Improving the Accuracy-Robustness Trade-Off of Classifiers via Adaptive Smoothing*

Yatong Bai[1], Brendon G. Anderson[1], Aerin Kim[2], Somayeh Sojoudi[1]
[1] *University of California, Berkeley* [2] *Scale AI*
{yatong_bai, bganderson, sojoudi}@berkeley.edu, aerinykim@gmail.com

– ***Abstract*** –

While prior research has proposed a plethora of methods that build neural classifiers robust against adversarial robustness, practitioners are still reluctant to adopt them due to their unacceptably severe clean accuracy penalties. This paper significantly alleviates this accuracy-robustness trade-off by mixing the output probabilities of a standard classifier and a robust classifier, where the standard network is optimized for clean accuracy and is not robust in general. We show that the robust base classifier's confidence difference for correct and incorrect examples is the key to this improvement. In addition to providing intuitions and empirical evidence, we theoretically certify the robustness of the mixed classifier under realistic assumptions. Furthermore, we adapt an adversarial input detector into a mixing network that adaptively adjusts the mixture of the two base models, further reducing the accuracy penalty of achieving robustness. The proposed flexible method, termed "adaptive smoothing", can work in conjunction with existing or even future methods that improve clean accuracy, robustness, or adversary detection. Our empirical evaluation considers strong attack methods, including AutoAttack and adaptive attack. On the CIFAR-100 dataset, our method achieves an 85.21% clean accuracy while maintaining a 38.72% ℓ_∞-AutoAttacked ($\epsilon = 8/255$) accuracy, becoming the second most robust method on the RobustBench CIFAR-100 benchmark as of submission, while improving the clean accuracy by ten percentage points compared with all listed models. The code that implements our method is available at https://github.com/Bai-YT/AdaptiveSmoothing.

1. Introduction

Neural networks are vulnerable to adversarial attacks in various applications, including computer vision and audio [35, 60], natural language processing [33], and control systems [42]. Due to the widespread application of neural classifiers, ensuring their reliability in practice is paramount.

To mitigate this susceptibility, researchers have explored "adversarial training" (AT) [15, 16, 35, 50, 86], an empirical technique that trains neural networks on adversarial examples to build robust models. Since its initial introduction, multiple improved variants of adversarial training have been devised to enhance the performance of robust classifiers. On the theoretical side, researchers have also considered ensuring the certified robustness of neural classifiers, which mathematically

*This work is an extension of [14], and was supported by grants from ONR, NSF, and C3 AI.

guarantees a model's robustness to adversarial attacks [7, 9, 57] within a radius. "Randomized smoothing" (RS) is one such method that seeks to achieve certified robustness of an existing model at inference time [25, 53], with improved variants incorporating dimension reduction methods [67] and denoising diffusion components [19], thereby certifying much larger radii. Recent work [8] has demonstrated that a locally biased smoothing approach can improve over traditional data-blind randomized smoothing. However, this method is limited to binary classification problems and suffers from the performance bottleneck of its underlying one-nearest-neighbor classifier.

Despite the emergence of these proposed remedies to the adversarial robustness issue, many practitioners are reluctant to adopt them. As a result, existing publicly available services are still vulnerable [18, 43], presenting severe safety risks. One important reason for this reluctance is the potential for significantly reduced model performance on clean data. Specifically, some previous works have suggested a fundamental trade-off between accuracy and robustness [78, 84]. Since the sacrifice in unattacked performance is understandably unacceptable in real-world scenarios, developing robust classifiers with minimal clean accuracy degradation is crucial.

Fortunately, recent research has argued that it should be possible to simultaneously achieve robustness and accuracy on benchmark datasets [82]. To this end, variants of adversarial training that improve the accuracy-robustness trade-off have been proposed, including TRADES [84], Interpolated Adversarial Training [51], and many others [13, 17, 68, 77, 80, 83]. However, despite these improvements, degraded clean accuracy is often an inevitable price of achieving robustness. Moreover, standard non-robust models often achieve enormous performance gains by pre-training on larger datasets. In contrast, the effect of pre-training on robust classifiers is less understood and may be less prominent [23, 31]. As a result, the performance gap between these existing works and the possibility guaranteed in [82] is still huge.

This work builds upon locally biased smoothing [8] and makes a theoretically disciplined step towards reconciling adversarial robustness and clean accuracy, significantly closing this performance gap and thereby providing practitioners additional incentives for deploying robust models. We summarize our contributions below.

- In Section 3, under the observation that the K-nearest-neighbor (K-NN) classifier, a crucial component of locally biased smoothing, becomes a bottleneck of the overall performance, we replace the K-NN classifier with a robust neural network that can be obtained via various existing methods, and modify the smoothing formulation accordingly. The resulting formulation (5) is a convex combination of the output probabilities of a standard neural network and a robust neural network. When the robust neural network has a certified Lipschitz constant or is based on randomized smoothing, the combined classifier also has a certified robust radius. This section, along with the corresponding experiment results in Appendix A.1 and B.2, are also presented in our conference submission [14].
- In Section 4, we propose to adaptively adjust the mixture of a standard model and a robust model by adopting a type of adversary detector as a "mixing network". The mixing network adjusts the convex combination of the output probabilities from the two base networks, further improving the accuracy-robustness trade-off. We then empirically verify the robustness of the proposed method using gray-box and white-box projected gradient descent (PGD) attack, AutoAttack, and adaptive attacks, demonstrating that the mixing network is robust against the adversary types it is trained with. When the mixing network is trained with a carefully

designed adaptive AutoAttack, the composite model significantly gains clean accuracy while sacrificing little robustness. This section and the corresponding experiment results are entirely new relative to our conference submission [14], and are crucial for achieving the much improved accuracy-robustness trade-off over existing works.

Note that adaptive smoothing is agnostic to the method of obtaining standard and robust base models. Thus, adaptive smoothing can take advantage of pre-training on large datasets via the standard base classifier and benefit from ongoing advancements in robust training methods via the robust base classifier.

2. Background and Related Works

2.1. Notations

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 a natural number c, $[c] = \{1, 2, \ldots, c\}$. 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 $\mathcal{S}$.

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 $f : \mathbb{R}^d \to [c]$ can be obtained via $f(x) \in \arg\max_{i\in[c]} g_i(x)$. 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 $h(\cdot)$). Throughout this paper, we regard $g(\cdot)$ and $h(\cdot)$ as the base classifier logits. To denote their output probabilities, we use $\sigma \circ g(\cdot)$ and $\sigma \circ h(\cdot)$. Similarly, $\sigma \circ g_i(\cdot)$ denotes the predicted probability of the i^{th} class from $g(\cdot)$. Moreover, we use $\mathcal{D}$ to denote the set of all validation input-label pairs (x_i, y_i).

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 use PGD_T to denote the T-step PGD attack.

2.2. Related Adversarial Attacks and Defenses

The fast gradient sign method (FGSM) and PGD attacks based on the first-order maximization of the cross-entropy loss have traditionally been considered classic and straightforward attacks [35, 58]. However, these attacks have been shown to be insufficient as defenses designed against them are often easily circumvented [11, 12, 21, 66]. To this end, various attack methods based on alternative loss functions, Expectation Over Transformation, and black-box perturbations have been proposed. Such efforts include MultiTargeted attack loss [38], AutoAttack [29], adaptive attack [76], minimal distortion attack [28], and many others, even considering attacking test-time defenses [27]. The diversity of attack methods has led to the creation of benchmarks such as RobustBench [26] and ARES-Bench [54] to unify the evaluation of robust models.

On the defense side, while adversarial training [58] and TRADES [84] have seen enormous success, such methods are often limited by a significantly larger amount of required training data [72]. Initiatives that construct more effective training data via data augmentation [36, 37, 69] and gen-

erative models [73, 81] have successfully produced more accurate and robust models. Improved versions of adversarial training [44, 63, 74, 79] have also been proposed.

Previous research has developed models that improve robustness by dynamically changing at test time. Specifically, Input-Adaptive Inference improves the accuracy-robustness trade-off by appending side branches to a single network, allowing for early-exit predictions [41]. Other initiatives that aim to enhance the accuracy-robustness trade-off include using the SCORE attack during training [64] and applying adversarial training for regularization [87].

Moreover, ensemble-based defenses, such as random ensemble [55], diverse ensemble [1, 3, 65], and Jacobian ensemble [24], have been proposed. In comparison, this work is distinct in that our mixing scheme uses two separate classifiers, incorporating one non-robust component while still ensuring the adversarial robustness of the overall design. By doing so, we take advantage of the high performance of modern pre-trained models, significantly alleviating the accuracy-robustness trade-off and achieving much higher overall performances. Additionally, unlike some previous ensemble initiatives, our formulation is deterministic and straightforward (in the sense of gradient propagation), making it easier to evaluate its robustness properly. The work [49] also explored assembling an accurate classifier and a robust classifier, but the method considered robustness against distribution shift in a non-adversarial setting and was based on different intuitions. After the submission of this paper, the work [85] also considered leveraging the power of a pair of standard and robust classifiers. However, instead of mixing the outputs, the authors proposed to distill a new model from the two base classifiers. While this approach also yielded impressive results, the distillation process is time-consuming.

2.3. Locally Biased Smoothing

Randomized smoothing, popularized by [25], achieves robustness at inference time by replacing the standard classifier $f(\cdot)$ with the smoothed classifier

$$\widetilde{f}(x) \in \arg\max_{i \in [c]} \mathbb{P}_{\delta \sim \mathcal{S}}\big[f(x+\delta) = i\big],$$

where $\mathcal{S}$ is a smoothing distribution, for which a common choice is a Gaussian distribution.

Note that $\mathcal{S}$ is independent of the input x and it is standard to use $\mathcal{S}$ that is zero-mean. The authors of [8] have shown that data-invariant smoothing enlarges the region of the input space at which the prediction of $\widetilde{f}(\cdot)$ stays constant. Such an operation may actually degrade both clean and robust accuracy (the limiting case is when $\widetilde{f}(\cdot)$ becomes a constant classifier). Furthermore, when $f(\cdot)$ is a linear classifier, the zero-mean restriction on $\mathcal{S}$ leaves $f(\cdot)$ unchanged. That is, randomized smoothing with a zero-mean distribution cannot help robustify even the most simple linear classifiers. To overcome these limitations, [8] allowed $\mathcal{S}$ to be input-dependent (denoted by $\mathcal{S}_x$) and nonzero-mean and searched for distributions $\mathcal{S}_x$ that best robustify $\tilde{f}(\cdot)$ with respect to the data distribution. The resulting scheme is "locally biased smoothing".

It is shown in [8] that, up to a first-order linearization of the base classifier, the optimal locally biased smoothing distribution $\mathcal{S}_x$ shifts the input point in the direction of its true class. Formally, for a binary classifier of the form $f(x) = \text{sign}(g(x))$ with continuously differentiable $g(\cdot)$, maximizing the robustness of $\widetilde{f}(\cdot)$ around x over all distributions $\mathcal{S}_x$ with bounded mean yields the optimal locally biased smoothing classifier given by

$$\widetilde{f}(x) = \text{sign}(\widetilde{g}(x)), \quad \text{where} \quad \widetilde{g}(x) = g(x) + \gamma y(x) \|\nabla g(x)\|_{p*},$$

where $y(x) \in \{-1, 1\}$ is the true class of x, and where $\gamma \geq 0$ is the (fixed) bound on the distribution mean (i.e., $\|\mathbb{E}_{\delta \sim \mathcal{S}_x}[\delta]\|_p \leq \gamma$).

Intuitively, this optimal locally biased smoothing classifier shifts the input along the direction $\nabla g(x)$ when $y(x) = 1$ as a means to make the classifier more likely to label x into class 1, and conversely shifts the input along the direction $-\nabla g(x)$ when $y(x) = -1$. Of course, during inference, the true class $y(x)$ is generally unavailable, and therefore [8] uses a "direction oracle" $h(x) \in \{-1, 1\}$ as a surrogate for $y(x)$, resulting in the locally biased smoothing classifier

$$f^\gamma(x) = \text{sign}(h^\gamma(x)), \quad \text{where} \quad h^\gamma(x) = g(x) + \gamma h(x) \|\nabla g(x)\|_{p*}. \tag{1}$$

Notice that unlike randomized smoothing, the computation (1) is deterministic, which is a consequence of the closed-form optimization over $\mathcal{S}_x$.

In contrast to data-invariant randomized smoothing, the direction oracle $h(\cdot)$ is learned from data as a means to incorporate the data distribution directly into the manipulation of the locally biased randomized smoothing classifier's decision boundaries. This allows for increases in nonlinearity when the data implies that such nonlinearities are beneficial for robustness, resolving a fundamental limitation of data-invariant smoothing. In general, the direction oracle should come from an inherently robust classifier. Since such a robust classifier $h(\cdot)$ is often less accurate, the value γ can be viewed as a trade-off parameter, as it encodes the amount of trust put into the direction oracle. The authors of [8] show that when the direction oracle is chosen to be a one-nearest-neighbor classifier, locally biased smoothing outperforms traditional randomized smoothing in binary classification.

2.4. Adversarial Input Detectors

It has been shown that adversarial inputs can be detected via various methods. For example, [59] proposes to append an additional detection branch to an existing neural network, and uses adaptive adversarial data to train the detector in a supervised fashion. However, [20] has shown that it is possible to bypass this detection method. They constructed adversarial examples via the C&W attacks [21] and simultaneously targeted the classification branch and the detection branch by treating the two branches as an "augmented classifier". According to [20], 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 [59] can recognize the state-of-the-art AutoAttack adversaries with a high success rate.

The literature has also considered alternative detection methods that mitigate the above challenges faced by detectors trained in a supervised fashion [22]. Such initiatives include unsupervised detectors [4, 5] and re-attacking [2]. Since universally effective detectors have not yet been discovered, this paper focuses on transferring the properties of the existing detector toward better overall robustness. Future advancements in the field of adversary detection can further enhance the performance of our method.

