Robustness Verification with Non-Uniform Randomized Smoothing

Lucas Tecot
Department of Computer Science
University of California, Los Angeles
lucasmtecot@ucla.edu

Huan Zhang
Department of Computer Science
University of California, Los Angeles
huanzhang@ucla.edu

Cho-Jui Hsieh
Department of Computer Science
University of California, Los Angeles
chohsieh@ucla.edu

Quanquan Gu
Department of Computer Science
University of California, Los Angeles
qgu@cs.ucla.edu

Abstract

Formal verification via randomized smoothing has become an effective method for verifying the robustness of machine learning models to adversarial attacks. In this work, we extend randomized smoothing by allowing noises with independent variances on each input element of a smoothed classifier, enabling the flexibility of using non-uniform perturbations rather than the standard ℓ_p norm perturbations. Furthermore, we provide practical optimization methods to find the optimal variance matrix through gradient descent. We evaluate our method on MNIST, Fashion-MNIST, CIFAR-10, ImageNet, and KITTI datasets and show that our optimized non-uniform smoothing noises can certify a region with guaranteed robustness magnitudes larger in volume than previous works. Additionally, our method can be used to evaluate the sensitivity of input features and identify non-robust and robust features.

1 Introduction

Neural networks have caused a massive leap in the capabilities of machine learning over the past few decades. However, neural networks have been shown to be vulnerable to adversarial attacks, in which seemingly inconsequential changes to the input data cause the model to drastically change their output undesirably [10; 30]. These types of attacks have been demonstrated in black-box attacks where the attacker only has access to the model's predictions [22; 2; 3], and succeed in real-world examples [29; 1]. Understandably, this is a massive security problem, especially for neural networks deployed in environments where manipulating features is easy, and changing their outputs can cause large amounts of real-world harm. As a result, there has been a large body of work in making models more robust to adversarial of attacks [21; 16; 13; 11; 31; 36].

Recently there has been the significant development on verification via randomized smoothing methods [17; 20; 5]. These methods instead verify the robustness of a "smoothed" model that returns the most probable class of a base model $f(\cdot)$ when the input is perturbed randomly. The robustness of each input point is certified within some ℓ_2-ball, often with much larger bounds than previous interval-propagation based methods [5]. Furthermore, this method only samples $f(\cdot)$ instead of looking at the architecture itself, and as such can be plugged into and efficiently run on any $f(\cdot)$ to get a certified robust model.

Preprint. Under review.

We provide further improvements to this method. Specifically, past works have used this model with a single variance to be used for all elements of the $f(\cdot)$ base model input. These single-variance methods can only certify a ℓ_2 ball with radius r. But in practice, features are not always equally robust. For instance, in datasets like MNIST, models are likely more robust around the images edges where there is no useful information. As such, certifying a uniform ball will lead to a small radius to accommodate the sensitive features. To overcome this issue, we provide a modification that allows for this method to be used with independent variances per each model input element.

Furthermore, we propose a method to find an optimal variance vector for the inputs. Previous randomized smoothing methods usually empirically try different variances and select the best one. However, with our approach there will be numerous independent variances that need to be selected. We show that variance selection can be posed as an optimization problem to maximize some predefined criteria (for instance, maximum volume of the certified region), and the gradient can be back-propagated to the variances via the reparameterization trick.

We provide results to help illustrate where and how our new method is useful. We are able to achieve significantly improved results over the state of the art for datasets with non-uniformly robust features. For datasets with more uniformly robust features, we are still able to achieve better results in some cases, although the improvements are much more modest. Furthermore, the variance vectors produced by our method allow for insights into the robustness of each feature for the underlying model.

2 Previous Work

The crux of what we are interested in is finding the *minimum adversarial distortion*, which is the minimum change required to the input of the network in order to cause a correct classification to become an incorrect classification. There are a number of methods for finding this exact value [12; 8]. However, unfortunately for common non-linear activations such as ReLU, this is believed to be a NP-Complete problem [12; 28] and is typically too computationally intensive for anything beyond small models. As such, there have been a number of previous works that provide algorithms to efficiently derive lower bounds on the minimum adversarial distortion for multi-layer perceptrons and other common neural network architectures [13; 37; 25; 26; 33; 32; 4; 27]. The state-of-the-art algorithms mostly work in a similar fashion; we compute a volume in which we can guarantee that our network's output will fall, given that we confine our inputs to a specific p-norm ball around an input point x. Because it is required to "relax" the problem in order to make the computation efficient, it is not guaranteed that the model can output any value in this volume. But it is guaranteed that any output produced from the specified input space will be in the volume. Using this information, finally it is determined whether or not this output range may allow for an attacker to perturb the input x such that the model will change it's classification of this point.

