR-10 dataset, after only 10 epochs the pair $(L_0(f), H(\widehat{y}|y))$ starts to approach Fano's curve even though the error is still far from the lower bound p. This is an indication that the model is learning correctly. Note that the model eventually reaches less than 6% test error after 300 epochs [57]. These experiments show that observing the trajectory of entropy $H(\widehat{y})$, conditional entropy $H(\widehat{y}|y)$, and expected error $L_0(f)$ can serve as a method for verifying the correct learning of neural networks.

Now, we make use of Theorem 6.4.1 to predict the trajectory of DNNs in the informa-

tion plane. To that end, we keep the learning rate α and the batch size b fixed during the learning process, and study the effect of different hyperparameter configurations. In Figures 6.7 and 6.8, we simulate the learning trajectories of classifiers under various hyperparameter choices. We take the first point where $0.4 \log k \leq H(\widehat{y}) \leq 0.6 \log k$ as the initial point of Theorem 6.4.1 and use this result to predict the learning trajectory of the DNN under test. We observe that good hyperparameter configurations[8] tend to approximately follow the predicted trajectory. On the other hand, the trajectories that do not converge near the optimal point seem to deviate from the predicted trajectory from early on in training. Moreover, we can observe the impact that different hyperparameters have on these trajectories. Small learning rates produce smooth trajectories, but may not move towards the optimal point. On the other hand, extremely large learning rates prevent neural networks from learning, thus the trajectory in the information plane barely moves from its initial point. However, moderately large learning rates may produce non-smooth trajectories that roughly follow the predicted ones. Finally, the effect of the batch size is similar. Larger batch sizes lead to smoother trajectories, which follow the predicted behavior when an appropriate learning rate is chosen.

Our last experiment consists on investigating how underfitting and overfitting affects the trajectory of SGD. To that end, in Figure 6.9, we include ℓ_1-norm regularization term into the loss function controlled by a parameter λ. As expected, for sufficiently small λ the minimum error is attained. Interestingly, as we induce underfitting, by increasing this regularization coefficient, the obtained models move away from Fano's bound. This naturally leads to increased error values. On the other hand, in Figure 6.10 we increase the number of parameters of FCNN and reduce the dataset size to 10 000 in order to induce overfitting. While overfitting leads to larger error as well as underfitting, it can be distinguished by its trajectory in the entropy-error plane. Underfitting seems to push models away from Fano's bound while overfitting happens when a neural network is at this bound. We see in our experiments that learning the marginal of $\widehat{y}$ is often easy for classifiers, since it happens fast after few training iterations. From this observation, it seems that having a low $H(\widehat{y})$ value is an indication of underfitting. How does this look like in the entropy-error plane? Since

$$H(y|\widehat{y}) = H(\widehat{y}|y) - (H(y) - H(\widehat{y})) \leq \Psi(L_0(f)) - (H(y) - H(\widehat{y})) \ ,$$

we have that Fano's bound is only sharp if the classifier can learn the marginal distribution of y so that $H(y) = H(\widehat{y})$. As $H(\widehat{y})$ deviates from $H(y)$, the conditional entropy $H(y|\widehat{y})$ remains away from Fano's bound. However, one can learn the marginal of y and classify poorly, since $H(y) = H(\widehat{y})$ does not guarantee that $H(y|\widehat{y})$ is small. In the entropy-error plane this is equivalent to moving along Fano's bound. We expect a good learner to start decreasing $H(y|\widehat{y})$ once it has learned about the marginal of y. When a classifier increases $H(y|\widehat{y})$ despite having already learned the marginal of y, thus moving away from the objective, it is an indication of overfitting. In underfitting, a classifier moves towards the optimal point but its limited expressiveness prevents it from

[8] A good hyperparameter configuration is one that ultimately leads to low test error.

$H(\widehat{y})$ remains	$H(y\|\widehat{y})$ significantly	indicates
low	decreases	underfitting
low	increases	not possible
high	decreases	good fit
high	increases	overfitting

Table 6.2.: Summary of training problems that can be spotted using the information plane.

reaching it, while in overfitting the classifier starts moving away from it. The later can be seen using traditional training and test error, since the test error starts increasing. Nevertheless, the advantage of the entropy error plane is that we can observe this effect happening even before the error increases significantly, since $H(y|\widehat{y})$ may change more drastically than $L_0(f)$. In Table 6.2 we summarize how to use information theoretic quantities as indicators for the state of learning. Note that the case where $H(\widehat{y})$ is low and $H(y|\widehat{y})$ high is not possible since

$$H(y|\widehat{y}) = H(\widehat{y}|y) - (H(y) - H(\widehat{y})) \leq H(\widehat{y}|y) \leq H(\widehat{y})\,.$$

These experiments show that the information plane provides a richer view beyond train and test error that allow us to observe effects that were previously hidden. Further understanding about these trajectories is interesting since it may allow practitioners to monitor models during training, spot undesired behaviors, and possibly tune hyperparameters accordingly.