3. Using a Robust Neural Network as the Smoothing Oracle

Locally biased smoothing was designed for binary classification problems, making it restrictive in practice. Here, we first extend it to the multi-class setting by treating the output $h_i^\gamma(x)$ of each class independently, giving rise to:

$$h^\gamma_{\text{smo1},i}(x) \coloneqq g_i(x) + \gamma h_i(x)\|\nabla g_i(x)\|_{p*}, \quad \forall i \in [c]. \tag{2}$$

Note that if $\|\nabla g_i(x)\|_{p*}$ is large for some i, then $h^\gamma_{\text{smo1},i}(x)$ can be large even if both $g_i(x)$ and $h_i(x)$ are small, potentially leading to incorrect predictions. To remove the effect of the magnitude difference across the classes, we propose a normalized formulation as follows:

$$h^\gamma_{\text{smo2},i}(x) \coloneqq \frac{g_i(x) + \gamma h_i(x)\|\nabla g_i(x)\|_{p*}}{1 + \gamma\|\nabla g_i(x)\|_{p*}}, \quad \forall i \in [c]. \tag{3}$$

The parameter γ adjusts the trade-off between clean accuracy and robustness. When $\gamma = 0$, it holds that $h^\gamma_{\text{smo2},i}(x) \equiv g_i(x)$ for all x, i. When $\gamma \to \infty$, it holds that $h^\gamma_{\text{smo2},i}(x) \to h_i(x)$ for all x, i.

With the mixing procedure generalized to the multi-class setting, we now discuss the choice of the smoothing oracle $h_i(\cdot)$. While K-NN classifiers are relatively robust and can be used as the oracle, their representation power is too weak. On the CIFAR-10 image classification task [48], K-NN only achieves around 35% accuracy on clean test data. In contrast, an adversarially trained ResNet [39] can reach 50% accuracy on attacked test data [58]. This lackluster performance of K-NN becomes a significant bottleneck in the mixed classifier's accuracy-robustness trade-off. To this end, we replace the K-NN model with a robust neural network. The robustness of this network can be achieved via various methods, including adversarial training, TRADES, and traditional randomized smoothing.

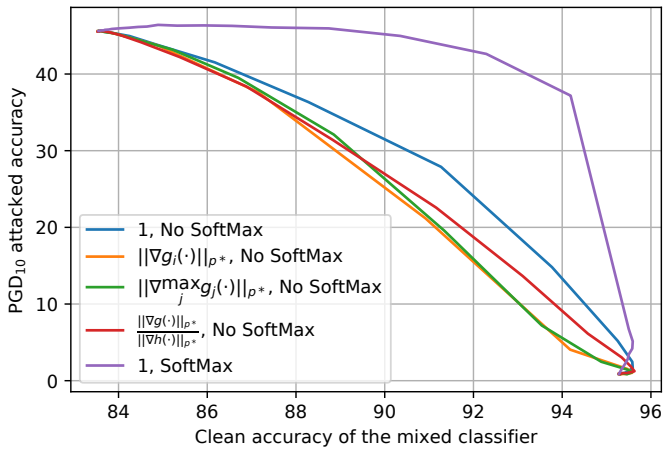
Further scrutinizing (3) leads to the question of whether $\|\nabla g_i(x)\|_{p*}$ is the best choice for adjusting the mixture of $g(\cdot)$ and $h(\cdot)$. In fact, this gradient magnitude term is a result of the assumption that $h(x) \in \{-1, 1\}$, which is the setting considered in [8]. Here, we no longer have this assumption. Instead, we assume both $g(\cdot)$ and $h(\cdot)$ to be differentiable. Thus, we further generalize the formulation to

$$h^\gamma_{\text{smo3},i}(x) \coloneqq \frac{g_i(x) + \gamma R_i(x) h_i(x)}{1 + \gamma R_i(x)}, \quad \forall i \in [c], \tag{4}$$

where $R_i(x)$ is an extra scalar term that can potentially depend on both $\nabla g_i(x)$ and $\nabla h_i(x)$ to determine the "trustworthiness" of the base classifiers. Here, we empirically compare four options for $R_i(x)$, namely, 1, $\|\nabla g_i(x)\|_{p*}$, $\|\nabla \max_j g_j(x)\|_{p*}$, and $\frac{\|\nabla g_i(x)\|_{p*}}{\|\nabla h_i(x)\|_{p*}}$. In Appendix B.1, we explain how these four options were designed.

Another design question is whether $g(\cdot)$ and $h(\cdot)$ should be the pre-softmax logits or the post-softmax probabilities. Since most attack methods are designed based on logits, the output of the mixed classifier should be logits rather than probabilities to avoid gradient masking, an undesirable phenomenon that makes it hard to evaluate the proposed method properly. Thus, we have the following two options that make the mixed model compatible with existing gradient-based attacks:

1. Use the logits for both $g(\cdot)$ and $h(\cdot)$.
2. Use the probabilities for both base classifiers, and then convert the mixed probabilities back to logits. The required "inverse-softmax" operator is given simply by the natural logarithm, and does not change the prediction.



- "No Softmax" represents Option 1, i.e., use the logits for $g(\cdot)$ and $h(\cdot)$.
- "Softmax" represents Option 2, i.e., use the probabilities for $g(\cdot)$ and $h(\cdot)$, with a natural log applied to the assembled $h^{\gamma}_{\text{smo3},i}(\cdot)$.

Figure 1. Comparing the "attacked accuracy versus clean accuracy" curve for various options for $R_i(x)$. Note that with the best formulation, high clean accuracy can be achieved with very little sacrifice on robustness.

Figure 1 visualizes the accuracy-robustness trade-off achieved by different choices for $R_i(x)$. Here, the base classifiers are a pair of standard and adversarially trained ResNet-18s. This "clean accuracy versus PGD_{10}-attacked accuracy" plot conclude sthat $R_i(x) = 1$ gives the best accuracy-robustness trade-off, and $g(\cdot)$ and $h(\cdot)$ should be the probabilities. Appendix B.2 confirms this selection by repeating Figure 1 with alternative model architectures, different methods to train the robust base classifiers, and various attack budgets.

Our selection of $R_i(x) = 1$ differs from $R_i(x) = \|g_i(x)\|_{p*}$ used in [8]. Intuitively, [8] used linear classifier examples to motivate estimating the trustworthiness of the base models with their gradient magnitudes. However, when the base classifiers are highly nonlinear neural networks as in our case, while the local Lipschitzness of a base classifier still correlates with its robustness, its gradient magnitude is not always a good estimator of the local Lipschitzness. Appendix B.2 provides additional discussions on this matter. Additionally, Section 3.1 offers theoretical intuitions for favoring mixing probabilities over mixing logits.

With these design choices implemented, the formulation (4) can be re-parameterized as

$$h^{\alpha}_i(x) \coloneqq \log\Big((1-\alpha)\sigma \circ g_i(x) + \alpha \cdot \sigma \circ h_i(x)\Big), \quad \forall i \in [c], \tag{5}$$

where $\alpha = \frac{\gamma}{1+\gamma} \in [0, 1]$. We take $h^{\alpha}(\cdot)$ in (5), which is a convex combination of base classifier probabilities, as our proposed mixed classifier. Note that (5) calculates the mixed classifier logits so that the mixed classifier acts as a drop-in replacement for existing models which usually produce logits. Removing the logarithm operator gives the output probabilities.

3.1. Theoretical Certified Robust Radius

In this section, we derive certified robust radii for $h^{\alpha}(\cdot)$ introduced in (5), given in terms of the robustness properties of $h(\cdot)$ and the mixing parameter α. The results ensure that despite being more sophisticated than a single model, $h^{\alpha}(\cdot)$ cannot be easily conquered, even if an adversary attempts to adapt its attack methods to its structure. Such guarantees are of paramount importance for reliable deployment in safety-critical control applications. Note that while the focus of

this paper is improved empirical accuracy-robustness trade-off and the existing literature often considers empirical and certified robustness separately, we will discuss how the certified results in this section provide important insights into the empirical performance, as the underlying assumptions are realistic and (approximately) verifiable for many empirically robust models.

Noticing that the base model probabilities satisfy $0 \leq \sigma \circ g_i(\cdot) \leq 1$ and $0 \leq \sigma \circ h_i(\cdot) \leq 1$ for all i, we introduce the following generalized and tightened notion of certified robustness.

Definition 1. Consider a model $h : \mathbb{R}^d \to \mathbb{R}^c$ and an arbitrary input $x \in \mathbb{R}^d$. Further consider $y = \arg\max_i h_i(x)$, $\mu \in [0, 1]$, and $r \geq 0$. Then, $h(\cdot)$ is said to be **certifiably robust at x with margin μ and radius r** if $\sigma \circ h_y(x + \delta) \geq \sigma \circ h_i(x + \delta) + \mu$ for all $i \neq y$ and all $\delta \in \mathbb{R}^d$ such that $\|\delta\|_p \leq r$.

Intuitively, Definition 1 ensures that all points within a radius from a nominal point have the same prediction as the nominal point, with the difference between the top and runner-up probabilities no smaller than a threshold. For practical classifiers, the robust margin can be straightforwardly estimated by calculating the confidence gap between the predicted and the runner-up classes at an adversarial input obtained with strong attacks. As shown in the experiments in Section 5.1.2, if a practical robust model is robust at an input with a given radius, it is likely to be robust with a large margin.

Lemma 1. *Let $x \in \mathbb{R}^d$ and $r \geq 0$. If it holds that $\alpha \in [\frac{1}{2}, 1]$ and $h(\cdot)$ is certifiably robust at x with margin $\frac{1-\alpha}{\alpha}$ and radius r, then the mixed classifier $h^\alpha(\cdot)$ is robust in the sense that $\arg\max_i h_i^\alpha(x + \delta) = \arg\max_i h_i(x)$ for all $\delta \in \mathbb{R}^d$ such that $\|\delta\|_p \leq r$.*

Proof. Suppose that $h(\cdot)$ is certifiably robust at x with margin $\frac{1-\alpha}{\alpha}$ and radius r. Since $\alpha \in [\frac{1}{2}, 1]$, it holds that $\frac{1-\alpha}{\alpha} \in [0, 1]$. Let $y = \arg\max_i h_i(x)$. Consider an arbitrary $i \in [c] \setminus \{y\}$ and $\delta \in \mathbb{R}^d$ such that $\|\delta\|_p \leq r$. Since $\sigma \circ g_i(x + \delta) \in [0, 1]$, it holds that

$$\begin{aligned}
&\exp\left(h_y^\alpha(x + \delta)\right) - \exp\left(h_i^\alpha(x + \delta)\right) \\
&\qquad = (1 - \alpha)(\sigma \circ g_y(x + \delta) - \sigma \circ g_i(x + \delta)) + \alpha(\sigma \circ h_y(x + \delta) - \sigma \circ h_i(x + \delta)) \\
&\qquad \geq (1 - \alpha)(0 - 1) + \alpha(\sigma \circ h_y(x + \delta) - \sigma \circ h_i(x + \delta)) \\
&\qquad \geq (\alpha - 1) + \alpha\left(\tfrac{1-\alpha}{\alpha}\right) = 0.
\end{aligned}$$

Thus, it holds that $h_y^\alpha(x + \delta) \geq h_i^\alpha(x + \delta)$ for all $i \neq y$, and thus $\arg\max_i h_i^\alpha(x + \delta) = y = \arg\max_i h_i(x)$. □

While most existing provably robust results consider the special case with zero margin, we will show that models built via common methods are also robust with non-zero margins. We specifically consider two types of popular robust classifiers: Lipschitz continuous models (Theorem 1) and RS models (Theorem 3). Lemma 1 builds the foundation for proving these two theorems, which amounts to showing that Lipschitz and RS models are robust with non-zero margins and thus the mixed classifiers built with them are robust. Lemma 1 can also motivate future researchers to develop margin-based robustness guarantees for base classifiers so that they immediately grant robustness guarantees for mixed architectures.

Lemma 1 additionally provides further justifications for using probabilities instead of logits in the smoothing operation. Intuitively, $(1 - \alpha)\sigma \circ g_i(\cdot)$ is bounded between 0 and $1 - \alpha$, so as long as

α is relatively large (specifically, at least $\frac{1}{2}$), the detrimental effect of $g(\cdot)$ when subject to attack can be overcome by $h(\cdot)$. Had we used the logits $g_i(\cdot)$, since this quantity cannot be bounded, it would have been much harder to overcome the vulnerability of $g(\cdot)$.

Since we do not make assumptions on the Lipschitzness or robustness of $g(\cdot)$, Lemma 1 is tight. To understand this, we suppose that there exists some $i \in [c] \setminus \{y\}$ and $\delta \neq 0$ such that $\|\delta\|_p \leq r$ that make $\sigma \circ h_y(x+\delta) - \sigma \circ h_i(x+\delta) \coloneqq h_d$ smaller than $\frac{1-\alpha}{\alpha}$, indicating that $-\alpha h_d > \alpha - 1$. Since the only information about $g(\cdot)$ is that $\sigma \circ g_i(x+\delta) \in [0,1]$ and thus the value $\sigma \circ g_y(x+\delta) - \sigma \circ g_i(x+\delta) \coloneqq g_d$ can be any number between -1 and 1, it is possible that $(1-\alpha)g_d$ is smaller than $-\alpha h_d$. By (5), when $(1-\alpha)g_d < -\alpha h_d$, it holds that $h_y^\alpha(x+\delta) < h_i^\alpha(x+\delta)$, and thus $\arg\max_i h_i^\alpha(x+\delta) \neq \arg\max_i h_i(x)$.

