ADVERSARIAL TRAINING WITH RECTIFIED REJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial training (AT) is one of the most effective strategies for promoting model robustness, whereas even the state-of-the-art adversarially trained models struggle to exceed 65% robust test accuracy on CIFAR-10 without additional data, which is far from practical. A natural way to improve beyond this accuracy bottleneck is to introduce a rejection option, where confidence is a commonly used certainty proxy. However, the vanilla confidence can overestimate the model certainty if the input is wrongly classified. To this end, we propose to use true confidence (T-Con) (i.e., predicted probability of the true class) as a certainty oracle, and learn to predict T-Con by rectifying confidence. Intriguingly, we prove that under mild conditions, a rectified confidence (R-Con) rejector and a confidence rejector can be coupled to distinguish any wrongly classified input from correctly classified ones. We also quantify that training R-Con to be aligned with T-Con could be an easier task than learning robust classifiers. In our experiments, we evaluate our rectified rejection (RR) module on CIFAR-10, CIFAR-10-C, and CIFAR-100 under several attacks, and demonstrate that the RR module is well compatible with different AT frameworks on improving robustness, with little extra computation.

1 INTRODUCTION

The adversarial vulnerability of machine learning models has been widely studied because of its counter-intuitive behavior and the potential effect on safety-critical tasks (Biggio et al., 2013; Goodfellow et al., 2015; Szegedy et al., 2014). Towards this end, many defenses have been proposed, but most of them can be evaded by adaptive attacks (Athalye et al., 2018; Tramer et al., 2020). Among the previous defenses, adversarial training (AT) is recognized as an effective defending approach (Madry et al., 2018; Zhang et al., 2019b). Nonetheless, as reported in RobustBench (Croce et al., 2020), the state-of-the-art AT methods still struggle to exceed 65% robust test accuracy on CIFAR-10 without extra data, even after exploiting large model architectures (Gowal et al., 2020; Rebuffi et al., 2021; Sehwag et al., 2021; Wu et al., 2020), which is far from practical requirements.

An improvement can be naturally achieved by incorporating a rejection or detection module along with the adversarially trained classifier, which enables the model to refuse to make predictions for abnormal inputs (Kato et al., 2020; Laidlaw and Feizi, 2019; Stutz et al., 2020). However, although previous rejectors trained via margin-based objectives or confidence calibration can capture some aspects of prediction certainty, they may overestimate the certainty, especially on wrongly classified samples (discussed in Section 5). Furthermore, Tramer (2021) argues that learning a robust rejector could suffer from a similar accuracy bottleneck as learning robust classifiers, which may be caused by data insufficiency (Schmidt et al., 2018) or poor generalization (Yang et al., 2020c).

To solve these problems, we first observe that the *true* cross-entropy loss $-\log f_\theta(x)[y]$ reflects how well the classifier $f_\theta(x)$ is generalized on the input x (Goodfellow et al., 2016), assuming that we can access its true label y. Thus, we propose to treat **true confidence (T-Con)** $f_\theta(x)[y]$, i.e., the predicted probability on the true label as a certainty oracle. Note that T-Con is different from the commonly used **confidence**, which is obtained by taking the maximum as $\max_l f_\theta(x)[l]$.

As we shall see in Table 1, executing the rejection based on T-Con can largely increase the test accuracy under a given true positive rate for both standardly and adversarially trained models. Another intriguing fact about T-Con is that *if we first threshold confidence by $\frac{1}{2}$, then T-Con can perfectly distinguish any wrongly classified input from correctly classified ones* (formally stated in Lemma 1). This inspires us that instead of employing a single metric, we can couple two connected metrics like confidence and T-Con to execute certified rejection options.

The property of T-Con is compelling, but its computation is unfortunately not realizable during inference since the absence of the true label y. This motivates us to construct the **rectified confidence (R-Con)** to learn to predict T-Con, by rectifying confidence via an auxiliary function. We prove that if R-Con is trained to be aligned with T-Con within ξ-error, then *a ξ-error R-Con rejector and a $\frac{1}{2-\xi}$ confidence rejector can be coupled to distinguish any wrongly classified input from correctly classified ones*, as formally described in Section 4.2.

Technically, as illustrated in Fig. 1, we adopt a two-head structure to model the classifier and our rectified rejection (RR) module, while adversarially training them in an end-to-end manner. In particular, our rejection module is learned by minimizing an extra BCE loss between T-Con and R-Con. The design of a shared main body saves computation and memory costs. Stopping gradients on the confidence $f_\theta(x)[y^m]$ when $y^m = y$ can avoid focusing on easy examples and keep the optimal solution of classifier unbiased.

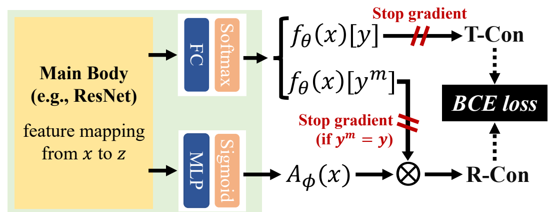


Figure 1: Construction of the objective $\mathcal{L}_{\mathrm{RR}}$ in Eq. (4) for training the RR module, which is the binary cross-entropy (BCE) loss between T-Con and R-Con.

Empirically, we evaluate the performance of our RR module on CIFAR-10, CIFAR-10-C, and CIFAR-100 (Hendrycks and Dietterich, 2019; Krizhevsky and Hinton, 2009) with extensive experiments. In Section 4, we verify the certified rejection options obtained by coupling confidence and R-Con. To fairly compare with previous baselines, we also use R-Con alone as the rejector, and report both the accuracy for a given true positive rate and the ROC-AUC scores in Section 6. We perform ablation studies on the construction of R-Con, and design adaptive attacks to evade our RR module. Our results demonstrate that the RR module is well compatible with different AT frameworks, and can consistently facilitate the returned predictions to achieve higher robust accuracy under several attacks and threat models, with little computational burden, and is easy to implement.

2 Related work

In the literature of standard training, Cortes et al. (2016) first propose to *jointly* learn the classifier and rejection module, which is later extended to deep networks (Geifman and El-Yaniv, 2017; 2019). Recently, Laidlaw and Feizi (2019) and Kato et al. (2020) jointly learn the rejection option during adversarial training (AT) via margin-based objectives, whereas they abandon the ready-made information from confidence that is shown to be a simple but good solution of rejection for PGD-AT (Wu et al., 2018). On the other hand, Stutz et al. (2020) propose confidence-calibrated AT (CCAT) by adaptive label smoothing, leading to preciser rejection on unseen attacks. However, this calibration acts on the true classes in training, while the confidences obtained by the maximal operation during inference may not follow the calibrated property, especially on the misclassified inputs. In contrast, we exploit true confidence (T-Con) as a certainty oracle (detailed in Section 3.1), and propose to learn T-Con by rectifying confidence, in an adversarially end-to-end manner. As seen in our experiments (e.g., Table 2 and Table 3), our RR module is compatible with CCAT, where R-Con is trained to be aligned with the calibrated T-Con. In Appendix B, we introduce more backgrounds on adversarial training and detection methods, where several representative methods are involved as our baselines.

3 Classification with a rejection option

Consider a data pair (x, y), with $x \in \mathbb{R}^d$ as the input and y as the true label. We refer to $f_\theta(x) : \mathbb{R}^d \to \Delta^L$ as a classifier parameterized by θ, where Δ^L is the probability simplex of L classes. Following Geifman and El-Yaniv (2019), a classifier with a rejection module $\mathcal{M}$ can be formulated as

$$(f_\theta, \mathcal{M})(x) \triangleq \begin{cases} f_\theta(x), & \text{if } \mathcal{M}(x) \geq t; \\ \text{don't know}, & \text{if } \mathcal{M}(x) < t, \end{cases} \tag{1}$$

where t is a threshold, and $\mathcal{M}(x)$ is a certainty proxy computed by auxiliary models or statistics.

What to reject? The design of $\mathcal{M}$ is principally decided by what kinds of inputs we intend to reject. In the adversarial setting, most of the previous detection methods aim to reject adversarial examples, which are usually misclassified by standardly trained models (STMs) (Carlini and Wagner, 2017a). In this case, the misclassified and adversarial characters are considered as associated by default. However, for adversarially trained models (ATMs) on CIFAR-10, more than 50% adversarial inputs are correctly classified (Croce and Hein, 2020). Hence, it is more reasonable to execute rejection depending on whether the input will be misclassified rather than adversarial.

3.1 TRUE CONFIDENCE (T-CON) AS A CERTAINTY ORACLE

To reject misclassified inputs, there are many ready-made choices for computing $\mathcal{M}(x)$. We use $f_\theta(x)[l]$ to represent the returned probability on the l-th class, and denote the predicted label as

$$y^m = \arg\max_l f_\theta(x)[l], \tag{2}$$

where $f_\theta(x)[y^m]$ is usually termed as **confidence** (Goodfellow et al., 2016). In the standard setting, confidence is shown to be one of the best certainty proxies for a trained network (Geifman and El-Yaniv, 2017), which is often used by practitioners. However, the confidence returned by STMs can be adversarially fooled (Moosavi-Dezfooli et al., 2016).

Different from confidence which is obtained by taking the maximum as $\max_l f_\theta(x)[l]$, we introduce **true confidence (T-Con)** defined as $f_\theta(x)[y]$, i.e., the returned probability on the true label y. When classifiers are trained by minimizing cross-entropy loss $\mathbb{E}[-\log f_\theta(x)[y]]$, the value of $-\log f_\theta(x)[y]$ can better reflect how well the model is generalized on a new input x during inference, compared to its empirical approximation $-\log f_\theta(x)[y^m]$, especially when x is misclassified (i.e., $y^m \neq y$).

Empirically in Table 1, we adversarially train a classifier on CIFAR-10, and evaluate the effects of confidence and T-Con as the rejection metric $\mathcal{M}$, respectively. We report the accuracy without rejection ('All'), and the accuracy when fixing the rejection threshold at 95% true positive rate ('TPR-95') w.r.t. confidence or T-Con[1], i.e., at most 5% correctly classified examples are rejected. As seen, thresholding on T-Con can largely improve the accuracy.

Table 1: Test accuracy (%) of ResNet-18.