However, these relaxing methods have significant drawbacks. They are slow to run, and with deep networks they give very loose bounds. To address this, recently formal verification via randomized smoothing methods have made significant progress [17; 20]. These methods instead verify the robustness of a "smoothed" model that returns the most probable class of a base model $f(\cdot)$ when the input is perturbed via random sampling from a Gaussian distribution ($f(x + \epsilon)$). Due to the nature of the noise used for these classifiers, a ℓ_2-ball robustness bound can be derived without any information of the inner workings of the base model. This method generally gives very good robustness certificates, and it has even been proved optimal for adversarial L-2 perturbations [5]. Additionally, various works have shown that the certified robustness of randomized smoothing can be further enhanced if $f(\cdot)$ is robustly trained [24]. Beyond ℓ_2 ball certification, several variants of randomized smoothing methods were recently proposed for $\ell_0, \ell_1, \ell_\infty$ balls [19; 7; 18; 15]. However, the certified bounds are generally tighter in the ℓ_2 case due to nice properties of the Gaussian distribution.

The drawbacks of randomized smoothing is that because it is impossible to directly sample the smoothed classifier, it must be sampled via approximation processes. This means that these models can only be certified to a high probability, and not complete certainty. Furthermore, because this method is certifying the smoothed classifier and not the original one, anyone using the model who wants these safety guarantees must sample the base model with random noise multiple times for a single use of the model.

3 Randomized Smoothing with Non-Isotropic Gaussian

Before we introduce our new theorem, let us first provide some important introductory definitions:

Definition 3.1 *Let $f : \mathbb{R}^D \to \gamma$ be a multi-class classification model with $\gamma = \{1, \ldots, C\}$. We defined the smoothed classifier g as:*

$$g(x) = \underset{c \in \gamma}{\operatorname{argmax}} \, \mathbb{P}(f(x + \epsilon) = c),$$

where ϵ is some random perturbation added to the input space.

Definition 3.2 *Let $c_a \in \gamma$, and let $\underline{p_A}$ and $\overline{p_B}$ satisfy the following:*

$$\mathbb{P}(f(x + \epsilon) = c_a) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c \neq c_a} \mathbb{P}(f(x + \epsilon) = c).$$

In the method proposed by [5], they assume $\epsilon \sim \mathcal{N}(0, \Sigma)$ where $\Sigma = \sigma^2 I$, and they are able to show:

$$\|\delta\| < \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B}))$$
$$\|\delta\| < \sigma \Phi^{-1}(\underline{p_A}) \quad (\text{if } \overline{p_B} = 1 - \underline{p_A}).$$

In this work, we show the randomized smoothing approach can be extended to non-isotropic Gaussian noise. We assume the co-variance matrix Σ is a diagonal matrix with vector $\sigma^{\circ 2}$ as the diagonal. (Where $\cdot^{\circ 2}$ is the element-wise square.) Using this assumption, we can prove that:

Theorem 3.1 *$g(x + \delta) = c_a$ for all δ vectors that satisfy the following inequality:*

$$\|\delta \oslash \sigma\| < \frac{1}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B}))$$
$$\|\delta \oslash \sigma\| < \Phi^{-1}(\underline{p_A}) \quad (\textit{if } \overline{p_B} = 1 - \underline{p_A}),$$

where Φ^{-1} is the inverse of the standard Gaussian CDF, and $\oslash$ is the Hadamard (element-wise) division.

The proof for this theorem can be found in the appendix. It closely follows the proof of [5]; we simply change the assumption of the structure of Σ and follow the mathematical implications of such a change through the entire proof.

