

Appendix

A. Related work

Randomized smoothing (Cohen et al., 2019) proposes to first convert any base neural network into a smoothed classifier by injecting spherical Gaussian noises to inputs followed by majority voting, then provides a robust certificate that can guarantee the prediction of the resulting smoothed classifier is constant within some ℓ_2 -norm ball for any given input. Compared with other robustness certification methods, the biggest advantage of the randomized smoothing framework is its scalability to large neural networks and large-scale datasets such as classification task for ImageNet. In particular, (Cohen et al., 2019) provided a tight robustness guarantee for randomized smoothing with ℓ_2 perturbations. Later, SmoothADV (Shafahi et al., 2019) improved the proposed training method in (Cohen et al., 2019) by designing an adaptive attack on the smoothed classifier using adversarial training and first-order approximations. In addition, MACER (Zhai et al., 2020) developed a more direct way which directly optimizes the smoothed classifiers' certified radius with respect to correctly-classified samples using margin based loss and achieves better robustness and accuracy trade-off than previous methods.

Cost-sensitive learning deals with the situation where different misclassifications will induce different costs (Domingos, 1999; Elkan, 2001). For example, misclassifying a malicious tumor to benign (Khan et al., 2017) will bring more harmful consequences to the patient than the reverse. In adversarial training, it's also valuable to make the classifier adapt to the cost-sensitive setting so that adversarial transformations with high costs will be less likely to happen. Most of the cost-sensitive robust training methods are either could only be employed on linear classifiers or are empirical training methods without any robust certification (Chen et al., 2021; Khan et al., 2017). (Zhang & Evans, 2018) firstly trained a cost-sensitive certifiable classifier using convex optimization certifiable methods, however, it can not scale to large neural network or large datasets such as ImageNet. Our work combines randomized smoothing and cost-sensitive learning to provide more scalable classifiers with good certifiable robustness under cost-sensitive scenarios.

B. Hyperparameter Tuning

This section further studies the effect of the hyperparameters of the proposed method in Section 3.2 on the two objectives, certified overall accuracy and cost-sensitive robustness, with respect to γ_1 and γ_2 . Note that our goal is to improve cost-sensitive robustness without sacrificing overall accuracy, where γ_1 controls the margin of normal classes and γ_2 controls the margin of sensitive classes. In particular, we report the parameter tuning results on CIFAR-10. Here, the cost matrix is selected as a seed-wise cost matrix with a sensitive seed class "Cat". We choose the specific "Cat" class only for the purpose of illustration, as we observe similar trends in our experiments for other cost matrices, similar to the results shown in Table 1.

In addition, we consider two comparison baselines:

1. MACER (Zhai et al., 2020) with $\gamma = 8$, restricting only on correctly classified examples.
2. Our method with $\gamma_1 = 8$ and $\gamma_2 = 8$, the only difference with MACER is that our method contains misclassified examples for sensitive classes.

Below, we show the effect of γ_1 and γ_2 on the performance of our method, respectively.

Effect of γ_1 . Note that γ_1 is used to restrict the certified radius with respect to normal data points. Figure 4 illustrates the influence of varying $\gamma_1 \in \{4, 6, 8\}$ and fixed $\gamma_2 = 10$ for our method, with comparisons to the two baselines, on both overall accuracy and cost-sensitive robustness.

For the original implementation of MACER, γ is selected as 8 for the best overall performance. Although it achieves good overall robustness, it does not work for cost-sensitive settings, which suggests the possibility of a trade-off space, where different classes can be balanced to achieve our desired goal of cost-sensitive robustness. The second baseline is our method with $\gamma_1 = 8$ and $\gamma_2 = 8$. By incorporating misclassified samples for sensitive seed class, the cost-sensitive performance substantially improves. This results shows the significance of including misclassified sensitive samples during the optimization process of the certified radius.

Moreover, we can observe from Figure 4(b) that as we reduce the value of γ_1 , the robustness performance of the cost-sensitive

