Efficient and Robust Classification for Sparse Attacks

Mark Beliaev, Payam Delgosha, Hamed Hassani, and Ramtin Pedarsani, *Senior Member, IEEE*

***Abstract*—Over the past two decades, the rise in adoption of neural networks has surged in parallel with their performance. Concurrently, we have observed the inherent fragility of these prediction models: small changes to the inputs can induce classification errors across entire datasets. In the following study, we examine perturbations constrained by the ℓ_0–norm, a potent attack model in the domains of computer vision, malware detection, and natural language processing. To combat this adversary, we introduce a novel defense technique comprised of two components: "truncation" and "adversarial training". Subsequently, we conduct a theoretical analysis of the Gaussian mixture setting and establish the asymptotic optimality of our proposed defense. Based on this obtained insight, we broaden the application of our technique to neural networks. Lastly, we empirically validate our results in the domain of computer vision, demonstrating substantial enhancements in the robust classification error of neural networks.**

***Index Terms*—Robust classification, sparse attack, adversarial training, neural networks.**

I. Introduction

PRESENTLY, machine learning tackles an array of applications with significant safety implications, such as computer vision, autonomous vehicle navigation, and virtual assistance. This trend aligns with the rise in popularity of deep neural networks, which have demonstrated nearly human-equivalent proficiency in the realm of image recognition [2], alongside notable achievements in game playing [3], [4], and natural language processing [5]. However, the unexpected frailty of these neural networks when exposed to adversarial attacks presents a paradoxical contrast to their otherwise impressive performance.

Manuscript received 20 October 2023; revised 23 January 2024 and 17 April 2024; accepted 1 May 2024. Date of publication 6 May 2024; date of current version 22 May 2024. This work was supported by the National Science Foundation under Award 2342253, Award 2236483, Award 1943064, and Award 2236484. A preliminary version of this work was presented at the 2022 IEEE-ISIT at Aalto University in Espoo, Finland 1. *(Corresponding author: Mark Beliaev.)*

Mark Beliaev is with the Graduate School of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: mbeliaev@ucsb.edu).

Payam Delgosha is with the Faculty of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL 61801 USA (e-mail: delgosha@illinois.edu).

Hamed Hassani is with the Faculty of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: hassani@seas.upenn.edu).

Ramtin Pedarsani is with the Faculty of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: ramtin@ucsb.edu).

Digital Object Identifier 10.1109/JSAIT.2024.3397187

Adversarial attacks represent techniques designed to deceive predictive models, corrupting their inputs with small perturbations. The potency of such attacks in causing prediction errors across diverse machine learning models was initially exhibited in various studies [6], [7], [8]. Subsequent to these findings, significant effort has been expended to formulate progressively intricate adversaries that can leverage a minimal quantity of semantic-retaining alterations while maintaining the capacity to mislead a classifier [9], [10], [11], [12]. In the realm of image classification, this is typically accomplished by constraining the perturbations using an ℓ_p–norm, with the majority of applications employing either ℓ_∞ [7], [9], [10], [13], [14], [15], [16], ℓ_2 [10], [17], [18], [19], [20], or ℓ_1 [21], [22]. As of now, the most empirically effective defense against such attacks involves iterative retraining with adaptively generated adversarial examples [9]. Despite adversarial training's potential to enhance robustness, there exists a fundamental tradeoff between clean accuracy and robustness, in addition to a lack of generalization across diverse attacks [23], [24], [25], [26], [27], [28], [29].

This study is primarily concerned with adversaries constrained using the ℓ_0–norm, a scenario that has captured substantial interest [10], [22], [30], [31], [32], [33] due to its relevance in NLP [34] and object detection [35], [36]. Within these settings, robust guarantees against ℓ_0–attacks are particularly vital due to the implied limit on the quantity of input features allowed to be altered. In the aforementioned settings, the adversary could alter all input elements while complying with the specified constraint. In contrast, when using the ℓ_0–norm, the adversary has a fixed budget k which restricts them to modifying a maximum of k coordinates within the input. Put simply, the adversary can perturb the input within the ℓ_0–ball of radius k, where k is typically small compared to the input dimension, hence the term *sparse attacks*. Additionally, in contrast to ℓ_p–balls ($p \geq 1$), the ℓ_0–ball exhibits a complex geometry: it is unbounded, highly non-smooth, and non-convex.

Concurrently, the intrinsic discrete structure of the ℓ_0–ball introduces fundamental challenges missing in other adversarial settings explored in existing literature, rendering many techniques from previous studies non-applicable. Importantly, piece-wise linear classifiers, such as neural networks with ReLU activations, have been demonstrated to fail in this context [37], and empirical studies have shown the efficacy of ℓ_0–attacks on images [10], [11], [30], [31], [38]. Therefore, it becomes imperative to reassess our current architectural

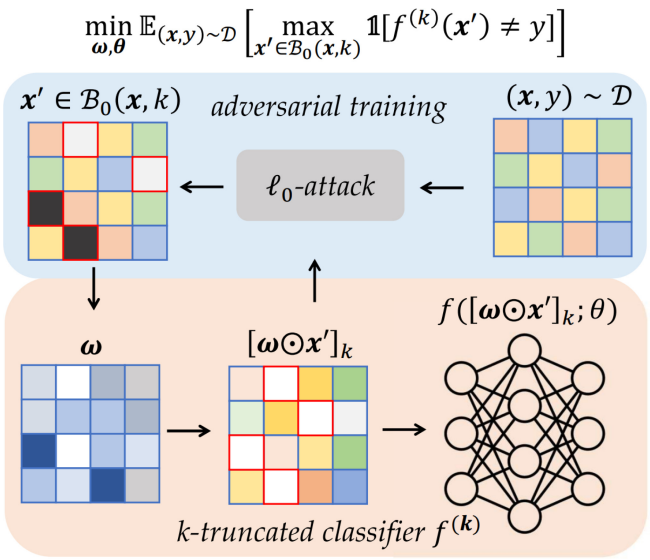

Fig. 1. We illustrate the two key components of our framework: truncation and adversarial training. The bottom diagram depicts truncation - given an input $\vec{x}$ and a weight $\vec{w}$, we *truncate* the input by deleting the top and bottom k coordinates of the element-wise product between $\vec{w}$ and $\vec{x}$. Given any classifier $f(\vec{x};\vec{\theta}) = y$ parameterized by $\vec{\theta}$, we can replace its input $\vec{x}$ with this k–truncated counterpart $[\vec{w} \odot \vec{x}]_k$ forming the k–truncated classifier $f^{(k)}$. The top diagram depicts adversarial training - using an ℓ_0–attack append adversarial examples $\vec{x}'$ to the training data. This procedure is used as a proxy for deriving the weights $\vec{w}$ and $\vec{\theta}$ that solve the minimax problem of finding a classifier $f^{(k)}$ that attains the minimum robust classification error.

designs and learning methodologies in light of the unique geometry of the ℓ_0–norm.

Building upon our previous work [39], we design an algorithm that directly addresses the ℓ_0–adversary and demonstrate that in the Gaussian mixture scenario, asymptotic optimality can be attained. Inspired by these theoretical insights, we extend our methodology to the realm of image classification. Our proposed framework, depicted in Figure 1, can be used in conjunction with any neural network classifier to improve its robustness against the ℓ_0–adversary. Leveraging `sparse-rs` [32], a state-of-the-art sparse attack, in addition to the widely employed `Pointwise Attack` [31], our findings indicate that while standalone adversarial training may prove insufficient in bolstering against ℓ_0–attacks, our approach exhibits substantial performance with regard to robustness and computational efficiency when evaluated using the MNIST [40] and CIFAR [41] datasets.

Our main contributions are outlined below:

- We propose a novel defense algorithm against ℓ_0–bounded attacks that is based on two key components: truncation and adversarial training. Essentially, we leverage truncation to mitigate adversarial perturbations, and combine it with adversarial training to identify the best classifier in the set of truncated classifiers.
- We formalize the efficacy of our proposed method in the theoretical setting of binary Gaussian mixtures. More specifically, we prove that the optimal truncated classifier is near optimal in the class of all classifiers, and is indeed asymptotically optimal as the input dimension grows large.
- We provide extensive experiments that demonstrate the strength and efficiency of our proposed defense on the MNIST [40] and CIFAR [41] datasets. To the best of our knowledge, our proposed algorithm is the first to provide meaningful empirical results on the CIFAR dataset.

Before continuing, we outline the organization of this paper. First, Section II reviews two related works which also consider defense methods for classifiers in the ℓ_0–setting. Following this, Section III sets up our problem by introducing relevant notation and describing the robust classification setting. The proposed algorithm is generalized in Section IV, where the theoretical results specific to the Gaussian mixture setting are stated after in Section V. Finally, Section VI presents our numerical results derived by performing experiments using the MNIST [40] and CIFAR [41] datasets. We conclude the work by discussing our findings in Section VII, and provide an Appendix for proofs.[1]

II. Related Work

Aside from the general framework of adversarial training, which will be detailed in Section IV, there are two particularly significant studies that have proposed methodologies for defending against sparse attacks: *Analysis by Synthesis* (ABS) [31] and randomized ablation [33]. In the following, we describe these methods in brief and subsequently compare them to our proposed defense strategy.

Relying on optimization-based inference, the ABS model employs variational autoencoders (VAEs) to learn a generative distribution for a given input sample under each class. These generative distributions allow one to estimate a lower bound on the log-likelihood of a given sample belonging to a particular class. The method used for computing this lower bound is computationally expensive, requiring one to repeat 50 steps of gradient descent 1000 times for a single prediction. Although the authors show promising results on the MNIST dataset, they state that their implementation of the ABS model with one VAE per class neither scales efficiently to more classes nor to more complex datasets such as CIFAR-10.