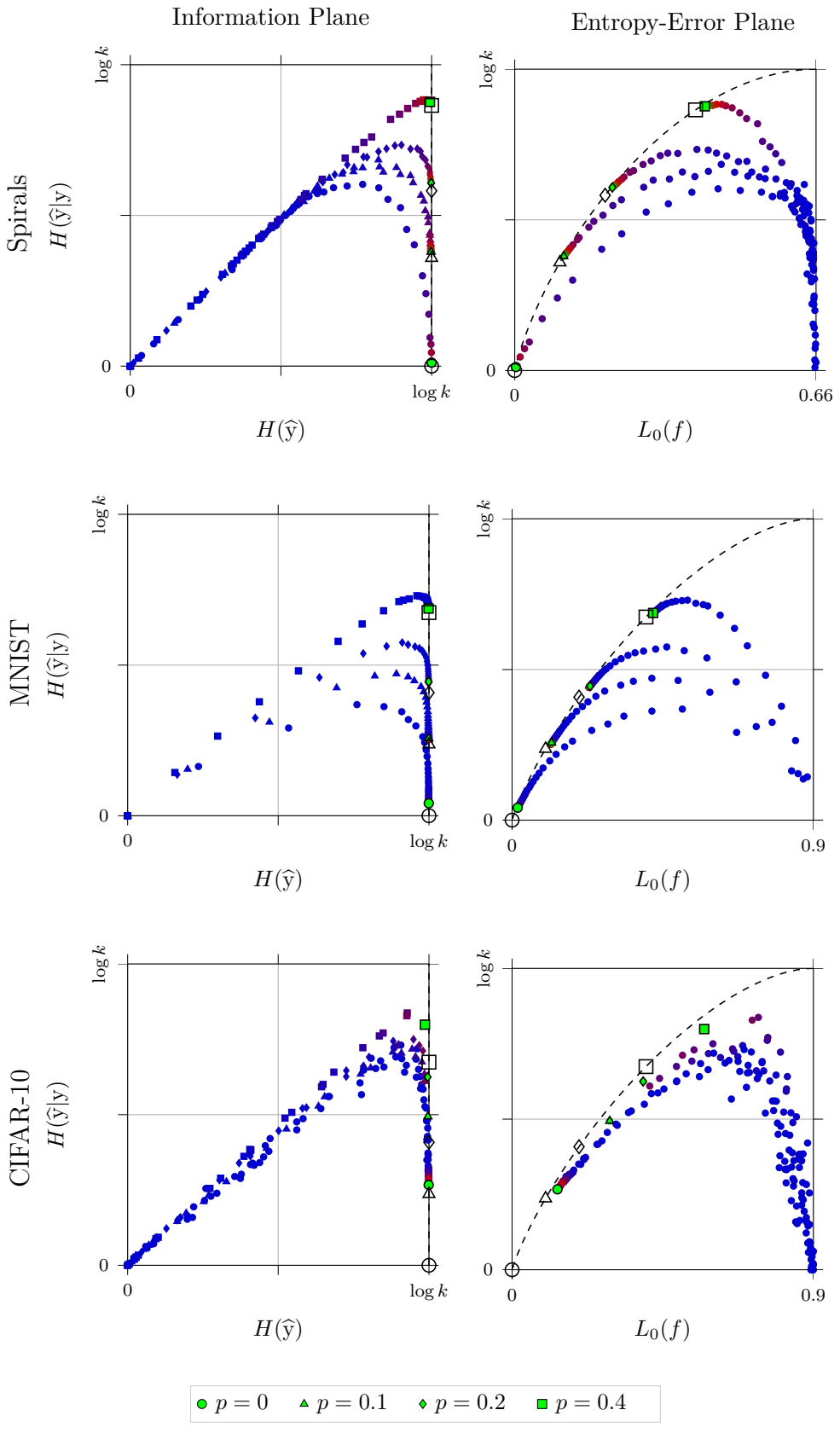


Figure 6.6.: Information theoretic quantities during the learning process. The black dashed lines depict Fano's bound (*i.e.*, Lemma 6.3.1). Various marker shapes are used to distinguish between experiments with different values of p. Color coding is done as in Figure 6.5.

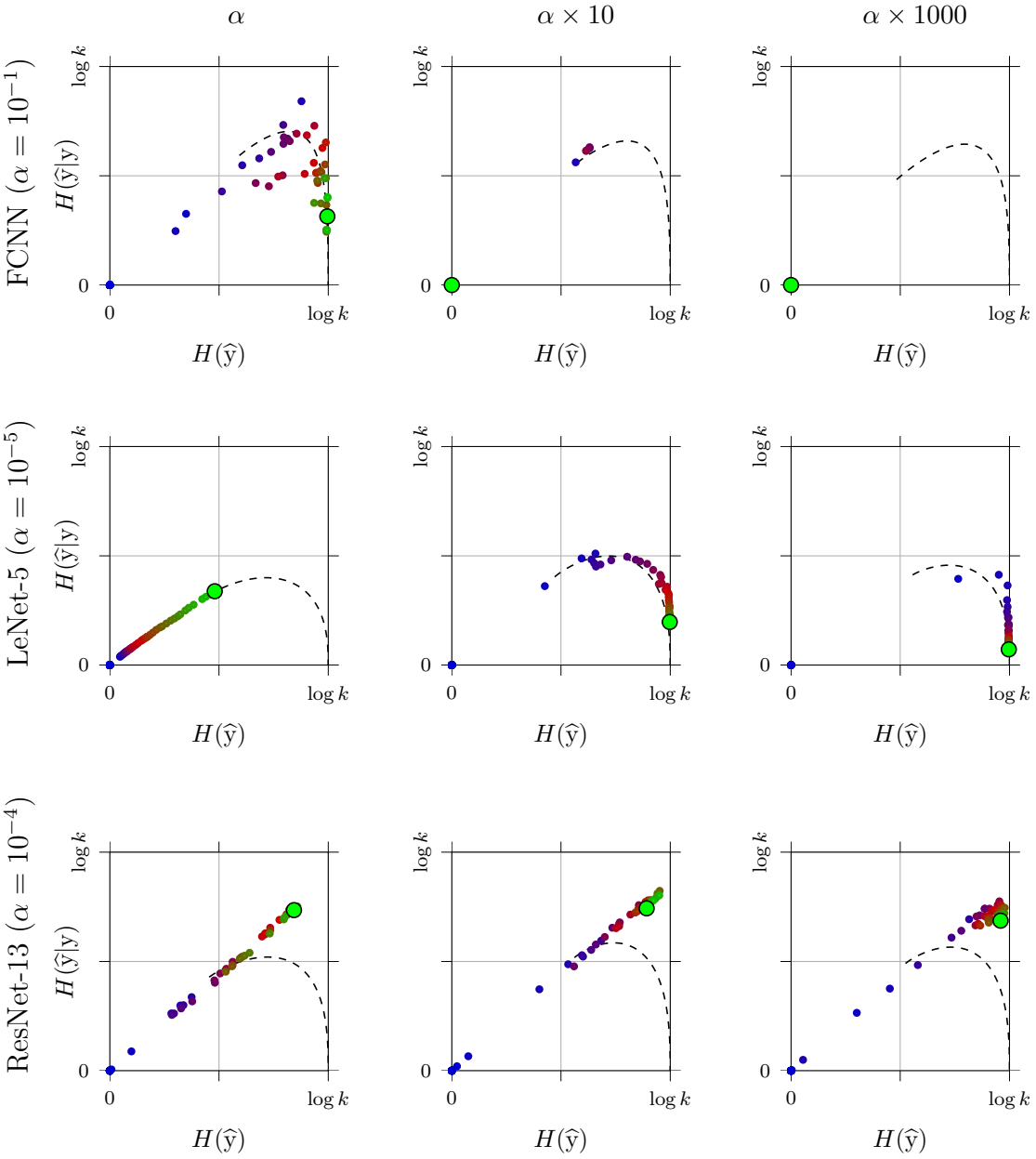


Figure 6.7.: Trajectory in the information plane, with different learning rates α, for FCNN, LeNet-5 and ResNet-13 in the Spirals, MNIST and CIFAR-10 datasets respectively. The dashed line shows the trajectory predicted using Theorem 6.4.1. Color coding is done as in Figure 6.5.