Definition 2. A function $f \colon \mathbb{R}^d \to \mathbb{R}$ is called *ℓ_p-Lipschitz continuous* if there exists $L \in (0, \infty)$ such that $|f(x') - f(x)| \leq L\|x' - x\|_p$ for all $x', x \in \mathbb{R}^d$. The **Lipschitz constant** of such f is defined to be

$$\mathrm{Lip}_p(f) \coloneqq \inf \left\{ L \in (0,\infty) : |f(x') - f(x)| \leq L\|x' - x\|_p \text{ for all } x', x \in \mathbb{R}^d \right\}.$$

Assumption 1. The base model $h(\cdot)$ is robust in the sense that, for all $i \in \{1, 2, \ldots, n\}$, $\sigma \circ h_i(\cdot)$ is ℓ_p-Lipschitz continuous with Lipschitz constant $\mathrm{Lip}_p(h_i)$.

Theorem 1. *Suppose that Assumption 1 holds, and let $y = \arg\max_i h_i(x)$, where $x \in \mathbb{R}^d$ be arbitrary. Then, if $\alpha \in [\frac{1}{2}, 1]$, it holds that $\arg\max_i h_i^\alpha(x+\delta) = y$ for all $\delta \in \mathbb{R}^d$ such that*

$$\|\delta\|_p \leq r_{\mathrm{Lip},p}^\alpha(x) \coloneqq \min_{i \neq y} \frac{\alpha\big(\sigma \circ h_y(x) - \sigma \circ h_i(x)\big) + \alpha - 1}{\alpha\left(\mathrm{Lip}_p(\sigma \circ h_y) + \mathrm{Lip}_p(\sigma \circ h_i)\right)}. \tag{6}$$

Proof. Suppose that $\alpha \in [\frac{1}{2}, 1]$, and let $\delta \in \mathbb{R}^d$ be such that $\|\delta\|_p \leq r_{\mathrm{Lip},p}^\alpha(x)$. Furthermore, let $i \in [c] \setminus \{y\}$. It holds that

$$\begin{aligned}
&\sigma \circ h_y(x+\delta) - \sigma \circ h_i(x+\delta) \\
&\qquad = \sigma \circ h_y(x) - \sigma \circ h_i(x) + \sigma \circ h_y(x+\delta) - \sigma \circ h_y(x) + \sigma \circ h_i(x) - \sigma \circ h_i(x+\delta) \\
&\qquad \geq \sigma \circ h_y(x) - \sigma \circ h_i(x) - \mathrm{Lip}_p(\sigma \circ h_y)\|\delta\|_p - \mathrm{Lip}_p(\sigma \circ h_i)\|\delta\|_p \\
&\qquad \geq \sigma \circ h_y(x) - \sigma \circ h_i(x) - \left(\mathrm{Lip}_p(\sigma \circ h_y) + \mathrm{Lip}_p(\sigma \circ h_i)\right) r_{\mathrm{Lip},p}^\alpha(x) \geq \tfrac{1-\alpha}{\alpha}.
\end{aligned}$$

Therefore, $h(\cdot)$ is certifiably robust at x with margin $\frac{1-\alpha}{\alpha}$ and radius $r_{\mathrm{Lip},p}^\alpha(x)$. Hence, by Lemma 1, the claim holds. □

Note that the ℓ_p norm that we certify can be arbitrary (e.g., ℓ_1, ℓ_2, or ℓ_∞), so long as the Lipschitz constant of the robust network $h(\cdot)$ is computed with respect to the same norm.

Assumption 1 is not restrictive in practice. For example, Gaussian RS with smoothing variance $\sigma^2 I_d$ yields robust models with ℓ_2-Lipschitz constant $\sqrt{\frac{2}{\pi\sigma^2}}$ [71]. Moreover, empirically robust methods such as adversarial training and TRADES often train Lipschitz continuous models, even though there may not be closed-form theoretical guarantees. The notion of Lipschitz continuity has even motivated novel robustness methods [61, 75].

Assumption 1 can be relaxed to the even less restrictive scenario of using local Lipschitz constants over a neighborhood (e.g., a norm ball) around a nominal input x (i.e., how flat $\sigma \circ h(\cdot)$ is near

x) as a surrogate for the global Lipschitz constants. In this case, Theorem 1 holds for all δ within this neighborhood. As a practical example, suppose that for an arbitrary input x and an ℓ_p attack radius ϵ, it holds that $|h_y(x+\delta) - h_y(x)| \leq \mathrm{Lip}_p(h_y) \cdot \epsilon$ and $|h_i(x+\delta) - h_i(x)| \leq \mathrm{Lip}_p(h_i) \cdot \epsilon$ for all $i \neq y$ and all perturbations δ such that $\|\delta\|_p \leq \epsilon$. Furthermore, suppose that $r^{\alpha}_{\mathrm{Lip},p}(x)$ as defined in (6) is not smaller than ϵ. Then, if the robust base classifier is correct at the nominal point x, then the mixed classifier is robust at x within the radius ϵ.

As shown in [82], the local Lipschitz constant of a given differentiable classifier can be easily estimated using an algorithm similar to the PGD attack. The authors of [82] also showed that many existing empirically robust models, including those trained with AT or TRADES, are in fact locally Lipschitz. Note that [82] evaluates the local Lipschitz constants of the logits, whereas we analyze the probabilities, whose Lipschitz constants are much smaller. Therefore, Theorem 1 provides important insights into the empirical robustness of the mixed classifier.

If $\alpha \to 1$, then $r^{\alpha}_{\mathrm{Lip},p}(x) \to \min_{i \neq y} \frac{h_y(x) - h_i(x)}{\mathrm{Lip}_p(h_y) + \mathrm{Lip}_p(h_i)}$, which is the standard (global) Lipschitz-based robust radius of $h(\cdot)$ around x (see, e.g., [32, 40] for further discussions on Lipschitz-based robustness). On the other hand, if α is too small in comparison to the relative confidence of $h(\cdot)$, namely, if there exists $i \neq y$ such that $\alpha \leq \frac{1}{1 + \sigma \circ h_y(x) - \sigma \circ h_i(x)}$, then $r^{\alpha}_{\mathrm{Lip},p}(x) \leq 0$, and in this case we cannot provide non-trivial certified robustness for $h^{\alpha}(\cdot)$. This is rooted in the fact that too small of an α value amounts to excess weight in the non-robust classifier $g(\cdot)$. If $h(\cdot)$ is 100% confident in its prediction, then $\sigma \circ h_y(x) - \sigma \circ h_i(x) = 1$ for all $i \neq y$, and therefore this threshold value of α becomes $\frac{1}{2}$, leading to non-trivial certified radii for $\alpha > \frac{1}{2}$. However, once we put over $\frac{1}{2}$ of the weight into $g(\cdot)$, a nonzero radius around x is no longer certifiable. Again, this is intuitively the best one can expect, since no assumptions on the robustness of $g(\cdot)$ around x have been made.

To summarize our certified robustness results, Lemma 1 shows the connection between the robust margin of the robust classifier and the robustness of the mixed classifier, while Theorem 1 demonstrates how general Lipschitz robust base classifiers exploit this relationship. Since empirically robust models often satisfy the conditions of these two results, they guarantee that adaptive attacks cannot easily circumvent our proposed robustification.

In Appendix A, we further tighten the certified radius in the special case when $h(\cdot)$ is a randomized smoothing classifier and the robust radii are defined in terms of the ℓ_2 norm by exploiting the stronger Lipschitzness of $x \mapsto \Phi^{-1}\big(\sigma \circ h_i(x)\big)$ arising from the unique structure and smoothness granted by Gaussian convolution operations (Φ^{-1} is the inverse Gaussian CDF function). We then visualize and compare the certified robustness of the mixed classifier to existing certifiably robust methods in Appendix A.1.

4. Adaptive Smoothing Strength with the Mixing Network

So far, α has been treated as a fixed hyperparameter. A more intelligent approach is to allow α to be different for each x by replacing the constant α with a function $\alpha(x)$. Here, we take $\alpha(x)$ to be deterministic, as stochastic defenses can be much harder to evaluate.

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 an attacked input targeting $g(\cdot)$, then the robust model $h(\cdot)$ should be used. However, as shown in Figure 2, if the target of the attack is $h(\cdot)$, then even

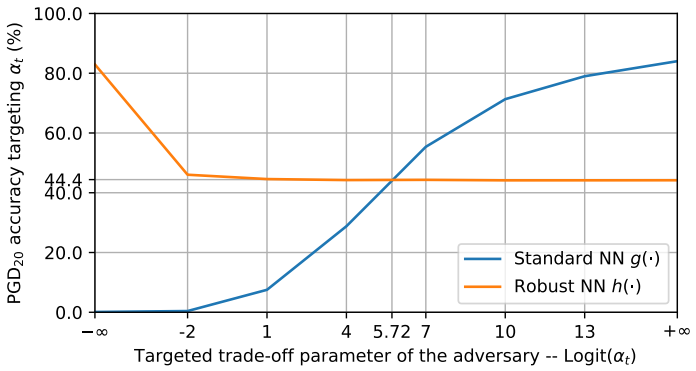


Figure 2. Attacked accuracy of the standard classifier $g(\cdot)$ and the robust classifier $h(\cdot)$ when the adversary targets different values of α_t. For better readability, we use $\text{Logit}(\alpha_t)$ as the horizontal axis labels, where $\text{Logit}(\cdot)$ denotes the inverse function of Sigmoid.

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 mixed classifier $h^{\alpha_t}(\cdot)$, as α_t varies, the optimal strategy also changes. We provide a visualization in Figure 2 based on the CIFAR-10 dataset. Specifically, we assemble a composite model $h^{\alpha_t}(\cdot)$ using a ResNet-18 standard classifier $g(\cdot)$ and a ResNet-18 robust classifier $h(\cdot)$ (both from [62]) via (5). Then, we attack $h^{\alpha_t}(\cdot)$ with different values of α_t via PGD_{20}, save the adversarial instances, and report the accuracy of $g(\cdot)$ and $h(\cdot)$ evaluated on these instances. When $\alpha_t \leq \text{Sigmoid}(5.72) = 0.9967$, the robust model $h(\cdot)$ performs better. When $\alpha_t > 0.9967$, the standard model $g(\cdot)$ is more suitable.

Throughout the remainder of this section, we overload the notation $h^{\alpha}(\cdot)$ even though $\alpha(\cdot)$ may be a function of the input, i.e., we define $h^{\alpha}(x) = h^{\alpha(x)}(x)$.

4.1. The Existence of $\alpha(x)$ that Achieves the Trade-Off

The following theorem shows that, under realistic conditions, there exists a function $\alpha(\cdot)$ that makes the combined classifier correct whenever either $g(\cdot)$ and $h(\cdot)$ makes the correct prediction, which further implies that the combined classifier matches the clean accuracy of $g(\cdot)$ and the attacked accuracy of $h(\cdot)$.

Theorem 2. *Let $\epsilon > 0$, $(x_1, y_1), (x_2, y_2) \sim \mathcal{D}$, and $y_1 \neq y_2$ (i.e., each input corresponds to a unique true label). Assume that $h_i(\cdot)$, $\|\nabla h_i(\cdot)\|_{p*}$, and $\|\nabla g_i(\cdot)\|_{p*}$ are all bounded and that there does not exist $z \in \mathbb{R}^d$ such that $\|z - x_1\|_p \leq \epsilon$ and $\|z - x_2\|_p \leq \epsilon$. Then, there exists a function $\alpha(\cdot)$ such that the assembled classifier $h^{\alpha}(\cdot)$ satisfies*

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

where $\mathcal{F}$ is an arbitrary distribution that satisfies $\mathbb{P}_{\delta\sim\mathcal{F}}\big[\|\delta\|_p > \epsilon\big] = 0$.

Proof. 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

$$\alpha(x+\delta) = \begin{cases} 0 & \text{if } \arg\max_{i\in[c]} g_i(x+\delta) = y, \\ 1 & \text{otherwise,} \end{cases}$$

satisfies that

$$\begin{aligned} \alpha(x+\delta) &= 0 \quad \text{if} \quad \arg\max_{i\in[c]} g_i(x+\delta) = y, \\ \alpha(x+\delta) &= 1 \quad \text{if} \quad \arg\max_{i\in[c]} g_i(x+\delta) \neq y \text{ and } \arg\max_{i\in[c]} h_i(x+\delta) = y. \end{aligned}$$

Therefore, it holds that

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

implying that

$$\arg\max_{i\in[c]} h_i^\alpha(x+\delta) = y \quad \text{if} \quad \big(\arg\max_{i\in[c]} g_i(x+\delta) = y \text{ or } \arg\max_{i\in[c]} h_i(x+\delta) = y\big),$$

which leads to the desired statement. □

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 distribution $\mathcal{F}$ is a Dirac measure centered at the origin, Theorem 2 implies that the clean accuracy of $h^\alpha(\cdot)$ is as good as the standard classifier $g(\cdot)$. Conversely, when $\mathcal{F}$ is a Dirac measure centered at the worst-case perturbation, the adversarial accuracy of $h^\alpha(\cdot)$ is not worse than the robust model $h(\cdot)$, implying that if $h(\cdot)$ is inherently robust, then $h^\alpha(\cdot)$ inherits the robustness. One can then conclude that there exists a $h^\alpha(\cdot)$ that matches the clean accuracy of $g(\cdot)$ and the robustness of $h(\cdot)$.

While Theorem 2 guarantees the existence of an instance of $\alpha(\cdot)$ that perfectly balances accuracy and robustness, finding an $\alpha(\cdot)$ that achieves this trade-off can be hard. However, we will use experiments to show that an $\alpha(\cdot)$ represented by a neural network can retain most of the robustness of $h(\cdot)$ while greatly boosting the clean accuracy. In particular, while we used the case of $\alpha(\cdot)$ being an indicator function to demonstrate the possibility of achieving the trade-off, Figure 1 has shown that letting α take an appropriate value between 0 and 1 also improves the trade-off. Thus, the task for the neural approximator is easier than representing the indicator function. Also note that if certified robustness is desired, one can enforce a lower bound on $\alpha(\cdot)$ and take advantage of Theorem 1 while still enjoying the mitigated trade-off.