	Inputs	All	TPR-95	
			Con.	T-Con
Stan.	Clean	95.36	98.40	**100.0**
	PGD-10	0.22	0.18	**100.0**
Adv.	Clean	82.67	87.39	**96.55**
	PGD-10	53.58	57.23	**88.75**
Availability			✓	✗

To explain the results, note that STMs tend to return high confidences, e.g., 0.95 on both clean and adversarial inputs (Nguyen et al., 2015), then if an input x is correctly classified, there is T-Con$(x) = 0.95$; otherwise T-Con$(x) < 1 - 0.95 = 0.05$. Thus it is reasonable to see that thresholding on T-Con for STMs can lead to TPR-95 accuracy of 100% as in Table 1. As a result, we treat T-Con as a certainty oracle, and confidence is actually a proxy of T-Con in inference when we cannot access the true label y. In Section 4, we propose a better proxy R-Con to approximate T-Con.

3.2 CERTIFIED SEPARABILITY BY COUPLING CONFIDENCE AND T-CON

Instead of using a single metric, we find an intriguing fact that properly coupling confidence and T-Con can certifiably separate wrongly and correctly classified inputs, as stated below:

Lemma 1. *(Certified separability) Given the classifier f_θ, $\forall x_1, x_2$ with confidences larger than $\frac{1}{2}$, i.e., $f_\theta(x_1)[y_1^m] > \frac{1}{2}$ and $f_\theta(x_2)[y_2^m] > \frac{1}{2}$. If x_1 is correctly classified as $y_1^m = y_1$, while x_2 is wrongly classified as $y_2^m \neq y_2$, then there is T-Con$(x_1) > \frac{1}{2} >$ T-Con(x_2).*

Proof. Since x_1 is correctly classified, i.e., $y_1^m = y_1$, we have $f_\theta(x_1)[y_1] = f_\theta(x_1)[y_1^m] > \frac{1}{2}$. On the other hand, since x_2 is wrongly classified, i.e., $y_1^m \neq y_1$, we have $f_\theta(x_1)[y_1] \leq 1 - f_\theta(x_1)[y_1^m] < \frac{1}{2}$. Thus we have T-Con$(x_1) > \frac{1}{2} >$ T-Con(x_2). □

Intuitively, Lemma 1 indicates that if we first threshold confidence to be larger than $\frac{1}{2}$, then for any x that pass the confidence rejector, there is T-Con$(x) < \frac{1}{2}$ if x is misclassified; otherwise T-Con$(x) > \frac{1}{2}$. Note that there is no constraint on how the misclassification is caused, i.e., wrongly classified inputs can be adversarial examples, generally corrupted ones, or just the clean samples.

[1]Here we assume that the true labels are known when computing T-Con.

4 LEARNING T-CON VIA RECTIFYING CONFIDENCE

In this section, we describe learning T-Con via rectifying confidence, and formally present the certified separability and the learning difficulty of rectified confidence. Proofs are provided in Appendix A.

4.1 CONSTRUCTION OF RECTIFIED CONFIDENCE (R-CON)

When the input x is correctly classified by f_θ, i.e., $y^m = y$, the values of confidence and T-Con become aligned. This inspires us to learn T-Con by rectifying confidence, instead of modeling T-Con from scratch, which facilitates optimization and is conducive to preventing the classifier and the rejector from competing for model capacity. Namely, we introduce an auxiliary function $A_\phi(x) \in [0, 1]$, parameterized by ϕ, and construct the **rectified confidence (R-Con)** as[2]

$$\text{R-Con}(x) = f_\theta(x)[y^m] \cdot A_\phi(x). \tag{3}$$

In training, we encourage R-Con to be aligned with T-Con. This can be achieved by minimizing the binary cross-entropy (BCE) loss (detailed implementation seen in Appendix C.1). Other alternatives like margin-based objectives (Kato et al., 2020) or mean square error can also be applied. The training objective of our rectified rejection (RR) module can be written as

$$\mathcal{L}_{\text{RR}}(x, y; \theta, \phi) = \text{BCE}\left(f_\theta(x)[y^m] \cdot A_\phi(x) \parallel f_\theta(x)[y]\right), \tag{4}$$

where the optimal solution of minimizing $\mathcal{L}_{\text{RR}}$ with respect to ϕ is $A^*_\phi(x) = \frac{f_\theta(x)[y]}{f_\theta(x)[y^m]}$. The auxiliary function $A_\phi(x)$ can be jointly learned with the classifier $f_\theta(x)$ during AT by optimizing

$$\min_{\theta,\phi} \mathbb{E}_{p(x,y)}\big[\underbrace{\mathcal{L}_{\text{T}}(x^*, y; \theta)}_{\textbf{classification}} + \lambda \cdot \underbrace{\mathcal{L}_{\text{RR}}(x^*, y; \theta, \phi)}_{\textbf{rectified rejection}}\big], \text{ where } x^* = \mathop{\arg\max}_{x' \in B(x)} \mathcal{L}_{\text{A}}(x', y; \theta). \tag{5}$$

Here λ is a hyperparameter, $B(x)$ is a set of allowed points around x (e.g., a ball of $\|x' - x\|_p \leq \epsilon$), $\mathcal{L}_{\text{T}}$ and $\mathcal{L}_{\text{A}}$ are the training and adversarial objectives for a certain AT method, respectively, where $\mathcal{L}_{\text{T}}$ and $\mathcal{L}_{\text{A}}$ can be either the same or chosen differently (Pang et al., 2020). Note that we can generalize Eq. (5) to involve clean inputs x in the outer minimization objective, which is compatible with the AT methods like TRADES. The inner maximization problem can also include ϕ.

Architecture of A_ϕ. We consider the classifier with a softmax layer as $f_\theta(x) = \mathbb{S}(Wz + b)$, where z is the mapped feature, W and b are the weight matrix and bias vector, respectively. We apply an extra shallow network to construct $A_\phi(x) = \text{MLP}_\phi(z)$, as illustrated in Fig. 1 and Appendix D.1. This two-head structure incurs little computational burden. Other more flexible architectures for A_ϕ can also be used, e.g., RBF networks (Sotgiu et al., 2020; Zadeh et al., 2018) or concatenating multi-block features that taking path information into account, and we do not further explore in this paper. Note that we stop gradients on the flows of $f_\theta(x)[y] \to$ BCE loss, and $f_\theta(x)[y^m] \to$ R-Con when $y^m = y$. These operations prevent the models from concentrating on correctly classified inputs, while facilitating $f_\theta(x)[y]$ to be aligned with $p_{\text{data}}(y|x)$, as detailed in Appendix C.1.

How well is A_ϕ learned? In practice, the auxiliary function $A_\phi(x)$ is usually trained to achieve the optimal solution $A^*_\phi(x)$ within a certain error. We introduce a definition on the *point-wise* error between $A_\phi(x)$ and $A^*_\phi(x)$, which admits two ways of measuring, either geometric or arithmetic:

Definition 1. *(point-wisely ξ-error) If at least one of the bounds holds at a point x:*

$$\text{Bound (i): } \left|\log\left(\frac{A_\phi(x)}{A^*_\phi(x)}\right)\right| \leq \log\left(\frac{2}{2-\xi}\right); \text{ Bound (ii): } \left|A_\phi(x) - A^*_\phi(x)\right| \leq \frac{\xi}{2}. \tag{6}$$

where $\xi \in [0, 1)$, then $A_\phi(x)$ is called ξ-error at input x.

We can show that given any A_ϕ that is better than a random guess at x, we can always find $\xi \in [0, 1)$ satisfying Definition 1. Specifically, assuming that A_ϕ simply performs random guess on x, i.e., $A_\phi(x) = \frac{1}{2}$. Since $A^*_\phi(x) \in [0, 1]$, there is $\left|A_\phi(x) - A^*_\phi(x)\right| = \left|\frac{1}{2} - A^*_\phi(x)\right| \leq \frac{1}{2}$, which means even a random-guess A_ϕ can satisfy Bound (ii) in Definition 1 with $\xi = 1$.

[2]It is also feasible to use an additive formula as $\text{R-Con}(x) = f_\theta(x)[y^m] - A_\phi(x)$.

4.2 CERTIFIED SEPARABILITY BY COUPLING CONFIDENCE AND R-CON

Recall that in Lemma 1 we present how to certifiably distinguish wrongly and correctly classified inputs, via referring to the values of confidence and T-Con. However, in practice we cannot compute T-Con without knowing the true label y. To this end, we substitute T-Con with R-Con during inference, and demonstrate that a $\frac{1}{2-\xi}$ confidence rejector and a R-Con rejector with ξ-error A_ϕ can be coupled to achieve certified separability, similar as the property of T-Con shown in Lemma 1.

Theorem 1. *(Certified separability) Given the classifier f_θ, for any pair of inputs x_1 and x_2 with confidences larger than $\frac{1}{2-\xi}$, i.e.,*

$$f_\theta(x_1)[y_1^m] > \frac{1}{2-\xi}, \text{ and } f_\theta(x_2)[y_2^m] > \frac{1}{2-\xi}, \tag{7}$$

where $\xi \in [0,1)$. If x_1 is correctly classified as $y_1^m = y_1$, while x_2 is wrongly classified as $y_2^m \neq y_2$, and A_ϕ is ξ-error at x_1, x_2, then there must be R-Con$(x_1) > \frac{1}{2} >$ R-Con(x_2).

Namely, after we first thresholding confidence by $\frac{1}{2-\xi}$, any misclassified input will obtain a R-Con value lower than any correctly classified one, as long as A_ϕ is trained to be ξ-error at these points. This property prevents adversaries from simultaneously fooling the predicted labels and R-Con values. As argued in Section 4.3, training A_ϕ to ξ-error could be easier than learning a robust classifier, which justifies the existence of wrongly classified but ξ-error points like x_2. In Fig. 2, we empirically verify Theorem 1 on a ResNet-18 (He et al., 2016) trained with the RR module on CIFAR-10. The test examples are perturbed by PGD-10 and filtered by a $\frac{1}{2-\xi}$ confidence rejector for each ξ. The remaining correctly and wrongly classified samples are separable w.r.t. the R-Con metric, even if we cannot compute ξ-error in practice without knowing true label y.

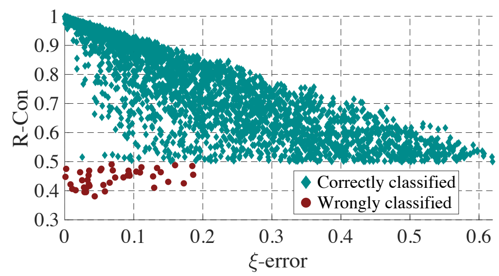


Figure 2: PGD-10 examples filtered by confidence value of $\frac{1}{2-\xi}$ for each ξ. R-Con can separate correctly and wrongly classified examples.

