Enhancing Robustness of Neural Networks through Fourier Stabilization

Netanel Raviv [1] **Aidan Kelley** [1] **Michael Guo** [1] **Yevgeny Vorobeychik** [1]

Abstract

Despite the considerable success of neural networks in security settings such as malware detection, such models have proved vulnerable to evasion attacks, in which attackers make slight changes to inputs (e.g., malware) to bypass detection. We propose a novel approach, *Fourier stabilization*, for designing evasion-robust neural networks with binary inputs. This approach, which is complementary to other forms of defense, replaces the weights of individual neurons with robust analogs derived using Fourier analytic tools. The choice of which neurons to stabilize in a neural network is then a combinatorial optimization problem, and we propose several methods for approximately solving it. We provide a formal bound on the per-neuron drop in accuracy due to Fourier stabilization, and experimentally demonstrate the effectiveness of the proposed approach in boosting robustness of neural networks in several detection settings. Moreover, we show that our approach effectively composes with adversarial training.

1. Introduction

Deep neural network models demonstrate human-transcending capabilities in many applications, but are often vulnerable to attacks that involve small (in ℓ_p-norm) adversarial perturbations to inputs (Szegedy et al., 2014; Goodfellow et al., 2015; Madry et al., 2017). This issue is particularly acute in security applications, where a common task is to determine whether a particular input (e.g., executable, twitter post) is malicious or benign. In these settings, malicious parties have a strong incentive to redesign inputs (such as malware) in order to *evade* detection by deep neural network-based detectors, and there have now been a series of demonstrations of successful evasion attacks (Grosse et al., 2016; Li & Vorobeychik, 2018; Laskov et al., 2014; Xu et al., 2016). In response, a number of approaches have been proposed to create models that are more robust to evasion attacks (Cohen et al., 2019; Lecuyer et al., 2019; Raghunathan et al., 2018; Wong & Kolter, 2018; Wong et al., 2018), with methods using adversarial training—where models are trained by replacing regular training inputs with their adversarially perturbed variants—remaining the state of the art (Goodfellow et al., 2015; Madry et al., 2017; Tong et al., 2019; Vorobeychik & Kantarcioglu, 2018). Nevertheless, despite considerable advances, increasing robustness of deep neural networks to evasion attacks typically entails a considerable decrease in accuracy on unperturbed (clean) inputs (Madry et al., 2017; Wu et al., 2020).

We propose a novel approach for enhancing robustness of deep neural networks with binary inputs to adversarial evasion that leverages Fourier analysis of Boolean functions (O'Donnell, 2014). Unlike most prior approaches for boosting robustness, which aim to refactor the entire deep neural network, say, through adversarial training, our approach is more fine-grained, applied at the level of individual neurons. Specifically, we start by treating neurons as linear classifiers over binary inputs, and considering their robustness as the problem of maximizing the average distance of *all inputs in the input space* from the neuron's decision boundary. We then derive a closed-form solution to this optimization problem; the process of replacing the original weights by their more robust variants, given by this solution, is called *Fourier stabilization of neurons*. Further, a bound for the per-neuron drop in accuracy due to this process is derived.

This idea applies to most common activation functions, such as $\mathrm{logistic}$, $\tanh$, erf, and ReLU (treating activation as a binary decision). Finally, we determine which subset of neurons in a neural network to stabilize. While this is a difficult combinatorial optimization problem, we develop several effective algorithmic approaches for it.

Our full approach, which we call *Fourier stabilization of a neural network* (abbrv. stabilization), applies only to neural networks with binary inputs, and is targeted at security applications, where binary inputs are common and, indeed, it is often the case that binarized inputs outperform real-valued

[1]Department of Computer Science and Engineering, Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63103. Correspondence to: Netanel Raviv <netanel.raviv@wustl.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

alternatives (Šrndić & Laskov, 2016; Tong et al., 2019). We emphasize that our approach is *complementary* to alternative defenses: it applies *post-training*, and can thus be easily composed with any defense, such as adversarial example detection (Xu et al., 2018) or adversarial training. Moreover, as our approach does not require any training data (as it stabilizes neurons directly), it can even apply to settings where one has a neural network that needs to be made more robust, but not training data, which is sensitive (e.g., in medical and cybersecurity applications where data contains sensitive or classified information). Access to training data, however, enables the additional benefit of estimating robustness and accuracy in practice; we use this approach in our experiments to decide which subset of neurons to stabilize.

We experimentally evaluate the proposed *Fourier stabilization* approach on several datasets involving detection of malicious inputs, including malware detection and hate speech detection. Our experiments show that our approach considerably improves neural network robustness to evasion in these domains, and effectively composes with adversarial training defense.

Our contribution We begin in Section 2 by familiarizing the reader with the formal definition of robustness (specifically, *prediction change* by (Diochnos et al., 2018)), its geometric interpretation, and provide some necessary background on Fourier analysis of Boolean functions. We proceed in Section 3 by formulating the stabilization of neurons as an optimization problem, and solving it analytically for the ℓ_1-metric in Section 3.1 (the solution for all other ℓ_p-metrics is given in (Raviv et al., 2021)). In Section 3.2 we employ probabilistic tools from (O'Donnell, 2014; O'Donnell & Servedio, 2011; Matulef et al., 2010) (among others) to bound the loss of accuracy that results from stabilization of a neuron, i.e., the fraction of inputs that would lie on the "wrong" side of its original decision boundary.

In Section 4 the discussion is extended to neural networks. It is observed that stabilizing the entire first layer might not be effective for improving robustness while maintaining accuracy. Instead, one should find an optimal subset of those, whose stabilization increases robustness the most, while maintaining bounded loss of accuracy. Since this combinatorial optimization problem is hard to solve in general, we suggest a few heuristics. The efficacy of these heuristics is demonstrated in Section 5 by showing improved accuracy-robustness tradeoff in classifying several commonly used cybersecurity datasets under state-of-the-art attacks. Further, it is also demonstrated that these techniques can be effectively used in conjunction with adversarial training. Future research directions are discussed in Section 6.