4.2. Attacking the Adaptive Classifier

When the combined model $h^\alpha(\cdot)$ is under adversarial attack, the function $\alpha(\cdot)$ provides an addition gradient flow path. Intuitively, the attack should be able to force α to be small through this additional gradient path, tricking the mixing network into favoring the non-robust $g(\cdot)$. Following the guidelines for constructing adaptive attacks [76], in the experiments, we consider the following types of attacks:

A Gray-box PGD_{20}: In this setting, the adversary has access to the gradients of $g(\cdot)$ and $h(\cdot)$ when performing first-order optimization, but is not given the gradient of the mixing network $\alpha(\cdot)$. We consider untargeted PGD attack with a fixed initialization.

B White-box PGD_{20}: Since the mixed classifier is end-to-end differentiable, following [76], we allow the adversary to access the end-to-end gradient, including the gradient of the mixing network.

C White-box AutoAttack: "AutoAttack" is a stronger attack formed by an ensemble of four automated attack algorithms [29]. The method considers APGD attacks generated via the untargeted cross-entropy loss and the targeted DLR loss, in addition to the targeted FAB attack and the black-box Square Attack (SA) [10]. Again, the end-to-end gradient of the mixed classifier is available to the adversary. AutoAttack requires much more computation budget than PGD_{20}.

D Adaptive white-box AutoAttack: Since the mixing network is a crucial component of the defense, we adapt AutoAttack to target the mixing network by adding an APGD loss component that aims to decrease α.

We will show that the adaptively smoothed model is robust against the attack that it is trained against. When trained using APGD_{75} attack with untargeted and targeted loss functions, our model becomes robust against AutoAttack while achieving a significant improvement in the accuracy-robustness trade-off. Additionally, in Section 5.1.1, we consider examples of transfer attacks.

4.3. The Mixing Network

In practice, we use a neural network $\alpha_\theta(\cdot) : \mathbb{R}^d \to [0, 1]$ to learn an effective mixing network that adjusts the outputs of $g(\cdot)$ and $h(\cdot)$. Here, θ represents the trainable parameters of the mixing network, and we refer to $\alpha_\theta(\cdot)$ as the "mixing network". To enforce an output range constraint, we apply a Sigmoid function to the mixing network output. Note that when training the mixing network $\alpha_\theta(\cdot)$, the base classifiers $g(\cdot)$ and $h(\cdot)$ are frozen. Freezing the base classifiers allows the mixed classifier to take advantage of existing accurate models and their robust counterparts, maintaining explainability and avoiding unnecessary feature distortions that the adversary can potentially exploit.

The mixing network's task of treating clean and attacked inputs differently is closely related to the adversary detection problem. To this end, we adapt the detection architecture introduced in [59] for our mixing network. This architecture is end-to-end differentiable, making training and evaluation easier, and is also known for its performance and ease of implementation. While [20] has argued that simultaneously attacking the base classifier and the adversary detector can bring the detection rate of the detection method proposed in [59] to near zero, we show that with several key modifications, the method is effective even against strong white-box attacks. Specifically, our mixing network $\alpha_\theta(\cdot)$ takes advantage of the two available models $g(\cdot)$ and $h(\cdot)$ by using the intermediate features of both networks via concatenation ([59] only has one base model). More importantly, we include stronger adaptive adversaries during training to generate much more diverse examples.

The mixing network structure is based on a ResNet-18, which is known to perform well for a wide range of computer vision applications and is often considered the go-to architecture. We make minimum changes to the ResNet-18 to fit into our framework. As the mixing network takes information

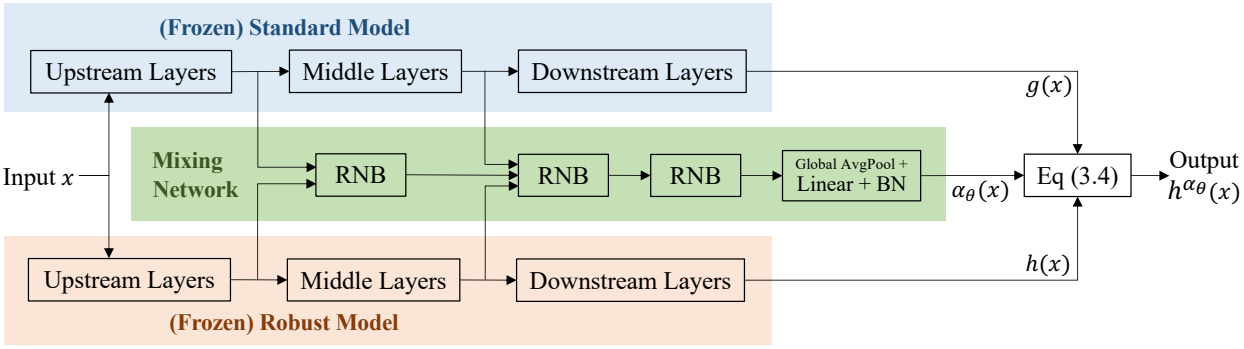


Figure 3. The overall architecture of the adaptively smoothed classifier introduced in Section 4. "RNB" stands for ResNetBlock and "BN" represents the 2D batch normalization layer.

from both $g(\cdot)$ and $h(\cdot)$, it uses the concatenated embeddings from the base classifiers. While [59] considers a single ResNet as the base classifier and uses the embeddings after the first ResNet block, to avoid the potential vulnerability against "feature adversaries" [70], we consider the embeddings from two different layers of the base model. Figure 3 demonstrates the modified architecture. The detailed implementations used in the experiment section are discussed in Appendix C.1.

Since Figure 1 shows that even a constant α can alleviate the accuracy-robustness trade-off, our method does not excessively rely on the performance of the mixing network $\alpha_\theta(\cdot)$. In Section 5.2, we provide empirical results demonstrating that the above modifications help the overall mixed network defend against strong attacks.

4.4. Training the Mixing Network

Consider the following two loss functions for training the mixing network $\alpha_\theta(\cdot)$:

- **Multi-class cross-entropy:** We minimize the multi-class cross-entropy loss of the combined classifier, which is the ultimate goal of the mixing network:

$$\min_\theta \mathbb{E}_{\substack{(x,y)\sim\mathcal{D} \\ \delta\sim\mathcal{F}}} \Big[\ell_{\mathrm{CE}}\big(h^{\alpha_\theta}(x+\delta), y\big) \Big], \tag{7}$$

  where ℓ_{CE} is the cross-entropy (CE) loss for logits and $y \in [c]$ is the label corresponding to x. The base classifiers $g(\cdot)$ and $h(\cdot)$ are frozen and not updated. Again, δ denotes the perturbation, and the distribution $\mathcal{F}$ is arbitrary. In our experiments, to avoid overfitting to a particular attack radius, $\mathcal{F}$ is formed by perturbations with randomized radii.

- **Binary cross-entropy:** The optimal $\alpha^\star$ that minimizes ℓ_{CE} in (7) can be estimated for each training point. Specifically, depending on whether the input is attacked and how it is attacked, either $g(\cdot)$ or $h(\cdot)$ should be prioritized. Thus, we treat the task as a binary classification problem and solve the optimization problem

$$\min_\theta \mathbb{E}_{\substack{(x,y)\sim\mathcal{D} \\ \delta\sim\mathcal{F}}} \Big[\ell_{\mathrm{BCE}}\big(\alpha_\theta(x+\delta), \widetilde{\alpha}\big) \Big],$$

  where ℓ_{BCE} is the binary cross-entropy (BCE) loss for probabilities and $\widetilde{\alpha} \in \{0, 1\}$ is the "pseudo label" for the output of the mixing network that approximates $\alpha^\star$.

Using only the multi-class loss suffers from a distributional mismatch between the training set and the test set. The robust classifier $h(\cdot)$ may achieve a low loss on adversarial training data but a high loss on adversarial test data. For example, with the CIFAR-10 dataset and our ResNet-18 robust classifier, the PGD_{10} adversarial training accuracy is 93.01% while the PGD_{10} test accuracy is 45.55%. As a result, approximating (7) with empirical risk minimization on the training set does not effectively optimize the true risk. When the adversary attacks a test input x targeting $h(\cdot)$, 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 mixing network does not receive a strong incentive to choose $g(\cdot)$ when it detects an attack targeting $h(\cdot)$.

The binary loss, however, does not capture the potentially different sensitivity of each input. Certain inputs can be more vulnerable to adversarial attacks, and ensuring the correctness of the mixing network on these inputs is more crucial.

To this end, we combine the above two components into a composite loss function, incentivizing the mixing network to select the standard classifier $g(\cdot)$ when appropriate, while forcing it to remain conservative. The composite loss for each data-label pair (x, y) is given by

$$\begin{aligned}\ell_{\text{composite}}\big(\theta, (x, y, \widetilde{\alpha})\big) = c_{\text{CE}} \cdot \ell_{\text{CE}}\big(h^{\alpha_\theta}(x+\delta), y\big) + c_{\text{BCE}} \cdot \ell_{\text{BCE}}\big(\alpha_\theta(x+\delta), \widetilde{\alpha}\big) \\ + c_{\text{prod}} \cdot \ell_{\text{CE}}\big(h^{\alpha_\theta}(x+\delta), y\big) \cdot \ell_{\text{BCE}}\big(\alpha_\theta(x+\delta), \widetilde{\alpha}\big),\end{aligned} \tag{8}$$

where the hyperparameters $c_{\text{CE}}, c_{\text{BCE}}$, and c_{prod} control the weights of the loss components. Appendix C.2 discusses how these hyperparameters affect the performance of the trained mixing model.

5. Numerical Experiments

5.1. Robust Neural Network Smoothing with a Fixed Strength

Here, we consider the case where the smoothing strength α is a fixed value, and use the CIFAR-10 dataset to evaluate the performance of the mixed classifier $h^\alpha(\cdot)$ with various values of α.

5.1.1 α's Influence on Robustness

We first analyze how the accuracy of the mixed classifier changes with the mixing strength α under various settings. Specifically, we consider PGD_{20} attacks that target $g(\cdot)$ and $h(\cdot)$ individually (abbreviated as STD and ROB attacks), in addition to the adaptive PGD_{20} attack generated using the end-to-end gradient of $h^\alpha(\cdot)$, denoted as the MIX attack. Note that the STD and ROB attacks, which share the inspiration of [34], correspond to the "transfer attack" setting, a common black-box attack strategy designed for defenses with unavailable or unreliable gradients. Note that the models with the best transferability with the mixed classifier $h^\alpha(\cdot)$ would likely be its base classifiers $g(\cdot)$ and $h(\cdot)$, precisely corresponding to the STD and ROB attack settings.

We use a ResNet18 model trained on clean data as the standard model $g(\cdot)$ and use another ResNet18 trained on PGD_{20} data as the robust model $h(\cdot)$. The test accuracy corresponding to each α value is presented in Figure 4. As α increases, the clean accuracy of $h^\alpha(\cdot)$ 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 targeting $h(\cdot)$ becomes more benign

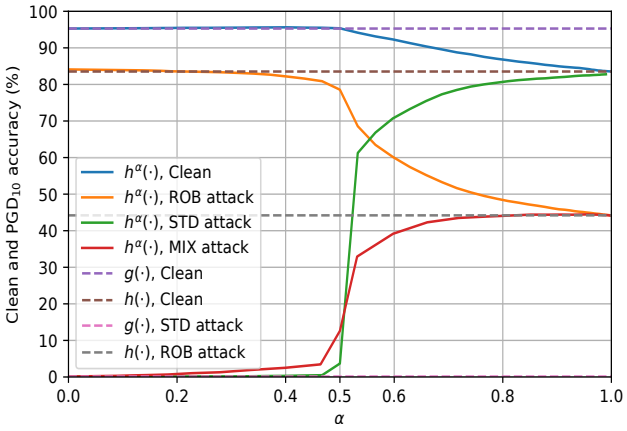


Figure 4. The performance of the mixed classifier $h^\alpha(\cdot)$. "STD attack", "ROB attack", and "MIX attack" refer to the PGD_{20} attack generated using the gradient of $g(\cdot)$, $h(\cdot)$, and $h^\alpha(\cdot)$ respectively, with ϵ set to $8/255$.

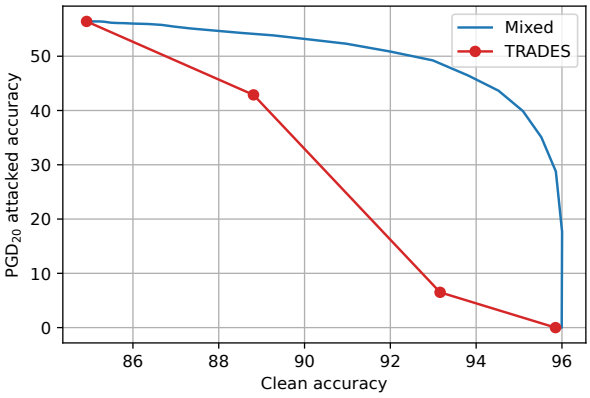


Figure 5. Accuracy-robustness trade-off comparison between our mixed classifier and TRADES models. "Mixed" stands for our mixed classifier $h^\alpha(\cdot)$. All TRADES models and the base models of $h^\alpha(\cdot)$ use the WideResNet-34-10 architecture as in [84].

when the mixed classifier emphasizes $g(\cdot)$. When the attack targets $h^\alpha(\cdot)$, the attacked accuracy increases with α.