Defenses based on randomized ablation take a different approach by adapting the technique of randomized smoothing to the ℓ_0–setting. Unlike the ℓ_1 and ℓ_2 settings where additive noise is used, the authors propose a certifiably robust defense against sparse adversarial attacks that relies on randomly ablating input features. Specifically, their method trains a smoothed classifier that takes as input $10,000$ randomly ablated versions of the original input, each with a small subset of unchanged coordinates. After this, a majority vote is taken over all $10,000$ ablated samples to determine the most probable class. Although this smoothed classifier is certifiably robust to ℓ_0-attacks of a certain magnitude, its reliance on large numbers of ablated samples is computationally impractical for complex datasets like CIFAR-10.

In contrast to both of the aforementioned defense methods, truncation provides a more efficient solution for defending against sparse attacks, and scales well to more complicated

[1]The code and implementation details used for our experiments can be found at https://github.com/mbeliaev1/robust-l0 [42].

datasets like CIFAR-10. The added computational complexity of incorporating truncation into a classifier is equivalent to adding a single layer in a deep neural network. We also note that our method is theoretically motivated by the pure ℓ_0–threat model, while both ABS and randomized ablation are designed to address the weaker $\ell_0+\ell_\infty$–adversary. Similar to the ℓ_0–setting, this adversary is still limited in the number of coordinates it can alter, but these alterations can no longer be arbitrarily large due to the additional boundary set by the ℓ_∞–norm. We realize that having this bound is a practical choice in the image domain, and hence test our algorithm under the same conditions. However, we note that when the ℓ_∞–norm is relaxed, our method can generalize its performance unlike the prior defense methods.

III. Problem Setup

In this study, we examine the M–class classification problem, where we strive to build a model that accurately predicts the label $y \in \{1, \ldots, M\}$ from the corresponding input $\vec{x} \in \mathbb{R}^d$. We can conceptualize the input and labels as samples generated by the distribution $(\vec{x}, y) \sim \mathcal{D}$, and the classifier as a member of the function family $\mathcal{F} : \mathbb{R}^d \to \{1, \ldots, M\}$. For a classifier $\mathcal{C}$, we employ the $0-1$ loss $\ell(\mathcal{C}; \vec{x}, y) = \mathbb{1}[\mathcal{C}(\vec{x}) \neq y]$ as a measure of the discrepancy between the label and the classifier's prediction for a specific input $\vec{x}$.

With this configuration, we introduce the ℓ_0–adversary, which perturbs the input $\vec{x}$ within the ℓ_0–ball of radius k:

$$\mathcal{B}_0(\vec{x}, k) := \{\vec{x}' \in \mathbb{R}^d : \left\|\vec{x} - \vec{x}'\right\|_0 \leq k\}, \tag{1}$$

where we define $\|\vec{x}\|_0 := \sum_{i=1}^d \mathbb{1}[x_i \neq 0]$ for $\vec{x} = (x_1, \ldots, x_d)$, and refer to k as the *budget* of the adversary. This states that the adversary is permitted to arbitrarily modify a maximum of k coordinates of $\vec{x}$ to generate $\vec{x}'$, which is then inputted into the classifier. With this setup, the *robust classification error* of a classifier $\mathcal{C}$ is defined by:

$$\mathcal{L}_{\mathcal{D}}(\mathcal{C}, k) = \mathbb{E}_{(\vec{x},y)\ \sim\mathcal{D}}\left[\max_{\vec{x}' \in \mathcal{B}_0(\vec{x},k)} \ell\big(\mathcal{C}; \vec{x}', y\big)\ \right], \tag{2}$$

where our goal is to design classifiers which attain the minimum *robust classification error*. With this objective in mind, we define the *optimal robust classification error* as the result of minimizing (2) over all possible classifiers:

$$\mathcal{L}^*_{\mathcal{D}}(k) := \inf_{\mathcal{C}} \mathcal{L}_{\mathcal{D}}(\mathcal{C}, k). \tag{3}$$

The intricate geometry of the ℓ_0–ball makes this a challenging problem. Indeed, we have already observed that all piecewise linear classifiers fail in this context [37]. To address this issue, the present architectural designs and learning procedures must be reevaluated, taking into account the geometry of the perturbation set. In this regard, it is important to mention that directly solving the optimization problem in (2) and determining the optimal robust error is intractable. Instead, drawing inspiration from robust statistics [43], we introduce truncation as the primary component of our classifier. Our objective is then to identify the most robust classifier within the set of truncated classifiers. This optimization can be analyzed in the Gaussian mixture scenario, and addressed through adversarial training in the broader deep learning context. As we demonstrate in Section V, the theoretical investigation of the Gaussian mixture model enables us to establish the optimality of our approach.

IV. The Proposed Algorithm

In this section we describe our proposed algorithm, beginning with the definition of *truncation*, followed by its extension to neural networks. Subsequently, we detail how adversarial training is used in our framework. Foreshadowing our theoretical and empirical results outlined in Section V and Section VI respectively, the coupling of truncation and adversarial training is integral to constructing robust classifiers resistant to ℓ_0–attacks

A. Truncation

We define truncation as an operation that acts on two vectors by computing their truncated inner product. Given vectors $\vec{w}, \vec{x} \in \mathbb{R}^d$ and an integer $0 \leq k \leq d/2$, we define the *k–truncated inner product* of $\vec{w}$ and $\vec{x}$ as the summation of the element-wise product between $\vec{w}$ and $\vec{x}$ after deleting the top and bottom k elements, denoting it by $\langle \vec{w}, \vec{x} \rangle_k$. Defining the element-wise product of $\vec{w}$ and $\vec{x}$ as $\vec{u} := \vec{w} \odot \vec{x} \in \mathbb{R}^d$, we let $u_{s(1)} \leq \ldots \leq u_{s(d)}$ represent the sorted elements of $\vec{u}$ under permutation s. This allows us to denote the k–truncated inner product by:

$$\langle \vec{w}, \vec{x} \rangle_k := \sum_{i=k+1}^{d-k} u_{s(i)}. \tag{4}$$

It is worth mentioning that when $k = 0$, the truncation operation in (4) simplifies to the standard inner product, represented by $\langle \vec{w}, \vec{x} \rangle$.

Truncation naturally facilitates the removal of "outliers" in the data, which may have been manipulated by an adversary affecting specific coordinates. Given that an ℓ_0–adversary with a budget of k can arbitrarily alter a maximum of k coordinates of the input, we anticipate the k–truncated inner product to exhibit resilience against this ℓ_0–adversary. We formalize this concept in Section V, demonstrating that truncation can be directly employed to construct the optimally robust classifier in the context of Gaussian mixture models subjected to an ℓ_0–adversary. In the meantime, our discussion will focus on how we use truncation to design robust neural networks.

B. Robust Neural Networks

Our goal is to design robust classifiers that deal with high dimensional and complex data domains, such as images. In the Gaussian mixture setting, the optimal Bayes classifier is known to be the linear classifier $\text{sgn}(\langle \vec{w}, \vec{x} \rangle)$,[2] and our theoretical results show that in the presence of an ℓ_0–adversary, the optimal classifier takes the form $\text{sgn}(\langle \vec{w}, \vec{x}' \rangle_k)$. We use this intuition to extend truncation for the general classification task where neural networks are typically employed.

Recall that we defined truncation in Eq. (4) as a composition of three operations: (1) an element wise product between

[2] see, for instance, [44, Section 9.2.2].

the input $\vec{x}$ and weight vector $\vec{w}$ (2) the removal of the top k and bottom k elements of this product, and (3) the summation of the remaining $d-2k$ elements. Since neural networks already contain layers that sum weighted subsets of their inputs, the third operation of truncation is redundant in this setting. Furthermore, we may want to preserve the spatial information provided by the input since our classifier may include convolutional layers. Because of this, we only utilize the first two operations of truncation by excluding the summation. More precisely, given $\vec{w}, \vec{x} \in \mathbb{R}^d$ and integer k such that $2k < d$, we define

$$\left[\vec{w} \odot \vec{x}\right]_k \coloneqq \vec{u}, \text{ where } u_i \coloneqq \begin{cases} w_i x_i \text{ if } k+1 \leq s^{-1}(i) \leq d-k \\ 0 \quad \text{otherwise} \end{cases}, \tag{5}$$

where $s(.)$ is the permutation derived by sorting the elements of $\vec{w} \odot \vec{x}$ in increasing order. As an example, for $d = 3$, $k = 1$, $\vec{x} = (1, -1, 2)^T$, and $\vec{w} = (3, -2, -1)^T$, we have $[\vec{w}, \vec{x}]_k = (0, 2, 0)^T$.

Using the configuration defined in Section III, we consider the family of classifiers $\mathcal{F} : \mathbb{R}^d \to \{1, \ldots, M\}$ that can be represented by any neural network architecture, e.g., fully connected, convolutional, and recurrent. We define a neural network as a function $f(\vec{x}; \vec{\theta}) = y$, parameterized by $\vec{\theta}$. This function accepts an input $\vec{x} \in \mathbb{R}^d$ and returns the predicted label $y \in \{1, \ldots, M\}$. Without loss of generality, we assume that the input $\vec{x}$ has been flattened to one vector, and that the classifier f transforms the input back to its original shape if needed.

Given any classifier $f \in \mathcal{C} : \mathbb{R}^d \to \{1, \ldots, M\}$, we define its k–truncated counterpart as:

$$f^{(k)}\left(\vec{x}; \vec{w}, \vec{\theta}\right) = f\left(\left[\vec{w} \odot \vec{x}\right]_k; \vec{\theta}\right), \tag{6}$$

where $\vec{w}$ is a set of learnable weights that scales the input coordinates, and $\vec{\theta}$ is the set of parameters corresponding to classifier f. By implementing truncation in the initial layer, the impact of the adversary is counteracted in the early stages of the network and is prevented from propagating through the subsequent layers.