The effects of temperature tuning. It is known that for a softmax layer $f_\theta(x) = \mathbb{S}(\frac{Wz+b}{\tau})$ with a temperature scalar $\tau > 0$, the true label y and the predicted label y^m are invariant to τ, but the values of confidence and T-Con are not guaranteed to be order-preserving with respect to τ among different inputs. For instance, if there is $f_\theta(x_1)[y_1] < f_\theta(x_2)[y_2]$ under $\tau = 1$, it is possible that for other values of τ the inequality is reversed (detailed in Appendix C.2). As seen in Fig. 3, after we lower down the temperature τ during inference, more PGD-10 examples can satisfy the conditions in Theorem 1, on which R-Con can provably distinguish correctly and wrongly classified inputs.

4.3 THE DIFFICULTY OF LEARNING $A_\phi(x)$

Tramer (2021) advocates that learning a rejector is nearly as hard as learning a classifier against adversarial examples. So it would be informative to quantify the difficulty of training a ξ-error R-Con rejector. As learning $A_\phi(x)$ is a regression task with $A_\phi(x)$ bounded in $[0,1]$ by model design, we can convert the task of learning ξ-error $A_\phi(x)$ to a substituted classification task as:

Theorem 2. *(Substituted learning task of $A_\phi(x)$) The task of learning a ξ-error $A_\phi(x)$ can be reconstructed into a classification task with number of classes as N_{sub}, where*

$$N_1 = \frac{\log \rho^{-1}}{\log\left(\frac{2}{2-\xi}\right)} + 1,\, N_2 = \frac{2}{\xi}, \text{ and } N_{sub} = \lceil \min(N_1, N_2) \rceil. \tag{8}$$

Here $\lceil \cdot \rceil$ is the ceil rounding function, and ρ is a preset rounding error for small values of $A_\phi^*(x)$.

Intuitively, Theorem 2 provides a way to approximate how many test samples are expected to satisfy ξ-error conditions. Under the similar data distribution, the classification problems with a larger number of classes are usually (not necessarily) more difficult to learn, i.e., achieve lower accuracy.

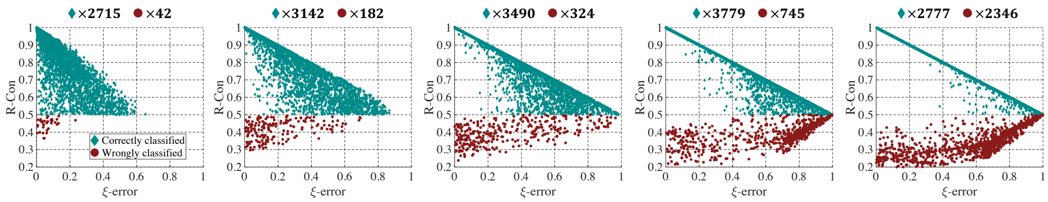


Figure 3: The PGD-10 examples crafted on $10,000$ test samples on CIFAR-10, and filtered by $\frac{1}{2-\xi}$ confidence threshold for each ξ. Here $\log_2 \tau = 0$ (i.e., $\tau = 1$) is the case shown in Fig. 2. Simply lower down the temperature τ can involve more samples into the area of certified separability.

For example, the same model that achieves 90% test accuracy on CIFAR-10 may only achieve 70% test accuracy on CIFAR-100. According to Theorem 2, if we want to obtain a 0.1-error A_ϕ on the CIFAR datasets, then this task can be regarded as a 20-classes classification problem, whose learning difficulty is expected to be between 10-classes one (e.g., CIFAR-10 task) and 100-classes one (e.g., CIFAR-100 task). Thus, the test accuracy of the 20-classes task is expected to be between 90% and 70%, which means about 70%∼90% test samples will satisfy ξ-error conditions with $\xi = 0.1$.

Similarly, Theorem 2 can also approximate the difficulty of learning a *robust* ξ-error A_ϕ, e.g., for any point x' in the ℓ_∞ ball around x, we have x' satisfy ξ-error conditions. This task can be converted into training a *certified* classifier (Wong and Kolter, 2018), and the ratio of test samples that achieve robust ξ-error A_ϕ can be approximated by the performance of existing certified defenses.

5 FURTHER DISCUSSION

Rectified rejection vs. binary rejection. In the limiting case of $\tau \to 0$, the returned probability vector will tend to one-hot, i.e., $f_\theta(x)[y^m]$ always equals to one, and the optimal solution A^*_ϕ becomes binary as $A^*_\phi(x) = 1$ if x is correctly classified; otherwise $A^*_\phi(x) = 0$. In this case, learning A_ϕ degenerates to a binary classification task, which has been widely studied and applied in previous work (Geifman and El-Yaniv, 2017; 2019; Gong et al., 2017; Kato et al., 2020). However, directly learning a binary rejector abandons the returned confidence that can be informative about the prediction certainty (Geifman and El-Yaniv, 2017; Wu et al., 2018). Besides, since a trained binary rejector $\mathcal{M}$ usually outputs continuous values in $[0, 1]$, e.g., after a sigmoid activation, its returned values will be overwhelmed by the optimization procedure under binary supervision. For example, two wrongly classified inputs x_1, x_2 may have $\mathcal{M}(x_1) < \mathcal{M}(x_2)$ only because $\mathcal{M}$ is easier to optimize on x_1 during training. This trend deviates $\mathcal{M}$ from properly reflecting the prediction certainty of $f_\theta(x)$, and induces suboptimal reject decisions during inference. In contrast, our RR module learns T-Con by rectifying confidence, where T-Con provides more distinctive supervised signals, and the rectified formula takes advantage of model sharing. It is easy to show that R-Con with a ξ-error A_ϕ is approximately order-preserving with respect to the T-Con values. This enables R-Con to stick to the certainty measure induced by T-Con, and make reasonable reject decisions.

Rectified confidence vs. calibrated confidence. Another concept related with T-Con and R-Con is confidence calibration (Guo et al., 2017). Typically, a classifier f_θ with calibrated confidence satisfies that $\forall c \in [0, 1]$, there is $p\big(y^m = y \big| f_\theta(x)[y^m] = c\big) = c$, where the probability is taken over the data distribution. For notation compactness, we let $q_\theta(c) \triangleq p\left(f_\theta(x)[y^m] = c\right)$ be the probability that the returned confidence equals to c. Then if we execute rejection option based on the calibrated confidence, the accuracy on returned predictions can be calculated by $\int_t^1 c \cdot q_\theta(c)\mathrm{d}c \big/ \int_t^1 q_\theta(c)\mathrm{d}c$, where t is the preset threshold. On the positive side, calibrated confidence certifies that the accuracy after rejection is no worse than t. However, since there is no explicit supervision on the distribution $q_\theta(c)$, the final accuracy still relies on the difficulty of learning task. In contrast, rejecting via T-Con with a 0.5 threshold will always lead to 100% accuracy, whatever the learning difficulty, which makes T-Con a more ideal supervisor when we aim to learn a generally well-behaved rejection module.

6 EXPERIMENTS

Our experiments are done on the datasets CIFAR-10, CIFAR-100, and CIFAR-10-C (Hendrycks and Dietterich, 2019). We choose two commonly used model architectures: ResNet-18 (He et al., 2016) and WRN-34-10 (Zagoruyko and Komodakis, 2016). Following the suggestions in Pang et al. (2021),

Table 2: TPR-95 accuracy (%) and ROC-AUC scores of the ResNet-18 models trained on CIFAR-10, evaluated by PGD-10 attacks. Here GDA* indicates using class-conditional covariance matrices.

AT	Rejector	Clean		ℓ_∞, 8/255		ℓ_∞, 16/255		ℓ_2, 128/255	
		TPR-95	AUC	TPR-95	AUC	TPR-95	AUC	TPR-95	AUC
PGD-AT	KD	82.59	0.618	53.44	0.587	32.23	0.537	64.91	0.599
	LID	84.02	0.712	55.12	0.661	33.09	0.622	66.32	0.666
	GDA	82.35	0.453	52.96	0.461	31.94	0.452	64.44	0.458
	GDA*	84.51	0.664	54.16	0.589	32.20	0.525	65.99	0.606
	GMM	85.44	0.703	54.55	0.606	32.22	0.530	66.74	0.634
CARL	Margin	85.54	0.682	51.93	0.539	30.69	0.517	66.20	0.647
ATRO	Margin	73.42	0.669	36.48	0.655	21.50	0.644	41.77	0.657
CCAT	Con.	92.44	0.806	51.88	0.637	45.30	0.683	67.34	0.770
TRADES	Con.	86.07	0.837	57.88	0.773	37.80	0.737	68.08	0.781
PGD-AT	SNet	84.19	0.796	56.63	0.729	35.65	0.692	67.83	0.740
PGD-AT	EBD	85.34	0.832	57.38	0.763	35.18	0.689	68.05	0.774
TRADES	**RR**	86.47	0.849	**58.71**	**0.786**	38.13	**0.746**	69.19	0.793
CCAT	**RR**	**94.12**	**0.909**	54.14	0.662	**48.14**	0.690	68.20	0.785
PGD-AT	**RR**	86.91	0.861	58.39	0.776	35.57	0.704	**70.36**	**0.794**

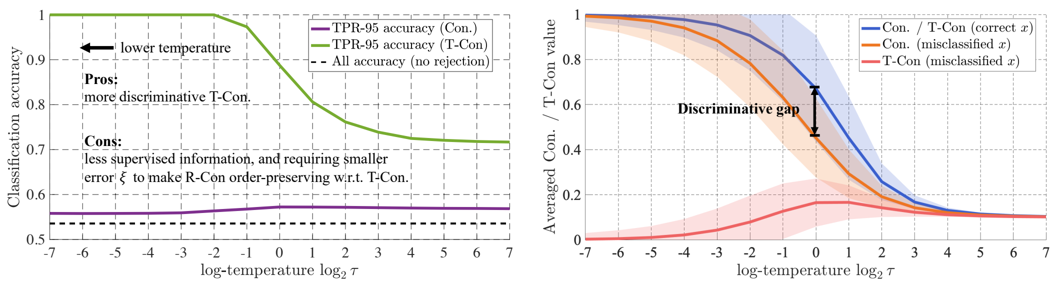


Figure 4: We quantify the effects of temperature τ. The model is adversarially trained on CIFAR-10 (no RR module used) and evaded by PGD-10. *Left*: TPR-95 accuracy with respect to confidence and T-Con. *Right*: Averaged confidence / T-Con value on correct / misclassified PGD-10 inputs.

for all the defenses, the default training settings include batch size 128; SGD momentum optimizer with the initial learning rate of 0.1; weight decay 5×10^{-4}. The training runs for 110 epochs with the learning rate decaying by a factor of 0.1 at 100 and 105 epochs. We report the results on the checkpoint with the best 10-steps PGD attack (PGD-10) accuracy (Rice et al., 2020).

