Improving 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. This "adaptive smoothing" method adapts an adversarial example detector into a policy network, which adaptively adjusts the mixture of a robust classifier and a standard network. We provide theoretical analysis to motivate the use of locally biased smoothing as well as the introduction of the policy network. The adaptively smoothed model noticeably improves the accuracy-robustness trade-off. Under moderate adversarial attacks, the smoothed model achieves the robustness of the robust network while suffering minor clean accuracy penalties compared with the standard network.

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., 2022a,b] 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., 2021, 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 policy network. 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

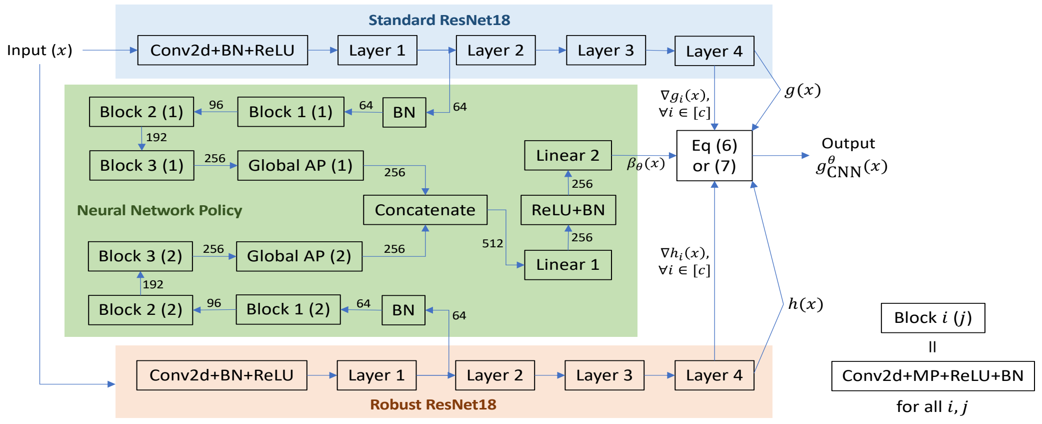


Figure 1: The overall architecture of the adaptive composite classifier introduced in Section 5 when applied to a pair of ResNet18 classifiers. "BN" represents 2D batch normalization, "MP" represents 2×2 2D max-pooling, and "AP" represents 2D average-pooling. All Conv2d layers in the policy network share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

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, "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, $\operatorname{sgn}(a) \in \{-1, 0, 1\}$ 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.

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.

2.2 ADVERSARIAL INPUT DETECTORS

It has been shown that adversarial inputs can be detected via various methods. For example, [Metzen et al., 2017] proposes to append an additional detection branch to an existing neural network, and use adaptive adversarial data to train the detector in a supervised fasion. Unfortunately, Carlini and Wagner [2017a] have shown that it is possible to bypass this detection method. They constructed adversarial examples via the C&W attacks [Carlini and Wagner, 2017b] and simultaneously target the classification branch and the detection branch by treating the two branches as an "augmented classifier". According to [Carlini and Wagner, 2017a], the detector is effective against the types of attack that it is trained with, but not necessarily the attack types that are absent in the training data. It is thus reasonable to expect the detector to be able to detect a wide range of attacks if it is trained using sufficiently diverse types of attacks (including those targeting the detector itself). While exhaustively covering the entire adversarial input space is intractable, and it is unclear to what degree one needs to diversify the attack types in practice, our experiments show that our modified architecture based on [Metzen et al., 2017] can recognize the state-of-the-art AutoAttack [Croce and Hein, 2020] adversaries with a high success rate.

To mitigate the above challenges faced by detectors obtained via supervised training, unsupervised detectors have been proposed [Aldahdooh et al., 2021a,b]. Other detection methods include [Carrara et al., 2019, Ahmadi et al., 2021]. Unfortunately, universally effective detectors have not been discovered to be best of our knowledge, so this paper focuses on transferring the properties of the existing detector towards better overall robustness. Future advancements in the field of adversary detection can further enhance the performance of our method.

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 in the binary classification setting, randomized smoothing with an unbiased distribution is suboptimal, and 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-NEIGHBOR 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:

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

To remove the effect of the different magnitude of the logit of each class, we propose a normalized model as follows:

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

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 $h(\cdot)$:

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

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

Certified robustness is a highly desirable property, since many defense methods without certification are quickly circumvented by novel attack methods. Here, we use certified robustness to show that adaptive smoothing preserves the inherent robustness of the base classifiers to some degree for each value of α, justifying the use of the smoothing operation introduced in (6).

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

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

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. *For a fixed x, 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

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

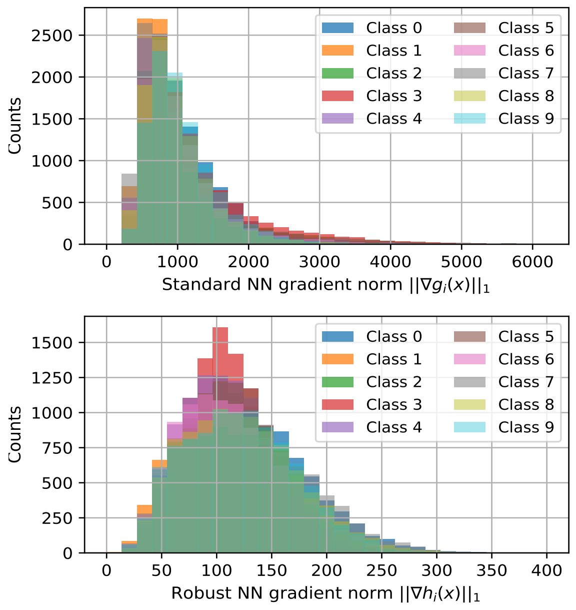


Figure 2: The distributions of $\|\nabla g_i(x)\|_1$ (upper) and $\|\nabla h_i(x)\|_1$ (lower) for each $i \in [10]$.