C. Adversarial Training

While truncation alone is anticipated to improve a classifier's robustness, we propose extending our approach by integrating our framework with adversarial training. Adversarial training is a common technique for improving the robust accuracy of neural network classifiers [9] - by training classifiers on adversarially perturbed samples from the original data, one aims to minimize the robust classification error. In the Gaussian mixture setting examined in Section V, we prove that the asymptotically optimal classifier necessitates both truncation and an optimization step resembling adversarial training to identify the optimal weights $\vec{w}$. We hypothesize that the extension of these theoretical findings to neural networks will help contribute to enhancing their robustness, and to this end we formalize the adversarial training procedure we utilize in our work when testing our claim.

We express our training algorithm generally for any classifier f, where we consider an ℓ_0–adversary that attacks some classifier f by using an ℓ_0–budget of k and a time budget of t. We define this attack as a function $g(\mathcal{X}; f, k, t) : \mathcal{X} \to \mathcal{X}'$ where $\mathcal{X} = \{\vec{x}_1, \ldots, \vec{x}_{|\mathcal{X}|}\}$ is a set of unperturbed data samples, and $\mathcal{X}' = \{\vec{x}'_1, \ldots, \vec{x}'_{|\mathcal{X}'|}\}$ is the derived set of adversarial examples which are all misclassified by f. Note that we use $|\mathcal{X}|$ to denote the cardinality of the set $\mathcal{X}$. Using this attack, we train on the appended dataset $\mathcal{X} \cup \mathcal{X}'$, emptying the adversarial set every epoch and recalculating it again $\mathcal{X}' = g(\mathcal{X}; f, k, t)$. Consequently, the adversarial examples are selected according to a process that is adaptive with respect to our model f. This procedure is utilized as a means to solve the minimax problem articulated in (3). The specific implementation details of the training framework used in our experiments is described in Section VI as they are problem specific and should be chosen accordingly.

V. Theoretical Framework

In this section, within the framework established in Section III, we consider a Gaussian mixture setting and demonstrate that our algorithm achieves near optimal robust classification error, i.e., we show that the deviation from optimality is asymptotically vanishing. Our theoretical analysis provides a key insight: both truncation and adversarial training are essential components for providing provable robustness against ℓ_0–attacks.

To elaborate, we consider a binary classification setting where the distribution $\mathcal{D}$ is defined as follows. We have $y \in \{\pm 1\}$ with $\mathbb{P}(y = +1) = \mathbb{P}(y = -1) = 1/2$, and $\vec{x} = y\vec{\mu} + \vec{z}$ conditional on y, where $\vec{\mu} \in \mathbb{R}^d$ and $\vec{z} \sim \mathcal{N}(0, \Sigma)$ is a Gaussian vector with zero mean and diagonal covariance matrix Σ. For ease of discussion, we assume that Σ has strictly positive diagonal entries $\sigma_1^2, \ldots, \sigma_d^2$.[3] It is well known that when the adversary is absent, the linear classifier $\text{sgn}(\langle \vec{w}, \vec{x} \rangle)$ with $\vec{w} = \Sigma^{-1}\mu$ is the optimal Bayes classifier. This optimal standard error in this setting is $\bar{\Phi}(\|\Sigma^{-1/2}\vec{\mu}\|_2)$, where $\bar{\Phi}(.)$ denotes the complementary CDF of the standard normal distribution. Therefore, to establish a baseline, we assume without loss of generality that $\|\Sigma^{-1/2}\vec{\mu}\|_2 = 1$ so that the optimal standard error is $\bar{\Phi}(1)$.

Given that the optimal Bayes classifier in this context is linear, and building on our discussion from Section IV, we focus our analysis on single layer neural networks. Specifically, we consider the family of k–truncated linear classifiers:

$$\mathcal{C}^{(k)}_{\vec{w}} : \vec{x}' \mapsto \text{sgn}\left(\langle \vec{w}, \vec{x}' \rangle_k\right).$$

Using our notation in (2), we denote the robust classification error of a classifier $\mathcal{C}$ (a general classifier, not necessarily a truncated linear classifier) by:

$$\mathcal{L}_{\vec{\mu}, \Sigma}(\mathcal{C}, k) = \mathbb{E}_{(\vec{x}, y)\ \sim \mathcal{D}}\left[\max_{\vec{x}' \in \mathcal{B}_0(\vec{x}, k)} \ell\left(\mathcal{C}; \vec{x}', y\right)\right],$$

where k denotes both the truncation parameter and the adversary's budget. Furthermore, as in (3), we denote the optimal

[3] Although we make the diagonal assumption in this section, we discuss a more general setting in Appendix A.

robust classification error by minimizing over all possible classifiers under a fixed k:

$$\mathcal{L}^*_{\vec{\mu},\Sigma}(k) \coloneqq \inf_{\mathcal{C}} \mathcal{L}_{\vec{\mu},\Sigma}(\mathcal{C}, k). \tag{7}$$

Note that here, minimization is taken over all possible classifiers $\mathcal{C}$, not necessarily truncated linear classifiers.

For the sake of brevity, we may omit the problem parameters $\vec{\mu}$ and Σ from the aforementioned notations and use $\mathcal{L}(\mathcal{C}^{(k)}_{\vec{w}}, k)$ and $\mathcal{L}^*(k)$ instead, provided that the context makes them unambiguous.

A. Asymptotic Optimality of Our Algorithm

To demonstrate that k–truncated linear classifiers are asymptotically optimal, we first revisit a result from our previous work [39]. Specifically, we recall the lower bound on the optimal robust classification error, which was established by developing an attack strategy for the adversary and proving that no classifier can achieve better performance.

Theorem 1 (Theorem 2 in [39]): Assume that Σ is diagonal and let $\vec{v} = \Sigma^{-1/2}\vec{\mu}$. Then for any $A \subseteq \{1, \ldots, d\}$, we have

$$\mathcal{L}^*(\|\vec{v}_A\|_1 \log d) \geq \bar{\Phi}(\|\vec{v}_{A^c}\|_2) - \frac{1}{\log d},$$

where $\vec{v}_A$ and $\vec{v}_{A^c}$ denote the coordinates of $\vec{v}$ in the sets A and A^c, respectively.

The main difference between the current work and [39] is that here, we use adversarial training in order to choose the model parameters within the class of truncated linear classifiers defined above. However, in [39], we start with the optimal model parameters when there is no adversary, and use truncation as well as another component called "filteration". Filtration refers to the procedure of removing some of the data coordinates that are most susceptible to adversarial perturbations. Even though this approach is proven to be asymptotically optimal in [39], from a practical point of view, filteration requires a complex discrete optimization which can be intractable in practice. However, in this work, we show that employing adversarial training together with truncation is sufficient and eliminates the need to apply filteration.

Recall from Section IV that adversarial training is used to identify the model weights. This is a proxy for optimizing $\vec{w}$ in the class of k–linear classifiers $\mathcal{C}^{(k)}_{\vec{w}}$. More precisely, let

$$\vec{w}^*(k) \in \arg\min_{\vec{w}} \mathcal{L}\left(\mathcal{C}^{(k)}_{\vec{w}}, k\right). \tag{8}$$

In the following, we demonstrate that when an adversary with ℓ_0–budget k is present, the performance of $\mathcal{C}^{(k)}_{\vec{w}^*(k)}$ is comparable to the optimal robust classification error, with an asymptotically vanishing deviation.

In order to do this, given an error threshold $\bar{\Phi}(1) \leq \varepsilon \leq 1/2$, we define

$$k^{\text{Trunc}}(\varepsilon) \coloneqq \max\left\{k : \mathcal{L}\left(\mathcal{C}^{(k)}_{\vec{w}^*(k)}, k\right) \leq \varepsilon\right\}, \tag{9}$$

which is the maximum adversarial budget that the class of truncated linear classifiers can tolerate to achieve a robust error of at most ε, when the truncation parameter is chosen to equal the adversary's budget. Here, ε is chosen to range between the standard error $\bar{\Phi}(1)$ and the error corresponding to a random guess. Moreover, let

$$k^*(\varepsilon) \coloneqq \max\{k : \mathcal{L}^*(k) \leq \varepsilon\}, \tag{10}$$

be the maximum adversarial budget that an optimal classifier can tolerate constrained on having a robust error of at most ε. By definition, we know $k^*(\varepsilon) \geq k^{\text{Trunc}}(\varepsilon)$.

As we formally demonstrate below, k^{Trunc} and k^* are close to each other up to multiplicative factors that are sublinear in d. As a results, to have a first order analysis and to focus on the behavior of the adversary's budget as a power of the dimension d, we define

$$\alpha^{\text{Trunc}}(\varepsilon) \coloneqq \log_d k^{\text{Trunc}}(\varepsilon), \tag{11}$$

and

$$\alpha^*(\varepsilon) \coloneqq \log_d k^*(\varepsilon). \tag{12}$$

The following theorem shows that α^{Trunc} is close to α^*, modulo some vanishing terms in d. More clearly, the class of linear truncation classifiers are asymptotically optimal for the above mixture Gaussian setting. Proof of Theorem 2 is provided in Appendix A.

Theorem 2: Given $\bar{\Phi}(1) + 1/\log d + \sqrt{2/\log d} < \varepsilon < \frac{1}{2}$, there are constants $c_i = c_i(\varepsilon, d)$, $i \in \{1, 2\}$, which do not depend on the parameters of the problem (i.e., $\vec{\mu}$ and Σ) such that $\lim_{d\to\infty} c_i(\varepsilon, d) = 0$ for $i \in \{1, 2\}$ and

$$\alpha^*(\varepsilon) \geq \alpha^{\text{Trunc}}(\varepsilon) \geq \alpha^*(\varepsilon - c_1) - c_2.$$

The essence of Theorem 2 is that, disregarding asymptotically negligible terms, the truncated classifier can endure an equivalent amount of adversarial budget as an optimal robust classifier. To substantiate this claim, we rely on Theorem 1, which certifies that no other classifier can attain superior asymptotic performance. As a consequence, our algorithm is asymptotically optimal.