2. Preliminaries

For $\mathbf{w} \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$, denote the hyperplane $\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x}\mathbf{w}^\intercal = \theta\}$ by $\mathcal{H}(\mathbf{w}, \theta)$. Our fundamental technique operates at the level of neurons in a neural network, which we treat as (generalized) linear models. We start by considering linear models of the form $h(\mathbf{x}) = \operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal - \theta)$ that map binary inputs $\mathbf{x} \in \{\pm 1\}^n$ to binary outputs; below, we discuss how the machinery we develop applies to a variety of activation functions. For $1 \leq p \leq \infty$ let d_p and $\|\cdot\|_p$ be the ℓ_p-distance and ℓ_p-norm, respectively. That is, for vectors $\mathbf{v} = (v_i)_{i=1}^n$ and $\mathbf{u} = (u_i)_{i=1}^n$ let $\|\mathbf{v}\|_p = (\sum_{i=1}^n |v_i|^p)^{1/p}$ (or $\max\{|v_i|\}_{i=1}^n$ if $p = \infty$) and $d_p(\mathbf{v}, \mathbf{u}) = \|\mathbf{v} - \mathbf{u}\|_p$. For real numbers $q, p \geq 1$, the norms ℓ_p and ℓ_q are called *dual* if $\frac{1}{p} + \frac{1}{q} = 1$. For example, the dual norm of ℓ_2 is itself, and the dual norm of ℓ_1 is ℓ_∞. In the remainder of this paper, ℓ_p and ℓ_q denote dual norms. We will make use of the following theorem:

Theorem 1. *(Melachrinoudis, 1997) (Sec. 5) For a hyperplane $\mathcal{H}(\mathbf{v}, \mu) \subseteq \mathbb{R}^n$, a point $\mathbf{z} \in \mathbb{R}^n$, and any $p \geq 1$, let $d_p(\mathbf{z}, \mathcal{H}(\mathbf{v}, \mu))$ denote the ℓ_p-distance of $\mathcal{H}(\mathbf{v}, \mu)$ from $\mathbf{z}$, i.e., $\min\{d_p(\mathbf{u}, \mathbf{z}) | \mathbf{u} \in \mathcal{H}(\mathbf{v}, \mu)\}$. Then, we have $d_p(\mathbf{z}, \mathcal{H}(\mathbf{v}, \mu)) = \frac{|\mathbf{z} \cdot \mathbf{v}^\intercal - \mu|}{\|\mathbf{v}\|_q}$.*

2.1. Definition of Robustness

We operate under the geometric interpretation of robustness, in which the adversary is given a random $\mathbf{x} \in \{\pm 1\}^n$, and would like to apply minimum ℓ_p-change to induce misclassification. Since we address binary inputs, we focus our attention on $p = 1$, even though our techniques are also applicable to $1 < p \leq \infty$. The case $p = 1$ simultaneously captures bit flips, where the adversary changes a the sign of an entry, and bit erasures, where the adversary changes an entry to zero. Notice that a bit flip causes ℓ_1-perturbation of 2, and a bit erasure causes ℓ_1-perturbation of 1.

We use one of the standard definitions of robustness of a classifier h at an input $\mathbf{x}$ as the smallest distance of $\mathbf{x}$ to the decision boundary (Diochnos et al., 2018). Formally, the *prediction change robustness* (henceforth, simply *robustness*) of a model h is defined as

$$\mathbb{E}_{\mathbf{x}} \inf \{r : \exists \mathbf{x}' \in \operatorname{Ball}_r^p(\mathbf{x}), h(\mathbf{x}') \neq h(\mathbf{x})\}, \quad (1)$$

where $\operatorname{Ball}_r^p(\mathbf{x})$ is the set of all elements of $\mathbb{R}^n$ that are of ℓ_p-distance at most r from $\mathbf{x}$. Note that in our setting, (1) is equivalent to the ℓ_p-distance from the decision boundary (hyperplane), i.e., $\mathbb{E}_{\mathbf{x}} d_p(\mathbf{x}, \mathcal{H}(\mathbf{w}, \theta))$. A natural goal for robustness is therefore to *maximize* the expected ℓ_p-distance to the decision boundary. This problem will be the focus of *Fourier stabilization of neurons* below.

2.2. Fourier analysis of Boolean functions.

Since subsequent sections rely on notions from Fourier analysis of Boolean functions, we provide a brief introduction. For a thorough treatment of the topic the reader if referred to (O'Donnell, 2014). Let $[n]$ denote the set $\{1,\ldots,n\}$. Every function $f\colon\{\pm 1\}^n\to\mathbb{R}$ can be represented as a linear combination over $\mathbb{R}$ of the functions $\{\chi_{\mathcal{S}}(\mathbf{x})\}_{\mathcal{S}\subseteq[n]}$, where $\chi_{\mathcal{S}}(\mathbf{x})=\prod_{j\in\mathcal{S}}x_j$ for every $\mathcal{S}\subseteq[n]$. The coefficient of $\chi_{\mathcal{S}}(\mathbf{x})$ in this linear combination is called the *Fourier coefficient* of f at $\mathcal{S}$, and it is denoted by $\hat{f}(\mathcal{S})$. Each Fourier coefficient $\hat{f}(\mathcal{S})$ equals the inner product between f and $\chi_{\mathcal{S}}$, defined as $\mathbb{E}_{\mathbf{x}}f(\mathbf{x})\chi_{\mathcal{S}}(\mathbf{x})$, where $\mathbf{x}$ is chosen *uniformly* at random. The inner product between functions f and g equals the inner product (in the usual sense) between their respective Fourier coefficients, a result known as Plancherel's identity: $\mathbb{E}_{\mathbf{x}}f(\mathbf{x})g(\mathbf{x})=\sum_{\mathcal{S}\subseteq[n]}\hat{f}(\mathcal{S})\hat{g}(\mathcal{S})$.

For brevity, we denote $\hat{f}(\{i\})=\hat{f}_i$ for every $i\in[n]$ and $\hat{f}_\varnothing=\hat{f}(\varnothing)$. We also define the vector $\hat{\mathbf{f}}\triangleq(\hat{f}_1,\ldots,\hat{f}_n)$. The entries of $\hat{\mathbf{f}}$, known as *Chow parameters*, play an important role in the analysis of Boolean functions in general, and of sign functions in particular (e.g., (O'Donnell & Servedio, 2011)). We also note that when the range of f is small (e.g. $f:\{\pm 1\}^n\to[-1,1]$, as in sigmoid functions), Hoeffding's inequality implies that any Fourier coefficient $\hat{f}(\mathcal{S})$ can be efficiently approximated by choosing many $\mathbf{x}$'s uniformly at random from $\{\pm 1\}^n$, and averaging the expressions $f(\mathbf{x})\chi_{\mathcal{S}}(\mathbf{x})$. Finally, in the sequel we make use of the following lemma, whose proof is given in (Raviv et al., 2021).

Lemma 1. *For* $h(\mathbf{x})=\operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal-\theta)$ *we have that* $\operatorname{sign}(\hat{h}_i)=\operatorname{sign}(w_i)$ *for every* $i\in[n]$.