AT frameworks used in our methods. We mainly apply three popular AT frameworks to combine with our RR module, involving PGD-AT (Madry et al., 2018), TRADES (Zhang et al., 2019b), and CCAT (Stutz et al., 2020). For PGD-AT and TRADES, we use PGD-10 during training, under ℓ_∞-constraint of $8/255$ with step size $2/255$. The trade-off parameter for TRADES is 6 (Zhang et al., 2019b), and the implementation of CCAT follows its official code. In the reported results, '**RR**' refers to the model adversarially trained by Eq. (5) with different AT frameworks, and using R-Con as the rejection metric; We set $\lambda = 1$ in Eq. (5) without tuning.

Baselines. We choose two kinds of commonly compared baselines (Bulusu et al., 2020). The first kind constructs statistics upon the learned features after training the classifier, including kernel density (KD) (Feinman et al., 2017), local intrinsic dimensionality (LID) (Ma et al., 2018), Gaussian discriminant analysis (GDA) (Lee et al., 2018), and Gaussian mixture model (GMM) (Zheng and Hong, 2018). The second kind jointly learns the rejector with the classifier, which involves SelectiveNet (SNet) (Geifman and El-Yaniv, 2019), energy-based detection (EBD) (Liu et al., 2020b), CARL (Laidlaw and Feizi, 2019), ATRO (Kato et al., 2020), and CCAT (Stutz et al., 2020). We emphasize that most of these baselines are originally applied to STMs, while *we adopt them to ATMs as stronger baselines by re-tuning their hyperparameters*, as detailed in Appendix D.2.

Adversarial attacks. We evaluate under PGD (Madry et al., 2018), C&W (Carlini and Wagner, 2017a), AutoAttack (Croce and Hein, 2020), multi-target attack (Gowal et al., 2019), GAMA attack (Sriramanan et al., 2020), and general corruptions in CIFAR-10-C (Hendrycks and Dietterich, 2019). More details on the attacking hyperparameters can be found in Appendix D.3.

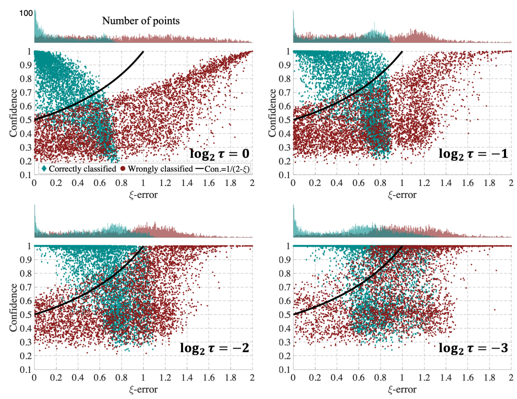
Table 3: TPR-95 accuracy (%) under common corruptions in **CIFAR-10-C**. The model architecture is ResNet-18, and the reported accuracy under each corruption is averaged across five severity.

AT	Rej.	CIFAR-10-C									
		Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contra	Elastic	JPEG
PGD-AT	SNet	77.74	75.52	78.72	79.77	75.81	61.32	81.75	42.97	78.59	82.08
PGD-AT	EBD	78.47	77.92	80.47	81.17	79.14	61.16	83.98	42.10	80.86	83.34
CARL	Margin	77.45	74.94	78.00	79.86	74.16	56.09	81.28	40.33	78.17	82.64
ATRO	Margin	55.36	53.74	54.59	50.84	41.12	42.82	50.13	33.54	54.48	56.82
CCAT	Con.	83.04	85.47	89.33	**89.38**	88.21	76.32	**92.71**	55.99	89.34	91.94
TRADES	Con.	79.89	78.48	80.92	78.75	71.61	63.53	80.97	45.22	80.53	84.50
PGD-AT	**RR**	80.87	79.42	81.90	81.89	76.95	63.49	84.02	44.03	82.18	85.12
CCAT	**RR**	**85.03**	**86.26**	**89.83**	89.22	**88.41**	**77.45**	92.62	**58.95**	**89.59**	**92.06**
TRADES	**RR**	80.03	79.15	81.00	80.16	74.18	63.55	82.13	45.99	80.98	84.64

Table 4: TPR-95 accuracy (%) on CIFAR-10, under multi-target attack and GAMA attacks. The model architecture is ResNet-18, and the threat model is $(\ell_\infty, 8/255)$.

AT	Rej.	Multi-target	GAMA (PGD)	GAMA (FW)
PGD-AT	SNet	55.02	55.79	51.37
PGD-AT	EBD	55.40	56.15	53.24
CARL	Margin	46.17	48.49	44.78
ATRO	Margin	32.53	31.74	28.31
CCAT	Con.	34.21	49.78	38.01
TRADES	Con.	53.69	56.89	50.88
PGD-AT	**RR**	**56.18**	57.57	**54.08**
CCAT	**RR**	36.48	51.30	40.72
TRADES	**RR**	54.83	**57.93**	51.48

Figure 5: Confidence values w.r.t. ξ-error values of ResNet-18 trained by PGD-AT+**RR** on CIFAR-10. Here ξ is calculated as the minimum value satisfying Definition 1. The settings are the same as in Fig. 3.



6.1 Performance against normal attacks

We report the results on defending normal attacks, i.e., those only target at fooling the classifiers. The results on CIFAR-10 are shown in Table 2 (results on CIFAR-100 are in Appendix D.4). The 'All' accuracy indicates the case with no rejection. As for the 'TPR-95' accuracy, we fix the thresholds to 95% true positive rate, which means at most 5% of correctly classified examples can be rejected. We evaluate under PGD-10 $(\ell_\infty, \epsilon = 8/255)$ which is seen during training, and unseen attacks with different perturbation constraint $(\epsilon = 16/255)$, threat model (ℓ_2), or steps (PGD-1000 in Table 8). We apply untargeted mode with 5 restarts. We can observe that our RR module can well incorporate with different AT frameworks, which outperform previous baselines and the vanilla versions of AT + confidence, with little extra computation and memory usage. Besides, the improvement on CIFAR-100 is more significant than it on CIFAR-10, which verifies our formulation on learning difficulty in Section 4.3. The poor performance of statistics-based baselines is affected by the irregular feature distributions in ATMs, as shown in Appendix D.5. We also investigate the performance of our methods against the out-of-distribution corruptions on CIFAR-10-C, as summarized in Table 3. In Table 4, we evaluate under multi-target attack and GAMA attacks. As to AutoAttack, we note that its algorithm returns crafted adversarial examples for successful evasions, while returns original clean examples otherwise. By using **RR** to train a ResNet-18, the All (TPR-95) accuracy (%) under AutoAttack is 48.62 (84.32) and 25.20 (70.99) on CIFAR-10 and CIFAR-100, respectively.

6.2 Ablation studies

Empirical effects of temperature τ. In addition to the certified separability described in Section 4.2, we show the curves of TPR-95 accuracy and averaged confidence / T-Con values in Fig. 4 w.r.t. the temperature scaling, while in Fig. 5 we visualize the sample distributions of ξ-error vs. confidence values. We can observe that the T-Con values become more discriminative for a lower temperature on rejecting misclassified examples, but numerically provide less supervised information and require smaller error ξ to make R-Con order-preserving w.r.t. T-Con. On the other hand, as the temperature τ gets larger above one, the discriminative power of confidence becomes weaker, making R-Con

Table 5: Ablation studies on the effect of temperature τ for **RR**. Note that in the objective Eq. (5), τ is only tuned in the term of $\mathcal{L}_{\mathrm{RR}}$, while the temperature for $\mathcal{L}_{\mathrm{T}}$ is kept to be 1.

$\log_2 \tau$	Clean inputs		PGD-10 inputs	
	TPR-95	AUC	TPR-95	AUC
-1	**86.86**	0.866	59.11	**0.770**
-2	86.62	0.865	60.63	0.762
-3	85.18	**0.868**	**61.12**	0.741
-4	80.22	0.836	55.15	0.740

Table 6: Ablation studies on rectified construction of R-Con in Eq. (3). Here '$f_\theta(x)[y^m]$' and '$A_\phi(x)$' indicate using confidence and auxiliary function to substitute R-Con in $\mathcal{L}_{\mathrm{RR}}$, respectively.

Rejector	Clean inputs		PGD-10 inputs	
	TPR-95	AUC	TPR-95	AUC
$A_\phi(x)$	85.77	0.844	56.97	0.765
RR	**86.91**	**0.861**	**58.39**	**0.776**
$f_\theta(x)[y^m]$	86.76	0.865	57.42	0.768
RR (Con.)	**87.12**	**0.868**	**58.49**	**0.777**

Table 7: Minimal perturbations required by successful evasions, searched by CW attacks. Here 'Normal (Nor.)' refers to fooling the classifier, and 'Adaptive (Ada.)' refers to *adaptively* fooling both the classifier and rejector.

Rej.	**CIFAR-10**				**CIFAR-100**			
	CW-ℓ_∞		CW-ℓ_2		CW-ℓ_∞		CW-ℓ_2	
	Nor.	Ada.	Nor.	Ada.	Nor.	Ada.	Nor.	Ada.
SNet	14.30	30.48	0.84	2.70	8.20	23.05	0.56	2.37
EBD	14.70	37.54	0.85	2.42	8.58	25.69	0.60	1.81
RR	14.99	**38.58**	0.87	**3.28**	8.53	**28.67**	0.61	**3.21**

Table 8: Classification accuracy (%) and ROC-AUC scores under PGD-1000 attacks, where the step size is $2/255$ and the perturbation constraint is $8/255$ under ℓ_∞ threat model.