VI. Experiments

Now that we established our theoretical results derived for the Gaussian mixture setting, we proceed to evaluate the practicality of our approach on the more complex image domain. This is accomplished by conducting experiments on the MNIST and CIFAR datasets using our proposed algorithm from Section IV. Since these datasets are in the image domain, it is natural to consider an ℓ_0 adversary which is restricted to keep the perturbed pixels within the permissible RGB range. We call this the $\ell_0 + \ell_\infty$ adversary. Note that the $\ell_0 + \ell_\infty$ adversary is weaker as compared to a pure ℓ_0 adversary. In fact, we can recover the ℓ_0 setting by relaxing the ℓ_∞ constraint. This can be done by setting the ℓ_∞ constraint bound to ∞. Motivated by our theoretical study in this paper, we also consider the pure ℓ_0 setting in our experiments by relaxing the ℓ_∞ constraint.

In both settings, we directly compare our algorithm with two well known defense methods against the sparse adversary: adversarial training [9] and the ABS model [31]. We omit comparisons against the randomized ablation defense [33] due

to the lack of computational resources.[4] Before going over the results of our experiments, we briefly summarize the two attack methods used.

A. Sparse Attacks

The experimental results in this paper primarily rely on `spare-rs` [32], a sparse black-box attack framework. Given a time budget t and pixel budget k, the `spare-rs` attack performs a random search to identify and modify a set of k pixels in the input image $\vec{x}$ in such a way that the resulting adversarial image $\vec{x}'$ is misclassified by the model f. In this framework, the attacker can only access the predicted scores of the classifier f, as opposed to white-box methods where gradient information can be utilized by the attacker. The authors of `sparse-rs` have demonstrated the superiority of their framework over other existing black-box and white-box attacks. Therefore, we employ this attack to evaluate the *robust accuracy* of our classifier, and use it as the adversarial training component in our defense framework. In the scope of our notation from Section IV-C, when training is finished we run the ℓ_0–attack $g(\mathcal{X}; f, k, t) = \mathcal{X}'$ on the entire testset $\mathcal{X}$, deriving a set of incorrectly classified samples $\mathcal{X}'$. We then calculate the robust accuracy as the portion of samples that are still correctly classified: $1 - |\mathcal{X}'|/|\mathcal{X}|$, where $|\mathcal{X}'|$ includes unperturbed samples $\vec{x} \in \mathcal{X}$ that were originally misclassified.

In addition, to compare our results with prior literature and test the transferability of our model's performance to an unseen adversary, we utilize the `Pointwise Attack` [31]. This type of attack aims to minimize the ℓ_0–norm in a greedy manner, first introducing salt-and-pepper noise, and then iteratively resetting perturbed pixels while maintaining the image misclassified. In contrast to the `sparse-rs` framework, we have no direct control over the number of allowed perturbations k, and hence cannot use it to evaluate the robust accuracy. Instead, we follow prior work and measure the *median adversarial attack magnitude*, denoting this value with ρ. Given a set of images $\mathcal{X}$, ρ is defined as the median value of pixels that need to be perturbed in order to misclassify an image $\vec{x} \in \mathcal{X}$, where originally misclassified images get a value of 0 and perturbed images that are still correctly classified get a value of ∞.

B. Results

In total, we conducted four sets of experiments to evaluate the robustness of our proposed method, reporting:

(1) the impact of increasing the ℓ_0–budget of an adversary on robust accuracy,

(2) the effects of relaxing the ℓ_∞–constraint of an adversary,

(3) the robust accuracy of different defense methods under a strong adversarial attack, and

[4]Using the implementation provided by the authors, we could not evaluate their CIFAR model since performing a forward pass with a batch size of one needed over 40GB of VRAM. For their MNIST model, using the pointwiwse attack on one image took 35 hours. Although we could utilize their models by decreasing the number of ablated samples, this negatively impacted the model's performance and hence we chose to not report such results.

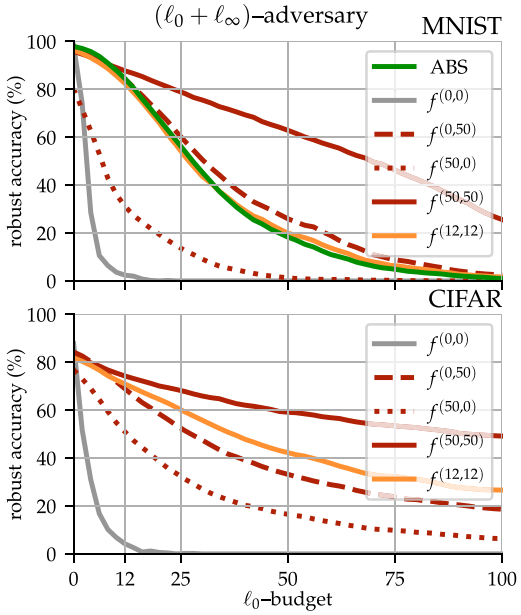

Fig. 2. We plot the robust accuracy of various classifiers when evaluated against the `sparse-rs` attack using 100 images from both MNIST (top) and CIFAR (bottom) datasets. In this experiment we use the original `sparse-rs` attack which bounds its pixel perturbations to the domain [0, 1], where the x–axis shows the ℓ_0–budget of the attack.

(4) the generalization performance of our method to an unseen attack. In all experiment except for the second one, we consider the $\ell_0 + \ell_\infty$ adversarial setting.

As suggested by our theoretical results, when training our k–truncated classifier $f^{(k)}$, we employ an adversarial training strategy that uses a sparse-adversary with an identical ℓ_0–budget of k. Since our experiments ablate both the truncation and adversarial training components of our algorithm, we introduce additional notation for clarity. Specifically, we refer to $f^{(k,r)}$ as the k–truncated classifier $f^{(k)}$ adversarially trained against a sparse-adversary with an ℓ_0–budget of r. This way, $f^{(0,r)}$ refers to adversarial training without truncation, $f^{(k,0)}$ refers to truncation without adversarial training, and $f^{(0,0)}$ refers to no adversarial training and no truncation. For our experiments, we treat k as a hyper parameter, and study how different choices of k affect the performance of our classifier on various attacks. In all experiments we set the time budget of the adversary to 500 queries when training. For CIFAR we used the VGG-16 [45] architecture without dropout layers, while for MNIST we used a similar but smaller CNN architecture. We provide the remaining implementation details online [42].

1) Varying the ℓ_0–Budget: We first test how varying the ℓ_0–budget of our attack impacted the robust accuracy of different classifiers. Here, we consider an $\ell_0 + \ell_\infty$ adversary where the ℓ_∞ budget of the adversary is set to be the permissible pixel range. We visualize these results in Figure 2 for both MNIST and CIFAR, where we used the `sparse-rs` adversary with a time budget of 500 queries and varied the ℓ_0-budget from 0 to 100 in increments of 2. The results showed that truncation and adversarial training improved the robust accuracy of the classifiers.

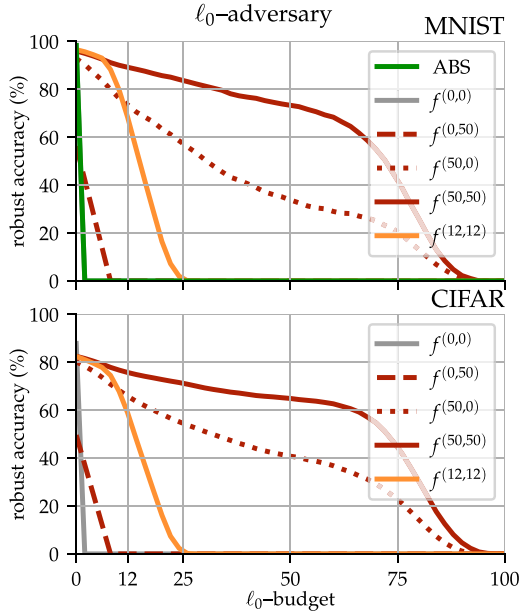

Fig. 3. We repeat the experiments shown in Figure 3, this time relaxing the ℓ_∞–constraint by letting the attack perturb pixels beyond the domain $[0, 1]$. Note that the results for $f^{(0,0)}$ and the ABS model are plotted on top of each other since they all show the same trivial performance - their robust accuracy drops to zero when the ℓ_0–budget increases to 1.

On MNIST, we see that the clean classifier without adversarial training $f^{(0,0)}$ fails once the ℓ_0–budget is increased. We also see that the ABS model performs similarly to $f^{(0,50)}$, which only employs adversarial training without truncation. Finally, we note that although our truncated classifier $f^{(12,12)}$ does not show improvement over the ABS model, when we increase the truncation parameter to $k = 50$, we see $f^{(50,50)}$ outperforms all other baselines. For CIFAR, we note that the plot for ABS is not shown because their method does not scale to the more complex dataset. We can see again that truncation and adversarial training improved the robust accuracy of our classifiers Notably, we can see in both settings that although using truncation $f^{(50,0)}$ or adversarial training $f^{(0,50)}$ independently enhances the original classifier $f^{(0,0)}$, applying both components in $f^{(50,50)}$ shows significant improvement. This underscores that combining truncation and adversarial training is essential for providing defense against ℓ_0–attacks, as proposed by our theoretical analysis.