When α is around 0.5, the MIX-attacked accuracy of $h^\alpha(\cdot)$ quickly increases from near zero to more than 30% (which is two-thirds of $h(\cdot)$'s attacked accuracy). This observation precisely matches the theoretical intuition provided by Theorem 1. When α is greater than 0.5, the clean accuracy gradually decreases at a much slower rate, leading to the noticeably alleviated accuracy-robustness trade-off. Note that this improved trade-off is achieved without any further training beyond the weights of $g(\cdot)$ and $h(\cdot)$. When α is greater than 0.55, neither STD attack nor ROB attack can reduce the accuracy of the mixed classifier below the end-to-end gradient-based attack (MIX attack), indicating that the considered transfer attack is weaker than gradient-based attack for practical α values, and implying that the robustness of $h^\alpha(\cdot)$ does not rely on obfuscated gradients. In Section 5.1.2, we will reveal that the source of $h^\alpha(\cdot)$'s robustness lies in $h(\cdot)$'s well-calibrated confidence properties.

5.1.2 The Relationship between $h^\alpha(\cdot)$'s Robustness and $h(\cdot)$'s Confidence

Our theoretical analysis (Lemma 1) has highlighted the relationship between the mixed classifier robustness and the robust base classifier $h(\cdot)$'s robust margin. For practical models, the margin at a given radius can be estimated with the confidence gap between the predicted and runner-up classes evaluated on strongly adversarial inputs, such as images returned from PGD_{20} or AutoAttack. Moreover, the improved accuracy-robustness trade-off of the mixed classifier, as evidenced by the difference in how clean and attacked accuracies change with α in Figure 4, can also be explained by the prediction confidence of $h(\cdot)$.

According to Table 1, the robust base classifier $h(\cdot)$ makes confident correct predictions even when under attack (average robust margin is 0.768 evaluated with PGD_{20} and 0.774 with AutoAttack[1]).

[1]The calculation details the AutoAttacked confidence gap are presented in Appendix C.3.

Table 1. Average gap between the probabilities of the predicted class and the runner-up class. $g(\cdot)$ and $h(\cdot)$ are the same ones used in Figure 4. The confidence difference highlighted by the bold numbers is crucial to the mitigated accuracy-robustness trade-off of the mixed classifier.

	Clean			PGD_{20}			AutoAttack		
	Accuracy	✓ Gap	✗ Gap	Accuracy	✓ Gap	✗ Gap	Accuracy	✓ Gap	✗ Gap
$g(\cdot)$	95.28%	0.982	0.698	0.10%	0.602	0.998	0.00%	–	0.986
$h(\cdot)$	83.53%	0.854	**0.434**	44.17%	**0.768**	0.635	40.75%	**0.774**	0.553

✓ Gap: The average gap between the confidences of the predicted class and the runner-up class among all correctly predicted validation data.
✗ Gap: The same quantity evaluated among all incorrectly predicted validation data.

Moreover, the robust margin of $h(\cdot)$ follows a long-tail distribution. Specifically, the median robust margin is 0.933 (same number when evaluated with PGD_{20} or AutoAttack), much larger than the 0.768/0.774 average margin. Thus, most attacked inputs correctly classified by $h(\cdot)$ are highly confident (i.e., robust with large margins), with only a tiny portion suffering from small robust margins. As Lemma 1 suggests, such property is precisely what adaptive smoothing relies on. Intuitively, once α becomes greater than 0.5 and gives $h(\cdot)$ more authority over $g(\cdot)$, $h(\cdot)$ can use its high confidence to correct $g(\cdot)$'s mistakes under attack.

On the other hand, $h(\cdot)$ is unconfident when producing incorrect predictions on clean data, with the top two classes' output probabilities separated by merely 0.434. This probability gap again forms a long-tail distribution (the median is 0.378 which is less than the mean), confirming that $h(\cdot)$ is generally unconfident when it mispredicts, rarely making confident incorrect predictions. Now, consider clean unattacked data which $g(\cdot)$ correctly classifies and $h(\cdot)$ mispredicts (recall that we assume $g(\cdot)$ to be a more accurate but less robust model so this scenario is common). Since $g(\cdot)$ is confident and $h(\cdot)$ is unconfident with high probability, even when $\alpha \geq 0.5$ and $g(\cdot)$ has less authority than $h(\cdot)$ in the mixture, $g(\cdot)$ can still correct some of the mistakes from $h(\cdot)$.

In summary, $h(\cdot)$ is confident when making correct predictions on attacked data, enjoying the large robust margin required by Lemma 1. At the same time, $h(\cdot)$ is unconfident when misclassifying clean data, and such a confidence property is the key source of the mixed classifier's improved accuracy-robustness trade-off. Additional analyses in Appendix B.2 with alternative base models imply that multiple existing robust classifiers share the favorable confidence property and thus help the mixed classifier improve the trade-off.

The standard non-robust classifier $g(\cdot)$ often does not have this desirable property: even though it is confident on clean data as are robust classifiers, it also makes highly confident mistakes under attack. Note that this does not undermine the mixed classifier robustness, since our formulation does not assume any robustness or smoothness from $g(\cdot)$.

5.1.3 Comparing the Accuracy-Robustness Trade-Off with TRADES

TRADES [84] is one of the most famous existing initiatives to improve the accuracy-robustness trade-off of classifiers. This subsection shows that our method achieves a much more benign trade-

off than TRADES. Specifically, TRADES proposed to train robust models by minimizing the risk

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\ell_{\text{CE}}\big(h(x),y\big)+\beta\max_{\|\delta\|\leq\epsilon}\ell_{\text{surrogate}}\big(h(x+\delta),h(x)\big)\Big],$$

where $\beta\geq 0$ is a trade-off parameter between the two loss components and $\ell_{\text{surrogate}}$ is the "surrogate loss" that promotes robustness. The larger β is, the more robust the trained model becomes at the expense of clean accuracy. By adjusting β, we can adjust the accuracy-robustness trade-off of TRADES similarly to adjusting α in our mixed classifier.

The authors of [84] reported that $\beta=6$ optimized the adversarial robustness and released the corresponding model. We use this model and train three additional models with β set to 0, 0.1, and 0.3. Here, $\beta=0$ is standard training, and the other two numbers were chosen so that the model accuracy spreads relatively uniformly between $\beta=0$ and $\beta=6$. For a fair comparison, we build mixed classifiers using the TRADES model trained with $\beta=0$ as $g(\cdot)$ and the $\beta=6$ model as $h(\cdot)$. We compare the relationship between the PGD_{20} accuracy and the clean accuracy of the two methods in Figure 5. Note that the trade-off curve of the mixed classifier intercepts the TRADES curve at the two ends (since the models are exactly the same at the two ends), and is significantly above the TRADES in the middle, indicating that the accuracy-robustness trade-off of the mixed classifier is better than that of TRADES.

Note that for TRADES, adjusting the trade-off requires training a new model, which is costly. For our mixing classifier, adjusting α does not require re-training and is thus much more flexible and efficient while achieving a better Pareto curve.

5.2. Robust Neural Network Smoothing with Adaptive Strength

Having validated the effectiveness of the mixing formulation described in (5), we are now ready to incorporate the mixing network for adaptive smoothing strength. As in Section 4, we denote the parameterized mixing network by $\alpha_\theta(\cdot)$, and slightly abuse notation by denoting the composite classifier with adaptive smoothing strength given by $\alpha_\theta(\cdot)$ by $h^{\alpha_\theta}(\cdot)$, which is defined by $h^{\alpha_\theta}(x)=h^{\alpha_\theta(x)}(x)$.

CIFAR-10 and CIFAR-100 are two of the most widely used benchmark datasets for evaluating adversarial robustness. Although recent progress in learning robust models has made the accuracy-robustness trade-off less noticeable for the easier CIFAR-10 dataset, with robust neural networks almost matching standard models on clean accuracy [36, 37, 69], the trade-off remains highly noticeable for more challenging tasks such as CIFAR-100, for which robust models suffer from significant accuracy degradation. As existing methods for improving standard model performance may not readily extend to robust models, mixing pre-trained classifiers via adaptive smoothing becomes particularly advantageous for these harder tasks. Therefore, we use CIFAR-10 models to perform ablation analyses and proof-of-concept demonstrations and show that our mixing classifiers reconcile the accuracy-robustness trade-off to an unprecedented level on the CIFAR-100 dataset.

In all experiments, we consider ℓ_∞ attacks and use the AdamW optimizer [46] for training the mixing network $\alpha_\theta(\cdot)$. The training data for $\alpha_\theta(\cdot)$ include clean images and the corresponding types of attacked images (attack settings A, B, and C presented in Section 4.2). For setting C (AutoAttack), the training data only include targeted and untargeted APGD attacks, with the other two AutoAttack components, FAB and Square, excluded during training in the interest of efficiency but included for evaluation. To alleviate overfitting, when generating training-time attacks, we

Table 2. CIFAR-10 results of adaptive smoothing models trained with three different settings.

CIFAR-10 base classifier performances

Model	Architecture	Clean	PGD_{20}	AutoAttack
$g(\cdot)$ (accurate)	ResNet-18 (Standard non-robust training)	95.28 %	0.12 %	0.00 %
$h(\cdot)$ (robust)	WideResNet-34-10 (TRADES model [84])	84.92 %	57.16 %	53.09 %

CIFAR-10 adaptive smoothing mixed classifier $h^{\alpha_\theta}(\cdot)$ performance

Evaluation Data \ Training Setting	**A**	**B**	**C**
Clean	92.05 %	92.07 %	91.51 %
A (gray-box PGD_{20})	57.22 %	57.25 %	56.30 %
B (white-box PGD_{20})	56.63 %	57.09 %	56.29 %
C (white-box AutoAttack)	40.04 %	40.02 %	42.78 %
D (adaptive AutoAttack)	39.85 %	39.70 %	42.66 %

randomize the attack radius and the number of steps, and add a randomly-weighted binary cross-entropy component that aims to decrease the mixing network output to the attack objective (thereby tricking it into favoring $g(\cdot)$). Additionally, Appendix C.1 discusses the details of implementing the architecture shown in Figure 3 for the ResNet base classifiers used in our experiments; Appendix C.2 performs an ablation study on the hyperparameters of the composite loss function (8).

5.2.1 Ablation Studies Regarding Attack Settings

We first use smaller base classifiers to analyze the behavior of adaptive smoothing by exploring various training and attack settings. The performance of the base models and the assembled mixed classifier are summarized in Table 2, where each column represents the performance of one mixed classifier. The results show that the adaptive smoothing model can defend against the attacks on which the underlying mixing network is trained. Specifically, for the attack setting A (gray-box PGD), $h^{\alpha_\theta}(\cdot)$ is able to achieve the same level of PGD_{20}-attacked accuracy as $h(\cdot)$ while retaining a similar level of clean accuracy as $g(\cdot)$. For the setting B (white-box PGD), the attack is allowed to follow the gradient path provided by $\alpha_\theta(\cdot)$ and deliberately evade the part of the adversarial input space recognized by $\alpha_\theta(\cdot)$. While the training task becomes more challenging, the improvement in the accuracy-robustness trade-off is still substantial. Furthermore, the composite model can generalize to examples generated via the stronger AutoAttack. For the setting C (AutoAttack), the difficulty of the training problem further escalates. While the performance of $h^{\alpha_\theta}(\cdot)$ on clean data slightly decreases, the mixing network can offer a more vigorous defense against AutoAttack data, still improving the accuracy-robustness trade-off.

Table 3 repeats the above analyses on the CIFAR-100 dataset. The results confirm that adaptive smoothing achieves even more significant improvements on the CIFAR-100 dataset. Notably, even for the most challenging attack setting C, $h^{\alpha_\theta}(\cdot)$ correctly classifies 1173 additional clean images compared with $h(\cdot)$ (cutting the error rate by a third) while making only 404 additional incorrect predictions on AutoAttacked inputs (increasing the error rate by merely 6.4 relative percent). Such results show that $\alpha_\theta(\cdot)$ is capable of approximating a robust high-performance mixing network when trained with sufficiently diverse attacked data. The fact that $h^{\alpha_\theta}(\cdot)$ combines the clean accuracy of $g(\cdot)$ and the robustness of $h(\cdot)$ highlights that our method significantly improves the accuracy-robustness trade-off.

Table 3. CIFAR-100 results of adaptive smoothing models trained with the three settings.

CIFAR-100 base classifier performances

Model	Architecture	Clean	PGD_{20}	AutoAttack
$g(\cdot)$ (accurate)	ResNet-152 (Based on BiT [47])	91.38 %	0.14 %	0.00 %
$h(\cdot)$ (robust)	WideResNet-70-16 (From [36])	69.17 %	40.86 %	36.98 %

CIFAR-100 adaptive smoothing mixed classifier $h^{\alpha_\theta}(\cdot)$ performance

Evaluation Data \ Training Setting	**A**	**B**	**C**
Clean	83.99 %	83.96 %	80.90 %
A (gray-box PGD_{20})	40.04 %	39.80 %	39.26 %
B (white-box PGD_{20})	30.59 %	34.48 %	38.92 %
C (white-box AutoAttack)	23.54 %	26.37 %	32.94 %
D (adaptive AutoAttack)	23.78 %	26.17 %	32.80 %

5.2.2 Comparisons Against Existing Works

This section uses state-of-the-art base classifiers to show that adaptive smoothing can produce excellent results and offer comparisons with existing methods. Since the literature has generally regarded AutoAttack [29] as one of the most reliable and appropriate robustness evaluation methods (weaker attacks such as PGD are known to be circumventable), we select existing works that use AutoAttack as the benchmark. For the accurate base classifier $g(\cdot)$, we select BiT [47] ResNet-152 models. Specifically, we fine-tune the publicly released BiT checkpoint (pre-trained on ImageNet-21k) on CIFAR-10 or CIFAR-100 following the recipe specified in [47], resulting in a 98.50% clean accuracy for CIFAR-10 and 91.38% for CIFAR-100. We highlight that the listed works should not be treated as competitors, since advancements in building robust classifiers can be incorporated into our framework as $h(\cdot)$, helping adaptive smoothing perform even better.