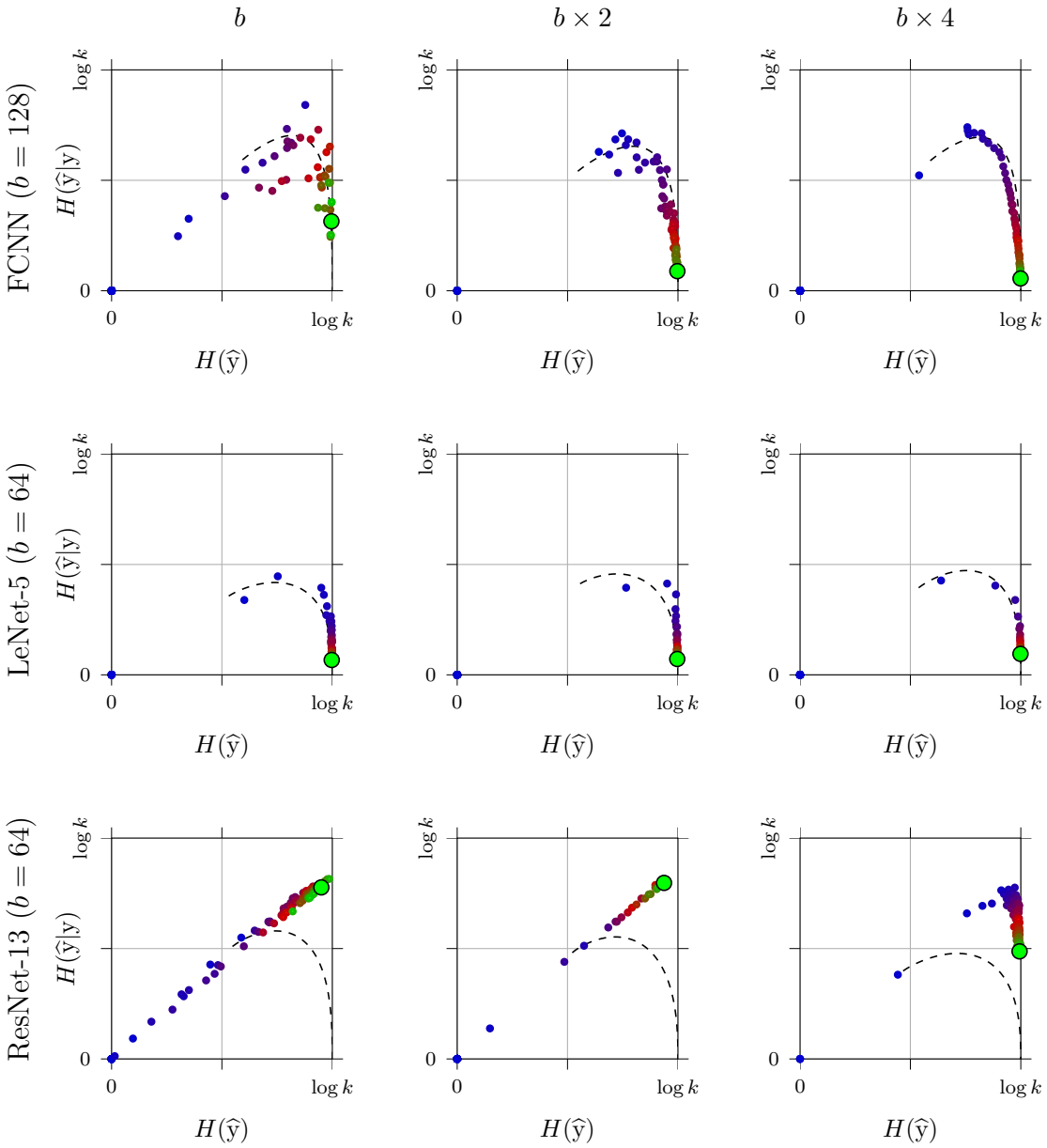


Figure 6.8.: Trajectory in the information plane, with different batch sizes b, for FCNN, LeNet-5 and ResNet-13 in the Spirals, MNIST and CIFAR-10 datasets respectively. The dashed line shows the trajectory predicted using Theorem 6.4.1. Color coding is done as in Figure 6.5.

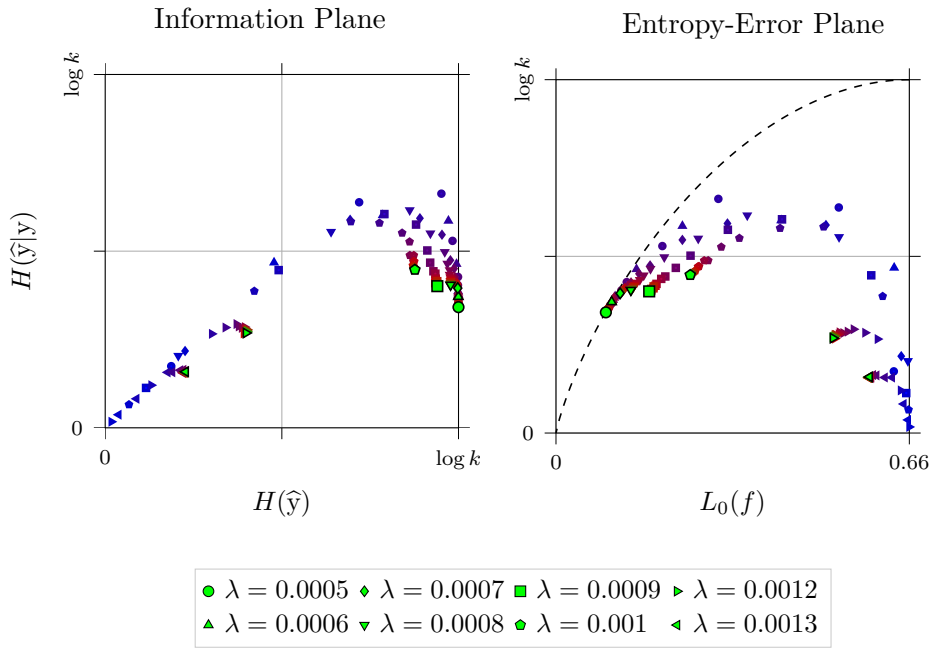


Figure 6.9.: Trajectory in the information plane for underfitted versions FCNN in the MNIST dataset. The parameter λ controls the amount of underfitting and $p = 0.1$. The dashed line shows Fano's bound. Color coding is done as in Figure 6.5.

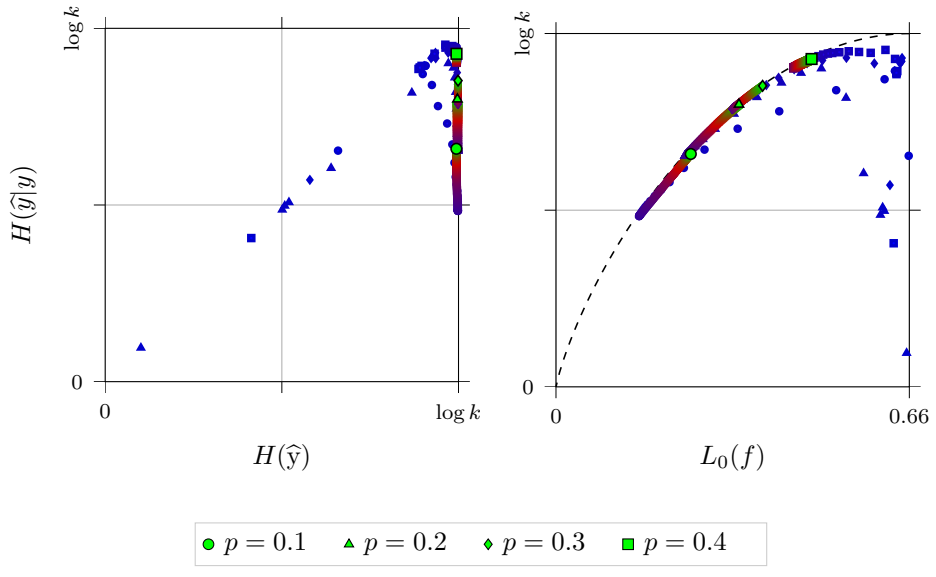


Figure 6.10.: Trajectory in the information plane for overfitted versions FCNN in the MNIST dataset. Different marker shapes correspond to different values of p. The dashed line shows Fano's bound. Color coding is done as in Figure 6.5.

7

Conclusions

The expanding availability of computational resources and hardware advances allows us to train larger and more sophisticated neural networks. However, theoretical advances for understanding these models are not keeping up with this pace of growth. Additionally, the problem of adversarial vulnerability of these networks is far from being solved, especially in real-world scenarios with high-dimensional data. In such scenarios, adversarial training is still computationally costly, which motivates the search for alternate solutions based on the theoretical understanding of adversarial robustness. Moreover, obtaining large amounts of data to train these models is costly and may come with low quality labels. This makes the study of learning under label noise a relevant task as well.