2) Relaxing the ℓ_∞–Constraint: In our previous experiment we used the original implementation of `sparse-rs`, which acts as an $\ell_0 + \ell_\infty$–adversary because the pixel perturbations are bounded to the original image domain of $[0, 1]$. We realize that having this bound is a practical choice, but because our theoretical results assumed a pure ℓ_0–adversary, we tested if our method could maintain performance when the ℓ_∞–constraint was relaxed. We did this by modifying the `sparse-rs` algorithm to allow perturbations in the expanded domain of $[-50, 50]$. We visualize these results in Figure 3 for both MNIST and CIFAR. We see that ABS, adversarial training alone $f^{(0,50)}$, as well as the original classifier $f^{(0,0)}$ show trivial performance - a few perturbed pixel cause misclassification in all 100 samples. On the other hand, our truncated classifiers $f^{(12,12)}$ and $f^{(50,50)}$ maintain robust accuracy until the attack budget becomes much higher than the truncation parameter.

Before moving on, we point out that the robust accuracy of both the $k = 50$ truncated classifiers with and without adversarial training, $f^{(50,50)}$ and $f^{(50,0)}$ respectively, are higher under the less restrictive ℓ_0–threat model compared to the $\ell_0 + \ell_\infty$–threat model. We note that under the same time budget, the less restrictive ℓ_0–threat model has a harder time finding adversarial examples due to the larger (unbounded) search space, which inherently favors the truncation component of our classifiers. Due to the heuristic nature of the attacks employed, we believe that the results of the two threat models should be viewed independently, as they are portrayed in the paper.

3) Measuring the Robust Accuracy: Although our previous experiments measured robust accuracy using `sparse-rs`, the time budget was kept small so we could run many evaluations using various ℓ_0–budgets. To accurately evaluate the performance of different defense methods, we setup `sparse-rs` as done in the original study [32]: using 5 restarts, with a time budget of $10,000$ queries, and an ℓ_0 budget of 12. Note that this refers to the aforementioned $\ell_0 + \ell_\infty$ setting. Using this attack, we compared the performance of ABS, the original classifier $f^{(0,0)}$, adversarial training alone $f^{(0,12)}$ and $f^{(0,50)}$, as well as our truncated classifiers $f^{(12,12)}$ and $f^{(50,50)}$. We list these results in Table I.

We can see from the results on MNIST that our truncated classifier $f^{(50,50)}$ outperformed ABS and all other models. Although we slightly sacrifice clean accuracy, utilizing truncation only decreased the base classifier's accuracy by 0.2%. As we used a small CNN consisting of only 4 convolutional layers, we believe the gap in clean accuracy compared to the ABS model is negligible. For both MNIST and CIFAR, we see our truncated classifiers $f^{(12,12)}$ and $f^{(50,50)}$ outperformed adversarial training alone, with a larger gap for the CIFAR experiments. We again note that the ABS model was not trained on the CIFAR dataset since the authors claimed preliminary experiments provided only a clean accuracy of 54%.

4) Generalizing to Unseen Attacks: Since our algorithm relies on adversarial training, we expect our final model to perform better against adversaries that were used during training. Hence for our final experiment we tested our model against an unseen attack, measuring the median adversarial attack magnitude ρ by using the `Pointwise Attack` over 100 image samples and 10 restarts. We show this result in Table I, where for MNIST we note that although both our truncated classifier $f^{(12,12)}$ and adversarial training alone $f^{(0,50)}$ outperformed ABS on this unseen attack, using truncation gave no improvement over adversarial training alone. On the other hand, for CIFAR we see that our truncated classifiers were able to generalize their performance to the unseen attack, outperforming adversarial training alone. As we will further discuss in the concluding section, we believe that the observed discrepancy in performance between the MNIST and CIFAR datasets can be attributed to the bimodal distribution of pixels present in the MNIST dataset. This distribution makes it challenging to eliminate outliers through truncation, as a

TABLE I
PERFORMANCE OF ABS [31], THE CLEAN CLASSIFIER $f^{(0,0)}$, ADVERSARIAL TRAINING $f^{(0,12)}$ AND $f^{(0,50)}$, AND OUR TRUNCATED CLASSIFIERS $f^{(12,12)}$ AND $f^{(50,50)}$ AGAINST BOTH THE `sparse-rs` AND `Pointwise` ATTACKS. NOTE THAT WHILE `sparse-rs` MEASURES ROBUST ACCURACY, `Pointwise` MEASURES THE MEDIAN ADVERSARIAL ATTACK MAGNITUDE ρ. FOR BOTH DATASETS WE UTILIZE 100 SAMPLES WHEN EVALUATING AGAINST ATTACKS

Setup			Attack Model	
Model	Data	Acc. (%)	`sparse-rs` [32] (%)	`Pointwise` [31] (# of pixels)
ABS	MNIST	**99.0**	37.0	19
$f^{(0,0)}$	MNIST	98.3	0.0	4
$f^{(0,12)}$	MNIST	98.6	23.0	16
$f^{(0,50)}$	MNIST	98.5	47	**27**
$f^{(12,12)}$	MNIST	98.3	29.0	22
$f^{(50,50)}$	MNIST	98.1	**50.0**	14
$f^{(0,0)}$	CIFAR	**87.9**	0.0	2
$f^{(0,12)}$	CIFAR	87.5	27.0	16
$f^{(0,50)}$	CIFAR	86.9	32.0	16
$f^{(12,12)}$	CIFAR	84.8	44.0	25
$f^{(50,50)}$	CIFAR	84.2	**64.0**	**51**

significant number of pixels already lie at the limit of their respective domain.

VII. DISCUSSION AND CONCLUSION

In this paper we introduced a novel defense method against sparse attacks, investigated its efficacy in the theoretical setting of Gaussian mixtures, and tested its application in the image domain. We conclude by discussing our results as well as potential avenues for future work.

Firstly, we observe that while our theoretical analysis is focused on binary classification, the insights gained from this theoretical discussion has enabled us to design an architecture that shows robustness in practical multiclass classification settings. We believe that in order to bridge this gap, extending our theoretical analysis to multiclass classification is an interesting direction for future research.

Also, it is important to mention that while only two black-box attacks were utilized for evaluating our method, the chosen `sparse-rs` framework demonstrated superior performance to all black- and white-box attacks. Furthermore, we did not design adaptive attacks for evaluating the robustness of our algorithm as suggested by prior works [46]. We believe that additional experiments with adaptive attacks will not improve our results as the general purpose `sparse-rs` framework utilized in our work was adept in causing misclassifications throughout our empirical analysis. As we showed in Table I, when using the `sparse-rs` framework with an ℓ_0–budget of 12 pixels, the robust accuracy was below 50% for 9 out of 11 evaluated models, with the highest being 50% and 64%, for MNIST and CIFAR respectively. Due to the strength of the ℓ_0–adversary, providing theoretical results and showing robustness against simple attack models is a significant first step in this line of research. Furthermore, we argue that without knowledge of the weights $\vec{w}$ used to perform our truncation operation, a true ℓ_0–attack provides the worst-case guarantees. However, we note that an adaptive attack that has access to the weights $\vec{w}$ can be more discreet when applying perturbations to the input, targeting coordinates without triggering the truncation operation.

Additionally, we add that while some prior works have considered the generalization properties between different ℓ_p–attacks [47], [48], [49] for $p \geq 1$, there is no theoretical justification for utilizing ℓ_1 or ℓ_2–adversaries when training classifiers (with or without truncation) to be robust against an ℓ_0–adversary. As demonstrated by our theoretical result, adversarial training wrt ℓ_0 is required for identifying the model weights $\vec{w}$ in our truncation operation. Hence we choose to only include the current attack model when training our truncated classifiers due to its relevance to the ℓ_0–threat model being studied in this paper. Furthermore, solely applying adversarial training together with ℓ_1 or ℓ_2–norms (without truncation) results in a classifier that belongs to the family of linear classifiers. However, we know from prior work [37] that linear classifiers do not behave well in the presence of ℓ_0 adversarial perturbations.

We also want to discuss our results displayed in Figures 2 and 3, which indicate a rapid decline in robust accuracy as the ℓ_0–budget increases, even for our k–truncated classifiers. Although there is an inherent limit to how much we can mitigate this, we showed that setting our truncation parameter to higher values could allow one to maintain performance without damaging clean accuracy. On top of this, additional truncation layers with decreasing k can be added beyond the first layer, as this would help prevent perturbed coordinates from propagating further into the network. Finally, a larger adversarial time budget can be used when performing training.

Additionally, we want to emphasize the efficiency of utilizing truncation as a defense against ℓ_0–attacks. For our truncated classifier $f^{(50,50)}$, we measured the increase of preforming a forward pass to be less than 25% compared to the base classifier, meanwhile the ABS model was more than 1000 times slower, and randomized ablation was not feasible to run on a cluster of 4–GPUs. Since truncation is a functional improvement that can be efficiently implemented alongside any classifier, we believe it to be a powerful tool against adversarial perturbations in the ℓ_0–setting.

Finally, we point out that truncation is not limited to the image domain, and may be more effective in the NLP domain. Since truncation is theoretically motivated by the ℓ_0–setting, it will scale better in domains where the adversary is less constrained by the ℓ_∞–norm bounding the perturbations. We showed in Figure 2 and Table I that truncation was relatively more effective on CIFAR than on MNIST. Since the MNIST dataset contains many black and white pixels whose values are close to the boundaries of the domain, perturbed pixels are harder to detect when performing the sorting operation of truncation. Images in CIFAR do not share this property, making it easier for truncation to detect outliers as the ℓ_∞–constraint is less prevalent. In the NLP domain, adversaries are typically allowed to change a fixed number of words by performing character operations or replacing words with synonyms [50]. Since there is no inherent limit on the perturbations, the adversary closely resembles an ℓ_0–attack.

For these reasons, we hypothesize that using truncation in the NLP domain may be an effective strategy against adversarial attacks.

APPENDIX A
PROOF OF THEOREM 2

Before giving the proof of Theorem 2, we need to make some definitions and state some lemmas. The proofs of the lemmas are provided at the end of this section.