3. Increasing Robustness of Individual Neurons

Recall that our goal is to increase robustness, quantified as the expected distance from the decision boundary, of individual neurons. Suppose for now that a neuron is a linear classifier $h(\mathbf{x})=\operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal-\theta)$. Then, by Theorem 1, the distance from the decision boundary for a given input $\mathbf{x}$ is

$$d_p(\mathbf{x},\mathcal{H}(\mathbf{w},\theta))=\frac{|\mathbf{x}\mathbf{w}^\intercal-\theta|}{\|\mathbf{w}\|_q}=\frac{\mathbf{x}\mathbf{w}^\intercal-\theta}{\|\mathbf{w}\|_q}\cdot h(\mathbf{x}). \tag{2}$$

In actuality, we wish to measure this distance with respect to *all* inputs in the input space. We can formalize this as the *average* distance over the input space (which is finite, since inputs are binary), or, equivalently if $\|\mathbf{w}\|_q=1$, as $\mathbb{E}_{\mathbf{x}}(\mathbf{x}\mathbf{w}^\intercal-\theta)\cdot h(\mathbf{x})$, where the expectation is with respect to the uniform distribution over inputs.[1]

Now, suppose that we are given a neuron parametrized by $(\mathbf{w},\theta)$ as input, and we wish to transform it in order to maximize its robustness—that is, average distance to the hyperplane—by choosing new weights and bias, $(\mathbf{v},\mu)$. We can formalize this as the following optimization problem:

The Neuron-Optimization Problem

Input: A neuron $h(\mathbf{x})=\operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal-\theta)$.
Variables: $\mathbf{v}=(v_1,\ldots,v_n)\in\mathbb{R}^n$.
Objective: Maximize $\mathbb{E}_{\mathbf{x}}(\mathbf{x}\mathbf{v}^\intercal-\mu)h(\mathbf{x})$.
Constraints:
- If $p>1$ (including $p=\infty$): $\|\mathbf{v}\|_q^q=1$.
- If $p=1$: $\|\mathbf{v}\|_\infty=1$.

However, an issue arises in finding the optimal bias μ^*: treating μ as an unbounded variable will result in an expression that can be made arbitrarily large by taking μ to either ∞ (if $\sum_{\mathbf{x}}h(\mathbf{x})>0$) or $-\infty$ (otherwise). Therefore, in what follows we treat μ as a constant, and discuss its optimal value with respect to the loss of accuracy in Section 3.2.

We briefly note here a connection to support vector machines (SVMs), which are based on an analogous margin maximization idea. The key distinction is that we aim to maximize margin with respect to the *entire* input space *given a fixed trained model*, whereas SVM maximizes margin with respect to a given dataset in order to train a model. Thus, our approach is about robust generalization rather than training.

3.1. Fourier Stabilization of Neurons

We now derive an analytic solution to the optimization problem above using Fourier analytic techniques. Since we use a uniform distribution over $\mathbf{x}\in\{\pm 1\}^n$, our objective function becomes

$$\mathbb{E}_{\mathbf{x}}(\mathbf{x}\mathbf{v}^\intercal-\mu)h(\mathbf{x})=\hat{\mathbf{h}}\mathbf{v}^\intercal-\hat{h}_\varnothing\mu,$$

by a straightforward application of Plancherel's identity. Therefore, the optimization problem reduces to linear maximization under equality constraints. In what follows, this maximization problem is solved analytically; we emphasize once more that $p=1$ is the focus of our attention, and yet the solution is stated in greater generality for completeness. Fourier stabilization for $p\neq 1$ is potentially useful in niche applications such as neural computation in hardware and adversarial noise in weights. We provide the proof for the case $p=1$, and the remaining cases ($1<p\le\infty$) are discussed in (Raviv et al., 2021).

Theorem 2. *Let* $h(\mathbf{x})=\operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal-\theta)$, *and* $\hat{\mathbf{h}}=(\hat{h}_1,\ldots,\hat{h}_n)$. *The solution* $\mathbf{w}^*=(w_1^*,\ldots,w_n^*)$ *to the*

[1] One may be concerned about the use of a uniform distribution over inputs. However, our experimental evaluation below demonstrates effectiveness for several real datasets. Additionally, we note that in some cases, a simple uniformization mechanism can be applied (see (Raviv et al., 2021)) as part of feature extraction.

neuron-optimization problem is

$$w_i^* = \begin{cases} \operatorname{sign}(\hat{h}_i) \cdot \left(\frac{|\hat{h}_i|}{\|\hat{\mathbf{h}}\|_p}\right)^{p-1} & \text{if } 1 \le p < \infty \\ 0 & \text{if } p = \infty \text{ and } |\hat{h}_i| < \|\hat{\mathbf{h}}\|_\infty \\ |\hat{h}_i| & \text{if } p = \infty \text{ and } |\hat{h}_i| = \|\hat{\mathbf{h}}\|_\infty \end{cases}$$

Further, the maximum value of the objective is $\|\hat{\mathbf{h}}\|_p - \hat{h}_\varnothing \mu$.

Proof for $p = 1$. Notice that the constraint $\|\mathbf{v}\|_\infty = 1$ translates to the n constraints $-1 \le v_i \le 1$, where at least one of which must be attained with equality; this is guaranteed since the optimum of a linear function over a convex polytope is always obtained on the boundary. Hence, the optimization problem reduces to a linear objective function under box constraints. Therefore, to maximize $\hat{\mathbf{h}}\mathbf{v}^\intercal - \hat{h}_\varnothing \mu$, it is readily verified that every v_i must be $\operatorname{sign}(\hat{h}_i)$. The solution in this case is $\mathbf{w}^* = (\operatorname{sign}(\hat{h}_i))_{i=1}^n$, and the resulting objective is $\hat{\mathbf{h}}\mathbf{v}^\intercal - \hat{h}_\varnothing \mu = \|\hat{\mathbf{h}}\|_1 - \hat{h}_\varnothing \mu$. □

We refer to the solution in Theorem 2 as *Fourier stabilization of neurons* (or simply *stabilization*), and the associated neuron as *stabilized*. If we fix $\mu = \theta$ it is easily proved (see (Raviv et al., 2021)) that stabilization increases robustness.

Lemma 2. *For every* $h(\mathbf{x}) = \operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal - \theta)$*, its stabilized counterpart* $h'(\mathbf{x}) = \operatorname{sign}(\mathbf{x}\mathbf{w}^{*\intercal} - \theta)$ *is at least as robust as* $h(\mathbf{x})$*. In particular:*

$$\mathbb{E}_{\mathbf{x}} d_p(\mathbf{x}, \mathcal{H}(\mathbf{w}, \theta)) \le \|\hat{\mathbf{h}}\|_p - \hat{h}_\varnothing \theta \le \mathbb{E}_{\mathbf{x}} d_p(\mathbf{x}, \mathcal{H}(\mathbf{w}^*, \theta)).$$

Notice that thanks to Lemma 1, for $p = 1$ it is not necessary to approximate the Fourier coefficients of h since their sign is given by the sign of the respective entries of $\mathbf{w}$. Notice also that in this case the resulting model is *binarized*, i.e., all its weights are $\{\pm 1\}$. Such models are popular as neurons in *binarized neural networks* (Hubara et al., 2016), and our results shed some light on their apparent increased robustness (Galloway et al., 2017).