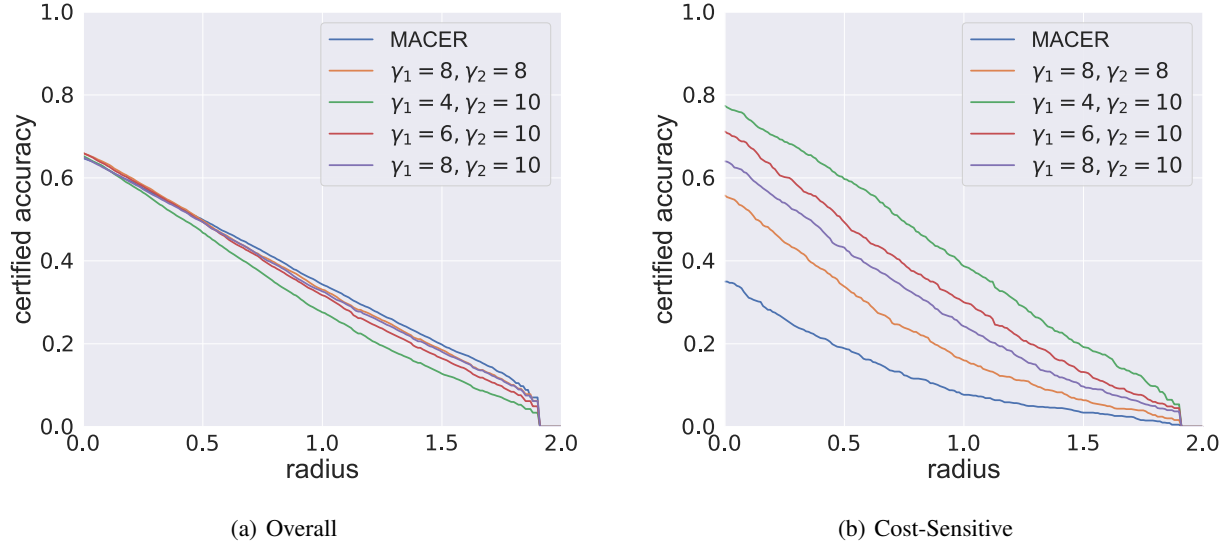


Figure 4. Visualizations of our method with $\gamma_1 \in \{4, 6, 8\}$ and fixed $\gamma_2 = 10$ with comparisons to baseline methods in terms of: (a) overall performance and (b) cost-sensitive performance. Here, the cost matrix is set as the matrix representing a single cost-sensitive seed class “Cat”.

seed class increases. This again confirms that by limiting the certified radius of normal classes to a small threshold in our method, the model can prioritize sensitive classes and enhance cost-sensitive robustness.

Effect of γ_2 . Figure 5 illustrates the influence of varying $\gamma_2 \in \{8, 12, 16\}$ with fixed $\gamma_1 = 4$ or fixed $\gamma_1 = 8$ for our method, with comparisons to the two baselines, on both overall accuracy and cost-sensitive robustness. Moreover, we can observe from Figure 5(b) and Figure 5(d) that as we increase the value of γ_2 , the robustness performance of the cost-sensitive seed class increases. This confirms that by optimizing the certified radius of sensitive classes to a large threshold in our method, the model can focus more on sensitive classes and enhance cost-sensitive robustness. Additionally, there is a slight increase in the overall certified accuracy. This can be attributed to the fact that the overall accuracy takes into account both the accuracy of sensitive samples and normal samples. As the certified accuracy of sensitive samples increases, it dominates the overall accuracy and leads to its overall improvement.

Table 3 demonstrates the impact of different combinations of hyperparameters of (γ_1, γ_2) on both the overall accuracy and cost-sensitive performance. The choice of γ_1 and γ_2 is crucial and requires careful consideration. For γ_2 , setting a value that is too small can greatly undermine the overall accuracy, even though it may improve cost-sensitive robustness. This is because the performance of normal classes deteriorates, resulting in a degradation of overall performance. On the other hand, if the value is too large such as $\gamma_2 = 8$, it may have a negative impact on cost-sensitive performance.

Regarding γ_1 , it is evident that increasing its value while keeping γ_2 fixed leads to a significant improvement in cost-sensitive robustness. It is worth noting that even though the cost-sensitive seed class represents only a single seed, accounting for only 10% of the total classes, enhancing its robustness has a positive effect on overall accuracy as well. For instance, let’s compare the combination $(\gamma_1 = 8, \gamma_2 = 4)$ to $(\gamma_1 = 8, \gamma_2 = 8)$. We observe that the former, which exhibits better cost-sensitive robustness, outperforms the latter in terms of both overall accuracy and cost-sensitive robustness. It achieves an approximate improvement of 1.52% in overall accuracy and a significant improvement of approximately 50% in cost-sensitive robustness.

This finding highlights the effectiveness of our sub-population-based methods. It demonstrates that by fine-tuning the optimization thresholds for the certified radius of sensitive classes and normal classes separately, we can achieve a better trade-off between overall accuracy and cost-sensitive robustness.