Rej.	**CIFAR-10**		**CIFAR-100**	
	TPR-95	AUC	TPR-95	AUC
SNet	55.83	0.725	32.69	0.744
EBD	56.12	0.763	33.35	0.769
RR	**57.57**	**0.773**	**34.48**	**0.776**

harder to distinguish misclassified inputs from correctly classified ones. In practice, we can trade-off between the learning difficulty and the effectiveness of R-Con by tuning τ. Namely, in Table 5 we study the effects of tuning temperature values for $f_\theta(x)[y]$ and $f_\theta(x)[y^m]$ in $\mathcal{L}_{\mathrm{RR}}$. We find that moderately lower down the temperature can benefit model robustness but sacrifice clean accuracy, while overly low temperature degenerates both clean and robust performance.

Formula of R-Con. In Table 6, we investigate the cases if there is no rectified connection (i.e., only use $A_\phi(x)$) or no auxiliary flexibility (i.e., only use $f_\theta(x)[y^m]$) in the constructed rejection module. As shown, our rectifying paradigm indeed promote the effectiveness.

6.3 Performance against adaptive attacks

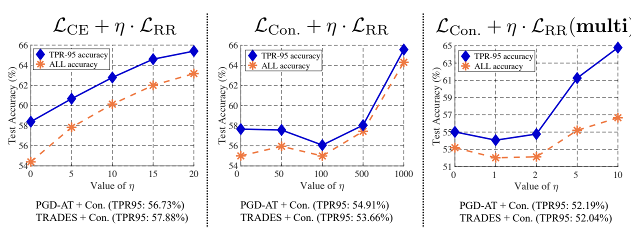


Figure 6: Accuracy (%) under adaptive PGD-500 (10 restarts) on CIFAR-10. The ResNet-18 is trained by PGD-AT+**RR**.

We design two kinds of adaptive attacks to evade the classifier model and rejection module simultaneously. The first one follows Carlini and Wagner (2017b), where we incorporate the loss term of the RR module into the original CW objective, and find the minimal distortion for a per-example successful evasion if the classifier is fooled and the rejector value is higher than the median value of the training set. The binary search steps are 9 with 1,000 iteration steps for each search. As in Table 7, adaptive attacks require larger minimal perturbations than normal attacks, and successfully evading our methods is harder than baselines. In the second adaptive attack, we fix the maximal perturbation size to $8/255$ under ℓ_∞-norm, and use adaptive objectives $\mathcal{L}_{\mathrm{CE}} + \eta \cdot \mathcal{L}_{\mathrm{RR}}$, $\mathcal{L}_{\mathrm{Con.}} + \eta \cdot \mathcal{L}_{\mathrm{RR}}$, and $\mathcal{L}_{\mathrm{Con.}} + \eta \cdot \mathcal{L}_{\mathrm{RR}}(\mathrm{multi})$, where $\mathcal{L}_{\mathrm{Con.}}$ is to directly optimize the confidence and *multi* refers to multi-target version. The results are in Fig. 6, where we also report the TPR-95 accuracy of baselines for reference. As seen, even under adaptive attacks, applying our RR module still outperforms the baselines.

7 Conclusion

We introduce T-Con as a certainty oracle, and use R-Con to mimic T-Con by training. Intriguingly, a ξ-error R-Con rejector and a $\frac{1}{2-\xi}$ confidence rejector can be coupled to provide certified separability, which demonstrates a promising prospect towards reliable predictions via coupling rejectors. We also empirically validate the effectiveness of our RR module by using R-Con alone as the rejector, which alleviates the overestimation of certainty, and is well compatible with different AT frameworks.

Ethics Statement

When deploying machine learning methods into practical systems, the adversarial vulnerability can cause a potential security risk, as well as the negative impact on the crisis of confidence by the public. To this end, this inherent defect raises the requirements for reliable, general, and lightweight strategies to enhance the model robustness against malicious, especially adversarial attacks. In this work, we provide an efficient strategy to couple two connected rejection metrics, which can certifiably distinguish correctly and wrongly classified inputs, prevents the model from outputting over-confident wrong predictions. Our methods contribute to the modules of constructing more reliable systems.

Reproducibility Statement

We include Appendix along with the main text, just after the references section. We provide code for implementing our experiments in the supplemental material as a .zip file.

References

Nilesh A Ahuja, Ibrahima Ndiour, Trushant Kalyanpur, and Omesh Tickoo. Probabilistic modeling of deep features for out-of-distribution and adversarial detection. *arXiv preprint arXiv:1909.11786*, 2019.

Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 12192–12202, 2019.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In *Advances in neural information processing systems (NeurIPS)*, 2020.

Rushil Anirudh, Jayaraman J Thiagarajan, Bhavya Kailkhura, and Peer-Timo Bremer. Mimicgan: Robust projection onto image manifolds with corruption mimicking. *International Journal of Computer Vision (IJCV)*, pages 1–19, 2020.

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 (ICML)*, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K Varshney, and Dawn Song. Anomalous instance detection in deep learning: A survey. *arXiv preprint arXiv:2003.06979*, 2020.

Qi-Zhi Cai, Chang Liu, and Dawn Song. Curriculum adversarial training. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3740–3747, 2018.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2017a.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security (AISec)*, 2017b.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C Duchi. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Fabio Carrara, Rudy Becarelli, Roberto Caldelli, Fabrizio Falchi, and Giuseppe Amato. Adversarial examples detection in features distance spaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Jinghui Chen and Quanquan Gu. Rays: A ray searching method for hard-label adversarial attack. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, 2020.

Kejiang Chen, Yuefeng Chen, Hang Zhou, Xiaofeng Mao, Yuhong Li, Yuan He, Hui Xue, Weiming Zhang, and Nenghai Yu. Self-supervised adversarial training. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2218–2222. IEEE, 2020a.

Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 699–708, 2020b.

Gilad Cohen, Guillermo Sapiro, and Raja Giryes. Detecting adversarial samples using influence functions and nearest neighbors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

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

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pages 67–82. Springer, 2016.

Francesco Crecchi, Marco Melis, Angelo Sotgiu, Davide Bacciu, and Battista Biggio. Fader: Fast adversarial example rejection. *arXiv preprint arXiv:2010.09119*, 2020.

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

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Abhimanyu Dubey, Laurens van der Maaten, Zeki Yalniz, Yixuan Li, and Dhruv Mahajan. Defense against adversarial images using web-scale nearest-neighbor search. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8767–8776, 2019.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in neural information processing systems (NeurIPS)*, 2017.

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International Conference on Machine Learning (ICML)*, 2019.

Lovedeep Gondara. Detecting adversarial samples using density ratio estimates. *arXiv preprint arXiv:1705.02224*, 2017.

Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

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

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.

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.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in neural information processing systems (NeurIPS)*, 2020.

Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning (ICML)*, 2019.

Haoming Jiang, Zhehui Chen, Yuyang Shi, Bo Dai, and Tuo Zhao. Learning to defense by learning to attack. *arXiv preprint arXiv:1811.01213*, 2018.

Masahiro Kato, Zhenghang Cui, and Yoshihiro Fukuhara. Atro: Adversarial training with a rejection option. *arXiv preprint arXiv:2010.12905*, 2020.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Cassidy Laidlaw and Soheil Feizi. Playing it safe: Adversarial robustness with an abstain option. *arXiv preprint arXiv:1911.11253*, 2019.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Bai Li, Shiqi Wang, Suman Jana, and Lawrence Carin. Towards understanding fast adversarial training. *arXiv preprint arXiv:2006.03089*, 2020.

Pengcheng Li, Jinfeng Yi, Bowen Zhou, and Lijun Zhang. Improving the robustness of deep neural networks via adversarial training with triplet loss. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Guanxiong Liu, Issa Khalil, and Abdallah Khreishah. Using single-step adversarial training to defend iterative adversarial examples. *arXiv preprint arXiv:2002.09632*, 2020a.

Weitang Liu, Xiaoyun Wang, John Owens, and Sharon Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020b.

Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. In *International Conference on Learning Representations (ICLR)*, 2019.

Jiajun Lu, Theerasit Issaranon, and David Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *International Conference on Computer Vision (ICCV)*, pages 446–454, 2017.

Chengcheng Ma, Baoyuan Wu, Shibiao Xu, Yanbo Fan, Yong Zhang, Xiaopeng Zhang, and Zhifeng Li. Effective and robust detection of adversarial examples via benford-fourier coefficients. *arXiv preprint arXiv:2005.05552*, 2020.

Shiqing Ma and Yingqi Liu. Nic: Detecting adversarial samples with neural network invariant checking. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*, 2019.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations (ICLR)*, 2018.

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 (ICLR)*, 2018.

Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning (ICML)*, pages 6640–6650. PMLR, 2020.

Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 478–489, 2019.

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

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A self-supervised approach for adversarial robustness. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 262–271, 2020.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.

Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4579–4589, 2018.

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

Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Hang Su, and Jun Zhu. Boosting adversarial training with hypersphere embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *International Conference on Learning Representations (ICLR)*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

Julien Perolat, Mateusz Malinowski, Bilal Piot, and Olivier Pietquin. Playing the game of universal adversarial perturbations. *arXiv preprint arXiv:1809.07802*, 2018.

Ambrish Rawat, Martin Wistuba, and Maria-Irina Nicolae. Adversarial phenomenon in the eyes of bayesian deep learning. *arXiv preprint arXiv:1711.08244*, 2017.

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.

Leslie Rice, Eric Wong, and J Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning (ICML)*, 2020.

Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. In *International Conference on Machine Learning (ICML)*, 2019.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations (ICLR)*, 2018.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5019–5031, 2018.

Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Improving adversarial robustness using proxy distributions. *arXiv preprint arXiv:2104.09425*, 2021.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Ali Shafahi, Mahyar Najibi, Zheng Xu, John P Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 5636–5643, 2020.

Fatemeh Sheikholeslami, Swayambhoo Jain, and Georgios B Giannakis. Minimum uncertainty based detection of adversaries in deep neural networks. *arXiv preprint arXiv:1904.02841*, 2019.

Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

Angelo Sotgiu, Ambra Demontis, Marco Melis, Battista Biggio, Giorgio Fumera, Xiaoyi Feng, and Fabio Roli. Deep neural rejection against adversarial examples. *EURASIP Journal on Information Security*, 2020:1–10, 2020.

Philip Sperl, Ching-Yu Kao, Peng Chen, and Konstantin Böttinger. Dla: Dense-layer-analysis for adversarial example detection. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, et al. Guided adversarial attack for evaluating and enhancing adversarial defenses. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *International Conference on Machine Learning (ICML)*, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Guanhong Tao, Shiqing Ma, Yingqi Liu, and Xiangyu Zhang. Attacks meet interpretability: Attribute-steered detection of adversarial samples. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Florian Tramer. Detecting adversarial examples is (nearly) as hard as classifying them. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021. URL https://openreview.net/forum?id=6pgY2PkoXb0.

Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5858–5868, 2019.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.

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

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research (JMLD)*, 9(11), 2008.

Huaxia Wang and Chun-Nam Yu. A direct approach to robust deep learning using adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *International Conference on Machine Learning (ICML)*, pages 6586–6595, 2019a.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations (ICLR)*, 2019b.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, pages 5283–5292, 2018.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations (ICLR)*, 2020.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

Xi Wu, Uyeong Jang, Jiefeng Chen, Lingjiao Chen, and Somesh Jha. Reinforcing adversarial robustness using model confidence induced by adversarial training. In *International Conference on Machine Learning (ICML)*, pages 5334–5342. PMLR, 2018.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

Huanrui Yang, Jingyang Zhang, Hongliang Dong, Nathan Inkawhich, Andrew Gardner, Andrew Touchet, Wesley Wilkes, Heath Berry, and Hai Li. Dverge: Diversifying vulnerabilities for enhanced robust generation of ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020a.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. Ml-loo: Detecting adversarial examples with feature attribution. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, 2020b.

Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020c.

Pourya Habib Zadeh, Reshad Hosseini, and Suvrit Sra. Deep-rbf networks revisited: Robust classification with rejection. *arXiv preprint arXiv:1812.03190*, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *The British Machine Vision Conference (BMVC)*, 2016.

Chiliang Zhang, Zuochang Ye, Yan Wang, and Zhimou Yang. Detecting adversarial perturbations with saliency. In *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, pages 271–275. IEEE, 2018.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019a.

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 (ICML)*, 2019b.

Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning (ICML)*, 2020.

Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations (ICLR)*, 2021.

Chenxiao Zhao, P Thomas Fletcher, Mixue Yu, Yaxin Peng, Guixu Zhang, and Chaomin Shen. The adversarial attack and detection under the fisher information metric. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5869–5876, 2019.

Zhihao Zheng and Pengyu Hong. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

A PROOF

In this section, we provide proofs for the proposed Theorem 1, and Theorem 2.

A.1 PROOF OF THEOREM 1

Proof. The conditions in Theorem 1 can be written as $f_\theta(x_1)[y_1^m] > \frac{1}{2-\xi}$, $y_1^m = y_1$ and $f_\theta(x_2)[y_2^m] > \frac{1}{2-\xi}$, $y_2^m \neq y_2$, where $\xi \in [0,1)$. Since $A_\phi(x)$ is ξ-error at x_1 and x_2, according to Definition 1, at least one of the bounds holds for x_1 and x_2, respectively:

$$\text{Bound (i): } \left|\log\left(\frac{A_\phi(x)}{A^*_\phi(x)}\right)\right| \leq \log\left(\frac{2}{2-\xi}\right);$$

$$\text{Bound (ii): } \left|A_\phi(x) - A^*_\phi(x)\right| \leq \frac{\xi}{2}.$$

For x_1, there is $A^*_\phi(x_1) = 1$. Then if bound (i) holds, we can obtain

$$\begin{aligned}\text{R-Con}(x_1) &= f_\theta(x_1)[y_1^m] \cdot A_\phi(x_1)\\ &> f_\theta(x_1)[y_1^m] \cdot \frac{2-\xi}{2}\\ &> \frac{1}{2-\xi} \cdot \frac{2-\xi}{2} = \frac{1}{2},\end{aligned}$$

and if bound (ii) holds, we can obtain

$$\begin{aligned}\text{R-Con}(x_1) &= f_\theta(x_1)[y_1^m] \cdot A_\phi(x_1)\\ &> f_\theta(x_1)[y_1^m] \cdot \left(1 - \frac{\xi}{2}\right)\\ &> \frac{1}{2-\xi} \cdot \frac{2-\xi}{2} = \frac{1}{2}.\end{aligned}$$

Similarly for x_2, there is $f_\theta(x_2)[y_2^m] \cdot A^*_\phi(x_2) = f_\theta(x_2)[y_2]$. Then if bound (i) holds, we can obtain

$$\begin{aligned}\text{R-Con}(x_2) &= f_\theta(x_2)[y_2^m] \cdot A_\phi(x_2)\\ &= f_\theta(x_2)[y_2^m] \cdot A^*_\phi(x_2) \cdot \frac{A_\phi(x_2)}{A^*_\phi(x_2)}\\ &< f_\theta(x_2)[y_2] \cdot \frac{2}{2-\xi}\\ &< \left(1 - \frac{1}{2-\xi}\right) \cdot \frac{2}{2-\xi}\\ &= \frac{2-2\xi}{(2-\xi)^2} < \frac{1}{2},\end{aligned}$$

where it is easy to verify that $\frac{2-2\xi}{(2-\xi)^2}$ is monotone decreasing in the interval of $\xi \in [0,1)$. If bound (ii) holds for x_2, we can obtain

$$\begin{aligned}&\text{R-Con}(x_2)\\ &= f_\theta(x_2)[y_2^m] \cdot A_\phi(x_2)\\ &< f_\theta(x_2)[y_2^m] \cdot \left(\frac{f_\theta(x_2)[y_2]}{f_\theta(x_2)[y_2^m]} + \frac{\xi}{2}\right)\\ &= f_\theta(x_2)[y_2] + f_\theta(x_2)[y_2^m] \cdot \frac{\xi}{2}\\ &= f_\theta(x_2)[y_2] \cdot \left(1 - \frac{\xi}{2}\right) + \left(f_\theta(x_2)[y_2] + f_\theta(x_2)[y_2^m]\right) \cdot \frac{\xi}{2}\\ &< \left(1 - \frac{1}{2-\xi}\right) \cdot \left(1 - \frac{\xi}{2}\right) + \frac{\xi}{2} = \frac{1}{2}.\end{aligned}$$

Thus we have proven $\text{R-Con}(x_1) > \frac{1}{2} > \text{R-Con}(x_2)$. □

A.2 PROOF OF THEOREM 2

Proof. Since $A^*_\phi(x)$ is naturally bounded in $[0,1]$ for any input x, and $A_\phi(x)$ is bounded in $[0,1]$ by model design, we denote $\{B_0, B_1, \cdots, B_S\}$ as $S+1$ points in $[0,1]$, where $B_0 = 0$ and $B_s = 1$. These $S+1$ points induce S bins or intervals, i.e., $I_s = [B_{s-1}, B_s]$ for $s = 1, \cdots, S$. When $A_\phi(x)$ is ξ-error at x, we consider the cases of bound (i) and bound (ii) hold, respectively, as detailed below:

Bound (i) holds. We construct the bins in a geometric manner, where $B_s = \frac{2}{2-\xi} \cdot B_{s-1}$ and we set $B_1 = \rho$ be a rounding error. Note that we have

$$\rho \cdot \left(\frac{2}{2-\xi}\right)^{S-2} < 1 \leq \rho \cdot \left(\frac{2}{2-\xi}\right)^{S-1},$$

thus we can derive that

$$S = \left\lceil \frac{\log \rho^{-1}}{\log\left(\frac{2}{2-\xi}\right)} \right\rceil + 1.$$

It is easy to find that if $A_\phi(x)$ and $A^*_\phi(x)$ locate in the same bin, then bound (i) holds. Therefore, this regression task can be substituted by a classification task of classes $N_1 = \left\lceil \frac{\log \rho^{-1}}{\log\left(\frac{2}{2-\xi}\right)} \right\rceil + 1$.

Bound (ii) holds. In this case, we construct the bins in an arithmetic manner, where $B_s = B_{s-1} + \frac{\xi}{2}$. Then we have

$$(S-1) \cdot \frac{\xi}{2} < 1 \leq S \cdot \frac{\xi}{2},$$

thus we can derive that

$$S = \left\lceil \frac{2}{\xi} \right\rceil.$$

It is easy to find that if $A_\phi(x)$ and $A^*_\phi(x)$ locate in the same bin, then bound (ii) holds. So this regression task can be substituted by a classification task of classes $N_2 = \left\lceil \frac{2}{\xi} \right\rceil$. ☐

B MORE BACKGROUNDS

Adversarial training. In recent years, adversarial training (AT) has become the critical ingredient for the state-of-the-art robust models (Chen and Gu, 2020; Croce et al., 2020; Dong et al., 2020). Many variants of AT have been proposed via adopting the techniques like ensemble learning (Pang et al., 2019; Tramèr et al., 2018; Yang et al., 2020a), metric learning (Li et al., 2019; Mao et al., 2019), generative modeling (Jiang et al., 2018; Wang and Yu, 2019), curriculum learning (Cai et al., 2018), semi-supervised learning (Alayrac et al., 2019; Carmon et al., 2019), and self-supervised learning (Chen et al., 2020a;b; Hendrycks et al., 2019; Naseer et al., 2020). Other efforts include tuning AT mechanisms by universal perturbations (Perolat et al., 2018; Shafahi et al., 2020), reweighting misclassified samples (Wang et al., 2019b; Zhang et al., 2021) or multiple threat models (Maini et al., 2020; Tramèr and Boneh, 2019). Accelerating the training procedure of AT is another popular research routine, where recent progresses involve reusing the computations (Shafahi et al., 2019; Zhang et al., 2019a), adaptive adversarial steps (Wang et al., 2019a; Zhang et al., 2020) or one-step training (Andriushchenko and Flammarion, 2020; Li et al., 2020; Liu et al., 2020a; Wong et al., 2020).

Adversarial detection. Instead of correctly classifying adversarial inputs, another complementary research routine aims to detect / reject them (Crecchi et al., 2020; Grosse et al., 2017; Liu et al., 2019; Lu et al., 2017; Metzen et al., 2017; Roth et al., 2019; Zhang et al., 2018). Previous detection methods mainly fall into two camps, i.e., statistic-based and model-based. Statistic-based methods stem from the features learned by standardly trained models. These statistics include density ratio (Gondara, 2017), kernel density (Feinman et al., 2017; Pang et al., 2018), prediction variation Xu et al. (2017), mutual information (Sheikholeslami et al., 2019; Smith and Gal, 2018), Fisher information (Zhao et al., 2019), local intrinsic dimension (Ma et al., 2018), activation invariance (Ma and Liu, 2019), and feature attributions (Tao et al., 2018; Yang et al., 2020b). As for the model-based methods, the auxiliary detector could be a sub-network (Carrara et al., 2018; Cohen et al., 2020; Sperl et al., 2020), a Gaussian mixture model (Ahuja et al., 2019; Lee et al., 2018; Ma et al., 2020), or an additional generative model (Anirudh et al., 2020; Dubey et al., 2019; Samangouei et al., 2018).

