Avoiding the Accuracy-Robustness Trade-off of Classifiers via Local Adaptive Smoothing

Yatong Bai[1] **Brendon G. Anderson**[1] **Somayeh Sojoudi**[1,2]

[1]Department of Mechanical Engineering, University of California, Berkeley, California, USA
[2]Department of Electrical Engineering and Computer Science, University of California, Berkeley, California, USA

Abstract

While the literature has shown that an accurate and robust classifier should exist for common datasets, existing methods that improve the adversarial robustness of classifiers often manifest an accuracy-robustness trade-off. We build upon recent advancements in data-driven "locally biased smoothing" to develop an adaptive classifier that treats benign and adversarial test data differently. The adaptive model uses a neural network policy to adaptively mix two base classifiers, achieving the robustness of a robust classifier while suffering minor clean accuracy penalties compared with a standard network. In addition to numerical experiments, we provide theoretical analysis on the robustness of the adaptive model.

1 INTRODUCTION

The vulnerability of neural networks to adversarial attacks has been observed in various applications, such as computer vision [Moosavi-Dezfooli et al., 2016, Goodfellow et al., 2015] and control systems [Huang et al., 2017]. In response, "adversarial training" [Kurakin et al., 2017, Goodfellow et al., 2015, Bai et al., 2021b, 2022] has been studied to alleviate the susceptibility. Adversarial training builds robust neural networks by training on adversarial examples.

A parallel line of work focuses on certified robustness. There are a number of techniques that provide robustness certifications to existing neural networks [Anderson et al., 2020, Ma and Sojoudi, 2021, Anderson and Sojoudi, 2020], while "randomized smoothing" seeks to achieve certified robustness at test time [Cohen et al., 2019, Li et al., 2019]. The recent work [Anderson and Sojoudi, 2021] has shown that a locally biased smoothing method provides an improvement over traditional data-blind randomized smoothing. However, Anderson and Sojoudi [2021] only focus on binary classification problems, significantly limiting the applications. Moreover, the method has a fixed balance parameter between clean accuracy and adversarial robustness, and an accuracy-robustness trade-off thus limits its performance.

While some works have shown that there exists a fundamental trade-off between accuracy and robustness [Tsipras et al., 2019, Zhang et al., 2019], recent research has argued that it should be possible to simultaneously achieve robustness and accuracy on benchmark datasets [Yang et al., 2020]. To this end, variants of adversarial training that improve the accuracy-robustness trade-off have been proposed, including TRADE [Zhang et al., 2019], Interpolated Adversarial Training (IAT) [Lamb et al., 2019], and many others [Bai et al., 2021a, Raghunathan et al., 2020, Wang and Zhang, 2019, Zhang and Wang, 2019, Tramèr et al., 2018, Balaji et al., 2019]. However, even with these improvements, clean accuracy is often an inevitable price of achieving robustness.

This work makes a theoretically disciplined step towards performing robust classification without sacrificing clean accuracy. We first extend the locally biased smoothing method to multi-class classification problems in Section 3. Then, we observe that the performance of the K-nearest-neighbor (K-NN) classifier, a crucial component of locally biased smoothing, becomes a bottleneck of the overall performance. To this end, in Section 4, we replace the K-NN classifier with a robust neural network, which can be obtained via the methods discussed above. We also prove theoretical bounds on the certified robust radii of the developed methods. Finally, in Section 5, we propose a data-aware adaptive smoothing procedure that adaptively adjusts the mixture of a standard model and a robust model with the help of a neural network policy. Figure 1 presents the architecture of the resulted adaptive model. The model achieves a clean accuracy on par with the standard model and an attacked accuracy similar to the robust model, significantly improving the accuracy-robustness trade-off.

Previous research has developed models that improve robustness by dynamically changing at test time. Specifically,

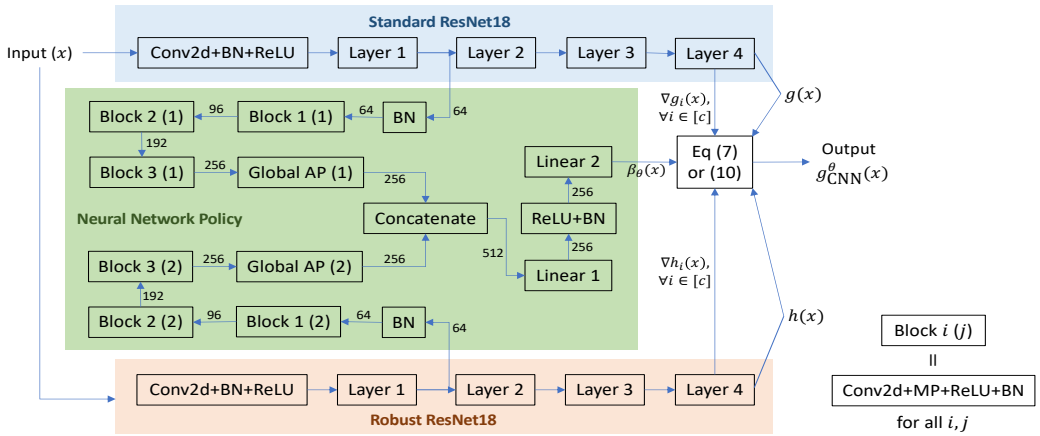


Figure 1: The overall architecture of the adaptive composite classifier introduced in Section 5 when applied to a pair of ResNet18 classifiers. "BN" represents 2D batch normalization, "MP" represents 2×2 2D max-pooling, and "AP" represents 2D average-pooling. All Conv2d layers in the neural network policy share the same filter size of 3×3.

"Input-Adaptive Inference (IAI)" improves the accuracy-robustness trade-off by appending side branches to a single network, allowing for early-exit predictions [Hu et al., 2020]. In comparison, this work considers two separate classifiers and uses their synergy to improve the accuracy-robustness trade-off, achieving much higher performances.

2 BACKGROUND

2.1 NOTATIONS

The projection onto a set $\mathcal{A}$ is denoted by $\Pi_{\mathcal{A}}(\cdot)$. The symbol $\|\cdot\|_p$ denotes the ℓ_p norm of a vector, while $\|\cdot\|_{p*}$ denotes its dual norm. The matrix I_d denotes the identity matrix in $\mathbb{R}^{d\times d}$. For a scalar a, $\text{sgn}(a) \in \{-1, 0, 1\}$ denotes its sign. For an event A, the indicator function $\mathbb{I}(A)$ evaluates to 1 if A takes place and 0 otherwise. The notation $\mathbb{P}_{X\sim\mathcal{S}}[A(X)]$ denotes the probability for an event $A(X)$ to occur, where X is a random variable drawn from the distribution $\mathcal{S}$. For a natural number c, we define $[c] = \{1, 2, \ldots, c\}$.

Consider a model $g : \mathbb{R}^d \to \mathbb{R}^c$, whose components are $g_i : \mathbb{R}^d \to \mathbb{R},\ i \in [c]$, where d is the dimension of the input and c is the number of classes. A classifier $f : \mathbb{R}^d \to \mathbb{R}$ can be obtained via $f(x) \in \arg\max_{i\in[c]} g_i(x)$, i. e., f outputs the class $i^\star$ such that $g_{i^\star}(x)$ is the largest among the c outputs of $g(\cdot)$. In this paper, we assume that $g(\cdot)$ does not have the desired level of robustness, and refer to it as a "standard classifier" (as opposed to a "robust classifier" which we denote as $h(\cdot)$). We denote the test dataset as $\mathcal{D}$, a set consisting of all input-label pairs (x_i, y_i) in the test set.

2.2 ADVERSARIAL ATTACKS AND ROBUSTNESS

In this work, we consider ℓ_p-norm-bounded attacks on differentiable neural networks. A classifier $f(\cdot)$ is considered robust against adversarial perturbations at an input data $x \in \mathbb{R}^d$ if it assigns the same label to all perturbed inputs $x + \delta$ such that $\|\delta\|_p \le \epsilon$, where $\epsilon \ge 0$ is the attack radius.

We assume that the adversary has access to the gradient $\nabla g_i(x)$ of the neural network and generates projected gradient descent (PGD) adversaries by performing the iterations

$$\widetilde{x}^{t+1} = \Pi_{\mathcal{X}}\left(\widetilde{x}^t + \gamma \cdot \text{sgn}\left(\nabla_x \ell_{\text{adv}}\big(g(x), y\big)\right)\right)$$

for $t = 0, 1, \cdots, T$, where $\widetilde{x}^t$ is the perturbed data vector at the t^{th} iteration, $\mathcal{X}$ is the perturbation set defined as $\{z : \|z - x\|_p \le \epsilon\}$, $\gamma > 0$ is the step size, ℓ_{adv} is a loss function (often the cross-entropy loss), and T is the total number of PGD steps. The initial step $\widetilde{x}^0$ is the clean input x. We use PGD_T to denote the attack that runs PGD for T steps.

2.3 LOCALLY BIASED SMOOTHING

Randomized smoothing achieves robustness at test time by replacing $f(x)$ with a smoothed classifier, given by $\widetilde{f}(x) \in \arg\max_{i\in[c]} \mathbb{P}_{\delta\sim\mathcal{S}}\big[f(x+\delta) = i\big]$, where $\mathcal{S}$ is a smoothing distribution. A common choice for $\mathcal{S}$ is a Gaussian distribution.

Anderson and Sojoudi [2021] have recently argued that data-invariant randomized smoothing does not always achieve robustness. They have shown that randomized smoothing with an unbiased distribution is suboptimal in the binary classification setting, and that an optimal smoothing procedure shifts the input point towards its true class. Since the true class is generally unavailable, a "direction oracle" is used as a surrogate. This "locally biased smoothing" method is no longer randomized and outperforms traditional data-blind randomized smoothing. The locally biased smoothed classifier $g^\mu(\cdot)$ is obtained via the calculation

$$g^\mu(x) = g(x) + \alpha h(x) \|\nabla g(x)\|_{p*},$$

where $h(x) \in \{1, -1\}$ is the direction oracle and $\alpha \geq 0$ is a trade-off parameter. Since locally biased smoothing aims to improve robustness, the direction oracle should come from an inherently robust classifier (which is often less accurate). In [Anderson and Sojoudi, 2021], this direction oracle is chosen to be a one-nearest-neighbor classifier. Intuitively, when $\|\nabla g(x)\|_{p*}$ is larger, $g(x)$ is more susceptible to adversarial attacks because perturbing the input by the same amount induces a larger output change. Thus, when $\|\nabla g(x)\|_{p*}$ is larger, we trust the direction oracle more.