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 $\|\delta\|_p \le \epsilon$.

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 $\|\delta\|_p \le \epsilon$.

4 USING ROBUST NEURAL NETWORK AS 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 (the original) adversarial training, TRADE, and traditional randomized smoothing.

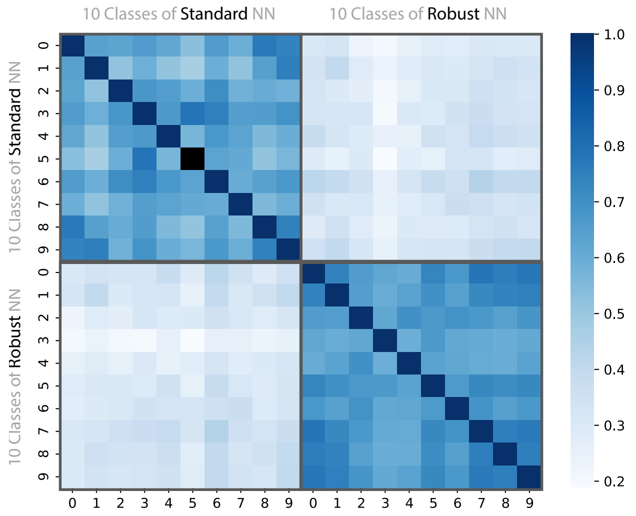


Figure 3: Correlation matrix of all $\|\nabla g_i(x)\|_1$ and all $\|\nabla h_i(x)\|_1$ where $i \in [10]$.

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:

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

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 $\|\nabla h_i(x)\|_1$ 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 $\|\nabla g_i(x)\|_1$ 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.3.

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

$$\begin{aligned} K_k^h &= \sup_{x' \in B_p(x,\epsilon)} |h_k(x')|, \\ m_k^h &= \inf_{x' \in B_p(x,\epsilon)} \|\nabla h_k(x')\|_{p*} > 0, \\ M_k^h &= \sup_{x' \in B_p(x,\epsilon)} \|\nabla h_k(x')\|_{p*}, \\ L_k^h &= \sup_{x' \in B_p(x,\epsilon)} \frac{\|\nabla h_k(x') - \nabla h_k(x)\|_{p*}}{\|x' - x\|_p}. \end{aligned}$$

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

$$C_k^{(1)}(x,\alpha) = \frac{M_k^g \left(1 + \alpha \frac{M_k^g}{m_k^h}\right) + \alpha \frac{K_k^g}{m_k^h} \left(L_k^h R_k(x) + L_k^g\right)}{\left(1 + \alpha \frac{m_k^g}{M_k^h}\right)(1 + \alpha R_k(x))},$$

$$C_k^{(2)}(x,\alpha) = \frac{\frac{M_k^g M_k^h}{m_k^h}(1 + \alpha R_k(x)) + \frac{K_k^h}{m_k^h}\left(L_k^h R_k(x) + L_k^g\right)}{\left(1 + \alpha \frac{m_k^g}{M_k^h}\right)(1 + \alpha R_k(x))}.$$

The certified radius is now presented.

Theorem 2. *For a fixed x, if* $\epsilon \le \frac{|g^\alpha_{CNN,i}(x) - g^\alpha_{CNN,j}(x)|}{\sum_{k \in \{i,j\}} \left(C_k^{(1)}(x,\alpha) + \alpha C_k^{(2)}(x,\alpha)\right)}$, *then*

$$\begin{aligned} \operatorname{sgn}\left(g^\alpha_{CNN,i}(x+\delta) - g^\alpha_{CNN,j}(x+\delta)\right) \\ = \operatorname{sgn}\left(g^\alpha_{CNN,i}(x) - g^\alpha_{CNN,j}(x)\right) \end{aligned}$$

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

Theorem 2 is proved in Appendix B.2.

5 ADAPTIVE SMOOTHING STRENGTH WITH THE POLICY NETWORK

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.

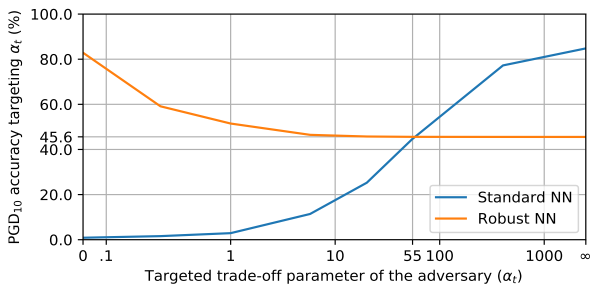


Figure 4: Attacked accuracy of the standard classifier $g(\cdot)$ and the robust classifier $h(\cdot)$ when the adversary targets different values of α_t.

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 $\alpha_t \le 55$, the robust model $h(\cdot)$ attains a higher accuracy. When $\alpha_t > 55$, the standard model $g(\cdot)$ becomes more suitable.