For CIFAR-10, we select the robust model checkpoint released in [81] as the robust base classifier $h(\cdot)$ and report the results in Table 4. Compared with the robust base classifier, our adaptive smoothing model retains 96.3 (relative) percent of the robust accuracy while reducing the clean data error rate by 29.3 (relative) percent. Among all models that were available on RobustBench at the time of the submission, our model achieves the third highest AutoAttacked accuracy, only behind [81] (used as $h(\cdot)$ in our model) and [45] (for which AutoAttack is unreliable and the best-known attacked accuracy is lower than ours). On the other hand, the clean accuracy of our mixed classifier is higher than all other CIFAR-10 models designed towards the ℓ_∞ adversarial attack setting available on RobustBench, and is even higher than the listed non-robust model that uses standard training.

While the above results demonstrate reconciled accuracy and robustness, the clean accuracy improvement over existing works is not highly prominent. This is because robust base classifiers are already highly accurate on the easier CIFAR-10 dataset, and there is not much improvement room for clean accuracy. As discussed earlier, adaptive smoothing is particularly advantageous for more challenging tasks where the accuracy-robustness trade-off is highly noticeable with existing models. We now support this claim with more significant improvements on the harder CIFAR-100 dataset.

For CIFAR-100, we assemble two instances of the mixed classifier using two different robust base classifiers and display the results in Table 5. Compared with their corresponding robust base models,

Table 4. The clean and AutoAttacked accuracy of adaptive smoothing on CIFAR-10 compared with existing works. The numbers for the existing works are reported by the authors.

Method	Model Architecture	Clean	AutoAttack
Adaptive Smoothing (this work) *	ResNet-152 + WRN-70-16	95.23 %	68.06 %
SODEF + TRADES [45]	WRN-70-16 + ODE Block	93.73 %	71.28 % †
Diffusion (EDM) + TRADES [81]	WideResNet-70-16	93.25 %	70.69 %
Diffusion (DDPM) + TRADES [69]	WideResNet-70-16	92.23 %	66.58 %
Unlabeled data + TRADES [36]	WideResNet-70-16	91.10 %	65.88 %
TRADES [36]	WideResNet-70-16	85.29 %	57.20 %

*: Uses "EDM + TRADES" [81] as the robust base model $h(\cdot)$.
†: AutoAttack raises the "potentially unreliable" flag (explained in the next page), and adaptive attack reduces the attacked accuracy to 64.20 %. AutoAttack does not raise this flag for our models.

Table 5. The clean and AutoAttacked accuracy of adaptive smoothing on CIFAR-100 compared with existing works.

Method	Model Architecture	Clean	AutoAttack
Adaptive Smoothing (this work) *	ResNet-152 + WRN-70-16	85.21 %	38.72 %
Adaptive Smoothing (this work) **	ResNet-152 + WRN-70-16	80.18 %	35.15 %
Diffusion (EDM) + TRADES [81]	WideResNet-70-16	75.22 %	42.67 %
Unlabeled data + TRADES [36]	WideResNet-70-16	69.17 %	36.98 %
TRADES‡ [30]	XCiT-L12 [6]	70.76 %	35.08 %
Diffusion (DDPM) + TRADES [69]	WideResNet-70-16	63.56 %	34.64 %
SCORE Loss AT [64]	WideResNet-70-16	65.56 %	33.05 %
Diffusion (DDPM) + AT [73]	WideResNet-34-10	65.93 %	31.15 %
TRADES [36]	WideResNet-70-16	60.86 %	30.03 %

*: Uses "EDM + TRADES" [81] as the robust base model $h(\cdot)$.
**: Uses "Unlabeled data + TRADES" [36] as the robust base model $h(\cdot)$.

both implementations of adaptive smoothing improve the clean accuracy by ten percentage points, with the AutoAttacked accuracy only deteriorating by four percentage points. Notably, as of the submission of this paper, on the CIFAR-100 dataset, the composite classifier based on the robust model introduced in [81] achieved an AutoAttacked accuracy better than any other methods listed on RobustBench [26], except [81] itself. Simultaneously, this mixed model offers an improvement of ten percentage points compared with any other listed models. These results demonstrate that adaptive smoothing significantly alleviates the accuracy-robustness trade-off of neural classifiers.

We also report that the SA component of AutoAttack, which performs gradient-free black-box attacks on images that gradient-based attack methods fail to perturb, only changes very few predictions. Specifically, AutoAttack will raise a "potentially unreliable" flag if SA further reduces the accuracy by at least 0.2 percentage points. This flag is not thrown for our models in Tables 4 and 5, indicating that the mixed classifiers' robustness is not a result of gradient obfuscation. Thus, gradient-based attacks in AutoAttack sufficiently evaluate our models.

6. Conclusions

This paper proposes "adaptive smoothing", a flexible framework that leverages the mixture of the probability outputs of an accurate model and a robust model to mitigate the accuracy-robustness trade-off of neural classifiers. We mathematically prove that the smoothed composite model can inherit the certified robustness of the robust base model under realistic assumptions. We then adapt an adversarial input detector into a (deterministic) mixing network, further improving the accuracy-robustness trade-off. Solid empirical results confirm that our method can simultaneously benefit from the high accuracy of modern pre-trained standard (non-robust) models and the robustness achieved via state-of-the-art robust classification methods.

Because our theoretical studies demonstrate the feasibility of leveraging the mixing network to entirely eliminate the accuracy-robustness trade-off, future advancements in adversary identification can reconcile this trade-off even more effectively via our framework. Furthermore, the proposed method can be conveniently extended to various attack types and budgets. Thus, this work paves the way for future research to focus on either accuracy or robustness without sacrificing the other.

References

[1] George Adam and Romain Speciel. Evaluating ensemble robustness against adversarial attacks. *arXiv preprint arXiv:2005.05750*, 2020.

[2] 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.

[3] Manaar Alam, Shubhajit Datta, Debdeep Mukhopadhyay, Arijit Mondal, and Partha Pratim Chakrabarti. Resisting adversarial attacks in deep neural networks using diverse decision boundaries. *arXiv preprint arXiv:2208.08697*, 2022.

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

[5] 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*, 2021.

[6] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. In *Advances in neural information processing systems*, 2021.

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

[8] Brendon G Anderson and Somayeh Sojoudi. Certified robustness via locally biased randomized smoothing. In *Annual Learning for Dynamics and Control Conference*, 2022.

[9] Brendon G Anderson and Somayeh Sojoudi. Data-driven certification of neural networks with random input noise. *IEEE Transactions on Control of Network Systems*, 2022.

[10] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *European Confer-*

ence on Computer Vision, 2020.

[11] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018.

[12] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International Conference on Machine Learning*, 2018.

[13] 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.

[14] Yatong Bai, Brendon G Anderson, and Somayeh Sojoudi. Mixing classifiers to alleviate the accuracy-robustness trade-off. In *IEEE Conference on Control Technology and Applications*, 2023.

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

[16] Yatong Bai, Tanmay Gautam, and Somayeh Sojoudi. Efficient global optimization of two-layer relu networks: Quadratic-time algorithms and adversarial training. *SIAM Journal on Mathematics of Data Science*, 2022.

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

[18] Jaydeep Borkar and Pin-Yu Chen. Simple transparent adversarial examples. *arXiv preprint arXiv:2105.09685*, 2021.

[19] Nicholas Carlini, Florian Tramer, J Zico Kolter, et al. (Certified!!) adversarial robustness for free! *arXiv preprint arXiv:2206.10550*, 2022.

[20] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017.

[21] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

[22] 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.

[23] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[24] Kenneth T Co, David Martinez-Rego, Zhongyuan Hau, and Emil C Lupu. Jacobian ensembles improve robustness trade-offs to adversarial attacks. In *Artificial Neural Networks and Machine Learning*, 2022.

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

[26] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized

adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

[27] Francesco Croce, Sven Gowal, Thomas Brunner, Evan Shelhamer, Matthias Hein, and Taylan Cemgil. Evaluating the adversarial robustness of adaptive test-time defenses. *arXiv preprint arXiv:2202.13711*, 2022.

[28] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2020.

[29] 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.

[30] Edoardo Debenedetti, Vikash Sehwag, and Prateek Mittal. A light recipe to train robust vision transformers. *arXiv preprint arXiv:2209.07399*, 2022.

[31] Lijie Fan, Sijia Liu, Pin-Yu Chen, Gaoyuan Zhang, and Chuang Gan. When does contrastive learning preserve adversarial robustness from pretraining to finetuning? In *Annual Conference on Neural Information Processing Systems*, 2021.

[32] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Annual Conference on Neural Information Processing Systems*, 2019.

[33] Ivan Fursov, Alexey Zaytsev, Pavel Burnyshev, Ekaterina Dmitrieva, Nikita Klyuchnikov, Andrey Kravchenko, Ekaterina Artemova, Evgenia Komleva, and Evgeny Burnaev. A differentiable language model adversarial attack on text classifiers. *IEEE Access*, 10:17966–17976, 2022.

[34] Xitong Gao, Cheng-Zhong Xu, et al. Mora: Improving ensemble robustness evaluation with model reweighing attack. In *Advances in Neural Information Processing Systems*, 2022.

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

[36] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

[37] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan A. Calian, and Timothy Mann. Improving robustness using generated data. *arXiv preprint arXiv:2110.09468*, 2021.

[38] Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. An alternative surrogate loss for pgd-based adversarial testing. *arXiv preprint arXiv:1910.09338*, 2019.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, 2016.

[40] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Annual Conference on Neural Information Processing Systems*, 2017.

[41] 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.

[42] 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.

[43] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, 2018.

[44] Xiaojun Jia, Yong Zhang, Baoyuan Wu, Ke Ma, Jue Wang, and Xiaochun Cao. LAS-AT: Adversarial training with learnable attack strategy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[45] Qiyu Kang, Yang Song, Qinxu Ding, and Wee Peng Tay. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. 2021.

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

[47] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (BiT): General visual representation learning. In *European Conference on Computer Vision*, 2020.

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

[49] Ananya Kumar, Tengyu Ma, Percy Liang, and Aditi Raghunathan. Calibrated ensembles can mitigate accuracy tradeoffs under distribution shift. In *The Conference on Uncertainty in Artificial Intelligence*, 2022.

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

[51] 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.

[52] Alexander Levine, Sahil Singla, and Soheil Feizi. Certifiably robust interpretation in deep learning. *arXiv preprint arXiv:1905.12105*, 2019.

[53] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *Annual Conference on Neural Information Processing Systems*, 2019.

[54] Chang Liu, Yinpeng Dong, Wenzhao Xiang, Xiao Yang, Hang Su, Jun Zhu, Yuefeng Chen, Yuan He, Hui Xue, and Shibao Zheng. A comprehensive study on robustness of image classification models: Benchmarking and rethinking. *arXiv preprint arXiv:2302.14301*, 2023.

[55] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *European Conference on Computer Vision*, 2018.

[56] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[57] 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.

[58] 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.

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

[60] 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.

[61] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

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

[63] Matteo Pagliardini, Gilberto Manunza, Martin Jaggi, and Tatjana Chavdarova. Improved generalization-robustness trade-off via uncertainty targeted attacks. Preprint, 2022.

[64] Tianyu Pang, Min Lin, Xiao Yang, Jun Zhu, and Shuicheng Yan. Robustness and accuracy could be reconcilable by (proper) definition. *arXiv preprint arXiv:2202.10103*, 2022.

[65] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, 2019.

[66] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM Asia Conference on Computer and Communications Security*, 2017.

[67] Samuel Pfrommer, Brendon G Anderson, and Somayeh Sojoudi. Projected randomized smoothing for certified adversarial robustness. *Transactions on Machine Learning Research*, 2023.

[68] 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.

[69] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

[70] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

[71] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Annual Conference on Neural Information Processing Systems*, 2019.

[72] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Annual Conference on Neural Information Processing Systems*, 2018.

[73] Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang,

and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *International Conference on Learning Representations*, 2022.

[74] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Annual Conference on Neural Information Processing Systems*, 2019.

[75] David Terjek. Adversarial lipschitz regularization. In *International Conference on Learning Representations*, 2020.

[76] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Annual Conference on Neural Information Processing Systems*, 2020.

[77] 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.

[78] 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.

[79] Haotao Wang, Tianlong Chen, Shupeng Gui, TingKuei Hu, Ji Liu, and Zhangyang Wang. Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. In *Annual Conference on Neural Information Processing Systems*, 2020.

[80] 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.

[81] Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. Better diffusion models further improve adversarial training. *arXiv preprint arXiv:2302.04638*, 2023.

[82] 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.

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

[84] 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.

[85] Shiji Zhao, Xizhe Wang, and Xingxing Wei. Mitigating the accuracy-robustness trade-off via multi-teacher adversarial distillation. *arXiv preprint arXiv:2306.16170*, 2023.

[86] Haizhong Zheng, Ziqi Zhang, Juncheng Gu, Honglak Lee, and Atul Prakash. Efficient adversarial training with transferable adversarial examples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[87] Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

A. Larger Certified Robust Radius for Randomized Smoothing Base Classifiers

In this section, we tighten the certified radius in the special case when $h(\cdot)$ is a randomized smoothing classifier and the robust radii are defined in terms of the ℓ_2 norm. This enables us to visualize and compare the certified robustness of the mixed classifier to existing certifiably robust methods in Appendix A.1.

Since randomized smoothing often operates on the probabilities and does not consider the logits, with a slight abuse of notation, we use $h(\cdot)$ to denote the probabilities throughout this section (as opposed to denoting the logits in the main text).