3 MULTI-CLASS CLASSIFICATION WITH K-NEAREST-NEIGHBOUR ORACLE

When multi-class classification is considered, we extend the locally biased smoothing approach by treating the output of each class $g_i(x)$ independently, giving rise to:

$$g^{\alpha}_{K\text{-NN},i}(x) = g_i(x) + \alpha h_i(x) \|\nabla g_i(x)\|_{p*}, \quad \forall i \in [c]. \tag{1}$$

Note that (1) increases $g^{\alpha}_{K\text{-NN},i}(x)$ for those indices i whose $\|\nabla g_i(x)\|_{p*}$ is large, resulting in a classifier that incorrectly outputs the most vulnerable class rather than the predicted class. To correct this bias, we normalize the result and use the following smoothed classifier as a surrogate:

$$g^{\alpha}_{K\text{-NN},i}(x) = \frac{g_i(x) + \alpha h_i(x) \|\nabla g_i(x)\|_{p*}}{1 + \alpha \|\nabla g_i(x)\|_{p*}}, \quad \forall i \in [c]. \tag{2}$$

The hyperparameter α adjusts the trade-off between clean accuracy and robustness. When $\alpha = 0$, it holds that $g^{\alpha}_{K\text{-NN},i}(x) \equiv g_i(x)$ for all i. When $\alpha \to \infty$, it holds that $g^{\alpha}_{K\text{-NN},i}(x) \to h_i(x)$ for all x and all i.

K-NN classifiers are generally robust against adversarial inputs targeting an independent neural network. The experiments in Subsection 6.1 provide empirical evidence, and the literature has provided theoretical analyses. For example, for binary classification problems, it has been shown that K-NN classifiers are robust if $K = 1$ and the training dataset is separable [Anderson and Sojoudi, 2021], or if $K = \Omega(\sqrt{dn \log n})$ where n is the training dataset size [Wang et al., 2018]. Therefore, a scaled K-NN classifier can be used as the direction oracle $h(\cdot)$:

$$h_i(x) = 2 \cdot \left(\tfrac{1}{K} \textstyle\sum_{k=1}^{K} \mathbb{I}(y_k = i) \right) - 1, \quad \forall i \in [c],$$

where $y_1, \cdots, y_K \in [c]$ are the true classes of the K training points with the smallest ℓ_2 distances to x.

3.1 THEORETICAL CERTIFIED ROBUST RADIUS

Consider two arbitrary classes, namely class i and class j. Consider an arbitrary unperturbed input $x \in \mathbb{R}^d$ and a radius $\epsilon > 0$. Suppose that $g(\cdot)$ is differentiable and has bounded gradients at all $x+\delta$ such that $\|\delta\|_p \leq \epsilon$. For each $k \in \{i, j\}$, we define the constant M_k to be $\sup_{\|\delta\|_p \leq \epsilon} \|\nabla g_k(x+\delta)\|_{p*}$. Moreover, we define the constant L_k as the smallest number such that

$$\frac{\|\nabla g_k(x') - \nabla g_k(x+\delta)\|_{p*}}{1 + \alpha \|\nabla g_k(x+\delta)\|_{p*}} \leq L_k \epsilon \tag{3}$$

for all δ such that $\|\delta\|_p \leq \epsilon$ and all x' on the line segment between x and $x + \delta$. The existence of a finite L_k is guaranteed due to the bounded gradient assumption. We further define the constant

$$b := \sum_{k \in \{i,j\}} \left(\frac{\alpha L_k |g_k(x)|}{1 + \alpha \|\nabla g_k(x)\|_{p*}} + 2\alpha L_k + \frac{M_k}{1 + M_k \alpha} \right).$$

Theorem 1. *Suppose that for all δ with the property $\|\delta\|_p \leq \epsilon$, it holds that*

$$h_i(x+\delta) = h_i(x), \quad h_j(x+\delta) = h_j(x).$$

If ϵ satisfies that

$$\epsilon \leq \frac{\sqrt{b^2 + 4(L_i + L_j)\left|g^{\alpha}_{K\text{-NN},i}(x) - g^{\alpha}_{K\text{-NN},j}(x)\right|} - b}{2(L_i + L_j)}, \tag{4}$$

then, it holds that

$$\operatorname{sgn}\left(g^{\alpha}_{K\text{-NN},i}(x+\delta) - g^{\alpha}_{K\text{-NN},j}(x+\delta)\right) = \operatorname{sgn}\left(g^{\alpha}_{K\text{-NN},i}(x) - g^{\alpha}_{K\text{-NN},j}(x)\right)$$

for all $\delta \in \mathbb{R}^d$ satisfying $\|\delta\|_p \leq \epsilon$.

The proof is provided in Appendix B.1. Theorem 1 implies that under mild assumptions, when ϵ is small and the K-NN model $h(x)$ is robust, the classification between class i and class j is the same for x and all $x + \delta$ such that $\|\delta\|_p \leq \epsilon$.

4 USING A ROBUST NEURAL NETWORK FOR THE SMOOTHING ORACLE

While K-NN classifiers are relatively robust and can be used as the direction oracle to improve the robustness of neural networks, K-NN only performs well on relatively easy tasks. For common image classification tasks, K-NN suffers from its insufficient representation power. As shown in Figure 5, on the CIFAR-10 image classification problem [Krizhevsky, 2012], K-NN only achieves around 35% accuracy on clean test data. In contrast, an adversarially trained ResNet model can reach a 50.0% accuracy on PGD adversarial test data [Madry et al., 2018], higher than the clean accuracy of K-NN. Such a lackluster performance of K-NN becomes a significant bottleneck of the accuracy-robustness trade-off of the smoothed classifier. To this end, we replace the K-NN classifier with a robust neural network. The robustness of this network can be achieved via various methods, including

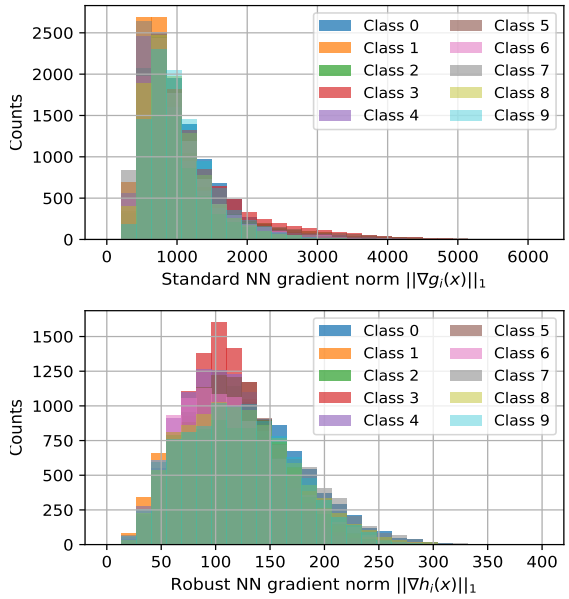


Figure 2: The distributions of $\|\nabla g_i(x)\|_1$ (upper) and $\|\nabla h_i(x)\|_1$ (lower) for each $i \in [10]$.

(the original) adversarial training, TRADE, and traditional randomized smoothing.

Specifically, we use the logits (before softmax) produced by a robust neural network as the oracle $h_i(\cdot)$. Since the function $h_i(\cdot)$ is differentiable, its gradient should also be utilized to determine the "trustworthiness" of the base classifier. Thus, we modify the smoothed classifier as:

$$g^{\alpha}_{\text{CNN},i}(x) = \frac{g_i(x) + \alpha R_i(x) h_i(x)}{1 + \alpha R_i(x)}, \quad \forall i \in [c] \tag{5}$$
$$\text{where } R_i(x) = \frac{\|\nabla g_i(x)\|_{p*}}{\|\nabla h_i(x)\|_{p*}}.$$

To avoid numerical issues, the denominator of $R_i(x)$ can be replaced with $\|\nabla h_i(x)\|_{p*} + \epsilon_c$ in practice, where ϵ_c is a small positive number. If $h_i(x)$, $\|\nabla h_i(x)\|_{p*}$, and $\|\nabla g_i(x)\|_{p*}$ are all bounded, then when $\alpha \to \infty$, it holds that $g^{\alpha}_{\text{CNN},i}(x) \to h_i(x)$ for all i and x. To simplify the analysis, we define $g^{\alpha}_{\text{CNN},i}(x)\big|_{\alpha=\infty} = h_i(x)$ for all x and i.

To show that the proposed smoothing procedure is different from naively combining the logits of a standard model and a robust model, we treat each $R_i(x)$ as a random variable and empirically demonstrate that the distribution of each $R_i(x)$ has a non-trivial variance. We use the ℓ_∞ attack as an example, and analyze the distributions of $\|\nabla g_i(x)\|_1$ and $\|\nabla h_i(x)\|_1$. Figure 2 demonstrates the variances of $\|\nabla g_i(x)\|_1$ and $\|\nabla h_i(x)\|_1$, and Figure 3 shows that the correlation between each $\|\nabla g_i(x)\|_1$ and each $\|\nabla g_i(x)\|_1$ is low. Therefore, $R_i(x)$ can be quite different at each x, and is thus a useful criteria of dynamically adjusting the smoothing strength. It can also be observed that $\|\nabla g_i(x)\|_1$

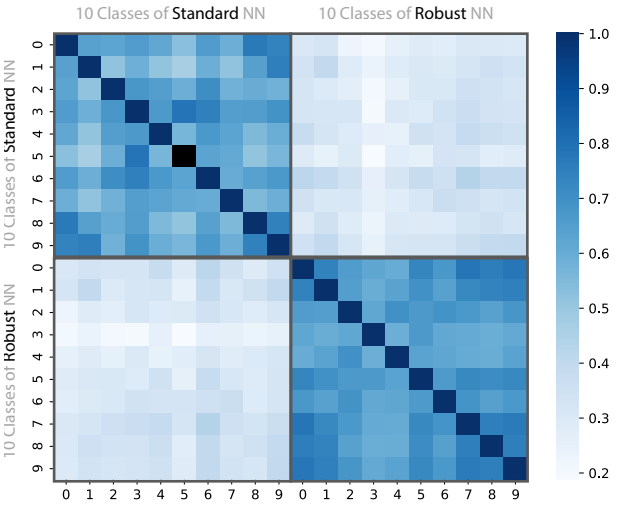


Figure 3: Correlation matrix of all $\|\nabla g_i(x)\|_1$ and all $\|\nabla h_i(x)\|_1$ where $i \in [10]$.

is usually greater than $\|\nabla h_i(x)\|_1$, supporting the inference that robustness and gradient magnitude are connected. The distributions of $\|\nabla h_i(x)\|_1$ and $\|\nabla g_i(x)\|_1$ with adversarial input data are shown in Appendix A.4.

4.1 THEORETICAL CERTIFIED ROBUST RADIUS

The certified radius is now generalized to arbitrary robust direction oracles. Again, let $i, j \in [c]$ and $\epsilon > 0$, and fix an arbitrary input $x \in \mathbb{R}^d$. For $k \in \{i, j\}$, assume that $h_k(\cdot)$ has nonzero bounded Lipschitz gradient on $B_p(x, \epsilon) := \{x' \in \mathbb{R}^d : \|x' - x\|_p \le \epsilon\}$, and define the constants

$$K^h_k = \sup_{x' \in B_p(x,\epsilon)} |h_k(x')|,$$
$$m^h_k = \inf_{x' \in B_p(x,\epsilon)} \|\nabla h_k(x')\|_{p*} > 0,$$
$$M^h_k = \sup_{x' \in B_p(x,\epsilon)} \|\nabla h_k(x')\|_{p*},$$
$$L^h_k = \sup_{x' \in B_p(x,\epsilon)} \frac{\|\nabla h_k(x') - \nabla h_k(x)\|_{p*}}{\|x' - x\|_p}.$$

Note that the bounded gradient assumption is always satisfied for continuously differentiable classifiers. We make analogous assumptions and definitions for $g_k(\cdot)$ with $k \in \{i, j\}$, and define

$$C^{(1)}_k(x, \alpha) = \frac{M^g_k \left(1 + \alpha \frac{M^g_k}{m^h_k}\right) + \alpha \frac{K^g_k}{m^h_k} \left(L^h_k R_k(x) + L^g_k\right)}{\left(1 + \alpha \frac{m^g_k}{M^h_k}\right)(1 + \alpha R_k(x))},$$
$$C^{(2)}_k(x, \alpha) = \frac{\frac{M^g_k M^h_k}{m^h_k}(1 + \alpha R_k(x)) + \frac{K^h_k}{m^h_k}\left(L^h_k R_k(x) + L^g_k\right)}{\left(1 + \alpha \frac{m^g_k}{M^h_k}\right)(1 + \alpha R_k(x))}.$$

The certified radius is now presented.