We first study the effect of truncation on the inner product. Lemma 1 below from [39] provides an upper bound on the deviation of the truncated inner product from the original inner product.

Lemma 1 (Lemma 1 in [39]): Given $\vec{x}, \vec{x}', \vec{w} \in \mathbb{R}^d$, for integer k satisfying $\|\vec{x} - \vec{x}'\|_0 \le k < d/2$, we have

$$|\langle \vec{w}, \vec{x}' \rangle_k - \langle \vec{w}, \vec{x} \rangle| \le 8k \|\vec{w} \odot \vec{x}\|_\infty.$$

Recall that in Section V, to simplify the discussion, we restrict ourselves to diagonal covariance matrices. However, in order to have a general setup, here we begin by proving an upper bound for the robust classification error of the family of truncated linear classifiers. In this case, we assume that the covariance matrix Σ is positive definite, but does not need to be diagonal. Lemma 2 below shows an upper bound for the robust classification error of the k–truncated linear classifier $\mathcal{C}^{(k)}_{\vec{w}}$.

Lemma 2: We have

$$\mathcal{L}_{\vec{\mu},\Sigma}\left(\mathcal{C}^{(k)}_{\vec{w}}, k\right) \le \frac{1}{\sqrt{2\log d}} + \bar{\Phi}\left(\frac{\langle \vec{w}, \vec{\mu} \rangle - 8k\|\widetilde{\Sigma}^{1/2}\vec{w}\|_\infty \left(1 + \sqrt{2\log d}\right)}{\|\Sigma^{1/2}\vec{w}\|_2}\right),$$

where $\widetilde{\Sigma}$ is the diagonal part of Σ.

As a direct consequence, this lemma implies the following bound for the diagonal regime.

Corollary 1: When the covariance matrix Σ is diagonal, we have

$$\mathcal{L}_{\vec{\mu},\Sigma}\left(\mathcal{C}^{(k)}_{\vec{w}}, k\right) \le \frac{1}{\sqrt{2\log d}} + \bar{\Phi}\left(\frac{\langle \widetilde{\vec{w}}, \vec{\nu} \rangle - 8k\|\widetilde{\vec{w}}\|_\infty \left(1 + \sqrt{2\log d}\right)}{\|\widetilde{\vec{w}}\|_2}\right),$$

where $\widetilde{\vec{w}} = \Sigma^{1/2}\vec{w}$ and $\vec{\nu} = \Sigma^{-1/2}\vec{\mu}$.

From this point forward, in order to prove Theorem 2, we assume that the covariance matrix Σ is diagonal with positive diagonal entries $\sigma_1^2, \ldots, \sigma_d^2$. We define

$$\vec{\nu} \coloneqq \Sigma^{-1/2}\vec{\mu}, \tag{13}$$

so that $\nu_i = \mu_i/\sigma_i$ is the signal to noise ratio associated to coordinate i. Without loss of generality, we may assume that

$$|\nu_1| \ge |\nu_2| \ge \cdots \ge |\nu_d|. \tag{14}$$

For $\bar{\Phi}(1) < \varepsilon < 1/2$, let $c(\varepsilon)$ be the unique solution of $\bar{\Phi}(\sqrt{1-c^2}) = \varepsilon$. Note that $c(\varepsilon) \in (0,1)$. Moreover, given $c \in (0,1)$, we define

$$\lambda_c \coloneqq \min\{\lambda : \|\vec{\nu}_{[1:\lambda]}\|_2 \ge c\}, \tag{15}$$

where $\vec{\nu}_{[1:\lambda]}$ corresponds to the vector containing the first λ elements of $\vec{\nu}$.

Note that since $c > 0$, we have $\lambda_c \ge 1$. Moreover, since $c < 1$ and $\|\vec{\nu}\|_2 = 1$, we have

$$\lambda_c < d. \tag{16}$$

Using Lemma 2 and in particular Corollary 1 in the diagonal regime, we can show the following bound on the robust classification error of the optimal k–truncated linear classifier $\mathcal{C}^{(k)}_{\vec{w}^*(k)}$.

Lemma 3: Assume that the covariance matrix Σ is diagonal. Given $\bar{\Phi}(1) < \varepsilon < 1/2$, for $k = a\|\vec{\nu}_{[1:\lambda_{c(\varepsilon)}]}\|_1$, we have

$$\mathcal{L}\left(\mathcal{C}^{(k)}_{\vec{w}^*(k)}, k\right) \le \varepsilon + a\frac{8\left(1 + \sqrt{2\log d}\right)}{\sqrt{2\pi}\sqrt{1 - c(\varepsilon)^2}} + \frac{1}{\sqrt{2\log d}}.$$

Furthermore, we can show the following lower bound on k^* which involves the class of all classifiers.

Lemma 4: For $\bar{\Phi}(1) + \frac{1}{\log d} < \varepsilon < \frac{1}{2}$, we have

$$k^*\left(\varepsilon - \frac{1}{\log d}\right) \le \|\vec{\nu}_{[1:\lambda_{c(\varepsilon)}]}\|_1 \log d.$$

We are finally ready to prove Theorem 2.

Proof of Theorem 2: Using Lemma 3 for $\bar{\Phi}(1) \le \varepsilon < 1/2$ and $k = a\|\vec{\nu}_{[1\,:\,\lambda_{c(\varepsilon)}]}\|_1$ with

$$a = \sqrt{1 - c(\varepsilon)^2}\frac{1}{16\log d},$$

we get

$$\mathcal{L}\left(\mathcal{C}^{(k)}_{\vec{w}^*(k)}, k\right) \le \varepsilon + \frac{1 + \sqrt{2\log d}}{2\sqrt{2\pi}\log d} + \frac{1}{\sqrt{2\log d}} \le \varepsilon + \sqrt{\frac{2}{\log d}}.$$

This means that

$$k^{\text{Trunc}}\left(\varepsilon + \sqrt{\frac{2}{\log d}}\right) \ge \frac{\sqrt{1 - c(\varepsilon)^2}}{16}\frac{\|\vec{\nu}_{[1:\lambda_{c(\varepsilon)}]}\|_1}{\log d} \quad \text{for} \quad \bar{\Phi}(1) < \varepsilon < \frac{1}{2} - \sqrt{\frac{2}{\log d}}. \tag{17}$$

On the other hand, from Lemma 4 we know that

$$k^*\left(\varepsilon - \frac{1}{\log d}\right) \le \|\vec{\nu}_{[1:\lambda_{c(\varepsilon)}]}\|_1 \log d \quad \text{for} \quad \bar{\Phi}(1) + \frac{1}{\log d} < \varepsilon < \frac{1}{2}.$$

Comparing this with (33), we realize that

$$k^{\text{Trunc}}\left(\varepsilon + \sqrt{\frac{2}{\log d}}\right) \ge \frac{\sqrt{1 - c(\varepsilon)^2}}{16\log^2 d}k^*\left(\varepsilon - \frac{1}{\log d}\right) \quad \text{for} \quad \bar{\Phi}(1) + \frac{1}{\log d} < \varepsilon < \frac{1}{2} - \sqrt{\frac{2}{\log d}}.$$

Equivalently, taking $\log_d$ from both sides, we realize that for $\bar{\Phi}(1)+1/\log d<\varepsilon<1/2-\sqrt{2/\log d}$,

$$\alpha^{\text{Trunc}}\left(\varepsilon+\sqrt{\frac{2}{\log d}}\right) \geq \alpha^*\left(\varepsilon-\frac{1}{\log d}\right)-\frac{2\log\log d}{\log d}+\frac{\log\left(\sqrt{1-c(\varepsilon)^2}/16\right)}{\log d}.$$

By shifting ε, we realize that for $\bar{\Phi}(1)+1/\log d+\sqrt{2/\log d}<\varepsilon<1/2$, we have

$$\alpha^{\text{Trunc}}(\varepsilon)\geq\alpha^*(\varepsilon-c_1(\varepsilon,d))-c_2(\varepsilon,d), \tag{18}$$

where

$$c_1(\varepsilon,d)\ :=\frac{1}{\log d}+\sqrt{\frac{2}{\log d}}, \tag{19}$$

and

$$c_2(\varepsilon,d)\ :=\frac{2\log\log d}{\log d}-\frac{\log\left(\frac{\sqrt{1-c\left(\varepsilon-\sqrt{2/\log d}\right)^2}}{16}\right)}{\log d}. \tag{20}$$

Observe that for $i\in\{1,2\}$, $c_i(\varepsilon,d)$ does not depend on the parameters of the problem and $\lim_{d\to\infty}c_i(\varepsilon,d)\ =0$. On the other hand, since $\alpha^*(\varepsilon)$ is obtained by optimizing over all classifiers while $\alpha^{\text{Trunc}}(\varepsilon)$ is obtained by optimizing over the class of linear truncated classifiers, we always have $\alpha^*(\varepsilon)\geq\alpha^{\text{Trunc}}(\varepsilon)$. This completes the proof. ∎

Finally, we give the proofs for Lemmas 2, 3, and 4.