Also notice that while our formal analysis pertains to $\operatorname{sign}(\cdot)$, similar reasoning can be applied as a heuristic to many other activation functions, and in particular to sigmoid functions (such as $\operatorname{logistic}(\cdot), \tanh(\cdot)$, etc.). For example, one can replace $\frac{1}{1+e^{-(\mathbf{x}\mathbf{w}^\intercal - \theta)}}$ by $\frac{1}{1+e^{-(\mathbf{x}\mathbf{w}^{*\intercal} - \theta)}}$, where $\mathbf{w}^*$ is the solution of the neuron-optimization problem when applied over $\operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal - \theta)$. Since the outputs of sigmoid functions are very close to ± 1 for most inputs, adversarial attacks attempt to push these inputs towards $\mathcal{H}(\mathbf{w}, \theta)$, a task which is made harder by stabilization. Furthermore, *one-sided* robustness is increased by stabilizing $\operatorname{ReLU}(\mathbf{x}) = \max\{0, \mathbf{x}\mathbf{w}^\intercal - \theta\}$; $\mathbf{x}$'s for which $\mathbf{x}\mathbf{w}^\intercal < \theta$ must be shifted across $\mathcal{H}(\mathbf{w}, \theta)$ for the output of the neuron to change. Hence, stabilizing $\operatorname{ReLU}(\cdot)$, i.e., replacing $\max\{0, \mathbf{x}\mathbf{w}^\intercal - \theta\}$ by $\max\{0, \mathbf{x}\mathbf{w}^{*\intercal} - \theta\}$, increases the robustness of attacking such inputs.

3.2. Bounding the Loss in Accuracy

In the above discussion we optimized for robustness, but were oblivious to the loss of accuracy, and did not specify the bias μ. In this section we again focus on $p = 1$, and the remaining cases are given in (Raviv et al., 2021). We now quantify the accuracy loss of a *single neuron*. Accuracy-loss of a neuron $h(\mathbf{x})$ is quantified in the following sense: we bound the fraction of $\mathbf{x}$'s such that $h(\mathbf{x}) \ne h'(\mathbf{x})$, i.e., they are on the wrong side of the original decision boundary $\mathcal{H}(\mathbf{w}, \theta)$ due to the stabilization. The bound is given as a function of the Fourier coefficients of h, and of the bias μ that can be chosen freely. The choice of μ manifests a robustness-accuracy tradeoff which we discuss subsequently (Corollary 1). Proving the bound requires the following technical lemmas.

Lemma 3. *Let* $\ell(\mathbf{x}) = \sum_{i=1}^n a_i x_i$*, with* $\sum_{i=1}^n a_i^2 = 1$ *and* $|a_i| \le \epsilon$*. If the entries of* $\mathbf{x}$ *are chosen uniformly at random, then there exist a constant* $C_0 \approx 0.47$ *such that for every* $\mu \ge 0$*,*

$$\Pr[|\ell(\mathbf{x}) - \mu| \le u] \le u\sqrt{\frac{2}{\pi}} + 2C_0\epsilon \text{ for every } u > 0.$$

Proof. Notice that

$$\begin{aligned} \Pr[|\ell(\mathbf{x}) - \mu| \le u] &= \Pr[\mu - u \le \ell(\mathbf{x}) \le \mu + u] \\ &\overset{(a)}{\le} \Pr[\mu - u \le N(0,1) \le \mu + u] + 2C_0\epsilon \\ &= \int_{\mu-u}^{\mu+u} \tfrac{1}{\sqrt{2\pi}} e^{-x^2/2} dx + 2C_0\epsilon \\ &\overset{(b)}{\le} u\sqrt{\frac{2}{\pi}} + 2C_0\epsilon, \end{aligned}$$

where (a) follows from The Berry-Esseen Theorem[2], and (b) follows since $e^{-x^2/2} \le 1$. □

Lemma 4. *Let* $Z_1, \ldots, Z_n$ *be independent and uniform* $\{\pm \frac{1}{\sqrt{n}}\}$ *random variables, let* $\mathbf{z} = (Z_1, \ldots, Z_n)$*, and let* $S = \sum_{i=1}^n Z_i$*.*

A. *For every* $\mathbf{a} \in \{\pm 1\}$ *the random variables* S *and* $\mathbf{a}\mathbf{z}^\intercal$ *are identically distributed.*
B. *For every* μ *we have* $\mathbb{E}[|S - \mu|] = \alpha(\mu)$*, where*

$$\alpha(\mu) = \frac{1}{2^n} \cdot \sum_{i \in \{-n, -n+2, \ldots, n\}} \binom{n}{\frac{n-i}{2}} \cdot |i\sqrt{n} - \mu|$$

Proof.

[2] A parametric variant of the central limit theorem.

A. Since each Z_i is uniform over $\{\pm\frac{1}{\sqrt{n}}\}$, it follows that the random variables Z_i and $-Z_i$ are identically distributed for every i, which implies the claim since the Z_i's are independent.
B. Follows by a straightforward computation of the expectation. $\square$

We mention that the proof of the following theorem is strongly inspired by a well-known $p = 2$ counterpart, that appeares repeatedly in the theoretical computer science literature (e.g., (Matulef et al., 2010) (Thm. 26, Thm. 34, Thm. 49), (O'Donnell & Servedio, 2011) (Thm. 8.1), and (O'Donnell, 2014) ($\frac{2}{\pi}$-Thm.), among others).

Theorem 3. *For $h(\mathbf{x}) = \operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal - \theta)$ let $\ell(\mathbf{x}) = \frac{1}{\sqrt{n}} \cdot \mathbf{x}\mathbf{w}^{*\intercal}$, where $\mathbf{w}^*$ is given by Theorem 2, and for any μ let*

$$\gamma = \gamma(\mu) = \left|\tfrac{1}{\sqrt{n}}\|\hat{\mathbf{h}}\|_1 - \hat{h}_\varnothing\mu - \alpha(\mu)\right|, \tag{3}$$

where $\alpha(\mu)$ is defined in Lemma 4. Then,

$$\Pr(\operatorname{sign}(\ell(\mathbf{x}) - \mu) \neq h(\mathbf{x})) \leq \tfrac{3}{2}\left(\tfrac{C_0}{\sqrt{n}} + \sqrt{\tfrac{C_0^2}{n} + \sqrt{\tfrac{2}{\pi}}\cdot\gamma}\right).$$

Proof. According to Plancherel's identity, we have that

$$\mathbb{E}[h(\mathbf{x})(\ell(\mathbf{x}) - \mu)] = \sum_{\mathcal{S}\subseteq[n],\mathcal{S}\neq\varnothing} \hat{h}(\mathcal{S})\hat{\ell}(\mathcal{S}) - \hat{h}_\varnothing\mu$$
$$= \tfrac{1}{\sqrt{n}}\sum_{i=1}^{n} \hat{h}_i \operatorname{sign}(\hat{h}_i) - \hat{h}_\varnothing\mu = \tfrac{1}{\sqrt{n}}\|\hat{\mathbf{h}}\|_1 - \hat{h}_\varnothing\mu. \tag{4}$$