Theorem 2. *If* $\epsilon \leq \frac{|g^{\alpha}_{CNN,i}(x) - g^{\alpha}_{CNN,j}(x)|}{\sum_{k\in\{i,j\}} \left(C^{(1)}_k(x,\alpha) + \alpha C^{(2)}_k(x,\alpha)\right)}$, *then*

$$\operatorname{sgn}\left(g^{\alpha}_{CNN,i}(x+\delta) - g^{\alpha}_{CNN,j}(x+\delta)\right) = \operatorname{sgn}\left(g^{\alpha}_{CNN,i}(x) - g^{\alpha}_{CNN,j}(x)\right)$$

for all $\delta \in \mathbb{R}^d$ *satisfying* $\|\delta\|_p \leq \epsilon$.

Theorem 2 is proved in Appendix B.2.

4.2 ATTACKING THE COMPOSITE CLASSIFIER

Since the composite classifier $g^{\alpha}_{\text{CNN}}(\cdot)$ is differentiable, the adversary can adapt to the smoothing procedure. We assume that the adversary chooses a target $\alpha_t \geq 0$ and generates adversarial examples by running the iterations below:

$$\widetilde{x}^{t+1} = \Pi_{\mathcal{X}}\left(\widetilde{x}^t + \gamma \cdot \operatorname{sgn}\left(\nabla_x \ell_{\text{adv}}\left(g^{\alpha_t}_{\text{CNN}}(x), y\right)\right)\right). \quad (6)$$

In general, α_t may be different from the true α, but setting α_t to α should give the worst-case attack. When α_t is set to 0, the adversary is equivalent to the PGD attack targeting $g(\cdot)$. When α_t is set to ∞, the adversary attacks $h(\cdot)$. Note that $R_i(x)$ is a function of x, providing an additional gradient path to the adversary.

5 ADAPTIVE SMOOTHING STRENGTH WITH NEURAL NETWORK POLICY

So far, α has been treated as a fixed hyperparameter. As a result, the accuracy-robustness trade-off between $g(\cdot)$ and $h(\cdot)$ limits the performance of the composite classifier. At each data point x, the model emphasizes either the standard model $g_i(x)$ or the robust model $h_i(x)$. A more intelligent and effective approach is to allow α to be different for each x by replacing the constant α with a function $\alpha(x)$.

One motivation for adopting the adaptive trade-off parameter $\alpha(x)$ is that the optimal $\alpha^\star$ can vary when x changes. For example, when x is clean and unperturbed, the standard model $g(\cdot)$ outperforms the robust model $h(\cdot)$. If x is a PGD-attacked input targeting $g(\cdot)$, then the robust model $h(\cdot)$ should be used. However, as shown in Figure 4, if the target of the attack is $h(\cdot)$, then even though $h(\cdot)$ is robust, a better choice is to feed x into $g(\cdot)$. This is because the loss landscapes of $g(\cdot)$ and $h(\cdot)$ differ enough that an adversarial perturbation targeting $h(\cdot)$ is benign to $g(\cdot)$.

When the PGD adversary targets a smoothed classifier $g^{\alpha_t}_{\text{CNN}}(\cdot)$, as α_t varies, the optimal strategy also changes. Figure 4 is obtained with the CIFAR-10 dataset, a ResNet18 standard classifier $g(\cdot)$, and a ResNet18 robust classifier $h(\cdot)$[1]. When $\alpha_t \leq 55$, the robust model $h(\cdot)$ attains a higher accuracy. When $\alpha_t > 55$, the standard model $g(\cdot)$ becomes more suitable.

[1]The ResNet classifiers are pretrained models from Na [2020].

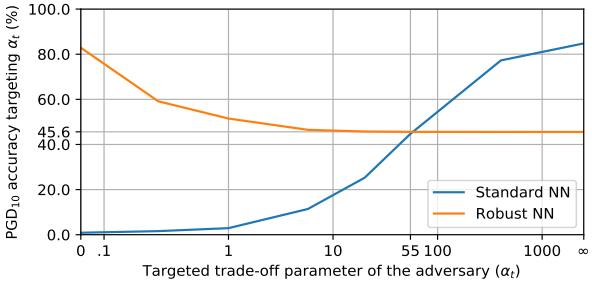


Figure 4: Attacked accuracy of the standard classifier $g(\cdot)$ and the robust classifier $h(\cdot)$ when the adversary targets different values of α_t.

Theorem 3. *Assume that there does not exist* $z \in \mathbb{R}^d$ *such that* $\|z - x_1\|_p \leq \epsilon$ *and* $\|z - x_2\|_p \leq \epsilon$, *where* $(x_1, y_1) \in \mathcal{D}$, $(x_2, y_2) \in \mathcal{D}$. *Also, assume that* $h_i(x)$, $\|\nabla h_i(x)\|_{p*}$, *and* $\|\nabla g_i(x)\|_{p*}$ *are all bounded.*

Then, there exists a function $\alpha(x)$ *such that the assembled classifier* $g^{\alpha}_{CNN}(x)$ *satisfies that*

$$\mathbb{P}_{(x,y)\sim\mathcal{D},\delta\sim\mathcal{F}}\left[\arg\max_{i\in[c]} g^{\alpha}_{CNN,i}(x+\delta) = y\right] \geq \max\left\{\begin{matrix}\mathbb{P}_{(x,y)\sim\mathcal{D},\delta\sim\mathcal{F}}\left[\arg\max_{i\in[c]} g_i(x+\delta) = y\right], \\ \mathbb{P}_{(x,y)\sim\mathcal{D},\delta\sim\mathcal{F}}\left[\arg\max_{i\in[c]} h_i(x+\delta) = y\right]\end{matrix}\right\},$$

where $\mathcal{F}$ *is any distribution such that* $\mathbb{P}_{\delta\sim\mathcal{F}}\left[\|\delta\|_p > \epsilon\right] = 0$.

The proof of Theorem 3 is shown in Appendix B.3. Note that when the probability density function (PDF) of $\mathcal{F}$ is a dirac delta at zero, Theorem 3 implies that the clean accuracy of $g^{\alpha}_{\text{CNN}}(\cdot)$ is as good as the standard classifier $g(\cdot)$. When the PDF of $\mathcal{F}$ is a dirac delta at the worst-case perturbation, Theorem 3 shows that the adversarial accuracy of $g^{\alpha}_{\text{CNN}}(\cdot)$ is not worse than the robust model $h(\cdot)$. Thus, $g^{\alpha}_{\text{CNN}}(\cdot)$ avoids the accuracy-robustness trade-off.

Since the true label y is unknown for the test data, we use a neural network policy $\alpha_\theta(x) : \mathbb{R}^d \to \mathbb{R}$ to learn the optimal strategy. Here, θ represents the trainable parameters of the policy. Note that when α becomes smaller, the trade-off in (5) becomes more sensitive to α. Therefore, we re-parameterize $\alpha_\theta(x)$ as $\exp\left(\beta_\theta(x)\right)$. The adaptive composite classifier is then

$$g^{\theta}_{\text{CNN},i}(x) = \frac{g_i(x) + \exp\left(\beta_\theta(x)\right) R_i(x) h_i(x)}{1 + \exp\left(\beta_\theta(x)\right) R_i(x)}, \quad \forall i \in [c]$$
$$\text{where } R_i(x) = \frac{\|\nabla g_i(x)\|_{p*}}{\|\nabla h_i(x)\|_{p*}}. \quad (7)$$

Since the forward pass of $g^{\theta}_{\text{CNN}}(x)$ calculates the Jacobians of $g(\cdot)$ and $h(\cdot)$, the time complexity is linear in c. In Appendix A.1, we show that an accelerated variant with constant complexity is almost as accurate and robust as (7). The variant predicts about half as fast as the standalone $g(\cdot)$.

5.1 ATTACKING THE ADAPTIVE CLASSIFIER

When $g^{\theta}_{\text{CNN}}(\cdot)$ is considered, since $\beta_\theta(x)$ is also a function of x, the gradient can flow through β_θ as well. The introduction of the additional gradient path could perplex the loss landscape for the adversary, because throughout the PGD iterations, $\beta_\theta(\widetilde{x}^0), \ldots, \beta_\theta(\widetilde{x}^T)$ can differ.

When the adversary performs white-box attack and has complete information about $g^{\theta}_{\text{CNN}}(\cdot)$, it can still choose a particular α_t, instead of back-propagating through the policy. We will consider both adversaries (with or without the gradient through the policy) in the experiment section.

5.2 NEURAL NETWORK POLICY ARCHITECTURE

The neural network policy $\beta_\theta(\cdot)$ should treat clean and attacked inputs differently, and a natural question is whether a neural network is capable of detecting adversarial attacks. The answer is affirmative, and the literature has proposed various methods for the detection task [Metzen et al., 2017, Carrara et al., 2019, Ahmadi et al., 2021]. Metzen et al. [2017] use a separate neural network to predict the nature of the input, demonstrating a high success rate. We adapt their detection architecture for our neural network policy by taking advantage of the two available models $g(\cdot)$ and $h(\cdot)$. The modified architecture is shown in Figure 1. Our task of estimating the optimal α can be more challenging than detecting adversaries, as our policy should be more flexible and need to differentiate attacks targeting different α_t's.

5.3 TRAINING THE POLICY

Consider the following two methods that aim to train the policy $\beta_\theta(\cdot)$:

- **Direct back-propagation:** Directly optimize the loss function via the following optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D},\delta\sim\mathcal{F}}\Big[\ell_{\text{CE}}\big(g^{\theta}_{\text{CNN}}(x+\delta), y\big)\Big], \qquad (8)$$

  where ℓ_{CE} is the cross-entropy loss for logits and $y \in [c]$ is the label corresponding to x. This optimization can be directly solved via gradient descent back-propagation because $g^{\theta}_{\text{CNN}}(\cdot)$ is differentiable. Note that the base classifiers $g(\cdot)$ and $h(\cdot)$ are not updated.