Proof of Lemma 2: We may write

$$\begin{aligned}
&\mathcal{L}_{\vec{\mu},\Sigma}\left(\mathcal{C}_{\vec{w}}^{(k)},k\right)\\
&=\mathbb{E}_{(\vec{x},y)\ \sim\mathcal{D}}\left[\max_{\vec{x}'\in\mathcal{B}_0(\vec{x},k)}\ell\left(\mathcal{C}_{\vec{w}}^{(k)};\vec{x}',y\right)\ \right]\\
&=\mathbb{P}\left(\exists\vec{x}'\in\mathcal{B}_0(\vec{x},k){:}\mathcal{C}_{\vec{w}}^{(k)}(\vec{x}'\neq y)\ \right)\\
&=\mathbb{P}\left(\exists\vec{x}'\in\mathcal{B}_0(\vec{x},k){:}\text{sgn}(\langle\vec{w},\vec{x}'\rangle_k)\neq y\right)\\
&\overset{(*)}{=}\mathbb{P}\left(\exists\vec{x}'\in\mathcal{B}_0(\vec{x},k){:}\text{sgn}(\langle\vec{w},\vec{x}'\rangle_k)\neq 1|y=1\right)\\
&=\mathbb{P}\left(\exists\vec{x}'\in\mathcal{B}_0(\vec{x},k){:}\langle\vec{w},\vec{x}'\rangle_k\leq 0|y=1\right)
\end{aligned} \tag{21}$$

where $(*)$ uses the symmetry in distribution $\mathcal{D}$. Using Lemma 1, for all $\vec{x}'\in\mathcal{B}_0(\vec{x},k)$, we have

$$\langle\vec{w},\vec{x}'\rangle_k\geq\langle\vec{w},x\rangle-8k\|\vec{w}\odot\vec{x}\|_\infty.$$

Using this in (21), we get

$$\mathcal{L}_{\vec{\mu},\Sigma}\left(\mathcal{C}_{\vec{w}}^{(k)},k\right)\ =\mathbb{P}(\langle\vec{w},\vec{x}\rangle\leq 8k\|\vec{w}\odot\vec{x}\|_\infty|y=1). \tag{22}$$

Note that conditioned on $y=1$, we have $\vec{x}=\vec{\mu}+\vec{z}$ where $\vec{z}\sim\mathcal{N}(0,\Sigma)$. Let $\widetilde{\Sigma}$ be the diagonal matrix consisting of the diagonal entries in Σ. Since $\widetilde{\Sigma}$ is diagonal, we may write

$$\begin{aligned}
\|\vec{w}\odot\vec{x}\|_\infty&=\|\vec{w}\odot\vec{\mu}+\vec{w}\odot\vec{z}\|_\infty\\
&\leq\|\vec{w}\odot\vec{\mu}\|_\infty+\|\vec{w}\odot\vec{z}\|_\infty\\
&\leq\|\left(\widetilde{\Sigma}^{1/2}\vec{w}\right)\odot\left(\widetilde{\Sigma}^{-1/2}\vec{\mu}\right)\|_\infty\\
&\quad+\|\left(\widetilde{\Sigma}^{1/2}\vec{w}\right)\odot\left(\widetilde{\Sigma}^{-1/2}\vec{z}\right)\|_\infty\\
&\leq\|\widetilde{\Sigma}^{1/2}\vec{w}\|_\infty\left(\|\widetilde{\Sigma}^{-1/2}\vec{\mu}\|_\infty+\|\widetilde{\Sigma}^{-1/2}\vec{z}\|_\infty\right).
\end{aligned} \tag{23}$$

We now bound the infinity norm of the vector $\vec{a}:=\widetilde{\Sigma}^{-1/2}\vec{\mu}$. With $\sigma_1^2,\ldots,\sigma_d^2$ denoting the diagonal entries in Σ, we have $a_i=\mu_i/\sigma_i$. Note that $\bar{\Phi}(|\mu_i|/\sigma_i)$ is the optimal Bayes classification error of y given x_i only, which cannot be smaller than the optimal Bayes classifier of y given the whole vector $\vec{x}$, which is in turn equal to $\bar{\Phi}(\|\Sigma^{-1/2}\vec{\mu}\|_2)=\bar{\Phi}(1)$. This means that $|a_i|=|\mu_i|/\sigma_i\leq 1$, and in particular

$$\|\vec{a}\|_\infty=\|\widetilde{\Sigma}^{-1/2}\vec{\mu}\|_\infty\leq 1. \tag{24}$$

Next, we bound the infinity norm of the random vector $\vec{b}:=\widetilde{\Sigma}^{-1/2}\vec{z}$. Note that $b_i\sim\mathcal{N}(0,1)$. Therefore, using the union bound, we may write

$$\begin{aligned}
\mathbb{P}\left(\|\widetilde{\Sigma}^{-1/2}\vec{z}\|_\infty\geq\sqrt{2\log d}\right)&\leq d\bar{\Phi}\left(\sqrt{2\log d}\right)\\
&\leq d\frac{1}{\sqrt{2\pi}\sqrt{2\log d}}e^{-\log d}\\
&\leq\frac{1}{\sqrt{2\log d}}.
\end{aligned} \tag{25}$$

Using this together with (24) back into (23), we realize that

$$\begin{aligned}
&\mathbb{P}\left(\|\vec{w}\odot\vec{x}\|_\infty\leq\|\widetilde{\Sigma}^{1/2}\vec{w}\|_\infty(1+\sqrt{2\log d})\right)\\
&\geq 1-\frac{1}{\sqrt{2\log d}}.
\end{aligned}$$

This together with (22) implies that

$$\begin{aligned}
\mathcal{L}_{\vec{\mu},\Sigma}\left(\mathcal{C}_{\vec{w}}^{(k)},k\right)\ &\leq\frac{1}{\sqrt{2\log d}}\\
&\quad+\mathbb{P}\left(\langle\vec{w},\vec{x}\rangle\leq 8k\|\widetilde{\Sigma}^{1/2}\vec{w}\|_\infty(1+\sqrt{2\log d})\ \middle|\ y=1\right).
\end{aligned} \tag{26}$$

Again, using the fact that $\vec{x}=\vec{\mu}+\vec{z}$ conditioned on $y=1$, we have

$$\begin{aligned}
&\mathbb{P}\left(\langle\vec{w},\vec{x}\rangle\leq 8k\|\widetilde{\Sigma}^{1/2}\vec{w}\|_\infty(1+\sqrt{2\log d})\ \middle|\ y=1\right)\\
&=\mathbb{P}\left(\langle\vec{w},\vec{z}\rangle\leq 8k\|\widetilde{\Sigma}^{1/2}\vec{w}\|_\infty(1+\sqrt{2\log d})-\langle\vec{w},\vec{\mu}\rangle\right)\\
&=\mathbb{P}\left(\frac{\langle\vec{w},\vec{z}\rangle}{\|\Sigma^{1/2}\vec{w}\|_2}\leq\frac{8k\|\widetilde{\Sigma}^{1/2}\vec{w}\|_\infty(1+\sqrt{2\log d})-\langle\vec{w},\vec{\mu}\rangle}{\|\Sigma^{1/2}\vec{w}\|_2}\right)\\
&=\bar{\Phi}\left(\frac{\langle\vec{w},\vec{\mu}\rangle-8k\|\widetilde{\Sigma}^{1/2}\vec{w}\|_\infty\left(1+\sqrt{2\log d}\right)}{\|\Sigma^{1/2}\vec{w}\|_2}\right).
\end{aligned}$$

Substituting this into (26) completes the proof of Lemma 2. ∎

Proof of Lemma 3: We define $\widetilde{\vec{w}}\in\mathbb{R}^d$ as follows

$$\widetilde{w}_i=\begin{cases}0 & i<\lambda_{c(\varepsilon)}\\ v_i & i\geq\lambda_{c(\varepsilon)}\end{cases}$$

With this, let $\vec{w}=\Sigma^{-1/2}\widetilde{\vec{w}}$ and note that since $\vec{w}^*(k)$ is obtained by optimizing for $\mathcal{L}(\mathcal{C}_{\vec{w}}^{(k)},k)$, we have

$$\mathcal{L}\left(\mathcal{C}_{\vec{w}^*(k)}^{(k)},k\right)\ \leq\mathcal{L}\left(\mathcal{C}_{\vec{w}}^{(k)},k\right), \tag{27}$$

with $\vec{w}$ defined above. From Corollary 1, we have

$$\begin{aligned}
\mathcal{L}\left(\mathcal{C}_{\vec{w}}^{(k)},k\right)&\leq\frac{1}{\sqrt{2\log d}}\\
&\quad+\bar{\Phi}\left(\frac{\langle\widetilde{\vec{w}},\vec{v}\rangle-8k\|\widetilde{\vec{w}}\|_\infty\left(1+\sqrt{2\log d}\right)}{\|\widetilde{\vec{w}}\|_2}\right).
\end{aligned} \tag{28}$$

Note that

$$\langle \widetilde{\vec{w}}, \vec{v} \rangle = \sum_{i=\lambda_{c(\varepsilon)}}^{d} v_i^2 = \|\vec{v}_{[\lambda_{c(\varepsilon)}:d]}\|_2^2. \tag{29}$$

Likewise,

$$\|\widetilde{\vec{w}}\|_2 = \sqrt{\sum_{i=\lambda_{c(\varepsilon)}}^{d} v_i^2} = \|\vec{v}_{[\lambda_{c(\varepsilon)}:d]}\|_2. \tag{30}$$

Recall that $\lambda_{c(\varepsilon)}$ by definition is the smallest λ such that $\|\vec{v}_{[1:\lambda]}\|_2 \geq c(\varepsilon)$. This implies that $\|\vec{v}_{[1:\lambda_{c(\varepsilon)}-1]}\|_2 < c(\varepsilon)$ and

$$\|\vec{v}_{[\lambda_{c(\varepsilon)}:d]}\|_2^2 = \|\vec{v}\|_2^2 - \|\vec{v}_{[1:\lambda_{c(\varepsilon)}-1]}\|_2^2 \geq 1 - c(\varepsilon)^2. \tag{31}$$

Comparing this with (29) and (31), we realize that

$$\frac{\langle \widetilde{\vec{w}}, \vec{v} \rangle}{\|\widetilde{\vec{w}}\|_2} = \|\vec{v}_{[\lambda_{c(\varepsilon)}:d]}\|_2 \geq \sqrt{1 - c(\varepsilon)^2}. \tag{32}$$