Theorem 3. *Assume that there does not exist $z \in \mathbb{R}^d$ such that $\|z - x_1\|_p \le \epsilon$ and $\|z - x_2\|_p \le \epsilon$, where $(x_1, y_1) \in \mathcal{D}$, $(x_2, y_2) \in \mathcal{D}$, and $y_1 \ne y_2$ (i. e. each input corresponds to a unique true label). 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

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

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 the distribution $\mathcal{F}$ is arbitrary, implying that the test data can be clean data, any type of adversarial data, or some combination of both. As a special case, 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 $g(\cdot)$. Conversely, when the PDF of $\mathcal{F}$ is a Dirac delta at the worst-case perturbation,

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

the adversarial accuracy of $g^{\alpha}_{\text{CNN}}(\cdot)$ is not worse than the robust model $h(\cdot)$, implying that if $h(\cdot)$ is inherently robust, then $g^{\alpha}_{\text{CNN}}(\cdot)$ inherits the robustness. Since there exists a $g^{\alpha}_{\text{CNN}}(\cdot)$ that produces the correct prediction whenever either $g(\cdot)$ or $h(\cdot)$ is correct (again, $\mathcal{F}$ is arbitrary), it can be concluded that there exists a $g^{\alpha}_{\text{CNN}}(\cdot)$ that resolves the accuracy-robustness trade-off between $g(\cdot)$ and $h(\cdot)$.

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

$$g^{\theta}_{\text{CNN},i}(x) = \frac{g_i(x) + \exp\big(\beta_\theta(x)\big) R_i(x) h_i(x)}{1 + \exp\big(\beta_\theta(x)\big) R_i(x)}, \quad \forall i \in [c],$$
$$\text{where } R_i(x) = \frac{\|\nabla g_i(x)\|_{p*}}{\|\nabla h_i(x)\|_{p*}}. \tag{6}$$

5.1 ACCELERATED ADAPTIVE MODEL

Since the forward pass of $g^{\theta}_{\text{CNN}}(x)$ constructed in (6) calculates the Jacobians of $g(\cdot)$ and $h(\cdot)$, the time complexity is linear in c, making it time-consuming to training the policy. To this end, we now propose a simplification that reduces this complexity to constant. We will demonstrate that the accelerated variant predicts about half as fast as the standalone $g(\cdot)$ and achieves accuracy and robustness simultaneously.

From the correlation matrix of the gradient magnitudes shown in 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, and $\nabla h_i(x)$ and $\nabla h_j(x)$ are also strongly correlated. Therefore, an accelerated version of (6) can be obtained by using the gradient magnitude of the predicted classes as an estimation of the gradient magnitudes of all classes. The adaptive smoothing operation then becomes:

$$g^{\theta}_{\text{CNN},i}(x) = \frac{g_i(x) + \exp\big(\beta_\theta(x)\big) R(x) h_i(x)}{1 + \exp\big(\beta_\theta(x)\big) R(x)}, \quad \forall i \in [c]$$
$$\text{where } R(x) = \frac{\|\nabla g_{k_1}(x)\|_{p*}}{\|\nabla h_{k_2}(x)\|_{p*}}, \tag{7}$$
$$k_1 = \operatorname*{arg\,max}_{k \in [c]} g_k(x), \quad k_2 = \operatorname*{arg\,max}_{k \in [c]} h_k(x).$$

The procedure (7) no longer requires calculating the full Jacobians of $g(\cdot)$ and $h(\cdot)$, and is much faster than (6) when c is large. Regarding the theoretical analyses, Theorem 3 directly extends to (7), and Theorem 2 extends with minor modifications.

5.2 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. In the experiments, we consider the following four types of attacks:

- **Gray-box PGD:** In this setting, the adversary has access to the gradients of both $g(\cdot)$ and $h(\cdot)$, but is not given the gradient of the policy network. We use untargeted PGD with a fixed initialization (also known as BIM) to generate the attacks.
- **PGD with a presumed α_t:** When the adversary has complete information about $g^{\theta}_{\text{CNN}}(\cdot)$, it can also assume a particular value for the smoothing strength α_t when calculating the gradient. In this case, the attacker generates the adversarial inputs with $g^{\alpha_t}_{\text{CNN}}(\cdot)$ and transfer them to attack $g^{\theta}_{\text{CNN}}(\cdot)$. Note that α_t can be different from the α value returned by the policy. When α_t is set to 0, the effectively attacking $g(\cdot)$. When α_t is set to ∞, the adversary attacks $h(\cdot)$.
- **White-box PGD:** Since the smoothed classifier is end-to-end differentiable, following the guideline outlined in [Tramèr et al., 2020], we allow the adversary to access the end-to-end gradient, including the gradient of the policy network. Note that $\beta_\theta(\widetilde{x}^0), \ldots, \beta_\theta(\widetilde{x}^T)$ can differ throughout the PGD iterations.
- **White-box AutoAttack:** [Croce and Hein, 2020] has proposed to use an ensemble of four automated attack algorithms to form a stronger attack. The method considers PGD attacks generated via the untargeted cross-entropy loss as well as the targeted DLR loss, in addition to the targeted FAB attack and the black-box Square attack [Andriushchenko et al., 2020]. Again, the end-to-end gradient of the smoothed classifier is available to the adversary.

5.3 POLICY NETWORK ARCHITECTURE

Since the policy network $\beta_\theta(\cdot)$ should treat clean and attacked inputs differently, its task is closely related to adversary detection. We adapt the detection architecture introduced in Metzen et al. [2017] for our policy network. Our policy takes advantage of the two available models $g(\cdot)$ and $h(\cdot)$ by using the intermediate features of both networks. The modified architecture is shown in Figure 1.