- **Supervised learning:** The optimal $\alpha^\star$ that minimizes ℓ_{CE} in (8) can be estimated for each training point. Thus, $\beta_\theta(\cdot)$ can be optimized via supervised learning:

$$\min_{\theta} \mathbb{E}_{\substack{(x,y)\sim\mathcal{D}\\ \delta\sim\mathcal{F}}}\Big[\ell_{\text{supervise}}\big(\beta_\theta(x+\delta), \log\alpha^\star(x+\delta)\big)\Big],$$

  where $\ell_{\text{supervise}}$ is a supervised learning loss function. While it is possible to estimate the scalar $\alpha^\star(x)$ with the Monte Carlo method, our experiments show that

$$\widetilde{\alpha}(x+\delta) = \begin{cases} +\infty & \text{if } \alpha_t \le \alpha_0, \\ 0 & \text{otherwise,} \end{cases}$$

  approximates $\alpha^\star(x)$ sufficiently well, where α_t is the target parameter of the PGD adversary that generates $x+\delta$. For the CIFAR-10 dataset and the ResNet18 base classifiers used in this paper, a good choice for α_0 is 55, i. e., the intersection of the two lines in Figure 4.

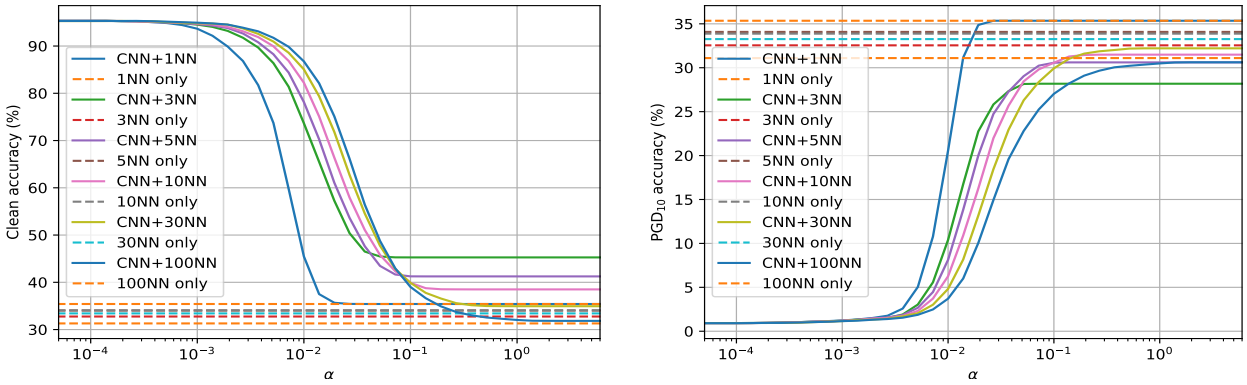
The direct back-propagation method suffers from a distributional mismatch between the training set and the test set. When the robust classifier $h(\cdot)$ is optimized via adversarial training, $h(\cdot)$ achieves a low loss on adversarial training data but a high loss on adversarial test data. For example, with the CIFAR-10 dataset and our ResNet18 robust classifier, the PGD_{10} adversarial training accuracy is 93.01% while the PGD_{10} test accuracy is 45.55%. As a result, approximating (8) with empirical risk minimization on the training set does not effectively optimize the true risk. For example, when the adversary attacks the input x targeting the robust model $h(\cdot)$, the standard prediction $g(x)$ yields a lower loss than $h(x)$. However, if x is an attacked example in the training set, then the losses of $g(x)$ and $h(x)$ are similar, and the neural network policy does not receive a strong incentive to choose $g(\cdot)$ when it detects an attack targeting $h(\cdot)$. As a result, performing direct back-propagation on the training set does not generalize well to the test set. Note that the policy can still recognize attacks that target $h(\cdot)$, but does not learn to avoid using $h(\cdot)$ for these inputs.

The supervised learning approach also possesses limitations, as the training labels can sometimes be difficult to determine. Since the composite model g^{θ}_{CNN} is differentiable, the adversary can adapt to the neural network policy $\beta_\theta(\cdot)$. While it is relatively easy to provide labels for attacks targeting $g^{\alpha_t}_{\text{CNN}}(\cdot)$ with a fixed α_t by referencing Figure 4, the optimal α for an adaptive attack targeting g^{θ}_{CNN} may not be clear.

For the above reasons, we propose a loss function that combines the above two approaches, providing incentives for the policy to choose the standard classifier $g(\cdot)$ when appropriate, while forcing the policy to remain conservative when facing adaptive attacks. The composite loss function for each data-label pair (x, y) is given by:

$$\ell_{\text{composite}}\big(\theta, (x,y)\big) = \Big(\ell_{\text{CE}}\big(g^{\theta}_{\text{CNN}}(x), y\big) + c_1\Big) \times \Big(c_2 + \big(c_3 - \text{sgn}(\log\alpha^\star(x) - \beta_0)\beta_\theta(x)\big)_+\Big), \qquad (9)$$

where the first group of terms is the direct loss component, the second group of terms is the supervised loss component, and $\beta_0 = \log\alpha_0$ is a threshold constant that determines the targets for $\beta_\theta(\cdot)$. The hyperparameters c_1, c_2, and c_3 control the trade-off between the two components. Intuitively, the goal is to decrease $\beta_\theta(\cdot)$ and prefer $g(\cdot)$ when $\alpha_t < \exp(\beta_0)$, and increase $\beta_\theta(\cdot)$ and prefer $h(\cdot)$ when $\alpha_t > \exp(\beta_0)$.



(a) Clean accuracy for each α for various values of K values. (b) PGD_{10} adversarial accuracy for each α for various values of K.

Figure 5: The performance of the smoothed neural network based on K-NN classifiers. The attack radius is $\epsilon = 0.031$.

6 NUMERICAL EXPERIMENTS

6.1 MULTI-CLASS K-NN SMOOTHING

We use the CIFAR-10 dataset to evaluate the performance of the smoothing procedure based on K-NN classifiers and analyze the effect of the trade-off parameter α on $g^{\alpha}_{K\text{-NN}}(\cdot)$. We use a ResNet18 model trained on clean data as the standard classifier $g(\cdot)$. For the K-NN classifier[2] used as the robust direction oracle $h(\cdot)$, we explore different values of K ranging from 1 to 100. The accuracy for the above K values at various choices of α is shown in Figure 5.

Figures 5 confirms that the K-NN classifiers are robust against adversaries targeting a neural network, as the clean and adversarial accuracies of a K-NN model are almost identical. Moreover, the figures verify that when α goes to zero, the smoothed neural network $g^{\alpha}_{K\text{-NN}}(\cdot)$ converges to the unsmoothed neural network $g(\cdot)$. When α is large, the smoothed neural network outperforms the K-NN classfier on clean data. The reason for this gap is that a K-NN classifier may result in ties (i. e., the K closest neighbors are from K different classes), which are then resolved when the neural network logits are added upon the K-NN results. As a result, the smoothed neural network takes advantage of the high clean accuracy of $g(\cdot)$. Since the clean and adversarial data result in different neural network outputs, the K-NN ties are resolved differently. In Appendix A.3, we repeat the experiment using distance-weighted K-NN classifiers that usually do not produce ties and show that the performance gaps vanish for large values of α.

6.2 ROBUST NEURAL NETWORK SMOOTHING WITH A FIXED STRENGTH

Similarly, we use the CIFAR-10 dataset to evaluate the performance of the composite models $g^{\alpha}_{\text{CNN}}(\cdot)$ with different fixed values of α. Specifically, we use a ResNet18 model trained on clean data as the standard model $g(\cdot)$ and use another ResNet18 trained on PGD_{10} data as the robust model $h(\cdot)$. We verify the performance of $g^{\alpha}_{\text{CNN}}(\cdot)$ with various settings for α. We consider PGD_{10} attacks that target $g(\cdot)$ and $h(\cdot)$, in addition to the adaptive PGD_{10} attacks targeting $g^{\alpha_t}_{\text{CNN}}$ produced with (6), where α_t is set to the true α of the corresponding model.

The test accuracy of each composite model is presented in Figure 6. As α increases, the clean accuracy of $g^{\alpha}_{\text{CNN}}(\cdot)$ converges from the clean accuracy of $g(\cdot)$ to the clean accuracy of $h(\cdot)$. In terms of the attacked performance, when the attack targets $g(\cdot)$, the adversarial accuracy increases with α. When the attack targets $h(\cdot)$, the adversarial accuracy decreases with α, showing that the attack becomes more benign to the composite model when α decreases (emphasizing $g(\cdot)$ more). When the adaptive attack targets $g^{\alpha}_{\text{CNN}}(\cdot)$, the adversarial accuracy increases with α.

6.3 ROBUST NEURAL NETWORK SMOOTHING WITH ADAPTIVE STRENGTH

Finally, we evaluate the performance of the adaptive composite model $g^{\theta}_{\text{CNN}}(\cdot)$ using the CIFAR-10 and the CIFAR-100 datasets. We consider ℓ_2 and ℓ_∞ attacks and use different robust neural networks for $h(\cdot)$. Specifically, the training inputs for the policy $\beta_\theta(\cdot)$ includes clean data, PGD_{10} adversarial data with $\alpha_t \in \{0, .316, 1.46, 6.81, 31.6, 147, 681, 3162, \infty\}$, and adaptive PGD_{10} data directly targeting $g^{\theta}_{\text{CNN}}(\cdot)$ with a random number of steps. For the adaptive PGD_{10} data, we only consider ℓ_{CE} in (8). For attacked data with fixed α_t settings, we use the composite loss function (9) with the parameters $c_1 = 0.5$, $c_2 = 0.8$, and $c_3 = 3.2$. The values of β_0 are reported in Table 1. Since the adaptive model $g^{\theta}_{\text{CNN}}(\cdot)$ requires exponentiating the policy output $\beta_\theta(\cdot)$, the output is cropped to $[-14, 14]$ to avoid potential numerical issues. The Adam optimizer [Kingma and Ba, 2015] is used to train the policy.

[2]The K-NN classifiers use images standardized per-channel.

Dataset	Type (ϵ)	$g(\cdot)$			$h(\cdot)$			$g^{\theta}_{\text{CNN}}(\cdot)$			
		Architecture	Clean	PGD_{20}	Architecture	Clean	PGD_{20}	β_0	Clean	PGD_{20}	PGD_{100}
CIFAR-10	ℓ_∞ (.031)	ResNet18†	95.28%	0.12%	AT ResNet18†	83.53%	45.15%	4	94.92%	45.94%	45.05%
CIFAR-10	ℓ_2 (.5)	ResNet18†	95.28%	2.57%	AT ResNet18	85.37%	64.92%	2.8	93.53%	64.55%	64.17%
CIFAR-10	ℓ_∞ (.031)	ResNet18†	95.28%	0.12%	TRADE WRN34‡	84.92%	57.16%	5.7	95.04%	57.93%	56.60%
CIFAR-100	ℓ_∞ (.031)	Mixup PRN18	78.46%	0.00%	AT ResNet18	50.37%	25.71%	3.8	78.01%	24.45%	24.32%

Dataset	Type (ϵ)	Work	Architecture	Clean Accuracy	PGD_T Accuracy (T)
CIFAR-10	ℓ_∞ (.031)	IAI [Hu et al., 2020]	ResNet38	83.62%	42.29% (20)
CIFAR-10	ℓ_∞ (.031)	IAT [Lamb et al., 2019]	Mixup PRN18	89.88%	38.38% (20)
CIFAR-100	ℓ_∞ (.031)	IAAT [Balaji et al., 2019]	ResNet18	63.90%	18.50% (10)

†: From [Na, 2020]. ‡: From [Zhang et al., 2019].