Furthermore, note what occurs when we take Theorem 3.1 and constrain the σ vector to contain only elements of the same value, which we will denote as σ':

$$\|\delta \oslash \sigma\| < \Phi^{-1}(\underline{p_A}) \Rightarrow \|\delta\| < \sigma' \Phi^{-1}(\underline{p_A}).$$

We recover the exact same certified radius as that of [5]. This means that when the σ vector in our theorem is constrained to have elements of the same value, it is identical to the bound of previous works.

4 Algorithm

Now that we have a theorem which allows us to certify a smoothed classifier with different variances per each input element, the question remains: how do we find an optimal variances vector? In previous works there is only a single parameter σ, so in practice they test many possible σ values and pick the most suitable one. However, this will not work for the non-isotropic case because there are numerous parameters. Here we define a reasonable optimization objective, and describe how we make said objective differentiable with respect to σ and hence optimizeable through gradient descent methods.

4.1 Certified Area

The major issue with this new formulation in Theorem 3.1 is that we no longer have a certified "radius", because the maximum δ perturbation we can guarantee robustness for depends on the

elements of σ. However, although we can't define a radius, we can instead define a correlated metric: certified *area*. Recall the equation introduced by Theorem 3.1:

$$\|\delta \oslash \sigma\| < \Phi^{-1}(\underline{p_A}).$$

Notice how this equation is in fact the equation of the interior of a n-dimensional hyper-ellipsoid if we re-arrange a few terms:

$$\sum_{i=1}^{D} \frac{\delta_i^2}{(\Phi^{-1}(\underline{p_A})\sigma_i)^2} < 1,$$

where D is the number of elements in σ, and the number of elements in the input to the smoothed model. As such, this equation defines a hyper-ellipsoid with semi-axes of length $\Phi^{-1}(\underline{p_A})\sigma_1, \Phi^{-1}(\underline{p_A})\sigma_2, \ldots, \Phi^{-1}(\underline{p_A})\sigma_D$. The equation for the volume of such a hyper-ellipsoid is:

$$V = \frac{2\pi^{D/2}}{D\Gamma(D/2)} \prod_{i=1}^{D} \Phi^{-1}(\underline{p_A})\sigma_i.$$

For the purposes of an optimizable objective, we can ignore the first term, as it is a constant with respect to the number of input elements. In order to avoid exponential scales and massive multiplications, we will instead decide to optimize the natural logarithm of the volume. And this finally leaves us with our *certified area objective*:

$$\mathcal{O}_{CA} := \sum_{i=1}^{D} \log(\sigma_i) + D \log(\Phi^{-1}(\underline{p_A})).$$

4.2 Optimization Objective

Now that we have a certified area formulation that will serve as a suitable objective to optimize for, we need to form an explicit loss function. The optimized loss function for all of our experiments will be:

$$\mathcal{L} := \frac{\sum_{i=1}^{N} \mathcal{O}_{CA}(g(x_i))\mathbb{1}(g(x_i) = c_i)}{\sum_{i=1}^{N} \mathbb{1}(g(x_i) = c_i)},$$

where $O_{CA}(g(x_i))$ is the certified area objective evaluated on g with input x_i, and x_i plus c_i are the features and label respectively from the dataset of length N. Notice that all we are doing is averaging the certified area objective values for all data points that can actually be certified properly. (If the smoothed classifier predicts the data point wrong, there is no hope of it being able to certify said point.)

This loss has an inherent trade-off; the fewer points that are certified, the more aggressively the certified area for those points can be increased. In order to control this trade-off, we found that it sufficed to simply initialize the entire σ vector to different amplitude constants such that the model had varied accuracy at initialization. Often the model would stay close to its accuracy at initialization during the optimization process, likely because the $\underline{p_A}$ term incentivises improving the model and prevents unbounded increases to σ. However, it would be beneficial for future works to devise a loss that makes this trade-off more explicit.

4.3 Differentiability

Note that the certified area objective contains terms that are easily differentiable with respect to σ, with the exception of the term $\Phi^{-1}(\underline{p_A})$. Here we'll outline how we backpropagate through this term.

At test time where we don't require gradients, we compute $\underline{p_A}$ identically to that of the methods used by [5]. We can simply use randomized sampling to approximate the Gaussian noise, count the number of times that the model predicts the correct class, and use an inverse binomial CDF to bound a value of $\underline{p_A}$ with a certain percentage probability. The only difference is that now there are different σ terms for each input element, whereas before noise was applied uniformly over the whole input.