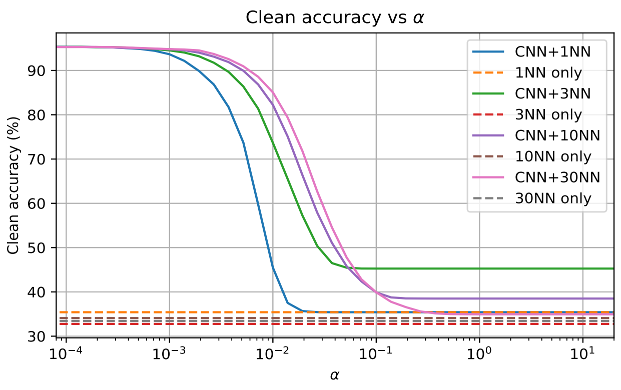
5.4 TRAINING THE POLICY

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

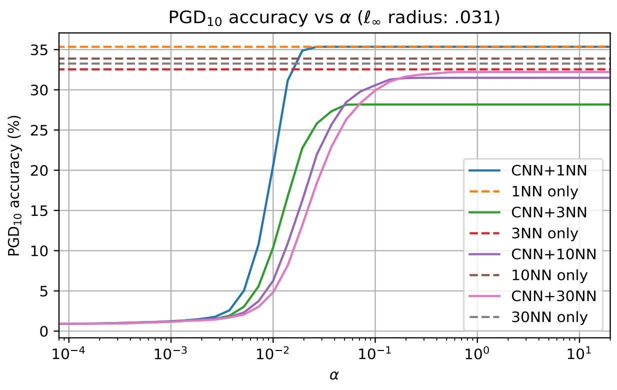
- **Direct back-propagation:** Directly optimize the loss function via the following optimization problem:

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

where ℓ_{CE} is the cross-entropy loss for logits and $y \in [c]$ is the label corresponding to x. This optimiza-



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

tion 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 and test sets. 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 a test 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 policy network does not receive a strong incentive to choose $g(\cdot)$ when it detects an attack targeting $h(\cdot)$. As a result, 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 "pseudo-label" $\widetilde{\alpha}$ is sometimes difficult to determine. Since the composite model g^{θ}_{CNN} is differentiable, the adversary can adapt to the policy network $\beta_\theta(\cdot)$. 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:

$$\ell_{\text{composite}}\big(\theta, (x,y)\big) = \Big(\ell_{\text{CE}}\big(g^{\theta}_{\text{CNN}}(x), y\big) + c_1\Big) \times \Big(c_2 + \big(c_3 - \text{sgn}(\alpha^\star(x) - \alpha_0)\beta_\theta(x)\big)_+\Big), \quad (9)$$

where the first group of terms is the direct loss component, the second group of terms is the supervised loss component, and α_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 $\alpha_t < \alpha_0$, and increase $\beta_\theta(\cdot)$ and prefer $h(\cdot)$ when $\alpha_t > \alpha_0$.

During training, we aim to approximate this optimal policy by using attacks that exploit the gradient of the policy network and diversifying the radii and the types of the attack.

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 used as the robust direction oracle $h(\cdot)$, we explore different values of K ranging from 1 to 100, and use images that are standardized per channel. The accuracy for the above K values at various choices of α is shown in Figure 5.

Figures 5 confirm 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 classifier 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 to 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.2, 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 generated using the end-to-end gradient of $g^{\alpha}_{\text{CNN}}(\cdot)$.

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 attacked accuracy increases with α. When the attack targets $h(\cdot)$, the attacked 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 attacked accuracy increases with α.

6.3 ROBUST NEURAL NETWORK SMOOTHING WITH ADAPTIVE STRENGTH

Finally, we evaluate the performance of the accelerated version of the adaptive composite model $g^{\theta}_{\text{CNN}}(\cdot)$ using the

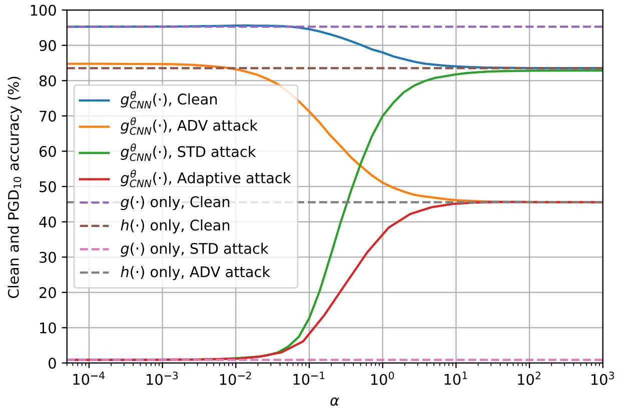


Figure 6: The performance of the smoothed model $g^{\alpha}_{\text{CNN}}(\cdot)$. "STD attack", "ADV attack", and "Adaptive attack" refer to the PGD$_{10}$ attack generated using the gradient of $g(\cdot)$, $h(\cdot)$, and $g^{\alpha}_{\text{CNN}}(\cdot)$ respectively, with ϵ set to 0.031.

CIFAR-10 and the CIFAR-100 datasets. We consider ℓ_2 and ℓ_∞ attacks and use different robust neural networks for $h(\cdot)$. During validation, the PGD$_{20}$ attack uses $\epsilon = 0.031$ for ℓ_∞ attacks and $\epsilon = 0.5$ for ℓ_2 attacks. In all experiments, the hyperparameters for the composite loss function (9) are $c_1 = 0.5$, $c_2 = 0.8$, and $c_3 = 3.2$. The value of α_0 for each experiment setting is reported in Table 1. The AdamW optimizer [Kingma and Ba, 2015] is used for optimization.

In all experiments, the training inputs for the policy $\beta_\theta(\cdot)$ includes clean data, and PGD$_{10}$ data with six different presumed α_t settings (same radius as the validation attack). Note that since $g(\cdot)$ and $h(\cdot)$ are not updated throughout training, these attacked data can be pre-computed. In addition, different adaptive training data are included for the following three attack settings. The test accuracy of $g^{\theta}_{\text{CNN}}(\cdot)$ in each setting is shown in Table 1. In the table, we also compare our method to the reported results of the existing works aiming to improve this trade-off.