Table 1: Summary of test accuracy achieved by the adaptive smoothing classifier $g^{\theta}_{\text{CNN}}(\cdot)$. WRN stands for WideResNet, PRN stands for PreActResNet, and AT stands for Adversarial Training.

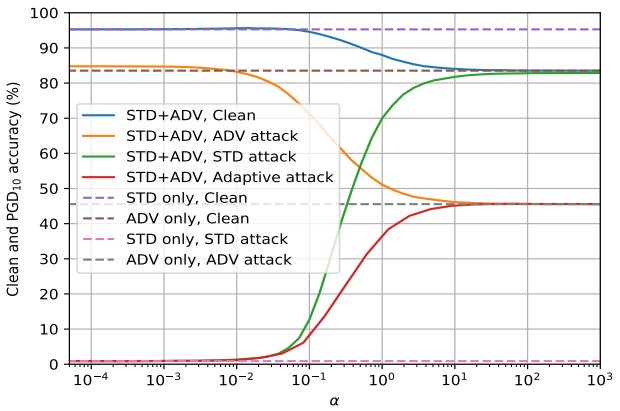


Figure 6: The performance of the composite model $g^{\alpha}_{\text{CNN}}(\cdot)$. All attacks are PGD_{10} with ϵ set to 0.031.

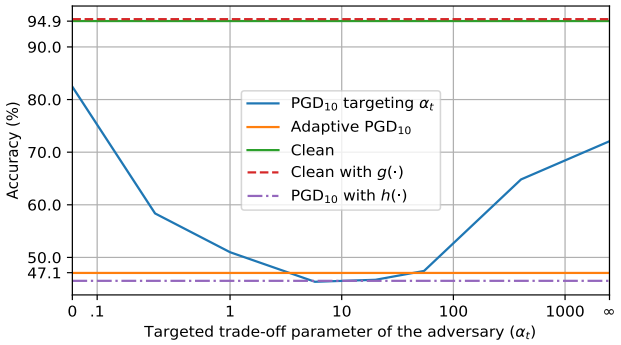


Figure 7: Clean and PGD_{10} adversarial accuracies with different target α_t's. The solid lines are the results of $g^{\theta}_{\text{CNN}}(\cdot)$.

The test accuracy of $g^{\theta}_{\text{CNN}}(\cdot)$ in each setting is shown in Table 1. In each case, g^{θ}_{CNN} achieves a clean accuracy close to the standard classifier $g(\cdot)$ and an adversarial accuracy similar to the robust model $h(\cdot)$. The above results show that the neural network policy $\beta_\theta(\cdot)$ is capable of approximating the optimal value of $\alpha^\star$ for each type of input data. The fact that $g^{\theta}_{\text{CNN}}(\cdot)$ combines the clean accuracy of $g(\cdot)$ and the robustness $h(\cdot)$ shows that $g^{\theta}_{\text{CNN}}(\cdot)$ largely avoids the accuracy-robustness trade-off. In the table, we also compare our method to the reported results of the existing works aiming to improve this trade-off. Moreover, in Appendix A.2, we show that the adaptive network trained on attacks with $\epsilon = 0.031$ does not fall apart when facing other attack radii.

Figure 7 compares the PGD_{10} attack that allows gradient to flow through $\beta_\theta(\cdot)$ and the attacks that assume a fixed value for α_t. The adaptive model from the first row of Table 1 is used. The adversary targeting $\alpha_t = 20$ provides the most powerful attack, decreasing the accuracy to 45.38%, lower than the accuracy on adaptive PGD data, but only 0.17% worse than the PGD_{10} accuracy of the robust model $h(\cdot)$. Figure 7 confirms the speculation that the composite structure of $g^{\theta}_{\text{CNN}}(\cdot)$ perplexes the optimization landscape for the PGD adversary, and can make it harder to generate worst-case perturbations. Note that the main source of the robustness of g^{θ}_{CNN} is the inherent robustness of $h(\cdot)$, rather than the sophisticated loss landscape.

Since the proposed method does not make assumptions on the implementation details of the two base models, previous advancements in the field of robustness can be incorporated into our framework. If a different type of attack needs to be considered, the training set for $\beta_\theta(\cdot)$ can be augmented with the corresponding adversarial data.

7 CONCLUSIONS

This paper proposes a flexible framework that significantly improves the accuracy-robustness trade-off of neural networks by adaptively adjusting the mixture of the outputs of an accurate model and a robust model. The developed theoretical analysis proves the existence of an accurate and robust classifier built via the proposed approach. Furthermore, certified robustness radii are proved for the cases of using a fixed smoothing strength α. Solid empirical results show that our method can simultaneously attain the high accuracy of modern standard models and the robustness achieved via state-of-the-art robust classification methods. This work allows future research to specialize on accuracy or robustness without fearing to compromise the other.

References

Morteza Ali Ahmadi, Rouhollah Dianat, and Hossein Amirkhani. An adversarial attack detection method in deep neural networks based on re-attacking approach. *Multimedia Tools and Applications*, 80(7):10985–11014, 2021.

Brendon Anderson and Somayeh Sojoudi. Certified robustness via locally biased randomized smoothing, 2021. URL `https://brendon-anderson.github.io/files/publications/anderson2021certified-long.pdf`.

Brendon Anderson, Ziye Ma, Jingqi Li, and Somayeh Sojoudi. Tightened convex relaxations for neural network robustness certification. In *IEEE Conference on Decision and Control*, 2020.

Brendon G. Anderson and Somayeh Sojoudi. Data-driven assessment of deep neural networks with random input uncertainty. *arXiv preprint arXiv:2010.01171*, 2020.

Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. In *International Joint Conference on Artificial Intelligence*, 2021a.

Yatong Bai, Tanmay Gautam, Yu Gai, and Somayeh Sojoudi. Practical convex formulation of robust one-hidden-layer neural network training. *arXiv preprint arXiv:2105.12237*, 2021b.

Yatong Bai, Tanmay Gautam, and Somayeh Sojoudi. Efficient global optimization of two-layer relu networks: Quadratic-time algorithms and adversarial training. *arXiv preprint arXiv:2201.01965*, 2022.

Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint arXiv:1910.08051*, 2019.

Fabio Carrara, Fabrizio Falchi, Roberto Caldelli, Giuseppe Amato, and Rudy Becarelli. Adversarial image detection in deep neural networks. *Multimedia Tools and Applications*, 78(3):2815–2835, 2019.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 2019.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. In *International Conference on Learning Representations*, 2020.

Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *International Conference on Learning Representations*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images, 2012. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In *ACM Workshop on Artificial Intelligence and Security*, 2019.

Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, 2019.

Ziye Ma and Somayeh Sojoudi. A sequential framework towards an exact SDP verification of neural networks. In *International Conference on Data Science and Advanced Analytics*, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *International Conference on Learning Representations*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Dongbin Na. Pytorch adversarial training on cifar-10. `https://github.com/ndb796/Pytorch-Adversarial-Training-CIFAR`, 2020.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In *International Conference on Machine Learning*, 2020.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

Jianyu Wang and Haichao Zhang. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In *International Conference on Computer Vision*, 2019.

Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. In *International Conference on Machine Learning*, 2018.

Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R. Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In *Annual Conference on Neural Information Processing Systems*, 2020.

Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *Annual Conference on Neural Information Processing Systems*, 2019.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, 2019.

A ADDITIONAL EXPERIMENTS

A.1 ACCELERATED ADAPTIVE MODEL

From Figure 2, it can be observed that for two arbitrary classes i and j, $\nabla g_i(x)$ and $\nabla g_j(x)$ are generally strongly correlated. Similarly, $\nabla h_i(x)$ and $\nabla h_j(x)$ are also strongly correlated. Therefore, an accelerated version of (7) can be obtained by using the gradient magnitude of the predicted classes to smooth all outputs. The adaptive smoothing operation then becomes:

$$g^{\theta}_{\text{CNN},i}(x) = \frac{g_i(x) + \exp\big(\beta_\theta(x)\big) R(x) h_i(x)}{1 + \exp\big(\beta_\theta(x)\big) R(x)}, \quad \forall i \in [c] \tag{10}$$

$$\text{where } R(x) = \frac{\|\nabla g_{k_1}(x)\|_{p*}}{\|\nabla h_{k_2}(x)\|_{p*}}, \quad k_1 = \arg\max_{k\in[c]} g_k(x), \quad k_2 = \arg\max_{k\in[c]} h_k(x).$$

The procedure (10) no longer requires calculating the full Jacobians of $g(\cdot)$ and $h(\cdot)$, and is much faster than (7) when c is large. Table 2 reports the test accuracy achieved with (10). On the CIFAR-100 dataset, testing 10000 clean examples with a batch size of 250 takes 4.16 seconds with an Nvidia V100 GPU using $g(\cdot)$, a PreActResNet18 model. Testing these images using the accelerated $g^{\theta}_{\text{CNN}}(\cdot)$ obtained via (10) requires 8.94 seconds.

Setting	CIFAR-10 AT ℓ_∞	CIFAR-10 AT ℓ_2	CIFAR-10 TRADE ℓ_∞	CIFAR-100 AT ℓ_∞
Clean Accuracy	95.15 %	94.36 %	95.25 %	78.35 %
PGD_{20} Accuracy	44.49 %	63.73 %	56.53 %	25.12 %

Table 2: Summary of test accuracy achieved by the accelerated adaptive smoothing classifier $g^{\theta}_{\text{CNN}}(\cdot)$ obtained via (10). The four columns of this table correspond to the four rows of Table 1.

Regarding the theoretical analyses, Theorem 3 directly extends to (10), and Theorem 2 extends with minor modifications.

A.2 EVALUATING THE ADAPTIVE POLICY WITH DIFFERENT ATTACK RADII

Since the neural network policy $\beta_\theta(\cdot)$ is trained using PGD adversarial data with a particular radius, it is natural to question whether the policy overfits this PGD setting. Figure 8 presents the accuracy of the adaptive model $g^{\theta}_{\text{CNN}}(\cdot)$ under the PGD_{20} attack with each different attack radius ϵ. While the policy was trained on PGD_{10} data with the radius $\epsilon = 0.031$, the adaptive classifier $g^{\theta}_{\text{CNN}}(\cdot)$ does not perform worse than the robust classifier $h(\cdot)$ in every setting.