Moreover, since Lemma 4 implies that

$$\mathbb{E}[|\ell(\mathbf{x}) - \mu|] = \alpha(\mu), \tag{5}$$

we have

$$\begin{aligned}&\mathbb{E}[(\ell(\mathbf{x}) - \mu)\cdot(\operatorname{sign}(\ell(\mathbf{x}) - \mu) - h(\mathbf{x}))] = \\ &= \mathbb{E}[|\ell(\mathbf{x}) - \mu|] - \mathbb{E}[h(\mathbf{x})(\ell(\mathbf{x}) - \mu)] \\ &\overset{(4),(5)}{=} \alpha(\mu) - \tfrac{1}{\sqrt{n}}\|\hat{\mathbf{h}}\|_1 + \hat{h}_\varnothing\mu \leq \gamma.\end{aligned} \tag{6}$$

In what follows, we bound $\Pr(\operatorname{sign}(\ell(\mathbf{x}) - \mu) \neq h(\mathbf{x}))$ by studying the expectation in (6). According to Lemma 3 with $\epsilon = \frac{1}{\sqrt{n}}$, it follows that for every $u > 0$ (a precise u will be given shortly)

$$\Pr(|\ell(\mathbf{x}) - \mu| \leq u) < u\sqrt{\tfrac{2}{\pi}} + \tfrac{2C_0}{\sqrt{n}} \triangleq \eta(u). \tag{7}$$

Assume for contradiction that $\Pr(\operatorname{sign}(\ell(\mathbf{x}) - \mu) \neq h(\mathbf{x})) > \frac{3}{2}\eta(u)$. Since $\Pr(|\ell(\mathbf{x}) - \mu| > u) \geq 1 - \eta(u)$ by (7), it follows that

$$\Pr(\operatorname{sign}(\ell(\mathbf{x}) - \mu) \neq h(\mathbf{x}) \text{ and } |\ell(\mathbf{x}) - \mu| > u) > \tfrac{\eta(u)}{2}. \tag{8}$$

Also, observe that

$$\begin{aligned}&\mathbb{E}[(\ell(\mathbf{x}) - \mu)(\operatorname{sign}(\ell(\mathbf{x}) - \mu) - h(\mathbf{x}))] = \\ &\tfrac{1}{2^n}\left(\sum_{\mathbf{x}\mid \operatorname{sign}(\ell(\mathbf{x})-\mu)>h(\mathbf{x})} 2(\ell(\mathbf{x}) - \mu) - \right. \\ &\left.\sum_{\mathbf{x}\mid \operatorname{sign}(\ell(\mathbf{x})-\mu)<h(\mathbf{x})} 2(\ell(\mathbf{x}) - \mu)\right).\end{aligned} \tag{9}$$

Since all summands in left summation in (9) are positive, and all summands in the right one are negative, by keeping in the left summation only summands for which $\ell(\mathbf{x}) - \mu > u$, and in the right summation only those for which $\ell(\mathbf{x}) - \mu < -u$, we get

$$\begin{aligned}(9) &\geq 2u \cdot \frac{\left|\left\{\mathbf{x} \,\middle|\, \begin{array}{c}\operatorname{sign}(\ell(\mathbf{x}) - \mu) \neq h(\mathbf{x}) \\ \text{and } |\ell(\mathbf{x}) - \mu| > u\end{array}\right\}\right|}{2^n} \\ &\overset{(8)}{>} u\cdot\eta(u).\end{aligned} \tag{10}$$

Combining (10) with (6), it follows that $u\cdot\eta(u) < \gamma$, which by the definition in (7) implies that

$$\sqrt{\tfrac{2}{\pi}}\cdot u^2 + \tfrac{2C_0}{\sqrt{n}}\cdot u - \gamma < 0. \tag{11}$$

We wish to find the smallest positive value of u which contradicts (11). By applying the textbook solution, we have that any positive u which complies with (11) must satisfy

$$u < \frac{-\frac{C_0}{\sqrt{n}} + \sqrt{\frac{C_0^2}{n} + \sqrt{\frac{2}{\pi}}\cdot\gamma}}{\sqrt{\frac{2}{\pi}}} \tag{12}$$

and hence setting u to the right hand side of (12) leads to a contradiction. Therefore,

$$\begin{aligned}\Pr(\operatorname{sign}(\ell(\mathbf{x}) - \mu) \neq h(\mathbf{x})) &\leq \tfrac{3}{2}\eta(u) \overset{(7)}{=} \tfrac{3}{2}(u\sqrt{\tfrac{2}{\pi}} + \tfrac{2C_0}{\sqrt{n}}) \\ &= \tfrac{3}{2}\left(\tfrac{C_0}{\sqrt{n}} + \sqrt{\tfrac{C_0^2}{n} + \sqrt{\tfrac{2}{\pi}}\cdot\gamma}\right).\end{aligned}$$
$\square$

Corollary 1. *Theorem 3 complements Theorem 2 in terms of the robustness-accuracy tradeoff in choosing the bias μ of the stabilized neuron. Given $h(\mathbf{x}) = \operatorname{sign}(\mathbf{x}\mathbf{w}^\intercal - \theta)$, choosing $\mu = \theta$ guarantees increased robustness of the stabilized model $h'(\mathbf{x}) = \operatorname{sign}(\mathbf{x}\mathbf{w}^{*\intercal} - \mu)$ by Lemma 2, and the accuracy loss is quantified by setting[3] $\mu = \theta$ Theorem 3. However, one is free to choose any other $\mu \neq \theta$, and obtain different accuracy and robustness. For any such μ, the robustness of the stabilized neuron is*

$$\mathbb{E}_{\mathbf{x}} d_p(\mathbf{x}, \mathcal{H}(\mathbf{w}^*, \mu)) = \sum_{i=1}^{n} w_i^*\hat{h}_i' - \hat{h}_\varnothing'\mu$$

[3] More precisely, setting $\mu = \theta/\sqrt{n}$, due to the additional normalization factor in Theorem 3.

by Plancherel's identity, and the resulting accuracy loss is given similarly by Theorem 3. In any case, the resulting accuracy and robustness should be contrasted with those of the non-stabilized model, where the accuracy loss is obviously zero, and the robustness is

$$\mathbb{E}_{\mathbf{x}} d_p(\mathbf{x}, \mathcal{H}(\mathbf{w}, \theta)) = \sum_{i=1}^{n} w_i \hat{h}_i - \hat{h}_{\varnothing}\theta.$$

4. Fourier Stabilization of Deep Neural Networks

Thus far, we were primarily focused on robustness and accuracy of individual neurons, modeled as linear classifiers. We now consider the problem of increasing robustness of neural networks, comprised of a collection of such neurons. The general idea is that by stabilizing individual neurons in the network we can increase the overall robustness. However, increased robustness comes almost inevitably at some loss in accuracy, and different neurons in a network will face a somewhat different robustness-accuracy tradeoff. Consequently, we will now consider the problem of stabilizing a neural network by selecting a subset of neurons to stabilize that best trades off robustness and accuracy.