- **A: Gray-box PGD:** We validate the smoothed model $g^{\theta}_{\text{CNN}}(\cdot)$ with a gray-box PGD attack. The training data thus additionally includes gray-box PGD attacks. The non-accelerated smoothing procedure (6) is applied.
- **B: White-box PGD:** We validate $g^{\theta}_{\text{CNN}}(\cdot)$ with a white-box PGD attack. Hence, we replace the training-time attack with a white-box PGD attack as well. Moreover, we randomize the radius and the number of steps of the training-time PGD attack to cover a larger portion of the adversarial input space. The detailed training settings are reported in Appendix A.1. The accelerated smoothing procedure (7) is applied.
- **C: White-box AutoAttack:** An AutoAttack adversary is used to evaluate $g^{\theta}_{\text{CNN}}(\cdot)$. During training, we use a cross-entropy-loss white-box PGD adversary and targeted-DLR-loss white-box PGD adversaries targeting all nine false classes. To further diversify the attack,

Scenario	Dataset	Type (ϵ)	$g(\cdot)$			$h(\cdot)$			
			Architecture	Clean	PGD$_{20}$	Architecture	Clean	PGD$_{20}$	AutoAttack
I	CIFAR-10	ℓ_∞ (.031)	ResNet18†	95.28%	0.12%	AT ResNet18†	83.53%	45.15%	41.33%
II	CIFAR-10	ℓ_2 (.5)	ResNet18†	95.28%	2.57%	AT ResNet18	85.37%	64.92%	
III	CIFAR-10	ℓ_∞ (.031)	ResNet18†	95.28%	0.12%	TRADE WRN34‡	84.92%	57.16%	
IV	CIFAR-100	ℓ_∞ (.031)	Mixup PRN18	78.46%	0.00%	AT ResNet18	50.37%	25.71%	

Scenario	α_0	Attack Setting A		Attack Setting B		Attack Setting C	
		Clean	PGD$_{20}$	Clean	PGD$_{20}$	Clean	AutoAttack
I	53	94.92%	45.94%	93.14%	42.93%	88.51%	34.88%
II	21	93.53%	64.55%	93.53%	64.49%		
III	335	95.04%	57.93%	94.11%	54.97%		
IV	45	78.01%	24.45%	76.66%	23.12%		

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]. The AutoAttack results for scenarios II, III, IV will be added in future versions.

Table 1: Summary of validation 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.

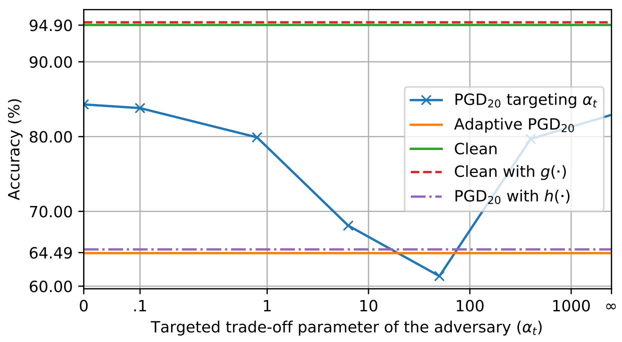


Figure 7: Clean and PGD$_{10}$ adversarial accuracies with different target α_t's. The solid lines are the results of $g^\theta_{\text{CNN}}(\cdot)$ under attack setting B.

we randomly initialize the PGD adversary and add a momentum term with a random strength, in addition to randomizing the radius and the step size. The accelerated smoothing procedure (7) is applied.

For setting A, the smoothed classifier is able to achieve the same level of attacked accuracy as $h(\cdot)$ while retaining a similar level of clean accuracy as $g(\cdot)$.

For setting B, the performance slightly deteriorates on both the accuracy side and the robustness side, as the attack is able to deliberately avoid the portion of the adversarial input space recognized by $\beta_\theta(\cdot)$, and thus the task becomes harder. However, the improvement in the robust-accuracy trade-off over a standalone $h(\cdot)$ is still substantial.

Additionally, Figure 7 compares the gray-box PGD$_{10}$ attacks and the PGD$_{10}$ attacks that assume a fixed value for α_t. The adaptive model from the second row of Table 1 is used. The adversary targeting $\alpha_t = 20$ provides the most powerful attack, decreasing the accuracy to 61.35%, lower than the accuracy on adaptive PGD data, but only 3.57% worse than the PGD$_{10}$ accuracy of the robust model $h(\cdot)$. This result confirms that the composite structure of $g^\theta_{\text{CNN}}(\cdot)$ perplexes the optimization landscape for the PGD adversary, making it harder to generate worst-case perturbations. Note that the main source of the robustness of g^θ_{CNN} is the inherent robustness of $h(\cdot)$ and the representability of the policy, rather than the sophisticated loss landscape.

For setting C, the difficulty of the training problem further escalates, as AutoAttack explores a much larger portion of the adversarial input space. While the overall performance of $g^\theta_{\text{CNN}}(\cdot)$ is not as outstanding as settings A and B, the smoothed network still achieves a non-trivial level of robustness and boosts the clean accuracy. Since the attacked accuracy of $g(\cdot)$ is near zero, achieving a robust accuracy that is about 77% of that of $h(\cdot)$ implies that the policy correctly identifies around 77% of the adversarial inputs that can be correctly predicted, contradicting the implication of [Carlini and Wagner, 2017a].