However, this procedure isn't sufficient for train time to maintain gradient information. We can make the outputs of the base model differentiable with respect to σ through a standard application of the reparameterization trick. However, when we count the output predictions to estimate $\underline{p_A}$, this destroys all gradient information. As such, we instead sum up the softmax outputs of all the randomly perturbed model outputs to estimate $\underline{p_A}$. Future works could get around this issue by using randomized smoothing formulations that instead rely on expectations of the direct output of the model [35]. However, in practice our estimate optimized our test metrics without issue, so we did not feel a need to modify further.

As such, for the train-time certified area objective we replace the $\Phi^{-1}(\underline{p_A})$ term as follows:

$$\Phi^{-1}(\underline{p_A}) \Rightarrow \Phi^{-1}\left(\frac{1}{n_0}\sum_{i=1}^{n_0} f(x+\sigma \odot z_i)_c\right),$$

where $z \sim \mathcal{N}(0, \mathbb{1})$ and $f(\cdot)_c$ is the softmax probability output of our model for the ground-truth label c of x. (Note that we decompose ϵ into $\sigma \odot z_i$ to make it possible to back-propagate to σ. Additionally we do not do any probability bounding for $\underline{p_A}$, as this guarantee is not important during training.)

Algorithm 1: Pseudocode for training of smoothed classifier σ vector

$\mathcal{L} \leftarrow 0, l \leftarrow 0$
for *x, c in dataset features, labels* **do**
  $z \sim \mathcal{N}(0, \mathbb{1})$
  $\widehat{p_A}_{SM} \leftarrow \frac{1}{n_0}\sum_{i=1}^{n_0} f(x+\sigma \odot z_i)_c$
  if $g(x) = c$ **then**
    $\mathcal{L} \leftarrow \mathcal{L} - \sum_{i=1}^{D} \log(\sigma_i) + D\log(\Phi^{-1}(\widehat{p_A}_{SM}))$
    $l \leftarrow l + 1$
  end
end
Perform gradient descent step on σ minimzing loss $\mathcal{L}/l$

5 Experiments

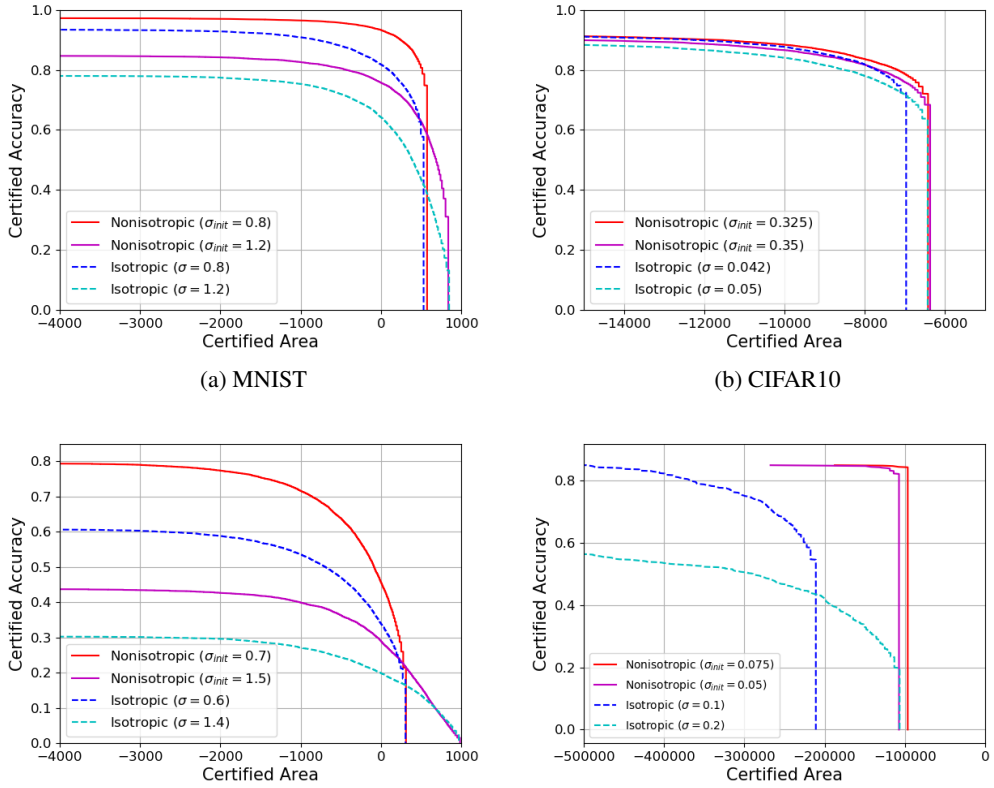
To help better understand our method and compare to previous methods, we produce two different plots: *certified accuracy vs. certified area* plots, and *clean accuracy vs. certified area* plots. The former is to look at a specific trained σ and understand how it performs on each point in a dataset; the latter is to help illustrate how differing methods and underlying models can perform as a whole. We produce these plots for the MNIST, Fashion-MNIST [34], CIFAR-10 [14], Imagenet [6], and KITTI [9] datasets.