To formalize this idea, let $\mathcal{S}$ denote the subset of neurons that are chosen for stabilization. Define $R(\mathcal{S})$ as robustness (for example, measured empirically on a dataset using any of the standard measures) and let $A(\mathcal{S})$ be the accuracy (again, measured empirically on unperturbed data) after we stabilize the neurons in set $\mathcal{S}$. Our goal is to maximize robustness subject to a constraint that accuracy is no lower than a predefined lower bound β:

The Network-Optimization Problem

Input: A neural network N with first-layer neurons $\{h_i(\mathbf{x}) = \mathrm{sign}(\mathbf{x}\mathbf{w}_i^\intercal - \theta_i)\}_{i=1}^t$, and accuracy bound β.
Variable: $\mathcal{S} \subseteq \{1, \ldots, t\}$.
Objective: Maximize $R(\mathcal{S})$
Constraint: $A(\mathcal{S}) \geq \beta$.

Observe that while in principle we can stabilize any subset of neurons, the tools we developed in Section 3.1 apply only to neurons with binary inputs, which is, in general, only true of the neurons in the first (hidden) layer of the neural network. Consequently, both the formulation above, and experiments below, focus on stabilizing a subset of the first-layer neurons.

There are two principal challenges in solving the optimization problem above. First, it is a combinatorial optimization problem in which neither $R(\mathcal{S})$ nor $A(\mathcal{S})$ are guaranteed to have any particular structure (e.g., they are not even necessarily monotone). Second, using empirical robustness $R(\mathcal{S})$ is typically impractical, as computing ℓ_1 adversarial perturbations on binary inputs is itself a difficult combinatorial optimization problem for which even heuristic solutions are slow (Papernot et al., 2016).

To address the first issue, we propose two algorithms. The first is *Greedy Marginal Benefit per Unit Cost (GMBC)* algorithm. Define $\Delta A(j|\mathcal{S}) = A(\mathcal{S}) - A(\mathcal{S} \cup \{j\})$ for any set of stabilized neurons $\mathcal{S}$; this is the marginal decrease in accuracy from stabilizing a neuron j in addition to those in $\mathcal{S}$. Similarly, define $\Delta R(j|\mathcal{S}) = R(\mathcal{S} \cup \{j\}) - R(\mathcal{S})$, the marginal increase in robustness from stabilizing j. We can greedily choose neurons to stabilize in decreasing order of $\frac{\Delta R(j|\mathcal{S})}{\Delta A(j|\mathcal{S})}$, until the accuracy "budget" is saturated (that is, as long as accuracy stays above the bound β). A second alternative algorithm we propose is *Greedy Marginal Benefit (GMB)*, which stabilizes neurons solely in the order of $\Delta R(j|\mathcal{S})$. If $A(S)$ is monotone decreasing in the number of neurons, we can show that *GMB* requires only a logarithmic number of accuracy evaluations (see Appendix E of Raviv et al. (2021)). In practice, we can also run both in parallel and choose the better solution of the two; indeed, if $R(\mathcal{S})$ and $A(\mathcal{S})$ are both monotone and submodular, with $A(\mathcal{S})$ having some additional structure, the resulting algorithm exhibits a known approximation guarantee (Zhang & Vorobeychik, 2016). However, we must be careful since in fact $A(\mathcal{S})$ is not necessarily monotone, and consequently $\Delta A(j|\mathcal{S})$ can be negative. To address this, we maintain a positive lower bound $\bar{a}$ on this quantity, and if $\Delta A(j|\mathcal{S}) < \bar{a}$ (including if it is negative), we simply set it to $\bar{a}$.

To address the second issue, we propose using an analytic proxy for $R(\mathcal{S})$, defining it as the sum of the increase in robustness from stabilizing the individual neurons in $\mathcal{S}$ (see Section 3.1).

5. Experiments

Datasets and Computing Infrastructure We evaluated the proposed approach using three security-related datasets: *PDFRate*, *Hidost*, and *Hate Speech*. The *PDFRate* dataset (Smutz & Stavrou, 2012) is a PDF malware dataset which extracts features based on PDF file metadata and content. The metadata features include the size of a file, author name, and creation date, while content-based features include position and counts of specific keywords. This dataset includes 135 total features, which are then binarized if not already binary. The *Hidost* dataset (Šrndić & Laskov, 2016) is a PDF malware dataset which extracts features based on the logical structure of a PDF document. Specifically, each binary feature corresponds to the presence of a particular *structural path*, which is a sequence of edges in the reduced (tree) *logical structure*, starting from the catalog dictionary and ending at this object (i.e., the shortest reference path to

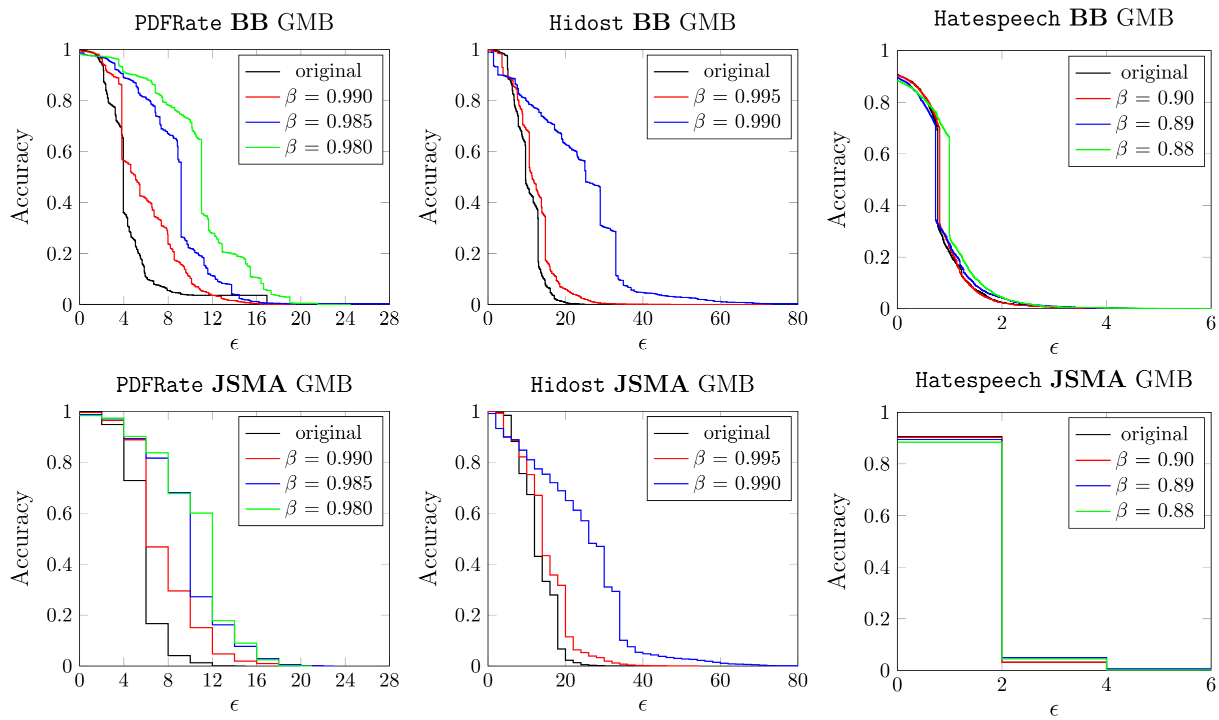

Figure 1: Robustness of original and stabilized neural network models (using *GMB*) on PDFRate, Hidost, and Hate Speech datasets (columns) against the BB (top row) and JSMA (bottom row) attacks. The x-axis shows varying levels of ℓ_1 perturbation bound ϵ for the attacks.