The above results show that the policy network $\beta_\theta(\cdot)$ 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 of $h(\cdot)$ shows that $g^\theta_{\text{CNN}}(\cdot)$ significantly improves the accuracy-robustness trade-off. While the performance of $g^\theta_{\text{CNN}}(\cdot)$ depends on the type of attack, our method offers an opportunity of balancing robustness and accuracy, especially in the setting of having modestly adversarial attacks.

On the CIFAR-100 dataset, testing 10000 clean examples with a batch size of 400 takes 2.246 seconds with an Nvidia

V100 GPU using a PreActResNet18 model. Testing these images using the accelerated $g^{\theta}_{\text{CNN}}(\cdot)$ obtained via (7) requires 5.845 seconds.

Since the proposed method does not make assumptions about 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 adapts an adversarial input detector into a policy. The policy improves the accuracy-robustness trade-off of neural networks against the attack types presented during training by adaptively adjusting the mixture of the outputs of an accurate model and a robust network. 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. Since the developed theoretical analysis proves the existence of an accurate and robust classifier built via our approach, future advancements in the detection of adversarial examples will enable the construction of robust classifiers that avoids the robustness-accuracy trade-off. Furthermore, certified robustness radii are proved for each value of the smoothing strength α. This work allows future research to specialize in either accuracy or robustness without fearing compromising 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.

Ahmed Aldahdooh, Wassim Hamidouche, and Olivier Déforges. Selective and features based adversarial example detection. *arXiv preprint arXiv:2103.05354*, 2021a.

Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza, and Olivier Déforges. Adversarial example detection for dnn models: A review and experimental comparison. *arXiv preprint arXiv:2105.00203*, 2021b.

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.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *Computer Vision - ECCV 2020 - 16th European Conference*, 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*, 2021.

Yatong Bai, Tanmay Gautam, Yu Gai, and Somayeh Sojoudi. Practical convex formulation of robust one-hidden-layer neural network training. *American Control Conference*, 2022a.

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*, 2022b.

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

Nicholas Carlini and David Wagner. *Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods*, page 3–14. 2017a.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pages 39–57, 2017b.

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.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020.

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.

Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems*, 2020.

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 DETAILED TRAINING SETTINGS AND HYPERPARAMETER TUNING GUIDELINES

This part of the appendix describes the detailed experiment settings used in Table 1. For all training data, we used "random horizontal flip" and "random crop" data augmentation techniques. The α_t settings for the non-adaptive PGD_{10} adversarial data are shown in Table 2. To avoid overfitting to the PGD attacked data in the training set, we use randomize the adaptive PGD attack directly targeting $g^{\theta}_{\text{CNN}}(\cdot)$. The attack radius is different for each training batch, and is randomly drawn from the uniform distribution supported on $[0.4 \times \epsilon_{\text{val}}, 1.1 \times \epsilon_{\text{val}}]$, where ϵ_{val} is the validation radius. We also draw a random number r from the χ^2 distribution with three degrees of freedom, and use floor$(r + 8)$ as the step size. The χ^2 distribution is chosen because it is non-negative, and has a long tail on the right side. Additionally, to accelerate convergence, we provide a pseudo-target $\alpha^\star = +\infty$ for the adaptive PGD data. 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 weight decay strength of the AdamW optimizer is set to 5×10^{-4}.

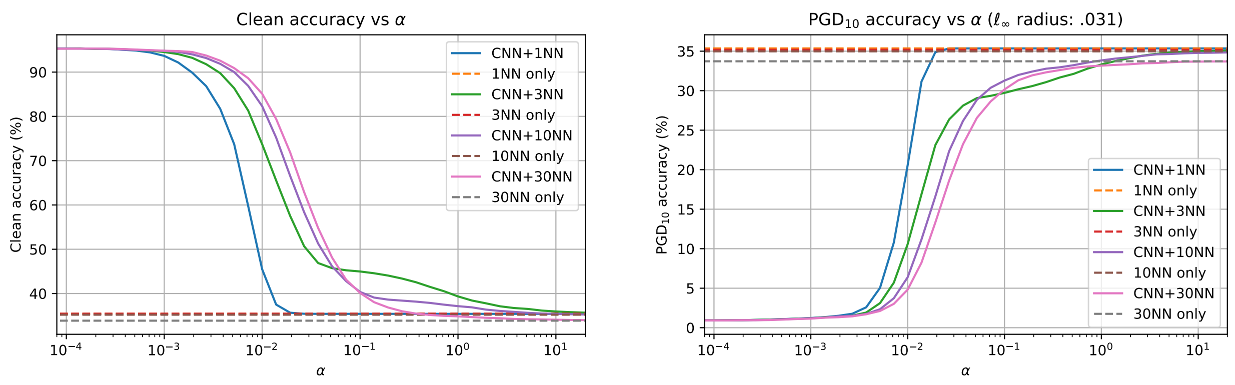
CIFAR-10 AT ℓ_∞	0	0.2512	1.819	13.18	95.50	691.8	5012	$+\infty$
CIFAR-10 AT ℓ_2	0	0.1	0.7943	6.310	50.12	398.1	3162	$+\infty$
CIFAR-10 TRADE ℓ_∞	0	0.2512	1.905	14.45	109.6	831.8	6310	$+\infty$
CIFAR-100 AT ℓ_∞	0	0.2512	1.819	13.18	95.50	691.8	5012	$+\infty$

Table 2: The α_t values of non-adaptive PGD training data. They are evenly drawn on a segment in the log space centered at α_0.