7.1. Summary of Contributions

First, in Chapter 3, we proposed a general framework based on convex approximations for generating adversarial examples with different desired constraints. Using this approach, we derived novel adversarial attacks for the context of classification and regression. For classification, our proposed attacks outperform consistently existing methods in the tested scenarios. For the case of regression, we showed that there exist many problems where DNNs are vulnerable to adversarial attacks. This fact was previously

overlooked in the literature, since existing works focused mainly on autoencoders, a regression problem that tends to be robust against adversarial perturbations.

In Chapter 4, we studied the relation between the adversarial robustness of a classifier and the simultaneous sparsity and low-rankness of its weight matrices. Empirically, we showed how adversarial training seems to induce these sparse and low-rank structures. We demonstrated that promoting low-rankness, on sparse classifiers, also leads to increased robustness. These findings are supported with a theorem on linear classifiers, which showed how low-rankness improves the robustness of effectively sparse classifiers without compromising standard accuracy.

Later, in Chapter 5, we have established adversarial risk bounds for DNNs under ℓ_∞ attacks. Our result has improved existing generalization bounds in terms of dependencies with the number of classes, the input dimension, and the norm of the inputs. This generalization bound has shown that effective sparsity does not only improve robustness, but results in better generalization of DNNs. Moreover, through empirical simulations, we found our notion of effective joint sparsity to be specially relevant for providing generalization guarantees under adversarial perturbations. This connection has not been discovered so far in existing works.

Finally, in Chapter 6, we studied the problem of learning with label noise using information theoretic quantities. To that end, we proposed to study the problem through a novel notion of information plane that requires information theoretic quantities which can be efficiently estimated. We derived several fundamental relations between these quantities and different types of error. Then, for the case of linear classifiers and linearly separable data, we characterized the trajectory of these classifiers in the information plane when the weights move towards the optimal point. Empirical simulations showed the resemblance of this behavior with neural networks, in more realistic scenarios, and showed the potential of using information theoretic quantities for detecting undesired learning phenomena.

7.2. Outlook

In this work, we proposed several techniques for generating adversarial noise. When adversarial examples exist, these methods seem to find them efficiently, but they are not guaranteed to do so. Constructing scenarios where we can provide theoretical guarantees for finding adversarial examples, using these algorithms, would increase our understating about the nature of the problem. While we showed that simultaneous low-rankness and sparsity is relevant for adversarial robustness, the effect of such combination of low-complexity structures on generalization was not studied. Using the compression technique from Chapter 5 to establish such results seems like a promising research direction, since imposing these low-complexity structures into the weight matrices of DNNs allows for compressing them efficiently. Another interesting direction is to use

the theoretical result from Chapter 5 to build regularization or optimization schemes, as in Chapter 4, that provide an alternative to adversarial training. Finally, the proposed information plane showed great potential for detecting learning problems. Using this information plane to design tracking methods for online hyperparameter tunning would explore further this idea.

A

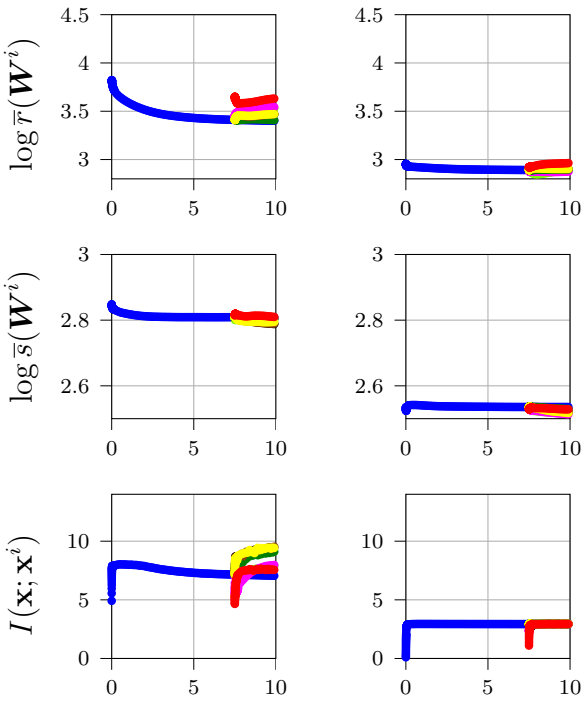
Appendix

A.1. Additional Experiments for Chapter 4

Tables A.1 and A.2 are self-explanatory. Figure A.1 shows the effective rank, effective sparsity, and mutual information corresponding to the same setup as in Figure 4.4, but on different layers.

Method	Test accuracy (non-adversarial)
Natural training	0.9508
FGSM adversarial training	0.9697
PGD adversarial training	0.9689
$\lambda_{1,1} = 0.01$	0.9527
$\lambda_{*,1} = 0.01$	0.9681
$\lambda_{1,1} = 0.01$, $\lambda_{*,1} = 0.01$	0.9446
$\lambda_{1,1} = 0.01$, $\lambda_{*,1} = 0.03$	0.9327

Table A.1.: Test accuracies of FCNN for the MNIST dataset.



(a) $i = 5$ **(b)** $i = 6$

Figure A.1.: Effective rank, effective sparsity, and mutual information corresponding to layers 5 and 6 of FCNN (as in Figure 4.4).

A.2. Reshaping of Convolutional Filters in Chapter 4

Assume a hidden representation composed of h_1 non-overlapping 3-dimensional patches represented by the tensors $\mathbf{X}_1, \ldots, \mathbf{X}_{h_1}$ of order 3. A convolution operation is performed using h_2 convolutional filters, corresponding to the 3D tensors $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{h_2}$ of the same size as the patches, which results in the following (vectorized) output of the convolution

$$\begin{pmatrix} \mathrm{vec}(\boldsymbol{W}_1)^\top \mathrm{vec}(\boldsymbol{X}_1) \\ \mathrm{vec}(\boldsymbol{W}_1)^\top \mathrm{vec}(\boldsymbol{X}_2) \\ \vdots \\ \mathrm{vec}(\boldsymbol{W}_{h_2})^\top \mathrm{vec}(\boldsymbol{X}_{h_1}) \end{pmatrix} = \underbrace{\left(\underbrace{\begin{pmatrix} \mathrm{vec}(\boldsymbol{W}_1)^\top \\ \mathrm{vec}(\boldsymbol{W}_2)^\top \\ \vdots \\ \mathrm{vec}(\boldsymbol{W}_{h_1})^\top \end{pmatrix}}_{=:\widetilde{\boldsymbol{W}}} \otimes \mathbf{I}_{h_2} \right)}_{=:\boldsymbol{W}} \begin{pmatrix} \mathrm{vec}(\boldsymbol{X}_1) \\ \mathrm{vec}(\boldsymbol{X}_2) \\ \ldots \\ \mathrm{vec}(\boldsymbol{X}_{h_2}) \end{pmatrix} \in \mathbb{R}^{h_1 h_2}.$$

Therefore, this convolution operation can be written as a standard fully connected layer with weight matrix $\boldsymbol{W} = \widetilde{\boldsymbol{W}} \otimes \mathbf{I}_{h_1}$. Note that $\boldsymbol{W}$ and $\widetilde{\boldsymbol{W}}$ have the same effective