a PDF object). This dataset is comprised of 658,763 PDF files and 961 features.

The *Hate Speech* dataset (Qian et al., 2019), collected from Gab, is comprised of conversation segments, with hate speech labels collected from Amazon Mechanical Turk workers. This dataset contains 33,776 posts, and we used a bag-of-words binary representation with 200 most commonly occurring words (not including stop words).

All datasets were divided into training, validation, and test subsets; the former two were used for training and parameter tuning, while all the results below are using the test data. We also used the validation set to select the subset of neurons $\mathcal{S}$ to be stabilized. For each dataset, we learned a two-layer sigmoidal fully connected neural network as a baseline. Experiments were run on a research computer cluster with over 2,500 CPUs and 60 GPUs, and the code is available online at https://github.com/AidanKelley/fourier-stabilization.

Attacks The robustness-accuracy tradeoff is quantified by the success rate of two state-of-the-art attacks, JSMA and ℓ_1-BB, under limited budget. *Jacobian-based Saliency Map Attack* (JSMA) (Papernot et al., 2016) (naturally adapted to the $\{\pm 1\}$ domain rather than $\{0, 1\}$), employs a greedy heuristic by which the bit with the highest impact is flipped. ℓ_1 *Brendel & Bethge* (ℓ_1-BB) (Brendel et al., 2019) is an attack that allows non-binary perturbations. It is radically different from JSMA in the sense that it requires an already-adversarial starting point which is then optimized. Given a clean point to attack, we select the adversarial starting point as the closest to it in ℓ_1-distance, among all points in the training set.

Adversarial Training In addition to the conventional baseline above, we also evaluated the use of neural network stabilization after adversarial training (*AT*) (Vorobeychik & Kantarcioglu, 2018), which is still a state-of-the-art general-purpose approach for defense against adversarial example attacks. We performed AT with the JSMA attack (ℓ_1-norm $\epsilon = 20$), which we adapted as follows: instead of minimizing the number of perturbed features to cause misclassification, we maximize loss subject to a constraint that we change at most ϵ features, still choosing which features to flip in the sorted order produced by JSMA.

5.1. Effectiveness of Neural Network Stabilization

We first evaluate the proposed Fourier stabilization approach for neural network models on neural networks trained in a regular way on the *PDFRate*, *Hidost*, and *Hate Speech* datasets. The results are shown in Figure 1 for the *GMB*

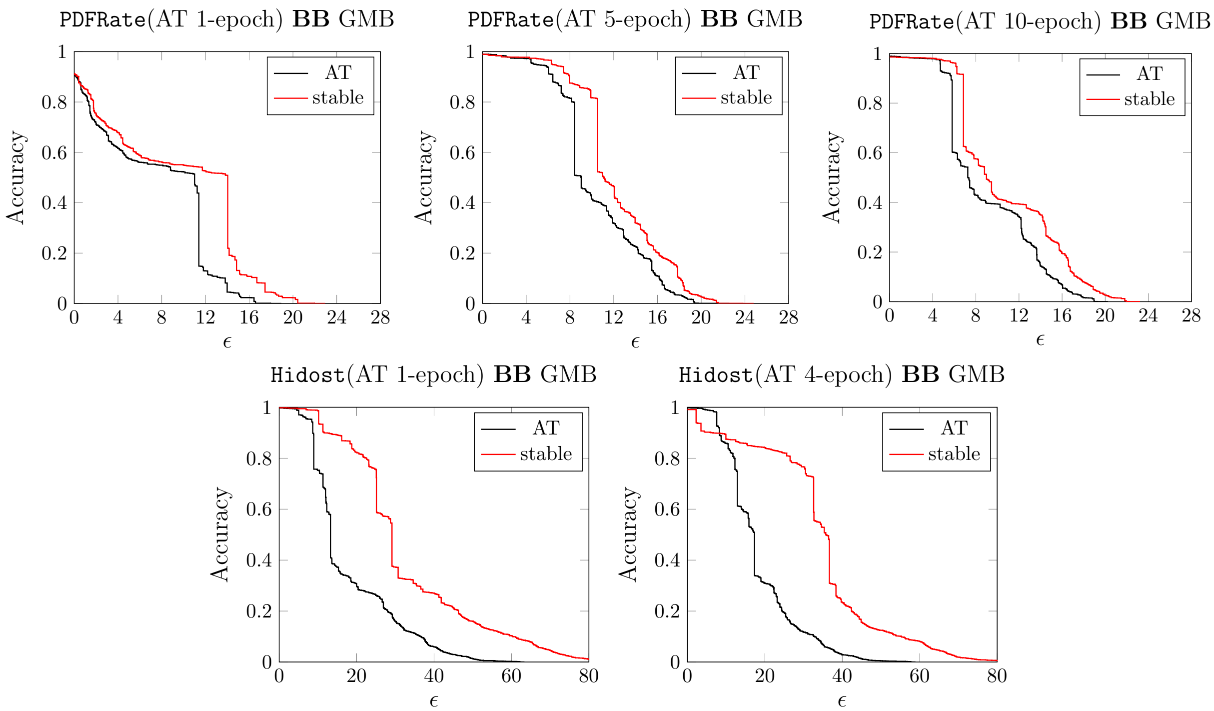

Figure 2: Robustness of adversarially trained neural networks and their stabilized variants (using *GMB*). Top row: PDFRate dataset, after 1, 5, and 10 epochs of adversarial training (from left to right). Bottom row: Hidost dataset after 1 (left) and 4 (right) epochs of adversarial training. The x-axis shows varying levels of ℓ_1 perturbation bound ϵ for the attacks.

algorithm, where the top three plots (one for each dataset) are for the BB attack, and the bottom three are for the JSMA attack; results for *GMBC* are provided in the supplement. The most significant impact on robustness is in the case of the PDFRate dataset, where an essentially negligible drop in accuracy is accompanied by a substantial increase in robustness. For example, for BB attack ℓ_1 perturbation of at most $\epsilon = 10$ (the x-axis), robust accuracy (y-axis) increases from nearly 0 to 70%, while clean data accuracy is 0.98. We can observe a similar impact for the JSMA attack, with robust accuracy increasing from 0 to 60%. Fourier stabilization has a similarly substantial impact on the Hidost data: with accuracy still at 99%, robust accuracy is increased from nearly 0 to 60% for both the BB and JSMA attacks. On the other hand, the impact is markedly small on the Hate Speech data, although even here we see an increase in robust accuracy for BB attacks on the stabilized version for $\beta = 0.88$ and $\epsilon = 1$ from 30% (baseline) to nearly 70% (Fourier stabilization).