The process of choosing c_1, c_2, and c_3 for the composite loss function (9) is guided via the following intuition: if the policy gives a very large $\beta(x)$ for all input x, then this is a clear sign of an insufficient motivation for the policy to choose the standard network $g(\cdot)$, and the supervised learning loss component should be emphasized more.

Additionally, the value for α_0 is the targeted smoothing strength at which $g(\cdot)$ and $h(\cdot)$ achieves the same accuracy on the validation set. The values reported in Table 1 are obtained by plotting the accuracy of $g(\cdot)$ and $h(\cdot)$ at each α_t setting and visually inspecting the intersection. Thus, the values are not very precise. However, during training, since we only use the sign of $\alpha_t - \alpha_0$ to determine the target for $\beta_\theta(\cdot)$, a rough estimation of α_0 is sufficient.

A.2 SMOOTHING WITH DISTANCE-WEIGHTED K-NN CLASSIFIERS



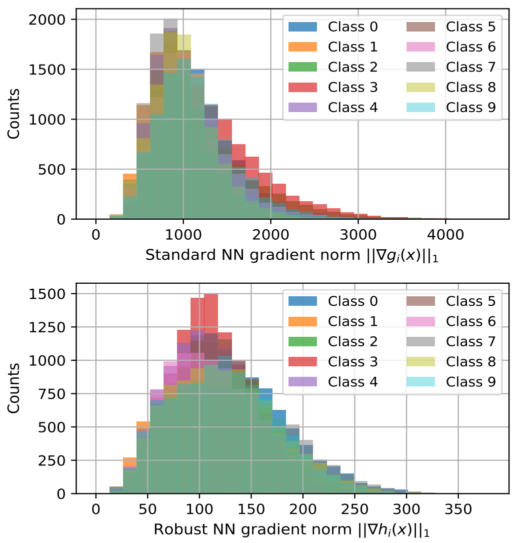
(a) Clean accuracy for each α for various values of K.

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

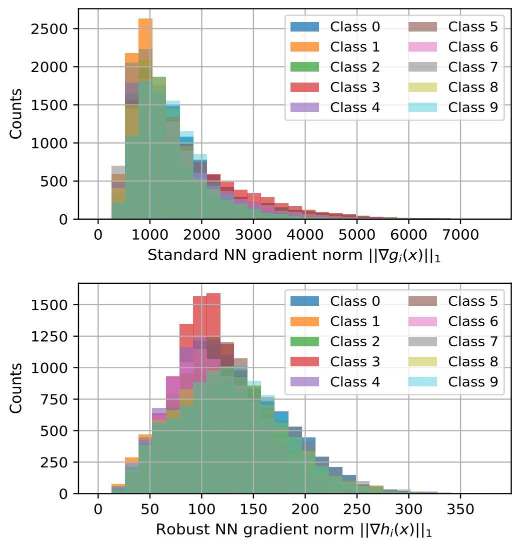
Figure 8: Evaluating the performance of the smoothed neural network based on distance-weighted K-NN classifiers.

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 8. 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 $h(\cdot)$.

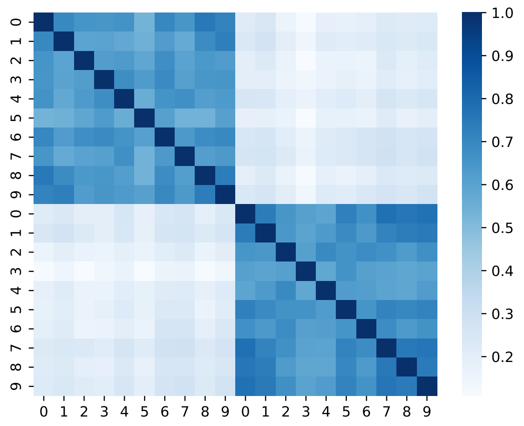
A.3 GRADIENT MAGNITUDE DISTRIBUTIONS WITH ADVERSARIAL DATA



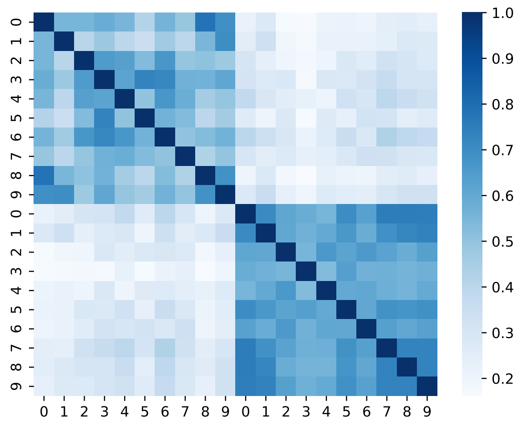
(a) The distributions with PGD_{10} adversarial data targeting $g(\cdot)$.



(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 9: 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 9a and 9c 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 9b and 9d 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 $g^{\alpha}_{\text{CNN}}(x)$.

A.4 EVALUATING THE ADAPTIVE POLICY WITH DIFFERENT ATTACK RADII

Figure 10 presents the accuracy of the adaptive model $g^{\theta}_{\text{CNN}}(\cdot)$ under the PGD_{20} attack with various attack radii ϵ_{val}. When ϵ_{val} is small, the efficacy of the adaptively smoothed model slightly drops, possibly because it is harder to differentiate an attacked image with a small radius from a clean image. However, when ϵ_{val} is large, the performance of the adaptively smoothed model is similar to that of the robust base model.