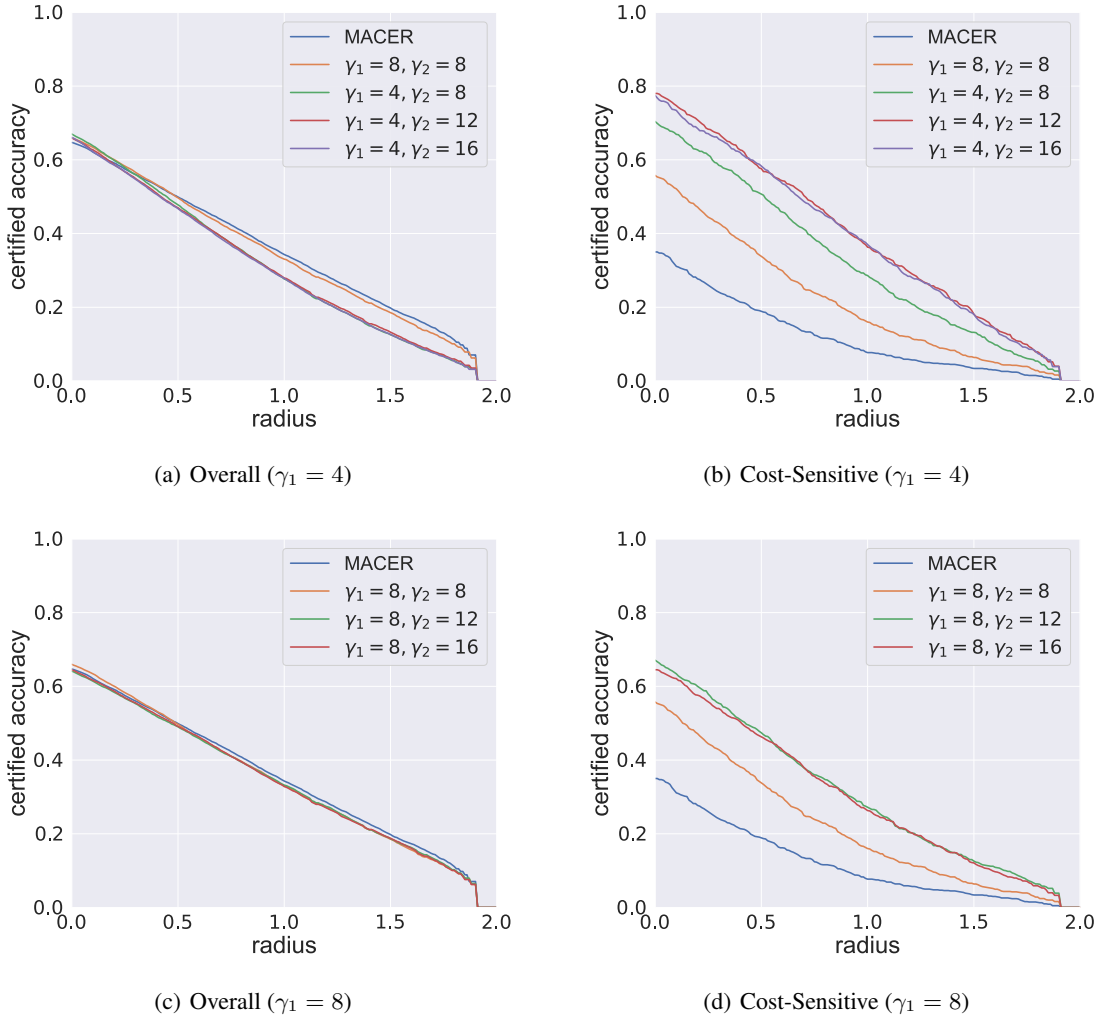


Figure 5. Visualizations of our method for two groups comparisons to baseline methods in terms of (a)(c) overall performance and (b)(d) cost-sensitive performance. The first with $\gamma_2 \in \{8, 12, 16\}$ and fixed $\gamma_1 = 4$, the second with $\gamma_2 \in \{8, 12, 16\}$ and fixed $\gamma_1 = 8$. The cost matrix is set as the matrix representing a single cost-sensitive seed class “Cat”.

Table 3. This table shows results for different parameter combinations. The evaluation metric contains the overall accuracy and sensitive robustness. Here, the cost matrix is set as the matrix representing a single cost-sensitive seed class “Cat”.

Method	sensitive	normal	Overall accuracy	Cost-Sensitive robustness
MACER	-	-	0.647	0.189
Ours	8	8	0.660	0.338
Ours	8	2	0.654	0.633
	10	2	0.634	0.687
	12	2	0.637	0.691
	16	2	0.630	0.705
Ours	8	4	0.670	0.507
	10	4	0.653	0.597
	12	4	0.659	0.576
	16	4	0.661	0.583
Ours	8	6	0.673	0.396
	10	6	0.660	0.493
	12	6	0.655	0.544
	16	6	0.649	0.552
Ours	8	8	0.660	0.338
	10	8	0.650	0.432
	12	8	0.641	0.474
	16	8	0.645	0.463

C. Comparisons with Convex Relaxation-based Method

Zhang et al. (Zhang & Evans, 2018) proposed a method to certify cost-sensitive robustness of any classifier based on convex relaxation method (Wong & Kolter, 2018), which provides a robustness guarantee for a given input via minimizing the worst-case loss within the relaxed convex outer polytype. Also, Zhang & Evans (2018) developed a training method for training provably cost-sensitive robust classifiers. In particular, their method incorporates different types of cost matrices into the convex optimization process to train cost-sensitive robust classifiers.