5.2. Stabilizing Adversarially Trained Models

In addition to demonstrating the value of stabilization for regularly trained neural networks (for example, when adversarial training is not an option, such as when datasets on which the original model was trained are sensitive), we now show that the approach also effectively composes with adversarial training (AT). Figure 2 presents the results of stabilization (using *GMB*; see the supplement for *GMBC*) performed after several epochs of AT. In all cases we see some improvement, and in a number of them the improvement over AT is considerable. For example, on the Hidost dataset after 4 epochs of AT, robust accuracy is considerably improved by AT compared to the original model in Figure 1, but then further improved significantly by the proposed stabilization approach. For example, for $\epsilon = 24$, robust accuracy increases from approximately 20% to 80%.

6. Discussion

We introduced Fourier stabilization, a harmonic-analysis inspired post-training defense against adversarial perturbations of randomly chosen binary inputs. It is natural to consider extensions of this work in several fronts, e.g., worst-case robustness, non-uniform binary inputs, and real-valued inputs. In worst-case robustness, correct computation is required for *every* input, i.e., $\mathbb{E}_{\mathbf{x}}$ in (1) is replaced by $\min_{\mathbf{x}}$. While average-case robustness is more suited for applications such as malware detection, worst-case robustness is relevant in critical applications such as neuromorphic computing. It was recently shown in Raviv et al. (2020) that worst-case robustness is impossible even against one bit erasure (i.e., setting $x_i = 0$ for some i), unless redundancy is added, and a simple methods of adding such redundancy was given.

Extensions for non-uniform-binary or real-valued inputs require developing new tools in harmonic analysis. In the binary case, one needs to study the coefficients which come up instead of the Fourier ones, and if Plancherel's identity holds. In the real-valued case, e.g., when the inputs are distributed by a multivariate Gaussian, *Hermite coefficients* can be used similarly, see (O'Donnell, 2014), Sec. 11.2. However, in this case every neuron is already stabilized (see (Matulef et al., 2010), Prop. 25.2), and hence we suggest to consider other input distributions that are common in the literature, such as Gaussian mixture, and study the resulting coefficients.

Acknowledgements

Vorobeychik was supported in part by the National Science Foundation grants no. IIS-1905558 and ECCS-2020289, Army Research Office grant no. W911NF1910241, and NVIDIA.

References

Brendel, W., Rauber, J., Kümmerer, M., Ustyuzhaninov, I., and Bethge, M. Accurate, reliable and fast robustness evaluation. In *Neural Information Processing Systems*, pp. 12861–12871, 2019.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 2019.

Diochnos, D., Mahloujifar, S., and Mahmoody, M. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In *Neural Information Processing Systems*, pp. 10359–10368, 2018.

Galloway, A., Taylor, G. W., and Moussa, M. Attacking binarized neural networks. *arXiv preprint arXiv:1711.00449*, 2017.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Grosse, K., Papernot, N., Manoharan, P., Backes, M., and McDaniel, P. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*, 2016.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *Neural Information Processing Systems*, pp. 4107–4115, 2016.

Laskov, P. et al. Practical evasion of a learning-based classifier: A case study. In *IEEE Symposium on Security and Privacy*, pp. 197–211, 2014.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy*, 2019.

Li, B. and Vorobeychik, Y. Evasion-robust classification on binary domains. *ACM Transactions on Knowledge Discovery from Data*, 12(4):1–32, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Matulef, K., O'Donnell, R., Rubinfeld, R., and Servedio, R. A. Testing halfspaces. *SIAM Journal on Computing*, 39(5):2004–2047, 2010.

Melachrinoudis, E. An analytical solution to the minimumlp-norm of a hyperplane. *Journal of Mathematical Analysis and Applications*, 211(1):172–189, 1997.

O'Donnell, R. *Analysis of boolean functions*. Cambridge University Press, 2014.

O'Donnell, R. and Servedio, R. A. The chow parameters problem. *SIAM J. on Computing*, 40(1):165–199, 2011.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy*, pp. 372–387, 2016.

Pinelis, I. On the nonuniform berry–esseen bound. In *Inequalities and Extremal Problems in Probability and Statistics*, pp. 103–138. Elsevier, 2017.

Qian, J., Bethke, A., Liu, Y., Belding, E., and Wang, W. Y. A benchmark dataset for learning to intervene in online hate speech. In *Conference on Empirical Methods in Natural Language Processing*, pp. 4755–4764, 2019.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

Raviv, N., Jain, S., Upadhyaya, P., Bruck, J., and Jiang, A. A. Codnn–robust neural networks from coded classification. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 2688–2693. IEEE, 2020.

Raviv, N., Kelley, A., Guo, M., and Vorobeychik, Y. Enhancing robustness of neural networks through Fourier stabilization. *arXiv preprint arXiv:2106.04435 [cs.LG]*, 2021.

Shevtsova, I. On the absolute constants in nagaev–bikelis-type inequalities. In *Inequalities and Extremal Problems in Probability and Statistics*, pp. 47–102. Elsevier, 2017.

Smutz, C. and Stavrou, A. Malicious pdf detection using metadata and structural features. In *Annual Computer Security Applications Conference*, pp. 239–248, 2012.

Šrndić, N. and Laskov, P. Hidost: a static machine-learning-based detector of malicious files. *EURASIP Journal on Information Security*, 2016(1):22, 2016.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Tong, L., Li, B., Hajaj, C., Xiao, C., Zhang, N., and Vorobeychik, Y. Improving robustness of ml classifiers against realizable evasion attacks using conserved features. In *USENIX Security Symposium*, pp. 285–302, 2019.

Vorobeychik, Y. and Kantarcioglu, M. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 2018.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *Neural Information Processing Systems*, 2018.

Wu, T., Tong, L., and Vorobeychik, Y. Defending against physically realizable attacks on image classification. In *International Conference on Learning Representations*, 2020.

Xu, W., Qi, Y., and Evans, D. Automatically evading classifiers. In *Network and Distributed Systems Security Symposium*, 2016.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Network and Distributed System Security Symposium*, 2018.

Zhang, H. and Vorobeychik, Y. Submodular optimization with routing constraints. In *AAAI Conference on Artificial Intelligence*, 2016.