Method	Test accuracy (non-adversarial)
Natural train.	0.9041
FGSM adv. train.	0.8639
PGD adv. train.	0.864
$\lambda_{1,1} = 0.01$, $\lambda_{1,2} = 0.01$, $\lambda_{*,2} = 0$	0.8917
$\lambda_{1,1} = 0.01$, $\lambda_{1,2} = 0.01$, $\lambda_{*,2} = 0.05$	0.8623
$\lambda_{1,1} = 0.01$, $\lambda_{1,2} = 0.01$, $\lambda_{*,2} = 0.1$	0.8614
$\lambda_{1,1} = 0.01$, $\lambda_{1,2} = 0.01$, $\lambda_{*,2} = 0.5$	0.8399

Table A.2.: Test accuracies of CNN for the F-MNIST dataset.

rank, and $\widetilde{\boldsymbol{W}}$ is a reshaped version of the 4D weight tensor composed of $W_1, \ldots, W_{h_2}$. Therefore, the matrix $\widetilde{\boldsymbol{W}}$ is used in (4.7) and (4.8) to regularize convolutional layers.

A.3. Proof of Lemma 6.4.2

Proof. For the sake of notation let us denote $P_{i,j} = \mathbb{P}[\widehat{y}^{(t)} = i | y = j]$ for $i, j = 1, \ldots, k$ and use the shorthand notation $\beta := \beta(t)$ and $\widehat{y} := \widehat{y}^{(t)}$. Then, for uniformly distributed y we can express $H(\widehat{y}|y)$ as

$$\begin{aligned} H(\widehat{y}^{(t)}|y) &= -\frac{1}{k}\sum_{j=1}^{k}\sum_{i=1}^{k}\mathbb{P}[\widehat{y} = i|y = j]\mathbb{P}[y = j]\log\mathbb{P}[\widehat{y} = i|y = j] \\ &= -\frac{1}{k}\sum_{j=1}^{k}\sum_{i=1}^{k}P_{i,j}\log P_{i,j}\,. \end{aligned}$$

Now, we can take its derivative with respect to β yielding

$$\begin{aligned} \frac{H(\widehat{y}^{(t)}|y)}{\partial\beta} &= \frac{\partial}{\partial\beta}\left[-\frac{1}{k}\sum_{j=1}^{k}\sum_{i=1}^{k}P_{i,j}\log P_{i,j}\right] \\ &= -\frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{k}\left[\frac{\partial P_{i,j}}{\partial\beta}(1+\log P_{i,j})\right] \\ &= -\frac{1}{k}\sum_{i=1}^{1}\sum_{j=j}\left[\frac{\partial P_{i,j}}{\partial\beta}(1+\log P_{i,j})\right] - \frac{1}{k}\sum_{i=1}^{1}\sum_{j\neq i}\left[\frac{\partial P_{i,j}}{\partial\beta}(1+\log P_{i,j})\right] \\ &\quad - \frac{1}{k}\sum_{i=2}^{k}\sum_{j=i}\left[\frac{\partial P_{i,j}}{\partial\beta}(1+\log P_{i,j})\right] - \frac{1}{k}\sum_{i=2}^{k}\sum_{i\neq j}\left[\frac{\partial P_{i,j}}{\partial\beta}(1+\log P_{i,j})\right]. \end{aligned}$$

Therefore, it sufices to obtain an expression for $P_{i,j}$ (and its derivative) in order to characterize $H(\widehat{y}^{(t)}|y)$.

The starting point $\mathcal{D}^{(0)}_{\widetilde{Y},\widehat{Y}}$ is assumed to be such that $H(\widehat{y}^{(0)}) = 0$. Without loss of generality let us assume that $\mathbb{P}[\widehat{y}^{(0)} = 1] = 1$. Then, since $\widetilde{y}$ is assumed to be uniformly distributed, this leads to

$$\mathbb{P}[\widehat{y} = i, \widetilde{y} = j] = (1-\beta)\frac{1}{k}\mathbb{1}_{(i=1)} + \beta\frac{1}{k}\mathbb{1}_{(i=j)}\,.$$

Knowing this joint distribution allows us to easily compute the marginal

$$\mathbb{P}[\widehat{y} = i] = \sum_j \mathbb{P}[\widehat{y} = i, \widetilde{y} = j] = (1-\beta)\mathbb{1}_{(i=1)} + \beta\frac{1}{k}\,.$$

On the other hand, since z is independent, we can rewrite the conditional probabilities $\mathbb{P}[y = 1|\widehat{y} = l]$ as

$$\begin{pmatrix}\mathbb{P}[y=1|\widehat{y}=l]\\ \vdots \\ \mathbb{P}[y=k|\widehat{y}=l]\end{pmatrix} = \begin{pmatrix}\mathbb{P}[\widetilde{y}\oplus z=1|\widehat{y}=l]\\ \vdots \\ \mathbb{P}[\widetilde{y}\oplus z=k|\widehat{y}=l]\end{pmatrix} = \begin{pmatrix}\mathbb{P}[\widetilde{y}=1|\widehat{y}=l]\\ \vdots \\ \mathbb{P}[\widetilde{y}=k|\widehat{y}=l]\end{pmatrix} \circledast \begin{pmatrix}\mathbb{P}[z=0]\\ \vdots \\ \mathbb{P}[z=k-1]\,,\end{pmatrix}$$

where $\circledast$ denotes the circular convolution operator. Then, for z distributed according to (6.7) we get

$$\begin{aligned}\begin{pmatrix}\mathbb{P}[y=1|\widehat{y}=i]\\ \vdots \\ \mathbb{P}[y=k|\widehat{y}=i]\end{pmatrix} &= \underbrace{((1-p-\frac{p}{k-1})\mathbf{I} + \frac{p}{k-1}\mathbf{1}\mathbf{1}^\top)}_{\text{circular convolution matrix of } \mathbb{P}[z=\cdot]}\begin{pmatrix}\mathbb{P}[\widetilde{y}=1|\widehat{y}=i]\\ \vdots \\ \mathbb{P}[\widetilde{y}=k|\widehat{y}=i]\end{pmatrix}\\ &= (1-p-\frac{p}{k-1})\begin{pmatrix}\mathbb{P}[\widetilde{y}=1|\widehat{y}=i]\\ \vdots \\ \mathbb{P}[\widetilde{y}=k|\widehat{y}=i]\end{pmatrix} + \frac{p}{k-1}\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}\,.\end{aligned}$$