C More analyses

In this section, we provide implementation details of the BCE loss, toy examples to intuitively illustrate the effects of temperature tuning, and analyze the role of T-Con in randomized classifiers.

C.1 Implementation of the BCE loss

For notation simplicity, we generally denote the BCE objective as

$$\text{BCE}(f \parallel g) = g_{\nmid} \cdot \log f + (1 - g_{\nmid}) \cdot \log (1 - f)\,, \tag{9}$$

where the subscript $\nmid$ indicates stopping gradients, an operation usually used to stabilize the training processes (Grill et al., 2020). We show that the stopping-gradient operations shown in Fig. 1 can lead to unbiased optimal solution for the classifier. Specifically, taking PGD-AT+RR as an example, the training objective is

$$\min_{\phi,\theta} \mathbb{E}_{p(x,y)} \left[\mathcal{L}_{\text{CE}} \left(f_\theta(x), y\right) + \text{BCE} \left(f_\theta(x)[y^m] \cdot A_\phi(x) || f_\theta(x)[y]\right)\right],$$

where we use $p(x, y)$ to represent adversarial data distribution. Note that the optimal solution of minimizing $\mathcal{L}_{\text{CE}} (f_\theta(x), y)$ is $f_\theta(x)[y] = p(y|x)$, but if we do not stop gradients of $f_\theta(x)[y]$ in the RR term (BCE loss), then the optimal θ of the entire PGD-AT+RR objective no longer satisfies $f_\theta(x)[y] = p(y|x)$, i.e., in this case RR will introduce bias on the optimal solution of classifier. Thus, stopping gradients on $f_\theta(x)[y]$ in the RR term can avoid affecting the training of classifier.

C.2 Toy examples on temperature tuning

Assume that there are three classes, and the confidence / T-Con on x_1 and x_2 are

$$\mathcal{M}(x_1; \tau) = \frac{e^{\frac{a_1}{\tau}}}{e^{\frac{a_1}{\tau}} + e^{\frac{b_1}{\tau}} + e^{\frac{c_1}{\tau}}}; \mathcal{M}(x_2; \tau) = \frac{e^{\frac{a_2}{\tau}}}{e^{\frac{a_2}{\tau}} + e^{\frac{b_2}{\tau}} + e^{\frac{c_2}{\tau}}}.$$

Let $a_1 = a_2 = 0$, $b_1 = 3$, $c_1 = -1000$, $b_2 = c_2 = 2$, it is easy to numerically compute that

$$\mathcal{M}(x_1; \tau = 1) < \mathcal{M}(x_2; \tau = 1);$$
$$\mathcal{M}(x_1; \tau = 2) > \mathcal{M}(x_2; \tau = 2).$$

This mimics the case of T-Con for misclassified inputs. We can simply choose $a_1 = a_2 = 0$, $b_1 = -1$, $c_1 = -1000$, $b_2 = c_2 = -2$ to mimic the case of confidence.

C.3 The role of T-Con in randomized classifiers

It has been shown that randomized classifiers like Bayesian neural networks (BNNs) (Liu et al., 2019; Rawat et al., 2017) and DNNs with randomized smoothing (Cohen et al., 2019) can benefit adversarial robustness. In practice, these methods are usually implemented by a Monte-Carlo ensemble with finite sampled weights or inputs. We construct an abstract classification process that involves both deterministic and randomized classifiers.

Specifically, the returned label y^s is sampled from a categorical distribution as $p(y^s = l) = f_\theta(x)[l]$, where in this case, $f_\theta(x)$ is a deterministic mapping either explicitly (e.g., for DNNs) or implicitly (e.g., for BNNs) defined. For example, considering a BNN $g_\omega(x)$ where $\omega \sim q_\theta(\omega)$, the induced $f_\theta(x)$ can be written as

$$f_\theta(x)[l] = p\left(l = \arg\max_{y_s} \sum_{n=1}^{N} g_{\omega_n}(y_s|x)\right), \tag{10}$$

which is the probability measure that the returned label is l from the Bayes ensemble $\sum_{n=1}^{N} g_{\omega_n}(y_s|x)$, under the distributions of $\omega_n \sim q_\theta(\omega)$, $n \in \{1, \cdots, N\}$. In practice, we can obtain empirical estimations on these implicitly defined $f_\theta(x)$ by sampling.

By presetting the temperature τ, the expected accuracy of the returned labels can be written as

$$\text{A}_\tau = \mathbb{E}_{p(x,y)} \mathbb{E}_{y^s} \left[\mathbf{1}_{y^s=y}\right] = \mathbb{E}_{p(x,y)} \left[f_\theta(x)[y]\right], \tag{11}$$

Table 9: Results of different hyperparameters for the KD and LID methods on CIFAR-10, under $(\ell_\infty, 8/255)$ threat model. For KD, we restore the features on $1,000$ correctly classified training samples in each class. For LID, we restore the features on totally $10,000$ correctly classified training samples.

Method	Hyperparameters	ROC-AUC	
		Clean	PGD-10
KD	$\sigma = 10^{-1}$	0.562	0.545
	$\sigma = 10^{-2}$	0.609	0.581
	$\sigma = 10^{-3}$	**0.618**	**0.587**
LID	$K = 100$	0.686	0.622
	$K = 200$	0.699	0.638
	$K = 300$	0.706	0.648
	$K = 400$	0.710	0.654
	$K = 500$	**0.712**	0.658
	$K = 600$	0.711	**0.661**
	$K = 700$	0.709	0.661
	$K = 800$	0.706	0.660
	$K = 1000$	0.695	0.653
	$K = 2000$	0.603	0.590

Table 10: Results of different hyperparameters for the KD and LID methods on CIFAR-100. The basic settings are the same as in Table 9, except that for KD, we restore 100 correctly classified training features in each class.

Method	Hyperparameters	ROC-AUC	
		Clean	PGD-10
KD	$\sigma = 10^{1}$	0.522	0.517
	$\sigma = 1$	**0.549**	**0.532**
	$\sigma = 10^{-1}$	0.500	0.479
	$\sigma = 10^{-2}$	0.473	0.453
	$\sigma = 10^{-3}$	0.477	0.457
LID	$K = 10$	0.662	0.652
	$K = 20$	**0.674**	**0.668**
	$K = 40$	0.672	0.667
	$K = 60$	0.668	0.661
	$K = 80$	0.659	0.652
	$K = 100$	0.652	0.644
	$K = 200$	0.615	0.607
	$K = 300$	0.584	0.578
	$K = 400$	0.559	0.551
	$K = 500$	0.537	0.529

where $\mathbf{1}_{y^s=y}$ is the indicator function, which equals to one if $y^s = y$ and zero otherwise. In the limiting case of $\tau \to 0$, the returned labels are deterministic, and the expected accuracy is $\mathrm{A}_0 = \mathbb{E}_{p(x,y)}[\mathbf{1}_{y^m=y}]$, which degenerates to the traditional definition of accuracy. Note that in the adversarial setting, the Bayes optimal classifier, i.e., $\tau = 0$ may not be an empirically optimal choice. For example, in the cases of $A_0 = 0$, we can still have $A_\tau > 0$ for the non-deterministic classifiers.

D More technical details and results

In this section, we provide more technical details and results. Our methods are implemented by Pytorch (Paszke et al., 2019), and run on GeForce RTX 2080 Ti GPU workers. The experiments of ResNet-18 are run by single GPU, while those on WRN-34-10 are run by two GPUs in parallel.

D.1 The MLP architecture of $A_\phi(x)$

In our experiments, $A_\phi(x)$ is implemented by the MLP as

$$A_\phi(x) = W_2(\mathbf{ReLU}(\mathbf{BN}(W_1 z + b_1))) + b_2, \tag{12}$$

where z is the feature vector shared with the classification branch, **BN** is an 1-D batch normalization operation, W_1, b_1 are the parameters of the first linear layer, and W_2, b_2 are the parameters of the second linear layer. For ResNet-18, there is $z \in \mathbb{R}^{512}$, $W_1 \in \mathbb{R}^{256\times512}$, $b_1 \in \mathbb{R}^{256}$, $W_2 \in \mathbb{R}^{1\times256}$, $b_2 \in \mathbb{R}^1$. For WRN-34-10, there is $z \in \mathbb{R}^{640}$, $W_1 \in \mathbb{R}^{320\times640}$, $b_1 \in \mathbb{R}^{320}$, $W_2 \in \mathbb{R}^{1\times320}$, $b_2 \in \mathbb{R}^1$.

Empirically, on ResNet-18, the average running time for PGD-AT is about 316 seconds per epoch, and it for PGD-AT+RR is about 320 seconds per epoch. As to the parameter sizes, saving a ResNet-18 model without/with RR branch uses 44.74 MB/45.27 MB, saving a WRN-34-10 model without/with RR branch uses 184.77 MB/185.59 MB.

D.2 Hyperparameters for baselines

For KD, we restore $1,000$ correctly classified training features in each class and use $\sigma = 10^{-3}$. For LID, we restore a total of $10,000$ correctly classified training features and use $K = 600$. We

Table 11: Results of different hyperparameters for the SelectiveNet and EBD methods on CIFAR-10. The AT framework is PGD-AT, and the evaluated PGD-10 adversarial inputs are crafted with $\epsilon = 8$.

Method	Hyperparameters	Accuracy (%)		ROC-AUC	
		Clean	PGD-10	Clean	PGD-10
SelectiveNet	$\lambda = 8, c = 0.7$	80.57	53.43	**0.796**	**0.730**
	$\lambda = 8, c = 0.8$	82.16	53.90	0.768	0.716
	$\lambda = 8, c = 0.9$	81.33	53.82	0.757	0.694
	$\lambda = 16, c = 0.7$	81.08	53.62	0.792	0.725
	$\lambda = 16, c = 0.8$	81.72	53.90	0.782	0.722
	$\lambda = 16, c = 0.9$	82.21	54.08	0.751	0.701
	$\lambda = 32, c = 0.7$	79.98	53.52	0.793	0.716
	$\lambda = 32, c = 0.8$	80.60	53.71	0.774	0.711
	$\lambda = 32, c = 0.9$	82.48	53.86	0.750	0.704
EBD	$m_{in} = -5, m_{out} = -23$	overflow			
	$m_{in} = 6, m_{out} = 0$	80.71	52.55	0.831	0.768
	$m_{in} = 6, m_{out} = 3$	81.98	53.89	0.832	0.763

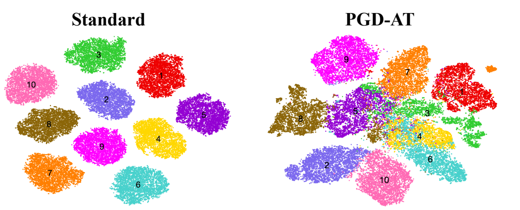


Figure 7: t-SNE visualization of the learned features on CIFAR-10. The irregular distributions of adversarially learned features make previous statistic-based detection methods less effective.