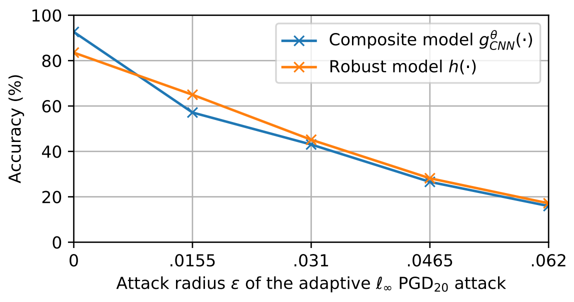


Figure 10: PGD$_{20}$ accuracies with different values of ϵ.

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

$$\|G_i^{x+\delta}(x') - G_i^{x+\delta}(x+\delta)\|_{p*} \le L_i\epsilon; \qquad \|G_j^{x+\delta}(x') - G_i^{x+\delta}(x+\delta)\|_{p*} \le L_j\epsilon. \tag{10}$$

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

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

and therefore,

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

$$\begin{aligned}
\leq & \Big| H_i(x,\delta) - H_j(x,\delta) \Big| + \Big| \big(G_i^{x+\delta}(x') - G_j^{x+\delta}(x') \big)^\top \delta \Big| \\
& + \alpha \Big| \big\| G_i^{x+\delta}(x+\delta) \big\|_{p*} - C_i(x,\delta) \big\| G_i^{x+\delta}(x) \big\|_{p*} \Big| + \alpha \Big| \big\| G_j^{x+\delta}(x+\delta) \big\|_{p*} - C_i(x,\delta) \big\| G_j^{x+\delta}(x) \big\|_{p*} \Big|, \quad (11)
\end{aligned}$$

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

The individual terms in (11) 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*

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

Lemma 5. *It holds that*

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

Lemma 6. *It holds that*

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

Following Lemmas 4, 5, and 6 (due to symmetry, Lemma 4 also holds for class j), the expression (11) can be bounded via

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

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

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

and thus,

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

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 $\operatorname{sgn}\big(g^\alpha_{K\text{-NN},i}(x+\delta) - g^\alpha_{K\text{-NN},j}(x+\delta) \big) = \operatorname{sgn}\big(g^\alpha_{K\text{-NN},i}(x) - g^\alpha_{K\text{-NN},j}(x) \big)$. $\square$

B.1.1 Proof of Lemma 4

Note that

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

where

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

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 (10). □

B.1.2 Proof of Lemma 5

It holds that

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

where the last inequality arises from the assumption (10). □

B.1.3 Proof of Lemma 6

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

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

where

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

where the second last inequality arises from the assumption (10) 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

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

which is the desired result. $\square$

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

$$\begin{aligned}
\big|R_k(x) - R_k(x+\delta)\big| &= \left|\frac{\|\nabla g_k(x)\|_{p*}}{\|\nabla h_k(x)\|_{p*}} - \frac{\|\nabla g_k(x+\delta)\|_{p*}}{\|\nabla h_k(x+\delta)\|_{p*}}\right| \\
&= \frac{\big|\|\nabla g_k(x)\|_{p*}\|\nabla h_k(x+\delta)\|_{p*} - \|\nabla g_k(x+\delta)\|_{p*}\|\nabla h_k(x)\|_{p*}\big|}{\|\nabla h_k(x)\|_{p*}\|\nabla h_k(x+\delta)\|_{p*}} \\
&\leq \frac{\|\nabla g_k(x)\|_{p*}\|\nabla h_k(x+\delta) - \nabla h_k(x)\|_{p*} + \|\nabla h_k(x)\|_{p*}\|\nabla g_k(x+\delta) - \nabla g_k(x)\|_{p*}}{\|\nabla h_k(x)\|_{p*}\|\nabla h_k(x+\delta)\|_{p*}} \\
&\leq \frac{\|\nabla g_k(x)\|_{p*}L_k^h\|\delta\|_p + \|\nabla h_k(x)\|_{p*}L_k^g\|\delta\|_p}{\|\nabla h_k(x)\|_{p*}\|\nabla h_k(x+\delta)\|_{p*}} \\
&= \frac{L_k^h R_k(x) + L_k^g}{\|\nabla h_k(x+\delta)\|_{p*}}\|\delta\|_p \\
&\leq \frac{L_k^h R_k(x) + L_k^g}{m_k^h}\|\delta\|_p.
\end{aligned} \tag{12}$$

It holds that

$$\begin{aligned}
|g_{\text{CNN},k}^\alpha(x+\delta) - g_{\text{CNN},k}^\alpha(x)| &= \left|\frac{g_k(x+\delta) + \alpha R_k(x+\delta)h_k(x+\delta)}{1+\alpha R_k(x+\delta)} - \frac{g_k(x) + \alpha R_k(x)h_k(x)}{1+\alpha R_k(x)}\right| \\
&\leq \underbrace{\left|\frac{g_k(x+\delta)}{1+\alpha R_k(x+\delta)} - \frac{g_k(x)}{1+\alpha R_k(x)}\right|}_{=:G_k(x,\delta,\alpha)} + \alpha\underbrace{\left|\frac{R_k(x+\delta)h_k(x+\delta)}{1+\alpha R_k(x+\delta)} - \frac{R_k(x)h_k(x)}{1+\alpha R_k(x)}\right|}_{=:H_k(x,\delta,\alpha)}.
\end{aligned} \tag{13}$$

Bounding the first term using (12) gives

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

Similarly, the second term is bounded using (12) as

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

Hence, (13) yields that

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

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

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

Therefore, it holds that

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

implying that

$$\arg\max_{i\in[c]} g^{\alpha}_{\text{CNN},i}(x+\delta) = y \quad \text{if} \quad \left(\arg\max_{i\in[c]} g_i(x+\delta) = y \;\text{ or }\; \arg\max_{i\in[c]} h_i(x+\delta) = y\right),$$

which leads to the desired statement. $\square$