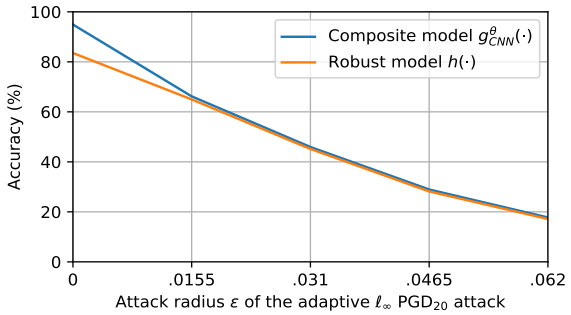
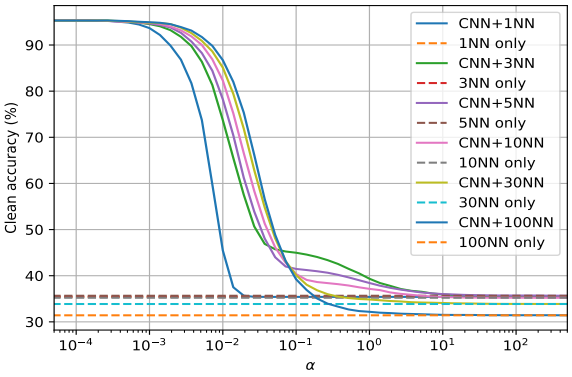


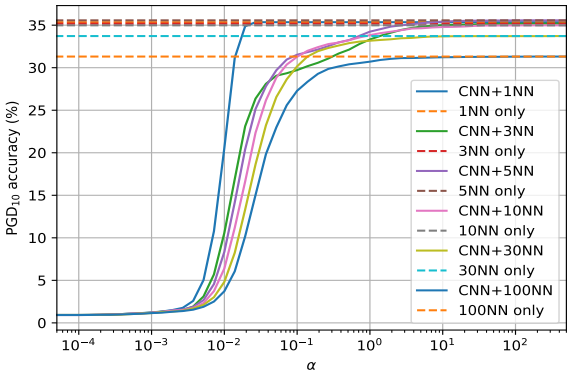
Figure 8: PGD_{20} accuracies with different values of ϵ.

A.3 SMOOTHING WITH DISTANCE-WEIGHTED K-NN CLASSIFIERS

We repeat the experiments in Subsection 6.1 with distance-weighted K-NN classifiers, which are far less likely to encounter ties. The results are shown in Figure 9. The results confirm that when α goes to $+\infty$, the performance of the smoothed classifier $g^{\alpha}_{K\text{-NN}}(\cdot)$ converges to the performance of the K-NN classifier $h(\cdot)$.



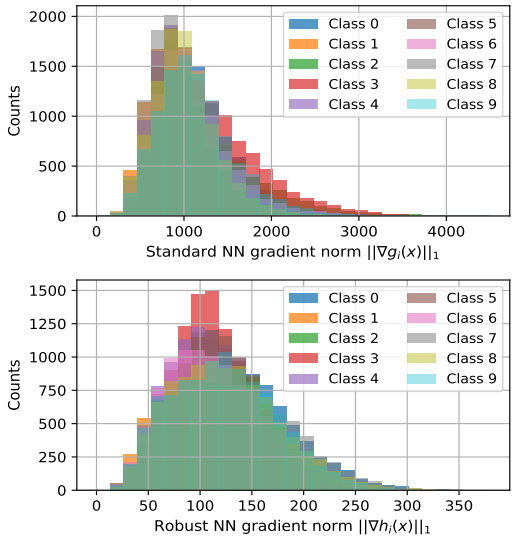
(a) Clean accuracy for each α for various values of K.



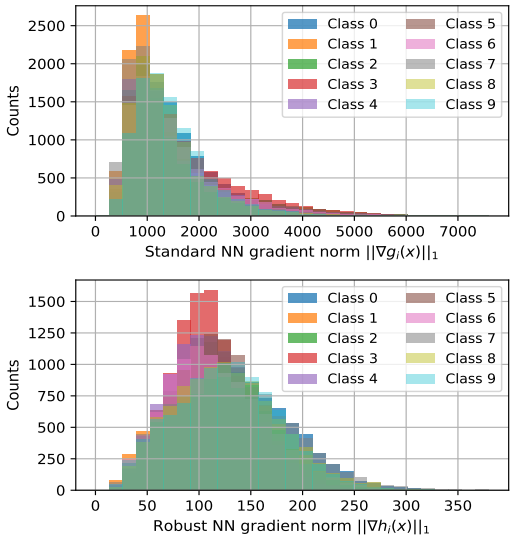
(b) PGD_{10} adversarial accuracy for each α for various values of K.

Figure 9: Evaluating the performance of the smoothed neural network based on distance-weighted K-NN classifiers.

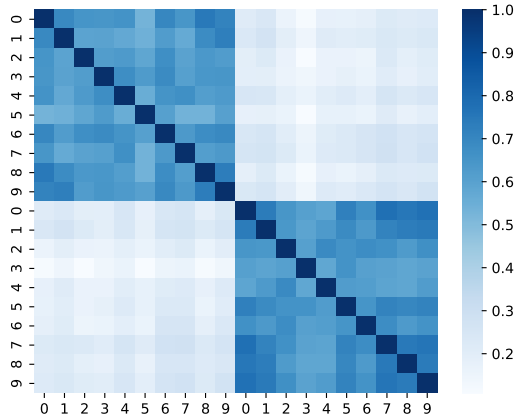
A.4 GRADIENT MAGNITUDE DISTRIBUTIONS WITH ADVERSARIAL DATA



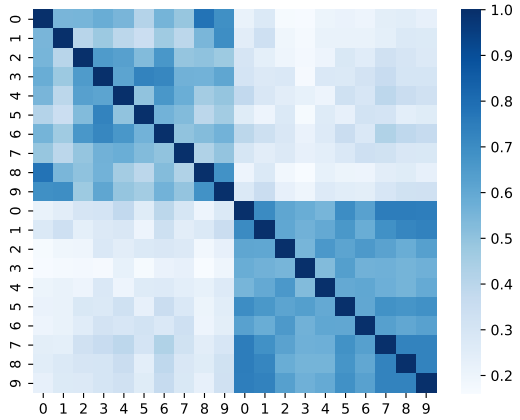
(a) The distributions with PGD_{10} adversarial data targeting $g(\cdot)$.



(b) The distributions with PGD_{10} adversarial data targeting $h(\cdot)$.



(c) Correlation matrix with adversarial data targeting $g(\cdot)$.



(d) Correlation matrix with adversarial data targeting $h(\cdot)$.

Figure 10: The distributions of $\|\nabla g_i(x)\|_1$ and $\|\nabla h_i(x)\|_1$ for each $i \in [10]$ with PGD_{10} adversarial data, as well as the correlation matrices of all $\|\nabla g_i(x)\|_1$ and all $\|\nabla h_i(x)\|_1$ with PGD_{10} adversarial data.

This subsection repeats the experiment in Figure 2 and Figure 3 with adversarial data. Figures 10a and 10c show the distributions and the correlations of $\|\nabla g_i(x)\|_1$ and $\|\nabla h_i(x)\|_1$ with PGD_{10} adversarial data targeting $g(\cdot)$. Figures 10b and 10d demonstrate the distributions and the correlations with PGD_{10} adversarial data targeting $h(\cdot)$. The results are highly similar compared with the clean data results in Figure 2, justifying the usage of $R_i(x)$ as a criteria for adjusting the mixture in $g^{\alpha}_{\text{CNN}}(x)$.

B PROOFS

B.1 PROOF OF THEOREM 1

Before diving into the proof, we introduce the following functions for each $i \in [c]$ for the sake of simplicity:

$$C_i(x,\delta) := \frac{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_i(x)\|_{p*}}, \qquad H_i(x,\delta) := \frac{\big(1-C_i(x,\delta)\big)g_i(x)}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}, \qquad G_i^{x+\delta}(\widehat{x}) := \frac{\nabla g_i(\widehat{x})}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}.$$

For the function $G_i^{x+\delta}(\widehat{x})$, the term $x+\delta$ in the denominator is a parameter and not a variable, whereas $\widehat{x} \in \mathbb{R}^d$ denotes the input to the function. For the other two functions, x and δ are variables. It follows from (3) that

$$\|G_i^{x+\delta}(x') - G_i^{x+\delta}(x+\delta)\|_{p*} \le L_i\epsilon; \qquad \|G_j^{x+\delta}(x') - G_i^{x+\delta}(x+\delta)\|_{p*} \le L_j\epsilon. \tag{11}$$

where x' is any point on the line segment between x and $x+\delta$. For an arbitrary $i \in [c]$, applying the Cauchy's mean-value theorem to g_i gives rise to that

$$\begin{aligned} g^{\alpha}_{K\text{-NN},i}(x+\delta) =& \frac{g_i(x+\delta)+\alpha h_i(x+\delta)\|\nabla g_i(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}} \\ =& \frac{g_i(x)+\nabla g_i(x_i')^\top\delta+\alpha h_i(x)\|\nabla g_i(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}} \end{aligned}$$

for some x_i' on the line segment between x and $x+\delta$, implying that

$$\begin{aligned} & g^{\alpha}_{K\text{-NN},i}(x+\delta) - g^{\alpha}_{K\text{-NN},i}(x) \\ =& \frac{g_i(x)+\nabla g_i(x_i')^\top\delta+\alpha h_i(x)\|\nabla g_i(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}} - \frac{g_i(x)+\alpha h_i(x)\|\nabla g_i(x)\|_{p*}}{1+\alpha\|\nabla g_i(x)\|_{p*}} \\ =& \frac{g_i(x)+\nabla g_i(x_i')^\top\delta+\alpha h_i(x)\|\nabla g_i(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}} - \frac{g_i(x)+\alpha h_i(x)\|\nabla g_i(x)\|_{p*}}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}\cdot C_i(x,\delta) \\ =& \frac{\big(1-C_i(x,\delta)\big)g_i(x)+\nabla g_i(x_i')^\top\delta+\alpha h_i(x)\|\nabla g_i(x+\delta)\|_{p*}-\alpha h_i(x)C_i(x,\delta)\|\nabla g_i(x)\|_{p*}}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}} \\ =& H_i(x,\delta)+G_i^{x+\delta}(x_i')^\top\delta+\alpha h_i(x)\|G_i^{x+\delta}(x+\delta)\|_{p*}-\alpha h_i(x)C_i(x,\delta)\|G_i^{x+\delta}(x)\|_{p*}, \end{aligned}$$

and therefore,

$$\begin{aligned} & \Big|\big(g^{\alpha}_{K\text{-NN},i}(x+\delta)-g^{\alpha}_{K\text{-NN},i}(x)\big)-\big(g^{\alpha}_{K\text{-NN},j}(x+\delta)-g^{\alpha}_{K\text{-NN},j}(x)\big)\Big| \\ =& \Big|H_i(x,\delta)-H_j(x,\delta)+\Big(G_i^{x+\delta}(x_i')-G_j^{x+\delta}(x_j')\Big)^\top\delta \\ & +\alpha h_i(x)\Big(\big\|G_i^{x+\delta}(x+\delta)\big\|_{p*}-C_i(x,\delta)\big\|G_i^{x+\delta}(x)\big\|_{p*}\Big) \\ & -\alpha h_j(x)\Big(\big\|G_j^{x+\delta}(x+\delta)\big\|_{p*}-C_j(x,\delta)\big\|G_j^{x+\delta}(x)\big\|_{p*}\Big)\Big| \\ \le& \Big|H_i(x,\delta)-H_j(x,\delta)\Big|+\Big|\big(G_i^{x+\delta}(x')-G_j^{x+\delta}(x')\big)^\top\delta\Big| \\ & +\alpha\Big|\big\|G_i^{x+\delta}(x+\delta)\big\|_{p*}-C_i(x,\delta)\big\|G_i^{x+\delta}(x)\big\|_{p*}\Big|+\alpha\Big|\big\|G_j^{x+\delta}(x+\delta)\big\|_{p*}-C_i(x,\delta)\big\|G_j^{x+\delta}(x)\big\|_{p*}\Big|, \end{aligned} \tag{12}$$

where the last line comes from the fact that $h_i(x), h_j(x) \in [-1,1]$.