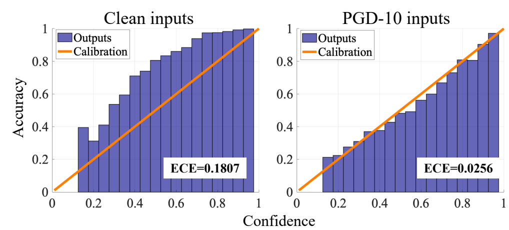


Figure 8: Reliability diagrams for an adversarially trained ResNet-18 on CIFAR-10, and the expected calibration error (ECE) (Guo et al., 2017). The model outputs are well calibrated.

calculate the mean and covariance matrix on all correctly classified training samples for GDA and GMM. For SelectiveNet, the $\lambda = 8$ and coverage is 0.7. For EBD, there is $m_{in} = 6$ and $m_{out} = 3$.

Kernel density (KD). In Feinman et al. (2017), KD applies a Gaussian kernel $K(z_1, z_2) = \exp(-\|z_1 - z_2\|_2^2/\sigma^2)$ to compute the similarity between two features z_1 and z_2. There is a hyperparameter σ controlling the bandwidth of the kernel, i.e., the smoothness of the density estimation. In Table 9 and Table 10, we report the ROC-AUC scores under different values of σ, where we restore the features of $1,000$/100 correctly classified training samples in each class on CIFAR-10/CIFAR-100, respectively.

Local intrinsic dimensionality (LID). In Ma et al. (2018), LID applies K nearest neighbors to approximate the dimension of local data distribution. Instead of computing LID in each mini-batch, we allow the detector to use a total of $10,000$ correctly classified training data points, and treat the number of K as a hyperparameter, as tuned in Table 9 and Table 10.

SelectiveNet (SNet). In Geifman and El-Yaniv (2019), the training objective consists of three parts, i.e., the prediction head, the selection head, and the auxiliary head. There are two hyperparameters in SelectiveNet, one is the coverage c, which is the expected value of selection outputs, another one is λ controlling the relative importance of the coverage constraint. In the standard setting, Geifman and El-Yaniv (2019) suggest $\lambda = 32$ and $c = 0.8$, while we investigate a wider range of λ and c when incorporating SelectiveNet with the PGD-AT framework, as reported in Table 11.

Energy-based detection (EBD). In Liu et al. (2020b), the discriminative classifier is implicitly treated as an energy-based model, which returns unnormalized density estimation. The two hyperparameters in EBD are m_{in} and m_{out}, controlling the upper and lower clipping bounds for correctly and

Table 12: Classification accuracy (%) and the ROC-AUC scores on CIFAR-10. The AT framework is PGD-AT and the model architecture is WRN-34-10. For KD, we restore $1,000$ correctly classified training features in each class and use $\sigma = 10^{-3}$. For LID, we restore totally $10,000$ correctly classified training features and use $K = 600$. We calculate mean and covariance matrix on all correctly classified training samples for GDA and GMM. For SNet, the $\lambda = 8$ and coverage is 0.7. For EBD, there is $m_{in} = 6$ and $m_{out} = 3$.

Rejector	Clean		$\ell_\infty, 8/255$		$\ell_\infty, 16/255$		$\ell_2, 128/255$	
	TPR-95	AUC	TPR-95	AUC	TPR-95	AUC	TPR-95	AUC
KD	85.51	0.759	57.26	0.674	34.87	0.605	67.55	0.695
LID	86.94	0.760	58.53	0.690	35.54	0.642	68.62	0.699
GDA	85.10	0.512	56.47	0.506	34.22	0.482	66.79	0.503
GDA*	87.16	0.694	57.62	0.627	34.66	0.561	68.23	0.637
GMM	88.36	0.747	57.98	0.650	34.79	0.568	68.87	0.667
SNet	88.30	0.803	60.07	0.733	**37.63**	0.695	70.14	0.730
EBD	89.63	0.860	60.96	0.778	36.92	**0.712**	70.97	0.792
RR	**90.74**	**0.897**	**61.48**	**0.783**	36.52	0.698	**72.00**	**0.809**

wrongly classified inputs, respectively. In Table 11, we tried the setting of $m_{in} = -5, m_{out} = -23$ as used in the original paper, which overflows on ATMs.

D.3 Details on attacking parameters

For **PGD attacks** (Madry et al., 2018), we use the step size of $2/255$ under ℓ_∞ threat model, and the step size of $16/255$ under ℓ_2 threat model. We apply untargeted mode with 5 restarts. For **CW attacks** (Carlini and Wagner, 2017a), we set the binary search steps to be 9 with the initial $c = 0.01$. The iteration steps for each c are $1,000$ with the learning rate of 0.005. Let x, x^* be the clean and adversarial inputs with the pixels scaled to $[0, 1]$. The values reported for CW-ℓ_∞ are $\|x - x^*\|_\infty \times 255$, while those for CW-$\ell_2$ are $\|x - x^*\|_2^2$. The default settings of **AutoAttack** (Croce and Hein, 2020) involve 100-steps APGD-CE/APGD-DLR with 5 restarts, 100-steps FAB with 5 restarts, $5,000$ query times for the square attack. For **multi-target attacks** (Gowal et al., 2019), we use 100 iterations and 20 restarts for each of the 9 targeted class, thus the number of total iteration steps on each data point is $100 \times 20 \times 9 = 18,000$. For **GAMA attacks**, we follow the default settings used in the offical code[3].

D.4 More results of WRN-34-10 and CIFAR-100

In Table 12, we use the larger model architecture of WRN-34-10 (Zagoruyko and Komodakis, 2016). We evaluate under PGD-10 ($\ell_\infty, \epsilon = 8/255$) which is seen during training, and unseen attacks with different perturbation constraint ($\epsilon = 16/255$), threat model (ℓ_2). As to the baselines, we choose SNet and EBD since they perform well in the cases of training ResNet-18. In Table 13, we experiment on CIFAR-100, and similarly evaluate under different variants of PGD-10 attacks. We report the results using both ResNet-18 and WRN-34-10 model architectures.

Moreover, to exclude gradient obstruction (Carlini et al., 2019), we do a sanity check by running PGD-10 against PGD-AT+**RR** on CIFAR-10 under $\epsilon = \{8, 16, 32, 64, 128\}/255$, where the model architecture is ResNet-18. The ALL accuracy (%) before rejection is $\{54.40, 33.56, 19.80, 6.71, 0.95\}$, which converges to zero.

D.5 Visualization of adversarially learned features

Although statistic-based detection methods like KD, LID, GDA, and GMM have achieved good performance on STMs against *non-adaptive* or *oblivious* attacks (Carlini et al., 2019), they perform much worse when combined with ATMs. To explain this phenomenon, we plot the t-SNE visualization (Van der Maaten and Hinton, 2008) in Fig. 7 on the standardly and adversarially learned

[3]https://github.com/val-iisc/GAMA-GAT

Table 13: Classification accuracy (%) and the ROC-AUC scores on CIFAR-100 under PGD-10 attacks. For KD, we restore the features on 100 correctly classified training samples in each class and use $\sigma = 1$. For LID, we restore the features on totally $10,000$ correctly classified training samples and use $K = 20$. For SNet, the $\lambda = 8$ and coverage is 0.7. For EBD, there is $m_{in} = 6$ and $m_{out} = 3$.

Rejector	Clean		$\ell_\infty, 8/255$		$\ell_\infty, 16/255$		$\ell_2, 128/255$	
	TPR-95	AUC	TPR-95	AUC	TPR-95	AUC	TPR-95	AUC
Architecture backbone: ResNet-18								
KD	58.20	0.549	30.23	0.532	16.39	0.510	40.67	0.539
LID	59.49	0.674	31.60	0.668	16.86	0.661	42.01	0.658
GDA	57.06	0.416	29.67	0.412	16.17	0.410	39.83	0.416
GDA*	58.98	0.599	31.40	0.593	17.04	0.588	42.10	0.596
GMM	58.06	0.518	30.48	0.505	16.69	0.508	40.68	0.511
SNet	59.68	0.729	33.12	0.743	19.48	0.759	42.72	0.726
EBD	61.44	0.795	34.56	0.776	**20.50**	0.762	44.22	0.777
RR	**64.44**	**0.837**	**35.52**	**0.782**	19.89	**0.767**	**47.03**	**0.802**
Architecture backbone: WRN-34-10								
KD	62.04	0.602	32.59	0.573	18.19	0.559	41.66	0.575
LID	63.17	0.705	33.27	0.672	18.97	0.652	42.97	0.672
GDA	60.12	0.436	31.64	0.426	17.75	0.421	40.52	0.423
GDA*	62.71	0.601	33.79	0.605	18.65	0.575	42.91	0.602
GMM	61.80	0.519	33.33	0.520	18.95	0.529	42.27	0.513
SNet	64.09	0.727	36.14	0.738	22.02	0.753	44.32	0.713
EBD	66.83	0.810	37.76	0.775	21.80	0.743	46.80	0.789
RR	**70.14**	**0.853**	**38.81**	**0.790**	**22.20**	**0.765**	**48.26**	**0.801**

features. As seen, ATMs have much more irregular feature distributions compared to STMs, while this fact breaks the statistic assumptions and rationale of previous statistic-based detection methods. For example, GDA applying a tied covariance matrix becomes unreasonable for ATMs, and this is why after using the conditional covariance matrix, GDA* performs better than GDA.

In Fig. 8, we also plot the reliability diagrams for an adversarially trained ResNet-18 on CIFAR-10, and we report the expected calibration error (ECE) (Guo et al., 2017). We can observe that the model trained by PGD-AT is well-calibrated, at least on the seen attack PGD-10, which is consistent with previous observations (Stutz et al., 2020; Wu et al., 2018).