Assumption 2. The robust classifier $h(\cdot)$ is a (Gaussian) randomized smoothing classifier, i.e., $h(x) = \mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I_d)}\left[\overline{h}(x+\xi)\right]$ for all $x \in \mathbb{R}^d$, where $\overline{h}\colon \mathbb{R}^d \to [0,1]^c$ is the output probabilities of a classifier that is non-robust in general. Furthermore, for all $i \in [c]$, $\overline{h}_i(\cdot)$ is not 0 almost everywhere or 1 almost everywhere.

Theorem 3. *Suppose that Assumption 2 holds, and let $x \in \mathbb{R}^d$ be arbitrary. Let $y = \arg\max_i h_i(x)$ and $y' = \arg\max_{i\neq y} h_i(x)$. Then, if $\alpha \in [\frac{1}{2}, 1]$, it holds that $\arg\max_i h_i^\alpha(x+\delta) = y$ for all $\delta \in \mathbb{R}^d$ such that*

$$\|\delta\|_2 \leq r_\sigma^\alpha(x) \coloneqq \frac{\sigma}{2}\Big(\Phi^{-1}\big(\alpha h_y(x)\big) - \Phi^{-1}\big(\alpha h_{y'}(x) + 1 - \alpha\big)\Big).$$

Proof. First, note that since every $\overline{h}_i(\cdot)$ is not 0 almost everywhere or 1 almost everywhere, it holds that $h_i(x) \in (0,1)$ for all i and all x. Now, suppose that $\alpha \in [\frac{1}{2}, 1]$, and let $\delta \in \mathbb{R}^d$ be such that $\|\delta\|_2 \leq r_\sigma^\alpha(x)$. Let $\mu_\alpha \coloneqq \frac{1-\alpha}{\alpha}$ (conversely, $\alpha = \frac{1}{\mu_\alpha+1}$). We construct a scaled classifier $\tilde{h}\colon \mathbb{R}^d \to \mathbb{R}^c$, whose i^{th} entry is defined as

$$\tilde{h}_i(x) = \begin{cases} \frac{\overline{h}_y(x)}{1+\mu_\alpha} = \alpha\overline{h}_y(x) & \text{if } i = y, \\ \frac{\overline{h}_i(x)+\mu_\alpha}{1+\mu_\alpha} = \alpha\overline{h}_i(x) + 1 - \alpha & \text{if } i \neq y. \end{cases}$$

Furthermore, define a scaled RS classifier $\hat{h}\colon \mathbb{R}^d \to \mathbb{R}^c$ based on $\tilde{h}_i(\cdot)$ by

$$\hat{h}(x) = \mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I_d)}\left[\tilde{h}(x+\xi)\right].$$

Then, since it holds that

$$\begin{aligned} \tilde{h}_y(x) &= \frac{\overline{h}_y(x)}{1+\mu_\alpha} \in \left(0, \frac{1}{1+\mu_\alpha}\right) \subseteq (0,1), \\ \tilde{h}_i(x) &= \frac{\overline{h}_i(x)+\mu_\alpha}{1+\mu_\alpha} \in \left(\frac{\mu_\alpha}{1+\mu_\alpha}, 1\right) \subseteq (0,1), \quad \forall i \neq y, \end{aligned}$$

it must be the case that $0 < \tilde{h}_i(x) < 1$ for all i and all x, and hence, for all i, the function $x \mapsto \Phi^{-1}\big(\hat{h}_i(x)\big)$ is ℓ_2-Lipschitz continuous with Lipschitz constant $\frac{1}{\sigma}$ (see [52, Lemma 1], or Lemma 2 in [71] and the discussion thereafter). Therefore,

$$\left|\Phi^{-1}\big(\hat{h}_i(x+\delta)\big) - \Phi^{-1}\big(\hat{h}_i(x)\big)\right| \leq \frac{\|\delta\|_2}{\sigma} \leq \frac{r_\sigma^\alpha(x)}{\sigma} \tag{9}$$

for all i. Applying (9) for $i = y$ yields that

$$\Phi^{-1}\big(\hat{h}_y(x+\delta)\big) \geq \Phi^{-1}\big(\hat{h}_y(x)\big) - \frac{r_\sigma^\alpha(x)}{\sigma}, \tag{10}$$

and, since Φ^{-1} is monotonically increasing and $\hat{h}_i(x) \leq \hat{h}_{y'}(x)$ for all $i \neq y$, applying (9) to $i \neq y$ gives that

$$\Phi^{-1}\big(\hat{h}_i(x+\delta)\big) \leq \Phi^{-1}\big(\hat{h}_i(x)\big) + \frac{r_\sigma^\alpha(x)}{\sigma} \leq \Phi^{-1}\big(\hat{h}_{y'}(x)\big) + \frac{r_\sigma^\alpha(x)}{\sigma}. \tag{11}$$

Subtracting (11) from (10) gives that

$$\Phi^{-1}\big(\hat{h}_y(x+\delta)\big) - \Phi^{-1}\big(\hat{h}_i(x+\delta)\big) \geq \Phi^{-1}\big(\hat{h}_y(x)\big) - \Phi^{-1}\big(\hat{h}_{y'}(x)\big) - \frac{2r_\sigma^\alpha(x)}{\sigma}$$

for all $i \neq y$. By the definitions of μ_α, $r_\sigma^\alpha(x)$, and $\hat{h}(x)$, the right-hand side of this inequality equals zero, and hence, since Φ is monotonically increasing, we find that $\hat{h}_y(x+\delta) \geq \hat{h}_i(x+\delta)$ for all $i \neq y$. Therefore,

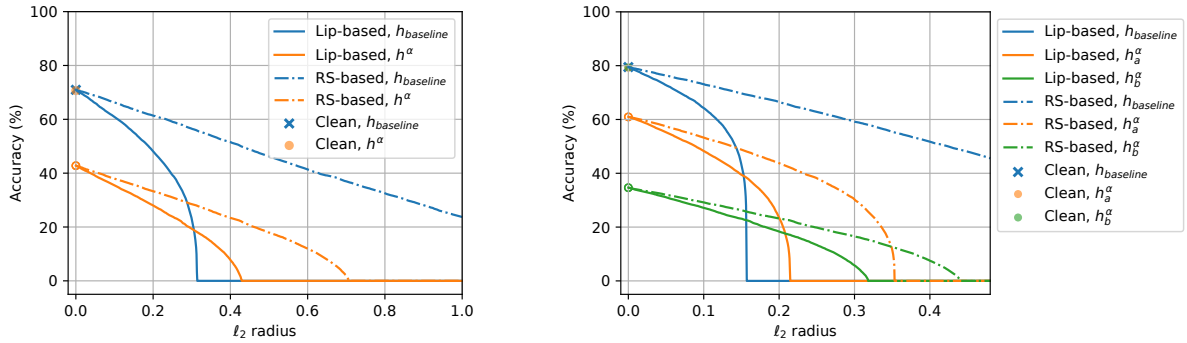
$$\begin{aligned}\frac{h_y(x+\delta)}{1+\mu_\alpha} =& \mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I_d)}\left[\frac{\overline{h}_y(x+\delta+\xi)}{1+\mu_\alpha}\right] = \hat{h}_y(x+\delta) \\ \geq& \hat{h}_i(x+\delta) = \mathbb{E}_{\xi\sim\mathcal{N}(0,\sigma^2 I_d)}\left[\frac{\overline{h}_i(x+\delta+\xi)+\mu_\alpha}{1+\mu_\alpha}\right] = \frac{h_i(x+\delta)+\mu_\alpha}{1+\mu_\alpha}.\end{aligned}$$

Hence, $h_y(x+\delta) \geq h_i(x+\delta) + \mu_\alpha$ for all $i \neq y$, so $h(\cdot)$ is certifiably robust at x with margin $\mu_\alpha = \frac{1-\alpha}{\alpha}$ and radius $r_\sigma^\alpha(x)$. Therefore, by Lemma 1, it holds that $\arg\max_i h_i^\alpha(x+\delta) = y$ for all $\delta \in \mathbb{R}^d$ such that $\|\delta\|_2 \leq r_\sigma^\alpha(x)$, which concludes the proof. □

A.1. Visualization of the Certified Robust Radii

In this section of the appendix, we visualize the certified robust radii presented in Theorem 1 and Theorem 3. Again, the smoothing strength α is a fixed value. Since a (Gaussian) RS model with smoothing covariance matrix $\sigma^2 I_d$ has an ℓ_2-Lipschitz constant $\sqrt{\frac{2}{\pi\sigma^2}}$, such a model can be used to simultaneously visualize both theorems, with Theorem 3 giving tighter certificates of robustness. Note that RS models with a larger smoothing variance certify larger radii but achieve lower clean accuracies, and vice versa. Here, we consider the CIFAR-10 dataset and select $g(\cdot)$ to be a ConvNeXT-T model with a clean accuracy of 97.25%, and use the RS models presented in [84] as $h(\cdot)$. For a fair comparison, we select an α value such that the clean accuracy of the constructed mixed classifier $h^\alpha(\cdot)$ matches that of another RS model $h_{\text{baseline}}(\cdot)$ with a smaller smoothing variance. The expectation term in the RS formulation is approximated with the empirical mean of 10,000 random perturbations[2] drawn from $\mathcal{N}(0, \sigma^2 I_d)$, and the certified radii of $h_{\text{baseline}}(\cdot)$ are calculated using Theorems 1 and 3 by setting α to 1. Figure 6 displays the calculated certified accuracies of $h^\alpha(\cdot)$ and $h_{\text{baseline}}(\cdot)$ at various attack radii. The ordinate "Accuracy" at a given abscissa "ℓ_2 radius" reflects the percentage of the test data for which the considered model gives a correct prediction and a certified radius at least as large as the ℓ_2 radius.

[2]The authors of [25] showed that 10,000 Monte Carlo samples are sufficient to provide representative results.



(a) $h_{\text{baseline}}(\cdot)$: RS with $\sigma = 0.5$.
$h^\alpha(\cdot)$ uses $\alpha = 0.76$;
$h(\cdot)$ is RS with $\sigma = 1$.

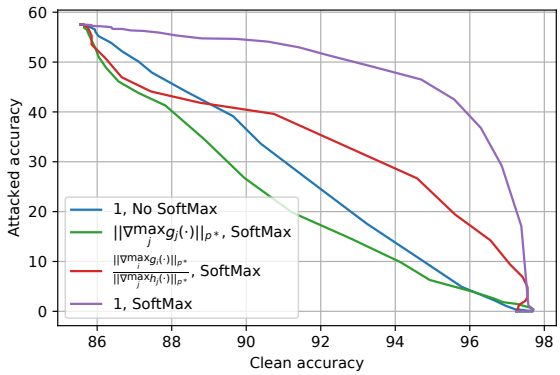
(b) $h_{\text{baseline}}(\cdot)$: RS with $\sigma = 0.25$.
Consider two mixed classifier examples:
$h_a^\alpha(\cdot)$ uses $\alpha = 0.76$ and $h_a(\cdot)$ is RS with $\sigma = 0.5$;
$h_b^\alpha(\cdot)$ uses $\alpha = 0.67$ and $h_b(\cdot)$ is RS with $\sigma = 1.0$.

Figure 6. Comparing the certified accuracy-robustness trade-off of RS models and our mixed classifier using both Lipschitz(Lip)-based certificates and RS-based certificates (Theorems 1 and 3, respectively). The clean accuracies are the same between $h_{\text{baseline}}(\cdot)$ and $h^\alpha(\cdot)$ in each subfigure, and the empty circles represent certified accuracy discontinuities at radius 0.

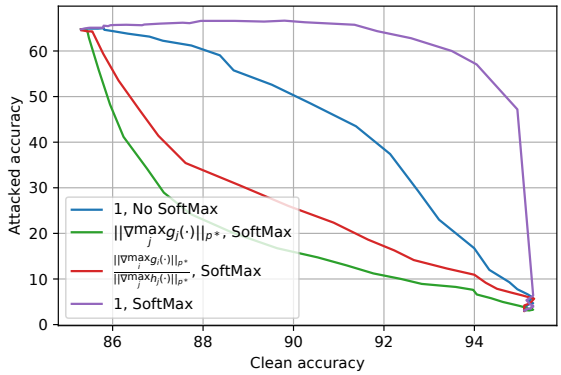
In both subplots of Figure 6, the certified robustness curves of $h^\alpha(\cdot)$ do not connect to the clean accuracy when α approaches zero. This is because Theorems 1 and 3 both consider robustness with respect to $h(\cdot)$ and do not issue certificates to test inputs at which $h(\cdot)$ makes incorrect predictions, even if $h^\alpha(\cdot)$ predicts correctly at some of these points. This is reasonable because we do not assume any robustness or Lipschitzness of $g(\cdot)$, and $g(\cdot)$ is allowed to be arbitrarily incorrect whenever the radius is non-zero.

The Lipschitz-based bound of Theorem 1 allows us to visualize the performance of the mixed classifier $h^\alpha(\cdot)$ when $h(\cdot)$ is an ℓ_2-Lipschitz model. In this case, the curves associated with $h^\alpha(\cdot)$ and $h_{\text{baseline}}(\cdot)$ intersect, with $h^\alpha(\cdot)$ achieving higher certified accuracy at larger radii and $h_{\text{baseline}}(\cdot)$ certifying more points at smaller radii. By adjusting α and the Lipschitz constant of $h(\cdot)$, it is possible to change the location of this intersection while maintaining the same level of clean accuracy. Therefore, the mixed classifier structure allows for optimizing the certified accuracy at a particular radius, while keeping the clean accuracy unchanged.