On the other hand, since we have assumed in (14), we have $\|\widetilde{\vec{w}}\|_\infty = |\vec{v}_{\lambda_{c(\varepsilon)}}|$. Furthermore, using (14), we have

$$\begin{aligned}\|\vec{v}_{[1:\lambda_{c(\varepsilon)}]}\|_1 |v_{\lambda_{c(\varepsilon)}}| &= \sum_{i=1}^{\lambda_{c(\varepsilon)}} |v_i| |v_{\lambda_{c(\varepsilon)}}| \\ &\leq \sum_{i=1}^{\lambda_{c(\varepsilon)}} |v_i|^2 \leq \|\vec{v}\|_2^2 = 1.\end{aligned}$$

This together with (29) and (30) implies that

$$\begin{aligned}&\frac{8k\|\widetilde{\vec{w}}\|_\infty \big(1 + \sqrt{2\log d}\big)}{\|\vec{w}\|_2} \\ &= \frac{8\big(1 + \sqrt{2\log d}\big) a \|\vec{v}_{[1:\lambda_{c(\varepsilon)}]}\|_1 |v_{\lambda_{c(\varepsilon)}}|}{\|\vec{v}_{[\lambda_{c(\varepsilon)}:d]}\|_2} \\ &\leq \frac{8a\big(1 + \sqrt{2\log d}\big)}{\sqrt{1 - c(\varepsilon)^2}}.\end{aligned}$$

Using this and (32) back into (28) and using the fact that $\bar{\Phi}(.)$ is decreasing and $1/\sqrt{2\pi}$-Lipschitz, we realize that

$$\begin{aligned}&\mathcal{L}\Big(\mathcal{C}_{\vec{w}}^{(k)}, k\Big) \\ &\leq \frac{1}{\sqrt{2\log d}} + \bar{\Phi}\left(\sqrt{1 - c(\varepsilon)^2} - \frac{8a\big(1 + \sqrt{2\log d}\big)}{\sqrt{1 - c(\varepsilon)^2}}\right) \\ &\leq \bar{\Phi}\Big(\sqrt{1 - c(\varepsilon)^2}\Big) + a\frac{8\big(1 + \sqrt{2\log d}\big)}{\sqrt{2\pi}\sqrt{1 - c(\varepsilon)^2}} + \frac{1}{\sqrt{2\log d}} \\ &= \varepsilon + a\frac{8\big(1 + \sqrt{2\log d}\big)}{\sqrt{2\pi}\sqrt{1 - c(\varepsilon)^2}} + \frac{1}{\sqrt{2\log d}}.\end{aligned}$$

This together with (27) completes the proof. ■

Proof of Lemma 4: From Theorem 1, we have

$$\mathcal{L}^*\Big(\|\vec{v}_{[1:\lambda_{c(\varepsilon)}]}\|_1 \log d\Big) \geq \bar{\Phi}\Big(\|\vec{v}_{[1+\lambda_{c(\varepsilon)}:d]}\|_2\Big) - \frac{1}{\log d}. \tag{33}$$

We have

$$\begin{aligned}\|\vec{v}_{[1+\lambda_{c(\varepsilon)}:d]}\|_2^2 &= \|\vec{v}\|_2^2 - \|\vec{v}_{[1:\lambda_{c(\varepsilon)}]}\|_2^2 \\ &= 1 - \|\vec{v}_{[1:\lambda_{c(\varepsilon)}]}\|_2^2 \\ &\leq 1 - c(\varepsilon)^2.\end{aligned}$$

where the last inequality uses the definition of $\lambda_{c(\varepsilon)}$ in (15). Using this back into (33), we get

$$\begin{aligned}\mathcal{L}^*\Big(\|\vec{v}_{[1:\lambda_{c(\varepsilon)}]}\|_1 \log d\Big) &\geq \bar{\Phi}\Big(\sqrt{1 - c(\varepsilon)^2}\Big) \\ &- \frac{1}{\log d} = \varepsilon - \frac{1}{\log d}.\end{aligned}$$

Note $\mathcal{L}^*(k)$ is nondecreasing in k, therefore this implies that $\mathcal{L}^*(k) \geq \varepsilon - 1/\log d$ for $k \geq \|\vec{v}_{[1:\lambda_{c(\varepsilon)}]}\|_1 \log d$ and completes the proof. ■

REFERENCES

[1] M. Beliaev, P. Delgosha, H. Hassani, and R. Pedarsani, "Efficient and robust classification for sparse attacks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 3150–3155.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA, 2012, pp. 1097–1105.

[3] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[4] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.

[5] D. Andor et al., "Globally normalized transition-based neural networks," 2016, *arXiv:1603.06042*.

[6] B. Biggio et al., "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2013, pp. 387–402. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40994-3_25

[7] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

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

[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2017, pp. 39–57.

[11] F. Croce and M. Hein, "Sparse and imperceivable adversarial attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4724–4732.

[12] A. Chaturvedi and U. Garain, "Mimic and fool: A task-agnostic adversarial attack," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1801–1808, Apr. 2021.

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

[14] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 274–283.

[15] Z. Marzi, S. Gopalakrishnan, U. Madhow, and R. Pedarsani, "Sparsity-based defense against adversarial attacks on linear classifiers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 31–35.

[16] Y. Bai, Y. Zeng, Y. Jiang, S.-T. Xia, X. Ma, and Y. Wang, "Improving adversarial robustness via channel-wise activation suppressing," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–19.

[17] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2016, pp. 2574–2582.

[18] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4322–4330.

[19] C. Xiao, P. Zhong, and C. Zheng, "Enhancing adversarial defense by k-winners-take-all," in *Proc. 8th Int. Conf. Learn. Represent.*, 2020, pp. 1–19.

[20] J. Lin, C. Gan, and S. Han, "Defensive Quantization: When efficiency meets robustness," *Artif. Intell., Commun., Imag., Navig., Sens. Syst.*, p. 8, 2019.

[21] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 1–8.

[22] A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "SparseFool: A few pixels make a big difference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9087–9096.

[23] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," 2018, *arXiv:1805.12152*.
[24] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, "Is robustness the cost of accuracy?—A comprehensive study on the robustness of 18 deep image classification models," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 631–648.
[25] A. Raghunathan, S. M. Xie, F. Yang, J. C. Duchi, and P. Liang, "Adversarial training can hurt generalization," 2019, *arXiv:1906.06032*.
[26] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7472–7482.
[27] S. Kanai, M. Yamada, H. Takahashi, Y. Yamanaka, and Y. Ida, "Relationship between nonsmoothness in adversarial training, constraints of attacks, and flatness in the input space," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 22, 2023, doi: 10.1109/TNNLS.2023.3244172.
[28] A. Javanmard, M. Soltanolkotabi, and H. Hassani, "Precise tradeoffs in adversarial training for linear regression," in *Proc. Conf. Learn. Theory*, 2020, pp. 2034–2078.
[29] B. Puranik, U. Madhow, and R. Pedarsani, "Generalized likelihood ratio test for adversarially robust hypothesis testing," *IEEE Trans. Signal Process.*, vol. 70, pp. 4124–4139, Aug. 2022.
[30] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Security Privacy*, 2016, pp. 372–387.
[31] L. Schott, J. Rauber, M. Bethge, and W. Brendel, "Towards the first adversarially robust neural network model on MNIST," 2018, *arXiv:1805.09190*.
[32] F. Croce, M. Andriushchenko, N. D. Singh, N. Flammarion, and M. Hein, "Sparse-RS: A versatile framework for query-efficient sparse black-box adversarial attacks," 2020, *arXiv:2006.12834*.
[33] A. Levine and S. Feizi, "Robustness certificates for sparse adversarial attacks by randomized ablation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 4585–4593.
[34] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is BERT really robust? Natural language attack on text classification and entailment," 2019, *arXiv:1907.11932*.
[35] J. Li, F. Schmidt, and Z. Kolter, "Adversarial camera stickers: A physical camera-based attack on deep learning systems," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3896–3904.
[36] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," 2016, *arXiv:1606.04435*.
[37] A. Shamir, I. Safran, E. Ronen, and O. Dunkelman, "A simple explanation for the existence of adversarial examples with small hamming distance," 2019, *arXiv:1901.10861*.
[38] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," 2016, *arXiv:1612.06299*.
[39] P. Delgosha, H. Hassani, and R. Pedarsani, "Robust classification under ℓ_0 attack for the Gaussian mixture model," *SIAM J. Math. Data Sci.*, vol. 4, no. 1, pp. 362–385, 2022. [Online]. Available: https://doi.org/10.1137/21M1426286
[40] Y. LeCun and C. Cortes. "MNIST handwritten digit database." 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[41] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10," Dataset, Canadian Institute for Advanced Research. [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html
[42] M. Beliaev. "robust-l0." May 2024. [Online]. Available: https://github.com/mbeliaev1/robust-l0
[43] P. J. Huber, *Robust Statistics*, vol. 523, Hoboken, NJ, USA: Wiley, 2004.
[44] K. Murphy, *Probabilistic Machine Learning: An Introduction* (Adaptive Computation and Machine Learning series). Cambridge, MA, USA: MIT, 2022. [Online]. Available: https://books.google.com/books?id=OyYuEAAAQBAJ
[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[46] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–44.
[47] D. Kang, Y. Sun, D. Hendrycks, T. Brown, and J. Steinhardt, "Testing robustness against unforeseen adversaries," 2019, *arXiv:1908.08016*.
[48] F. Croce and M. Hein, "Provable robustness against all adversarial l_p-perturbations for p≥1," in *Proc. ICLR*, 2020, pp. 1–22.
[49] F. Croce and M. Hein, "Adversarial robustness against multiple and single *l_p*-threat models via quick fine-tuning of robust classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 4436–4454.
[50] W. Wang, R. Wang, L. Wang, Z. Wang, and A. Ye, "Towards a robust deep neural network in texts: A survey," 2019, *arXiv:1902.07285*.