The individual terms in (12) can be bounded using the following Lemmas, whose proofs are presented in Appendix B.1.1, Appendix B.1.2, and Appendix B.1.3, respectively.

Lemma 4. *It holds that*

$$\left|\left\|G_i^{x+\delta}(x+\delta)\right\|_{p*} - C_i(x,\delta)\left\|G_i^{x+\delta}(x)\right\|_{p*}\right| \leq 2L_i\epsilon.$$

Lemma 5. *It holds that*

$$\left|H_i(x,\delta) - H_j(x,\delta)\right| \leq \frac{\alpha L_i|g_i(x)|}{1+\alpha\|\nabla g_i(x)\|_{p*}}\epsilon + \frac{\alpha L_j|g_j(x)|}{1+\alpha\|\nabla g_j(x)\|_{p*}}\epsilon.$$

Lemma 6. *It holds that*

$$\left|\left(G_i^{x+\delta}(x') - G_j^{x+\delta}(x')\right)^\top \delta\right| \leq \frac{M_i\epsilon}{1+\alpha M_i} + \frac{M_j\epsilon}{1+\alpha M_j} + (L_i+L_j)\epsilon^2.$$

Following Lemmas 4, 5, and 6 (due to symmetry, Lemma 4 also holds for class j), the expression (12) can be bounded via

$$\begin{aligned}
&\left|\left(g^\alpha_{K\text{-NN},i}(x+\delta) - g^\alpha_{K\text{-NN},i}(x)\right) - \left(g^\alpha_{K\text{-NN},j}(x+\delta) - g^\alpha_{K\text{-NN},j}(x)\right)\right| \\
&\quad\leq \frac{\alpha L_i|g_i(x)|}{1+\alpha\|\nabla g_i(x)\|_{p*}}\epsilon + \frac{\alpha L_j|g_j(x)|}{1+\alpha\|\nabla g_j(x)\|_{p*}}\epsilon + \frac{M_i}{1+\alpha M_i}\epsilon + \frac{M_j}{1+\alpha M_j}\epsilon + (L_i+L_j)\epsilon^2 + 2\alpha(L_i+L_j)\epsilon \\
&\quad= (L_i+L_j)\epsilon^2 + b\epsilon \\
&\quad\leq \left|g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x)\right|,
\end{aligned}$$

where the last inequality arises from the assumption given in (4). Therefore, it holds that

$$\begin{aligned}
&-\left|g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x)\right| \\
&\qquad\leq g^\alpha_{K\text{-NN},i}(x+\delta) - g^\alpha_{K\text{-NN},j}(x+\delta) - g^\alpha_{K\text{-NN},i}(x) + g^\alpha_{K\text{-NN},j}(x) \\
&\qquad\leq \left|g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x)\right|,
\end{aligned}$$

and thus,

$$\begin{aligned}
&g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x) - \left|g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x)\right| \\
&\qquad\leq g^\alpha_{K\text{-NN},i}(x+\delta) - g^\alpha_{K\text{-NN},j}(x+\delta) \\
&\qquad\leq g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x) + \left|g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x)\right|.
\end{aligned}$$

If $g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x) \geq 0$, then the left-hand inequality gives that

$$0 = g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x) - |g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x)| \leq g^\alpha_{K\text{-NN},i}(x+\delta) - g^\alpha_{K\text{-NN},j}(x+\delta),$$

whereas if $g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x) \leq 0$, then the right-hand inequality gives that

$$0 = g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x) + |g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x)| \geq g^\alpha_{K\text{-NN},i}(x+\delta) - g^\alpha_{K\text{-NN},j}(x+\delta).$$

In both cases, it holds that $\operatorname{sgn}\left(g^\alpha_{K\text{-NN},i}(x+\delta) - g^\alpha_{K\text{-NN},j}(x+\delta)\right) = \operatorname{sgn}\left(g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x)\right)$. □

B.1.1 Proof of Lemma 4

Note that

$$\begin{aligned}
\left|\left\|G_i^{x+\delta}(x+\delta)\right\|_{p*} - C_i(x,\delta)\left\|G_i^{x+\delta}(x)\right\|_{p*}\right| &= \left|\left\|G_i^{x+\delta}(x+\delta)\right\|_{p*} - \left\|G_i^{x+\delta}(x)\right\|_{p*} + \left(1-C_i(x,\delta)\right)\left\|G_i^{x+\delta}(x)\right\|_{p*}\right| \\
&\leq \left|\left\|G_i^{x+\delta}(x+\delta)\right\|_{p*} - \left\|G_i^{x+\delta}(x)\right\|_{p*}\right| + \left|\left(1-C_i(x,\delta)\right)\left\|G_i^{x+\delta}(x)\right\|_{p*}\right| \\
&\leq \left\|G_i^{x+\delta}(x+\delta) - G_i^{x+\delta}(x)\right\|_{p*} + \left|1-C_i(x,\delta)\right|\left\|G_i^{x+\delta}(x)\right\|_{p*},
\end{aligned}$$

where

$$\begin{aligned}
\big|1-C_i(x,\delta)\big|\big\|G_i^{x+\delta}(x)\big\|_{p*} =& \frac{\big|1+\alpha\|\nabla g_i(x)\|_{p*}-1-\alpha\|\nabla g_i(x+\delta)\|_{p*}\big|}{1+\alpha\|\nabla g_i(x)\|_{p*}}\cdot\left\|\frac{\nabla g_i(x)}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}\right\|_{p*}\\
=&\big\|\nabla g_i(x)\big\|_{p*}\frac{\alpha\big|\|\nabla g_i(x)\|_{p*}-\|\nabla g_i(x+\delta)\|_{p*}\big|}{\big(1+\alpha\|\nabla g_i(x)\|_{p*}\big)\big(1+\alpha\|\nabla g_i(x+\delta)\|_{p*}\big)}\\
\le&\alpha\big\|\nabla g_i(x)\big\|_{p*}\frac{\|\nabla g_i(x)-\nabla g_i(x+\delta)\|_{p*}}{\big(1+\alpha\|\nabla g_i(x)\|_{p*}\big)\big(1+\alpha\|\nabla g_i(x+\delta)\|_{p*}\big)}\\
\le&\frac{\|\nabla g_i(x)-\nabla g_i(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}\\
=&\big\|G_i^{x+\delta}(x)-G_i^{x+\delta}(x+\delta)\big\|_{p*}.
\end{aligned}$$

Combining the above two results gives rise to that

$$\Big|\big\|G_i^{x+\delta}(x+\delta)\big\|_{p*}-C_i(x,\delta)\big\|G_i^{x+\delta}(x)\big\|_{p*}\Big|\le 2\big\|G_i^{x+\delta}(x+\delta)-G_i^{x+\delta}(x)\big\|_{p*}\le 2L_i\epsilon,$$

where the last inequality arises from the assumption (11). □

B.1.2 Proof of Lemma 5

It holds that

$$\begin{aligned}
&\Big|H_i(x,\delta)-H_j(x,\delta)\Big|\\
=&\left|\frac{\big(1-C_i(x,\delta)\big)g_i(x)}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}-\frac{\big(1-C_j(x,\delta)\big)g_j(x)}{1+\alpha\|\nabla g_j(x+\delta)\|_{p*}}\right|\\
\le&\left|\frac{\big(1-\frac{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_i(x)\|_{p*}}\big)g_i(x)}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}}\right|+\left|\frac{\big(1-\frac{1+\alpha\|\nabla g_j(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_j(x)\|_{p*}}\big)g_j(x)}{1+\alpha\|\nabla g_j(x+\delta)\|_{p*}}\right|\\
=&\alpha\left|\frac{\big(\|\nabla g_i(x)\|_{p*}-\|\nabla g_i(x+\delta)\|_{p*}\big)g_i(x)}{\big(1+\alpha\|\nabla g_i(x+\delta)\|_{p*}\big)\big(1+\alpha\|\nabla g_i(x)\|_{p*}\big)}\right|+\alpha\left|\frac{\big(\|\nabla g_j(x)\|_{p*}-\|\nabla g_j(x+\delta)\|_{p*}\big)g_j(x)}{\big(1+\alpha\|\nabla g_j(x+\delta)\|_{p*}\big)\big(1+\alpha\|\nabla g_j(x)\|_{p*}\big)}\right|\\
\le&\alpha\frac{\|\nabla g_i(x)-\nabla g_i(x+\delta)\|_{p*}|g_i(x)|}{\big(1+\alpha\|\nabla g_i(x+\delta)\|_{p*}\big)\big(1+\alpha\|\nabla g_i(x)\|_{p*}\big)}+\alpha\frac{\|\nabla g_j(x)-\nabla g_j(x+\delta)\|_{p*}|g_j(x)|}{\big(1+\alpha\|\nabla g_j(x+\delta)\|_{p*}\big)\big(1+\alpha\|\nabla g_j(x)\|_{p*}\big)}\\
=&\big\|G_i^{x+\delta}(x)-G_i^{x+\delta}(x+\delta)\big\|_{p*}\cdot\frac{\alpha|g_i(x)|}{1+\alpha\|\nabla g_i(x)\|_{p*}}+\big\|G_j^{x+\delta}(x)-G_j^{x+\delta}(x+\delta)\big\|_{p*}\cdot\frac{\alpha|g_j(x)|}{1+\alpha\|\nabla g_j(x)\|_{p*}}\\
\le&\frac{\alpha L_i|g_i(x)|}{1+\alpha\|\nabla g_i(x)\|_{p*}}\epsilon+\frac{\alpha L_j|g_j(x)|}{1+\alpha\|\nabla g_j(x)\|_{p*}}\epsilon,
\end{aligned}$$

where the last inequality arises from the assumption (11). □

B.1.3 Proof of Lemma 6

By the Cauchy-Schwarz inequality for dual norms, it holds that

$$\big|\big(G_i^{x+\delta}(x')-G_j^{x+\delta}(x')\big)^\top\delta\big|\le\big\|G_i^{x+\delta}(x')-G_j^{x+\delta}(x')\big\|_{p*}\|\delta\|_p,$$

where

$$\begin{aligned}
&\big\|G_i^{x+\delta}(x')-G_j^{x+\delta}(x')\big\|_{p*}\\
\le&\Big\|G_i^{x+\delta}(x+\delta)-G_j^{x+\delta}(x+\delta)+\big(G_i^{x+\delta}(x')-G_i^{x+\delta}(x+\delta)\big)-\big(G_j^{x+\delta}(x')-G_i^{x+\delta}(x+\delta)\big)\Big\|_{p*}
\end{aligned}$$

$$
\begin{aligned}
&\leq \left\|G_i^{x+\delta}(x+\delta)\right\|_{p*} + \left\|G_j^{x+\delta}(x+\delta)\right\|_{p*} + \left\|G_i^{x+\delta}(x') - G_i^{x+\delta}(x+\delta)\right\|_{p*} + \left\|G_j^{x+\delta}(x') - G_i^{x+\delta}(x+\delta)\right\|_{p*} \\
&\leq \frac{\|\nabla g_i(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_i(x+\delta)\|_{p*}} + \frac{\|\nabla g_j(x+\delta)\|_{p*}}{1+\alpha\|\nabla g_j(x+\delta)\|_{p*}} + L_i\epsilon + L_j\epsilon \\
&\leq \frac{M_i}{1+\alpha M_i} + \frac{M_j}{1+\alpha M_j} + L_i\epsilon + L_j\epsilon,
\end{aligned}
$$