Now, we are ready to express $P_{i,j}$ as a funtion of β, that is

$$\begin{aligned}P_{i,j} &= \frac{\mathbb{P}[y=j|\widehat{y}=i]\mathbb{P}[\widehat{y}=i]}{\mathbb{P}[y=j]}\\ &= (1-p-\frac{p}{k-1})\cdot\frac{\mathbb{P}[\widetilde{y}=j|\widehat{y}=i]\mathbb{P}[\widehat{y}=i]}{1/k} + \frac{p}{k-1}\cdot\frac{\mathbb{P}[\widehat{y}=i]}{1/k}\\ &= (1-p-\frac{p}{k-1})\cdot\frac{\mathbb{P}[\widetilde{y}=j,\widehat{y}=i]}{1/k} + \frac{p}{k-1}\cdot\frac{\mathbb{P}[\widehat{y}=i]}{1/k}\\ &= (1-p-\frac{p}{k-1})\left[(1-\beta)\mathbb{1}_{(i=1)} + \beta\mathbb{1}_{(i=j)}\right] + \frac{kp}{k-1}\left[(1-\beta)\mathbb{1}_{(i=1)} + \frac{\beta}{k}\right]\\ &= \begin{cases}1-p-\frac{p}{k-1} + \frac{kp}{k-1}(1-\beta) + \frac{p}{k-1}\beta, & i=1, j=i\\ (1-p)\beta, & i\neq 1, j=i\\ (1-p-\frac{p}{k-1})(1-\beta) + \frac{kp}{k-1}(1-\beta) + \frac{p}{k-1}\beta, & i=1, i\neq j\\ \frac{p}{k-1}\beta, & i\neq 1, i\neq j\end{cases}.\end{aligned}$$

Therefore, we have that

$$\frac{\partial P_{i,j}}{\partial \beta} = \begin{cases} -p, & i = 1, j = i \\ 1 - p, & i \neq 1, j = i \\ \frac{p}{k-1} - 1, & i = 1, j \neq i \\ \frac{p}{k-1}, & i \neq 1, j \neq i \end{cases}.$$

Finally, joining these expresions together yields

$$\begin{aligned}
\frac{H(\widehat{y}^{(t)}|y)}{\partial \beta} &= -\frac{1}{k}\sum_{i=1}^{1}\sum_{j=1}\left[-p(1+\log(1-\beta+\beta(1-p)))\right] \\
&\quad -\frac{1}{k}\sum_{i=1}^{1}\sum_{j\neq i}\left[(\frac{p}{k-1}-1)(1+\log(1-\beta+\beta\frac{p}{k-1}))\right] \\
&\quad -\frac{1}{k}\sum_{i=2}^{k}\sum_{j=i}\left[(1-p)(1+\log(\beta(1-p)))\right] - \frac{1}{k}\sum_{i=2}^{k}\sum_{j\neq i}\left[\frac{p}{k-1}(1+\log(\beta\frac{p}{k-1}))\right] \\
&= \frac{1}{k}p\log(1-\beta p) - \frac{1}{k}(k-1)(\frac{p}{k-1}-1)(1+\log(1-\beta+\beta\frac{p}{k-1})) \\
&\quad -\frac{1}{k}(k-1)(1-p)(1+\log(\beta(1-p))) - \frac{1}{k}(k-1)p(1+\log(\beta\frac{p}{k-1})) \\
&\quad \frac{1}{k}\underbrace{\left(p-(k-1)(\frac{p}{k-1}-1)-(k-1)(1-p)-(k-1)p\right)}_{=0} \\
&= \frac{1}{k}\Big[p\log(1-\beta p) + (k-1-p)\log(1-\beta+\beta\frac{p}{k-1}) \\
&\quad -(k-1)(1-p)\log(\beta(1-p)) - (k-1)p\log(\beta\frac{p}{k-1})\Big].
\end{aligned}$$

Remark: For $p \to 0$, we now that the maximum the curve is at $\beta = \frac{1}{2}$, since

$$\begin{aligned}
\lim_{p\to 0}\frac{H(\widehat{y}^{(t)}|y)}{\partial\beta} \overset{!}{=} 0 \;&\Rightarrow\; \frac{1}{k}\left[(k-1)\log(1-\beta) - (k-1)\log\beta\right] = 0 \\
&\Rightarrow\; \log\frac{1-\beta}{t} = 0 \;\Rightarrow\; \beta = \frac{1}{2}.
\end{aligned}$$

□

List of Acronyms

i.i.d.	Independent and identically distributed
BIM	Basic Iterative Method
CCN	Class-Conditional random label Noise
CNN	Convolutional Neural Network
DNN	Deep Neural Network
ERM	Empirical Risk Minimization
FCNN	Fully Connected Neural Network
FGSM	Fast Gradient Sign Method
GNM	Gradient-based Norm-constrained Method
KDE	Kernel-based Density Estimator
PAC	Probably Approximately Correct
PGD	Projected Gradient Descent
PSNR	Peak Signal to Noise Ratio
RCN	Random Class Noise
R-FGSM	Randomized Fast Gradient Sign Method
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
SVM	Support Vector Machine
VC	Vapnik-Chervonenkis

List of Symbols and Notation

General Notation

$a, b, c, \ldots$	scalars
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$	column vectors
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$	matrices
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	sets
$\lvert\cdot\rvert$	absolute value of a scalar or cardinality of a set
$(\cdot)^\top$	matrix transpose
$\operatorname{vec}(\cdot)$	vectorization of a matrix
$\langle\cdot, \cdot\rangle$	Euclidean inner product between two vectors
$\lVert\cdot\rVert_{p,q}$	mixed (p, q)-norm of a matrix
$\lVert\cdot\rVert_{p\to q}$	operator (p, q)-norm of a matrix
$\lVert\cdot\rVert_p$	ℓ_p-norm of a vector, for matrices this is shorthand of $\lVert\cdot\rVert_{p\to p}$
$\lVert\cdot\rVert_{\mathrm{F}}$	Frobenius norm of a matrix
$\mathbb{1}_{(\cdot)}$	indicator function
$\mathbb{P}[\cdot]$	probability of an event
$\mathbb{E}[\cdot]$	expectation of a random process
$\operatorname{Var}[\cdot]$	variance of a random process

Predefined vectors, matrices and sets

$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real numbers
$\mathbf{I}$	identity Matrix
$\mathbf{0}$	all zeros vector
$\mathbf{1}$	all ones vector
$\mathbf{e}_i$	i-th standard basis vector (the i-th column of $\mathbf{I}$)

Domain Specific Notation

$\mathcal{X}$	input space, *i.e.*, the set all possible inputs
$\mathcal{Y}$	label space, *i.e.*, the set all possible labels
$\mathcal{D}$	$\mathcal{D} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ is the input data distribution
$(\mathbf{x}, y) \sim \mathcal{D}$	random input $\mathbf{x} \in \mathcal{X}$, with associated label $y \in \mathcal{Y}$, drawn from $\mathcal{D}$
$\mathcal{S}$	training set composed of independent realizations of $(\mathbf{x}, y) \sim \mathcal{D}$
m	number of samples in the training set, that is $m := \lvert\mathcal{S}\rvert$
n	input dimension, that is $\mathcal{X} \subseteq \mathbb{R}^n$
k	number of classes, that is $\mathcal{Y} = \{1, 2, \ldots, k\}$
d	number of layers of a multi-layer neural network
$\ell(\mathbf{x}, y)$	classification margin of the input $\mathbf{x}$ with associated label y
$c(\cdot)$	$c : \mathcal{X} \to \mathcal{Y}$ is the classifier function
f, g, h	score functions $\mathcal{X} \to \mathbb{R}^{\lvert\mathcal{Y}\rvert}$
$\mathcal{L}(\cdot, \cdot)$	$\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is the loss function used for training
$\mathcal{T}(\cdot, \cdot)$	$\mathcal{T} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the target loss function of an adversary
$L_\gamma(\cdot)$	expected risk with margin $\gamma \geq 0$
$\widehat{L}_\gamma(\cdot)$	empirical risk with margin $\gamma \geq 0$
$H(\cdot)$	Shannon entropy of a random variable
$I(\cdot;\cdot)$	mutual information between two random variables

Bibliography

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, 2012.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Information Processing Systems (NIPS)*, 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[7] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "Robustness of classifiers: From adversarial to random noise," in *Neural Information Processing Systems (NIPS)*, 2016.