However, the initial work of (Wong & Kolter, 2018) only focuses on ℓ_∞ -norm bounded perturbations and does not consider perturbations in ℓ_2 -norm. As a result, the proposed method in Zhang & Evans (2018) also did not address the cost-sensitive robustness for ℓ_2 perturbations. We note that in a follow-up work of (?), they extend the developed certification techniques to ℓ_2 perturbations. For fair comparisons with our method, we further extend the cost-sensitive robust learning method of (Zhang & Evans, 2018) to handle ℓ_2 -norm perturbations using the method of (?). We report their comparisons in Table 4, the certified cost-sensitive robustness for the convex-relaxation method is computed as the *cost-sensitive robust error* defined in (Zhang & Evans, 2018), which represents the fraction of test samples that are guaranteed to be robust to certain ℓ_2 perturbations.

Table 4. Comparison results of our method with convex relaxation based method (Zhang & Evans, 2018) for ℓ_2 perturbations, where a single pairwise cost-sensitive transformation ($3 \rightarrow 5$) is considered.

Method	ℓ_2 -perturbation	Cost-sensitive robustness	Overall accuracy
Convex relaxation (Zhang & Evans, 2018)	0.25	0.944	0.480
Ours	0.5	0.924	0.673

From Table 4, we observe our method is able to achieve much higher overall certified accuracy even against large ℓ_2 perturbations, suggesting better cost-sensitive robustness and overall accuracy trade-off. However, we find that the convex-relaxation method is not applicable to larger ℓ_2 perturbations (e.g., with $\epsilon = 0.5$), because of out-of-memory issues.

We remark that randomized smoothing techniques, as proposed in previous works, such as Cohen et al. (Cohen et al., 2019),

Li et al. (Li et al., 2019) and Jia et al. (Jia et al., 2019), primarily focus on defending against ℓ_2 -norm perturbations. As a result, our methods excel in achieving good cost-sensitive performance under ℓ_2 -norm attacks. However, they may have limitations when it comes to ℓ_1 -norm and ℓ_∞ -norm attacks. To overcome these limitations and extend our method to effectively handle these two types of attacks, further research and investigation are required.

D. Discussions of Algorithm 1

In this section, we provide further details and discussions of Algorithm 1 for certifying cost-sensitive robustness presented in Section 2.2. We follow the same sampling procedure of Cohen et al. (Cohen et al., 2019). To be more specific, the sampling function $\text{SAMPLEUNDERNOISE}(f, x, n, \sigma)$ is defined as:

1. Draw n i.i.d. samples of Gaussian noises $\delta_1 \dots \delta_n \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$.
2. Obtain the predictions $f(x + \delta_1), \dots, f(x + \delta_n)$ with base classifier f on noisy images.
3. Return the counts for each class, where the count for class c is $\sum_{i \in [n]} \mathbb{1}[f(x + \delta_i) = c]$.

Algorithm 1 introduces two approaches to compute certified radius for any cost-matrix: R_1 is computed using a lower $1 - \alpha$ confidence bound on p_A , while R_2 is computed using both a lower $1 - \alpha/2$ confidence bound of p_A and an upper $1 - \alpha/2$ confidence bound of p_B (see Theorem 1 in (Cohen et al., 2019)). According to Proposition 2 and Theorem 1 in (Cohen et al., 2019), we can easily show by union bounds that with probability at least $1 - \alpha$ over the randomness of sampling, the final output $R = \max(R_1, R_2)$ returned by Algorithm 1 defines a cost-sensitive certified radius for any given input.

We remark the certification algorithm proposed in (Cohen et al., 2019) only considers the first approach for computing the certified radius for certifying overall robustness. This is because choosing $1 - \alpha$ provides a better confidence bound of p_A , under condition that p_B is close to $1 - p_A$, which typically holds for overall robustness. For certain cost-sensitive matrices, however, it is possible that $R_2 > R_1$ for some input, which means $R_2 = \max\{R_1, R_2\}$ will be the final output of Algorithm 1. This is because the class probabilities for computing p_B with respect to Ω_y could be very small, which may not contain the second-highest probability class, especially when the number of cost-sensitive target classes $|\Omega_y|$ is small. To make sure we are producing certified radius as large as possible, Algorithm 1 selects the larger value between R_1 and R_2 as the certified radius for any cost-sensitive seed example.