where the second last inequality arises from the assumption (11) and the definitions of $G_i^{x+\delta}(\cdot)$ and $G_j^{x+\delta}(\cdot)$, and the last inequality holds because the function $f(x) = \frac{x}{1+\alpha x}$ is monotonously increasing when $x > 0$ and $\alpha > 0$. Thus, it holds that

$$
\left|\left(G_i^{x+\delta}(x') - G_j^{x+\delta}(x')\right)^\top \delta\right| \leq \left(\frac{M_i}{1+\alpha M_i} + \frac{M_j}{1+\alpha M_j} + (L_i + L_j)\epsilon\right)\|\delta\|_p \leq \frac{M_i\epsilon}{1+\alpha M_i} + \frac{M_j\epsilon}{1+\alpha M_j} + (L_i + L_j)\epsilon^2
$$

which is the desired result. □

B.2 PROOF OF THEOREM 2

Let $k \in \{i, j\}$, and start by noting that the bounded gradient assumption implies Lipschitz continuity:

$$
|g_k(x') - g_k(x)| = |\nabla g_k(x')^\top (x' - x)| \leq \|\nabla g_k(x')\|_{p*}\|x' - x\|_p \leq M_k^g\|x' - x\|_p,
$$

where the first equality follows from Cauchy's mean-value theorem. The same reasoning applies to h_k. Now, we begin by bounding

$$
\begin{aligned}
\left|R_k(x) - R_k(x+\delta)\right| &= \left|\frac{\|\nabla g_k(x)\|_{p*}}{\|\nabla h_k(x)\|_{p*}} - \frac{\|\nabla g_k(x+\delta)\|_{p*}}{\|\nabla h_k(x+\delta)\|_{p*}}\right| \\
&= \frac{\left|\|\nabla g_k(x)\|_{p*}\|\nabla h_k(x+\delta)\|_{p*} - \|\nabla g_k(x+\delta)\|_{p*}\|\nabla h_k(x)\|_{p*}\right|}{\|\nabla h_k(x)\|_{p*}\|\nabla h_k(x+\delta)\|_{p*}} \\
&\leq \frac{\|\nabla g_k(x)\|_{p*}\|\nabla h_k(x+\delta) - \nabla h_k(x)\|_{p*} + \|\nabla h_k(x)\|_{p*}\|\nabla g_k(x+\delta) - \nabla g_k(x)\|_{p*}}{\|\nabla h_k(x)\|_{p*}\|\nabla h_k(x+\delta)\|_{p*}} \\
&\leq \frac{\|\nabla g_k(x)\|_{p*}L_k^h\|\delta\|_p + \|\nabla h_k(x)\|_{p*}L_k^g\|\delta\|_p}{\|\nabla h_k(x)\|_{p*}\|\nabla h_k(x+\delta)\|_{p*}} \\
&= \frac{L_k^h R_k(x) + L_k^g}{\|\nabla h_k(x+\delta)\|_{p*}}\|\delta\|_p \\
&\leq \frac{L_k^h R_k(x) + L_k^g}{m_k^h}\|\delta\|_p.
\end{aligned} \tag{13}
$$

It holds that

$$
\begin{aligned}
|g^\alpha_{\text{CNN},k}(x+\delta) - g^\alpha_{\text{CNN},k}(x)| &= \left|\frac{g_k(x+\delta) + \alpha R_k(x+\delta)h_k(x+\delta)}{1+\alpha R_k(x+\delta)} - \frac{g_k(x) + \alpha R_k(x)h_k(x)}{1+\alpha R_k(x)}\right| \\
&\leq \underbrace{\left|\frac{g_k(x+\delta)}{1+\alpha R_k(x+\delta)} - \frac{g_k(x)}{1+\alpha R_k(x)}\right|}_{=:G_k(x,\delta,\alpha)} + \alpha \underbrace{\left|\frac{R_k(x+\delta)h_k(x+\delta)}{1+\alpha R_k(x+\delta)} - \frac{R_k(x)h_k(x)}{1+\alpha R_k(x)}\right|}_{=:H_k(x,\delta,\alpha)}.
\end{aligned} \tag{14}
$$

Bounding the first term using (13) gives

$$
\begin{aligned}
G_k(x,\delta,\alpha) &= \left|\frac{g_k(x+\delta) + \alpha R_k(x)g_k(x+\delta) - g_k(x) - \alpha R_k(x+\delta)g_k(x)}{(1+\alpha R_k(x+\delta)(1+\alpha R_k(x))}\right| \\
&\leq \frac{|g_k(x+\delta) - g_k(x)| + \alpha|R_k(x)g_k(x+\delta) - R_k(x+\delta)g_k(x)|}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)(1+\alpha R_k(x))} \\
&\leq \frac{M_k^g\|\delta\|_p + \alpha|g_k(x+\delta)|\cdot|R_k(x) - R_k(x+\delta)| + \alpha|g_k(x+\delta) - g_k(x)|\cdot|R_k(x+\delta)|}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)(1+\alpha R_k(x))}
\end{aligned}
$$

$$
\begin{aligned}
&\leq \frac{M_k^g\|\delta\|_p + \alpha K_k^g |R_k(x) - R_k(x+\delta)| + \alpha M_k^g |R_k(x+\delta)|\|\delta\|_p}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)(1+\alpha R_k(x))} \\
&\leq \frac{M_k^g\|\delta\|_p + \alpha K_k^g \frac{L_k^h R_k(x)+L_k^g}{m_k^h}\|\delta\|_p + \alpha M_k^g \frac{M_k^g}{m_k^h}\|\delta\|_p}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)(1+\alpha R_k(x))} \\
&\leq \frac{M_k^g\left(1+\alpha\frac{M_k^g}{m_k^h}\right) + \alpha\frac{K_k^g}{m_k^h}\left(L_k^h R_k(x)+L_k^g\right)}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)(1+\alpha R_k(x))}\|\delta\|_p \\
&= C_k^{(1)}(x,\alpha)\|\delta\|_p.
\end{aligned}
$$

Similarly, the second term is bounded using (13) as

$$
\begin{aligned}
H_k(x,\delta,\alpha) &= \left|\frac{R_k(x+\delta)h_k(x+\delta) + \alpha R_k(x)R_k(x+\delta)h_k(x+\delta) - R_k(x)h_k(x) - \alpha R_k(x+\delta)R_k(x)h_k(x)}{(1+\alpha R_k(x))(1+\alpha R_k(x+\delta))}\right| \\
&\leq \frac{|R_k(x+\delta)h_k(x+\delta) - R_k(x)h_k(x)| + \alpha R_k(x)R_k(x+\delta)|h_k(x+\delta) - h_k(x)|}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)\left(1+\alpha R_k(x)\right)} \\
&\leq \frac{R_k(x+\delta)|h_k(x+\delta) - h_k(x)| + |h_k(x)||R_k(x+\delta) - R_k(x)| + \alpha R_k(x)R_k(x+\delta)|h_k(x+\delta) - h_k(x)|}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)\left(1+\alpha R_k(x)\right)} \\
&\leq \frac{\left(1+\alpha R_k(x)\right)R_k(x+\delta)\left|h_k(x+\delta) - h_k(x)\right| + K_k^h\frac{L_k^h R_k(x)+L_k^g}{m_k^h}\|\delta\|_p}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)\left(1+\alpha R_k(x)\right)} \\
&\leq \frac{\left(1+\alpha R_k(x)\right)\frac{M_k^g}{m_k^h}M_k^h + K_k^h\frac{L_k^h R_k(x)+L_k^g}{m_k^h}}{\left(1+\alpha\frac{m_k^g}{M_k^h}\right)\left(1+\alpha R_k(x)\right)}\|\delta\|_p \\
&= C_k^{(2)}(x,\alpha)\|\delta\|_p.
\end{aligned}
$$

Hence, (14) yields that

$$
\begin{aligned}
\left|\left(g^\alpha_{\text{CNN},i}(x+\delta) - g^\alpha_{\text{CNN},i}(x)\right) - \left(g^\alpha_{\text{CNN},j}(x+\delta) - g^\alpha_{\text{CNN},j}(x)\right)\right| &\leq \sum_{k\in\{i,j\}}\left|g^\alpha_{\text{CNN},k}(x+\delta) - g^\alpha_{\text{CNN},k}(x)\right| \\
&\leq \sum_{k\in\{i,j\}}\left(G_k(x,\delta,\alpha) + \alpha H_k(x,\delta,\alpha)\right) \\
&\leq \sum_{k\in\{i,j\}}\left(C_k^{(1)}(x,\alpha) + \alpha C_k^{(2)}(x,\alpha)\right)\|\delta\|_p \\
&\leq \sum_{k\in\{i,j\}}\left(C_k^{(1)}(x,\alpha) + \alpha C_k^{(2)}(x,\alpha)\right)\epsilon \\
&\leq |g^\alpha_{\text{CNN},i}(x) - g^\alpha_{\text{CNN},j}(x)|.
\end{aligned}
$$

The remainder of the proof follows similarly to the final steps in the proof Theorem 1. □

B.3 PROOF OF THEOREM 3

Since it is assumed that the perturbation balls of the data are non-overlapping, the true label y corresponding to each perturbed data $x+\delta$ with the property $\|\delta\|_p \leq \epsilon$ is unique. Therefore, the indicator function

$$
\alpha(x+\delta) = \begin{cases} 0 & \text{if } \arg\max_{i\in[c]} g_i(x+\delta) = y, \\ +\infty & \text{otherwise.} \end{cases}
$$

satisfies that

$$\begin{aligned}
&\alpha(x+\delta) = 0 \qquad \text{if} \quad \arg\max_{i\in[c]} g_i(x+\delta) = y, \\
&\alpha(x+\delta) = +\infty \quad \text{if} \quad \arg\max_{i\in[c]} g_i(x+\delta) \neq y \text{ and } \arg\max_{i\in[c]} h_i(x+\delta) = y.
\end{aligned}$$

Therefore, it holds that

$$\begin{aligned}
&g^{\alpha}_{\mathrm{CNN},i}(x+\delta) = g_i(x+\delta) \quad \text{if} \quad \arg\max_{i\in[c]} g_i(x+\delta) = y, \\
&g^{\alpha}_{\mathrm{CNN},i}(x+\delta) = h_i(x+\delta) \quad \text{if} \quad \arg\max_{i\in[c]} g_i(x+\delta) \neq y \text{ and } \arg\max_{i\in[c]} h_i(x+\delta) = y,
\end{aligned}$$

implying that

$$\arg\max\nolimits_{i\in[c]} g^{\alpha}_{\mathrm{CNN},i}(x+\delta) = y \quad \text{if} \quad \Big(\arg\max\nolimits_{i\in[c]} g_i(x+\delta) = y \ \text{ or } \ \arg\max\nolimits_{i\in[c]} h_i(x+\delta) = y \Big),$$

which leads to the desired statement. □