[8] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, 2010.

[9] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, 2018.

[10] B. Wang, J. Gao, and Y. Qi, "A theoretical framework for robustness of (deep) classifiers against adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2017.

[11] A. Fawzi, O. Fawzi, and P. Frossard, "Fundamental limits on adversarial robustness," in *ICML Workshop on Deep Learning*, 2015.

[12] D. Angluin and P. Laird, "Learning from noisy examples," *Machine Learning*, 1988.

[13] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE transactions on neural networks and learning systems*, 2014.

[14] B. Frénay, A. Kabán, *et al.*, "A comprehensive introduction to label noise.," in *ESANN*, 2014.

[15] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *Conference on Artificial Intelligence (AAAI)*, 2017.

[16] A. K. Menon, B. van Rooyen, and N. Natarajan, "Learning from binary labels with instance-dependent noise," *Machine Learning*, 2018.

[17] J. A. Tropp, "Topics in sparse approximation," PhD thesis, 2004.

[18] E. R. Balda, A. Behboodi, and R. Mathar, "Perturbation analysis of learning algorithms: Generation of adversarial examples from classification to regression," *IEEE Transactions on Communications*, 2019.

[19] E. R. Balda, A. Behboodi, and R. Mathar, "On generation of adversarial examples using convex programming," in *Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, USA, 2018.

[20] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[21] S. Sarkar, A. Bansal, U. Mahbub, and R. Chellappa, "Upset and angri: Breaking high performance image classifiers," *arXiv preprint arXiv:1707.01159*, 2017.

[22] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] N. Rao, B. Recht, and R. Nowak, "Universal measurement bounds for structured sparse signal recovery," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[24] 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 (ICLR)*, 2018.

[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[26] 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 (ICML)*, 2018.

[27] A. Athalye and N. Carlini, "On the robustness of the CVPR 2018 white-box adversarial example defenses," *arXiv preprint arXiv:1804.03286*, 2018.

[28] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 2019.

[29] 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 (EuroS&P)*, 2016.

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

[31] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[32] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[33] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *IEEE International Conference on Computer Vision (CVPR)*, 2017.

[34] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," *arXiv preprint arXiv:1707.05373*, 2017.

[35] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *IEEE Military Communications Conference (MILCOM)*, IEEE, 2016.

[36] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *International Joint Conference on Artificial Intelligence (IJCAI)*, AAAI Press, 2017.

[37] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," in *IEEE Security and Privacy Workshops (SPW)*, 2018.

[38] P. Tabacof, J. Tavares, and E. Valle, "Adversarial images for variational autoencoders," *arXiv preprint arXiv:1612.00155*, 2016.

[39] T. Tanay and L. Griffin, "A boundary tilting perspective on the phenomenon of adversarial examples," *arXiv preprint arXiv:1608.07690*, 2016.

[40] A. Fawzi, S. M. Moosavi-Dezfooli, and P. Frossard, "The robustness of deep networks: A geometrical perspective," *IEEE Signal Processing Magazine*, 2017.

[41] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2018.

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

[43] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy (SP)*, IEEE, 2016.

[44] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017.

[45] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Neural Information Processing Systems (NIPS)*, 2017.

[46] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge Univ. Press, 2013.

[47] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing, Applied and Numerical Harmonic Analysis*. Birkhäuser, 2013.

[48] J. Rohn, "Computing the norm $\|A\|_{\infty,1}$ is NP-hard," *Linear and Multilinear Algebra*, 2000.

[49] D. Hartman and M. Hladík, *Tight Bounds on the Radius of Nonsingularity*. Springer, 2015.

[50] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, 1995.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[52] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," *arXiv preprint arXiv:1703.09387*, 2017.

[53] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[54] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[55] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," *Shape, Contour and Grouping in Computer Vision*, 1999.

[56] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[57] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[58] F. Baldassarre, D. G. Morín, and L. Rodés-Guirao, "Deep koalarization: Image colorization using cnns and inception-resnet-v2," *arXiv preprint arXiv:1712.03400*, 2017.

[59] M. Everingham and J. Winn, "The pascal visual object classes challenge 2012 (voc2012) development kit," *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 2011.

[60] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," in *International Conference on Learning Representations (ICLR)*, 2016.

[61] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," *arXiv preprint arXiv:1704.03453*, 2017.

[62] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," *Machine Learning*, 2018.

[63] A. Rozsa, M. Gunther, and T. E. Boult, "Towards robust deep neural networks with BANG," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[64] A. Rozsa, M. Günther, and T. E. Boult, "Are accuracy and robustness correlated," in *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016.

[65] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, "Robustness of classifiers to universal perturbations: A geometric perspective," in *International Conference on Learning Representations (ICLR)*, 2018.

[66] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *International Conference on Learning Representations (ICLR)*, 2017.

[67] S. Gopalakrishnan, Z. Marzi, U. Madhow, and R. Pedarsani, "Combating adversarial attacks using sparse representations," in *ICLR Workshop*, 2018.

[68] Z. Marzi, S. Gopalakrishnan, U. Madhow, and R. Pedarsani, "Sparsity-based defense against adversarial attacks on linear classifiers," in *IEEE International Symposium on Information Theory (ISIT)*, 2018.

[69] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse DNNs with improved adversarial robustness," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[70] L. Wang, G. W. Ding, R. Huang, Y. Cao, and Y. C. Lui, "Adversarial robustness of pruned neural networks," in *Submitted to ICLR Workshop*, 2018.

[71] A. Alemi, I. Fischer, J. Dillon, and K. Murphy, "Deep variational information bottleneck," in *International Conference on Learning Representations (ICLR)*, 2017.

[72] A. Sanyal, V. Kanade, and P. H. Torr, "Learning low-rank representations," *arXiv preprint arXiv:1804.07090*, 2018.

[73] S. Gopi, P. Netrapalli, P. Jain, and A. Nori, "One-bit compressed sensing: Provable support and vector recovery," in *International Conference on International Conference on Machine Learning (ICML)*, 2013.

[74] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *IEEE Information Theory Workshop (ITW)*, 2015.

[75] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[76] A. Kolchinsky and B. Tracey, "Estimating mixture entropy with pairwise distances," *Entropy*, 2017.

[77] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert, "Nonlinear information bottleneck," *arXiv preprint arXiv:1705.02436*, 2017.

[78] E. R. Balda, A. Behboodi, N. Koep, and R. Mathar, "Adversarial risk bounds for neural networks through sparsity based compression," *arXiv preprint arXiv:1906.00698*, 2019.

[79] H. Xu, C. Caramanis, and S. Mannor, "Robust regression and lasso," *IEEE Transactions on Information Theory*, 2008.

[80] H. Xu, C. Caramanis, and S. Mannor, "Robustness and regularization of support vector machines," *Journal of Machine Learning Research*, 2009.

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

[82] P. L. Bartlett, D. J. Foster, and M. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Neural Information Processing Systems (NIPS)*, 2017.

[83] N. Golowich, A. Rakhlin, and O. Shamir, "Size-independent sample complexity of neural networks," in *Conference on Learning Theory (COLT)*, 2018.