The RS-based bound from Theorem 3 captures the behavior of $h^\alpha(\cdot)$ when $h(\cdot)$ is an RS model. For both $h^\alpha(\cdot)$ and $h_{\text{baseline}}(\cdot)$, the RS-based bounds certify larger radii than the corresponding Lipschitz-based bounds. Nonetheless, $h_{\text{baseline}}(\cdot)$ can certify more points with the RS-based guarantee. Intuitively, this phenomenon suggests that RS models can yield correct but low-confidence predictions when under attack with a large radius, and thus may not be best-suited for our mixing operation, which relies on robustness with non-zero margins. In contrast, Lipschitz models, a more general and common class of models, exploit the mixing operation more effectively. Moreover, as shown in Figure 4, empirically robust models often yield high-confidence predictions when under attack, making them more suitable for the robust base classifier for $h^\alpha(\cdot)$.



(a) ConvNeXT-T and TRADES WideResNet-34 under ℓ_∞ PGD attack.



(b) Standard and AT ResNet-18s under ℓ_2 PGD attack.

Figure 7. Comparing the "attacked accuracy versus clean accuracy" curve of various options for $R_i(x)$ with alternative selections of base classifiers.

B. Additional Analyses Regarding $R_i(x)$

B.1. The Four Options for $R_i(x)$

Consider the four listed options of $R_i(x)$, namely 1, $\|\nabla g_i(x)\|_{p*}$, $\|\nabla \max_j g_j(x)\|_{p*}$, and $\frac{\|\nabla g_i(x)\|_{p*}}{\|\nabla h_i(x)\|_{p*}}$. The constant 1 is a straightforward option. $\|\nabla g_i(x)\|_{p*}$ comes from (3), which is a direct generalization from the locally biased smoothing (binary classification) formulation to the multi-class case. Note that $\|\nabla g_i(x)\|_{p*}$ is not practical for datasets with a large number of classes, since it requires the calculation of the full Jacobian of $g(x)$, which is very time-consuming. To this end, we use the gradient of the predicted class (which is intuitively the most important class) as a surrogate for all classes, bringing the formulation $\|\nabla \max_j g_j(x)\|_{p*}$. Finally, unlike locally biased smoothing, which only has one differentiable component, our adaptive smoothing has two differentiable base networks. Hence, it makes sense to consider the gradient from both of them. Intuitively, if $\|\nabla g_i(x)\|_{p*}$ is large, then $g(\cdot)$ is vulnerable at x and we should trust it less. If $\|\nabla h_i(x)\|_{p*}$, then $h(\cdot)$ is vulnerable and we should trust $h(\cdot)$ less. This leads to the fourth option, which is $\frac{\|\nabla g_i(x)\|_{p*}}{\|\nabla h_i(x)\|_{p*}}$.

B.2. Additional Empirical Supports and Analyses for Selecting $R_i(x) = 1$

In this section, we use additional empirical evidence (Figures 7a and 7b) to show that $R_i(x) = 1$ is the appropriate choice for the adaptive smoothing formulation, and that the post-SoftMax probabilities should be used for smoothing. While most of the experiments in this paper are based on ResNets, the architecture is chosen solely because of its popularity, and our method does not depend on any properties of ResNets. Therefore, for the experiment in Figure 7a, we select an alternative architecture by using a more modern ConvNeXT-T model [56] pre-trained on ImageNet-1k as $g(\cdot)$. We also use a robust model trained via TRADES in place of an adversarially-trained network for $h(\cdot)$. Moreover, while most of our experiments are based on ℓ_∞ attacks, our method applies to all ℓ_p attack budgets. In Figure 7b, we provide an example that considers the ℓ_2 attack. The experiment settings are summarized in Table 6.

Figures 7a and 7b demonstrate that setting $R_i(x)$ to the constant 1 achieves the best trade-off

Table 6. Experiment settings for comparing the choices of $R_i(x)$.

	PGD attack settings	$g(\cdot)$ Architecture	$h(\cdot)$ Architecture
Figure 1	ℓ_∞, $\epsilon = 8/255$, 10 Steps	Standard ResNet-18	ℓ_∞ AT ResNet-18
Figure 7a	ℓ_∞, $\epsilon = 8/255$, 20 Steps	Standard ConvNeXT-T	ℓ_∞ TRADES WideResNet-34
Figure 7b	ℓ_2, $\epsilon = 0.5$, 20 Steps	Standard ResNet-18	ℓ_2 AT ResNet-18

curve between clean and attacked accuracy. Moreover, smoothing using the post-SoftMax probabilities outperforms the pre-SoftMax logits. This result aligns with the conclusions of Figure 1 and our theoretical analyses, demonstrating that various robust networks share the property of being more confident when classifying correctly than when making mistakes.

The most likely reason for $R_i(x) = 1$ to be the best choice is that while the local Lipschitzness of a base classifier is a good estimator of its robustness and trustworthiness (as motivated in [8]), the gradient magnitude of this base classifier at the input is not always a good estimator of its local Lipschitzness. Specifically, local Lipschitzness, as defined in Definition 2, requires the classifier to be relatively flat within an ϵ-ball around the input, whereas the gradient magnitude only focuses on the nominal input itself and does not consider the landscape within the ϵ-ball. For example, the gradient magnitude of the standard base classifier $g(\cdot)$ may jump from a small value at the input to a large value at some nearby point within the ϵ-ball, which may cause $g(\cdot)$ to change its prediction around this nearby point. In this case, $\|\nabla g(x))\|$ may be small, but $g(\cdot)$ can have a high local Lipschitz constant.

As a result, while using $\|\nabla g(\cdot))\|$ as R_i seems to make sense at first glance, it does not work as intended and can make the mixed classifier trust $g(\cdot)$ more than it should. Therefore, within the ϵ-ball around a given x, the attacker may be able to find adversarial perturbations at which the gradient magnitude is small, thereby bypassing the defense.

In fact, as discussed in [8], the use of gradient magnitude results from approximating a neural classifier with a linear classifier. Our Figure 1 shows that such an approximation results in a large mismatch and therefore does not make sense in our setting.

We also note that even if some gradient-dependent options for $R_i(x)$ work better than the constant 1, unless it works significantly better, the constant 1 should still be favored since it avoids performing backward passes within the forward pass of the mixed classifier, making the mixing formulation more efficient and less likely to suffer from gradient masking.

C. Experiment Implementation Details

C.1. Implementation of the Mixing Network in Experiments

Since our formulation does not depend on the architecture of the base classifiers, Figure 3 presents the design of the mixing network in the context of general standard and robust classifiers. In the experiments presented in Section 5.2, Both $g(\cdot)$ and $h(\cdot)$ are based on variants of the ResNet family, which share the general structure of having four main blocks. Thus, we present the structure of the mixed classifier with ResNet-like base models in Figure 8. Following [59], we consider the initial Conv2D layer and the first ResNet block as the upstream layers. The embeddings extracted by the first Conv2D layers in $g(\cdot)$ and $h(\cdot)$ are concatenated before being provided to the mixing network

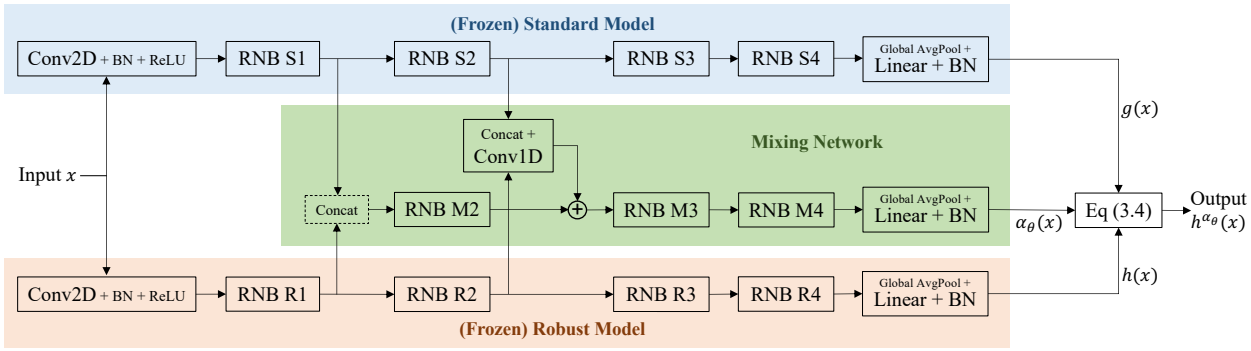


Figure 8. The architecture of the mixed classifier introduced in Section 4 when applied to a pair of ResNet base models.

$\alpha_\theta(\cdot)$. We further select the second ResNet block as the middle layers. For this layer, in addition to concatenating the embeddings from $g(\cdot)$ and $h(\cdot)$, we also attach a linear transformation layer (Conv1x1) to match the dimensions, reduce the number of features, and improve efficiency.

As mentioned in Section 4.1, the range of $\alpha_\theta(\cdot)$ can be constrained to be within $(\alpha_{\min}, \alpha_{\max}) \subseteq [0, 1]$ if certified robustness is desired. We empirically observe that setting $\alpha_{\max} - \alpha_{\min}$ to be 0.04 works well for CIFAR-10, whereas 0.1 or 0.15 works well for CIFAR-100. This observation coincides with Figure 4, which shows that a slight increase in α can greatly enhance the robustness at the most sensitive region. The value of $\alpha_{\min}$ can then be determined by enforcing a desired level of either clean validation accuracy or robustness. Following this guideline, we set the ranges of $\alpha_\theta(\cdot)$ to be $(0.96, 1)$ for the model discussed in Table 4. The range is $(0.84, 0.99)$ and $(0.815, 0.915)$ respectively for the two models demonstrated in Table 5. Note that this range is only applied during validation. When training $\alpha_\theta(\cdot)$, we use the full $(0, 1)$ range for its output, so that the training-time adversary can generate strong and diverse attacks that fully exploit $\alpha_\theta(\cdot)$, which is crucial for securing the robustness of the mixing network.

C.2. Ablation Study in Loss Function Hyperparameters

In this section, we discuss the effects of the constants c_{CE}, c_{BCE}, and c_{prod} in the composite loss function (8). Since multiplying the three weight constants by the same number is equivalent to using a larger optimizer step size and is not the focus of this ablation study (we focus on the loss function shape), we fix $c_{\text{CE}} + c_{\text{BCE}} = 1.5$. To avoid the issue of becoming excessively conservative and always prioritizing the robust base model (as described in Section 4.4), we add a batch normalization layer without trainable affine transform to the output of the mixing network. Additionally, note that since the mixing network has a single output, one can arbitrarily shift this output to achieve the desired balance between clean and attacked accuracies. For a fair and illustrative comparison, after training a mixing network $\alpha_\theta(\cdot)$ with each hyperparameter setting, we add an appropriate constant to the output of the $\alpha_\theta(\cdot)$ so that the clean accuracy of the overall model $h^{\alpha_\theta}(\cdot)$ is $90 \pm 0.02\%$, and compare the PGD_{20} attacked accuracy of $h^{\alpha_\theta}(\cdot)$ in Table 7. As a baseline, when the smoothing strength α is a constant, the PGD_{20} accuracy is 52.6% when the clean accuracy is tuned to be 90% (the corresponding α value is 1.763). The above results demonstrate that $c_{\text{CE}} = 0$, $c_{\text{BCE}} = 1.5$, and $c_{\text{prod}} = 0.2$ works the best.

Table 7. The PGD_{20} accuracy on CIFAR-10 with various loss hyperparameter settings. The setting is the same as in Table 2, and we consider both attack and defense in Setting B.

	$c_{\text{CE}} = 0$ $c_{\text{BCE}} = 1.5$	$c_{\text{CE}} = 0.5$ $c_{\text{BCE}} = 1$	$c_{\text{CE}} = 1$ $c_{\text{BCE}} = 0.5$	$c_{\text{CE}} = 1.5$ $c_{\text{BCE}} = 0$
$c_{\text{prod}} = 0$	54.5 %	52.8 %	53.8 %	54.4 %
$c_{\text{prod}} = 0.1$	54.3 %	54.1 %	54.0 %	54.1 %
$c_{\text{prod}} = 0.2$	55.1 %	54.2 %	54.3 %	53.9 %

Table 8. Compare selections of the mixing network's Sigmoid activation scaling factor.

Scale = 0.5	Scale = 1	Scale = 2	Scale = 4
55.1 %	55.5 %	55.7 %	55.6 %

Our results also show that a small positive c_{prod} is generally beneficial. This makes sense because the CE loss is low for a particular input if both $g(\cdot)$ and $h(\cdot)$ correctly predict its class. Thus, the smoothing strength should not matter for such input, and therefore the BCE loss is weighted by a small number. Compared with using only the BCE loss, the product term of the CE and the BCE components is lenient on inputs correctly classified by the mixed model $h^{\alpha_\theta}(\cdot)$, while assigning more weight to the data that are incorrectly predicted.

Recall that the output range of $\alpha_\theta(\cdot)$ is $[0, 1]$, which is enforced by appending a Sigmoid output activation function. In addition to shifting, one can arbitrarily scale the Sigmoid activation's input. By performing this scaling, we effectively calibrate the confidence of the mixing network. In Table 7, this scaling is set to the same constant for all settings. In Table 8, we select the best loss parameter and analyze the validation-time Sigmoid scaling. Again, we shift the Sigmoid input so that the clean accuracy is $90 \pm 0.02\%$. While a larger scale benefits the performance on clean/attacked examples that are confidently recognized by the mixing network, an excessively large scale makes $h^{\alpha_\theta}(\cdot)$ less stable under attack. Table 8 shows that applying a scaling factor of 2 yields the best result for the given experiment setting.

C.3. Calculating the AutoAttacked Robust Confidence Gap

The original AutoAttack implementation released with [29] does not return perturbations that fail to change the model prediction. To enable robustness margin calculation, we modify the code to always return the perturbation that achieves the smallest margin during the attack optimization. Since the FAB and Square components of AutoAttack are slow and do not successfully attack additional images on top of the APGD components, we only consider the two APGD components for the purpose of margin calculation.