[84] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "Towards understanding the role of over-parametrization in generalization of neural networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[85] X. Li, J. Lu, Z. Wang, J. Haupt, and T. Zhao, "On tighter generalization bound for deep neural networks: Cnns, resnets, and beyond," *arXiv preprint arXiv:1806.05159*, 2018.

[86] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "A pac-bayesian approach to spectrally-normalized margin bounds for neural networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[87] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," in *Neural Information Processing Systems (NIPS)*, 2017.

[88] V. Nagarajan and J. Z. Kolter, "Deterministic pac-bayesian generalization bounds for deep networks via generalizing noise-resilience," in *International Conference on Learning Representations (ICLR)*, 2019.

[89] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang, "Stronger generalization bounds for deep nets via a compression approach," in *International Conference on Machine Learning (ICML)*, 2018.

[90] I. Attias, A. Kontorovich, and Y. Mansour, "Improved generalization bounds for robust learning," in *International Conference on Algorithmic Learning Theory (ALT)*, 2018.

[91] D. I. Diochnos, S. Mahloujifar, and M. Mahmoody, "Adversarial risk and robustness: General definitions and implications for the uniform distribution," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[92] S. Mahloujifar and M. Mahmoody, "Can adversarially robust learning leverage computational hardness?" In *International Conference on Algorithmic Learning Theory (ALT)*, 2019.

[93] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[94] D. Cullina, A. N. Bhagoji, and P. Mittal, "Pac-learning in the presence of adversaries," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[95] O. Montasser, S. Hanneke, and N. Srebro, "VC classes are adversarially robustly learnable, but only improperly," *arXiv preprint arXiv:1902.04217*, 2019.

[96] F. Farnia, J. M. Zhang, and D. Tse, "Generalizable adversarial training via spectral normalization," in *International Conference on Learning Representations (ICLR)*, 2019.

[97] D. Yin, R. Kannan, and P. Bartlett, "Rademacher complexity for adversarially robust generalization," in *International Conference on Machine Learning (ICML)*, 2019.

[98] J. Khim and P.-L. Loh, "Adversarial risk bounds via function transformation," *arXiv preprint arXiv:1810.09519*, 2019.

[99] Z. Tu, J. Zhang, and D. Tao, "Theoretical analysis of adversarial learning: A minimax approach," *arXiv preprint arXiv:1811.05232*, 2018.

[100] E. R. Balda, A. Behboodi, and R. Mathar, "An information theoretic view on learning of artificial neural networks," in *International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2018.

[101] J. A. Aslam and S. E. Decatur, "On the sample complexity of noise-tolerant learning," *Information Processing Letters*, 1996.

[102] C. Gentile and D. P. Helmbold, "Improved lower bounds for learning from noisy examples: An information-theoretic approach," *Information and Computation*, 2001.

[103] B. van Rooyen, A. Menon, and R. C. Williamson, "Learning with symmetric label noise: The importance of being unhinged," in *Neural Information Processing Systems (NIPS)*, 2015.

[104] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE transactions on cybernetics*, 2013.

[105] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, 2015.

[106] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Neural Information Processing Systems (NIPS)*, 2013.

[107] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, "Cost-sensitive learning with noisy labels.," *Journal of Machine Learning Research*, 2017.

[108] A. Menon, H. Narasimhan, S. Agarwal, and S. Chawla, "On the statistical consistency of algorithms for binary classification under class imbalance," in *International Conference on Machine Learning (ICML)*, 2013.

[109] H. Narasimhan, R. Vaish, and S. Agarwal, "On the statistical consistency of plug-in classifiers for non-decomposable performance measures," in *Neural Information Processing Systems (NIPS)*, 2014.

[110] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on pattern analysis and machine intelligence*, 2015.

[111] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *IEEE transactions on pattern analysis and machine intelligence*, 2009.

[112] C. Scott, G. Blanchard, and G. Handy, "Classification with asymmetric label noise: Consistency and maximal denoising," in *Conference On Learning Theory (COLT)*, 2013.

[113] R. Wang, T. Liu, and D. Tao, "Multiclass learning with partially corrupted labels," *IEEE transactions on neural networks and learning systems*, 2017.

[114] A. Menon, B. Van Rooyen, C. S. Ong, and B. Williamson, "Learning from corrupted binary labels via class-probability estimation," in *International Conference on Machine Learning (ICML)*, 2015.

[115] G. Patrini, F. Nielsen, R. Nock, and M. Carioni, "Loss factorization, weakly supervised learning and label noise robustness," in *International Conference on Machine Learning (ICML)*, 2016.

[116] C. G. Northcutt, T. Wu, and I. L. Chuang, "Learning with confident examples: Rank pruning for robust classification with noisy labels," *arXiv preprint arXiv:1705.01936*, 2017.

[117] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[118] C. Scott, "A rate of convergence for mixture proportion estimation, with application to learning from noisy labels," in *Artificial Intelligence and Statistics (AISTATS)*, 2015.

[119] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[120] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 1999.

[121] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," in *International Conference on Learning Representations (ICLR)*, 2018.

[122] M. Gabrié, A. Manoel, C. Luneau, N. Macris, F. Krzakala, L. Zdeborová, *et al.*, "Entropy and mutual information in models of deep neural networks," 2018.

[123] D. Russo and J. Zou, "How much does your data exploration overfit? Controlling bias via information usage," *arXiv preprint arXiv:1511.05219*, 2015.

[124] A. Xu and M. Raginsky, "Information-theoretic analysis of generalization capability of learning algorithms," in *Neural Information Processing Systems (NIPS)*, 2017.

[125] A. Asadi, E. Abbe, and S. Verdú, "Chaining mutual information and tightening generalization bounds," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[126] M. Feder and N. Merhav, "Relations between entropy and error probability," *IEEE Transactions on Information Theory*, 1994.

[127] I. Csiszár and J. Körner, *Information theory: coding theorems for discrete memoryless systems*, 2nd ed. Cambridge ; New York: Cambridge University Press, 2011.

[128] G. A. Miller, "Note on the bias of information estimates," *Information theory in psychology: Problems and methods*, 1955.

[129] T. Schürmann, "Bias analysis in entropy estimation," *Journal of Physics A: Mathematical and General*, 2004.

[130] E. Archer, I. M. Park, and J. W. Pillow, "Bayesian entropy estimation for countable discrete distributions," *The Journal of Machine Learning Research*, 2014.

Curriculum Vitæ

Emilio Rafael Balda Cañizares

Sep. 15, 1991	Born in Guayaquil, Ecuador
Apr. 1998 - Dec. 2003	Elementary school "Unidad Educativa Torremar", Guayaquil, Ecuador
Apr. 2004 - Jan. 2010	Academic high school "Unidad Educativa Torremar", Guayaquil, Ecuador
Sep. 2010 - Jun. 2014	Bachelor Telecommunication Systems Engineering University of Navarra, San Sebastian, Spain
Oct. 2014 - Jan. 2017	Master of Science Communications and Signal Processing Ilmenau University of Technology, Ilmenau, Germany
Feb. 2017 - Oct. 2019	Research and Teaching Assistant Institute for Theoretical Information Technology RWTH Aachen University, Aachen, Germany
Nov. 2019 - *Present*	Artificial Intelligence (AI) Developer aiXbrain GmbH, Aachen, Germany