Furthermore, because we are using image datasets, our σ vectors can be used to produce images that represent the amount of noise each pixel has introduced when using the smoothed classifier. We will show a selection of these images and discuss their implications for robustness insights into each model.

For MNIST, Fashion-MNIST, and CIFAR-10 results, we trained with the standard train set and reported metrics from the standard test set as provided by the Pytorch [23] library functions. For Imagenet results, we constructed train and test sets via random sampling from the ILSVRC 2012 challenge. The train set has 5000 samples and the test set has 1000. For KITTI, we simplified the original 2D object detection dataset by choosing the object class that had the most occurances as the label for the whole image. (For example, if a picture had 4 pedestrians and 2 cars, it would be labeled as a pedestrian example.) We split the dataset into 90 percent training data and 10 percent test data.

For MNIST and Fashion-MNIST, we used a convolutional neural network with two convolution layers, a max-pool, and two fully-connected layers. For CIFAR-10 we used the Resnet 110 architecture. For Imagenet and KITTI we used the Resnet 50 architecture. (For the KITTI dataset, images were resized to the same size as the Imagenet dataset, 224 by 224 pixels.) All models we trained ourselves, with the exception of Imagenet where we used the Pytorch standard pre-trained model.

For all the following experiments, certified area refers to optimization objective $\mathcal{O}_{CA}$ defined in subsection 4.1. Furthermore, there are three important hyper-parameters used in all these experiments:



(a) MNIST

(b) CIFAR10

(c) Fashion-MNIST

(d) KITTI

Figure 1: Certified accuracy vs. certified area plots. Nonisotropic lines are our method, optimized from a σ vector initialized to the constant in the legend for all elements. Isotropic lines are [5]'s method, using a σ as stated in the legend. Certified accuracy is the percent of the test dataset that the smoothed classifier classifies correctly and has a certified area at or below the x-axis certified area value. Certified area is the optimization objective $\mathcal{O}_{CA}$ given in subsection 4.1. Note that this objective is a logarithm, and therefore can be negative.

n_0, n, and α. All of these parameters are utilized similarly to that of [5]'s method. n_0 is the number of samples taken to determine the class the smoothed classifier predicts. Additionally, it is the number of samples averaged during train-time to produce the softmax outputs of the smoothed classifier. n is the number of samples used to estimate $\underline{p_A}$ during test time. And α is the probability with which $\underline{p_A}$ is a true lower bound for the probability of the smoothed classifier predicting class A. (In other words, $\underline{p_A} < p_A$ with probability $1 - \alpha$.)

All of these experiments were run on Nvidia GeForce GTX 1080 GPUs. The hyper-parameter values and processes used to create each plot will be detailed in each sub-section. (Furthermore, even more detailed information, such as the run-times of these experiments, can be found in the appendix.)

5.1 Certified Accuracy vs. Certified Area

To produce each certified accuracy plot, we first select a dataset, an underlying model, and a σ. When using [5]'s method, σ is simply a vector with the the same value for each element. When using our method, σ is a vector that was obtained via optimization from a training session where σ was initialized to a vector with the same value in each element, as detailed in subsection 4.2. Using these three items, we create a smoothed classifier and obtain the certified area for each data point in the test set.

Using this information we obtain the *certified accuracy* of each certified area value. For a specific certified area value, this certified accuracy is the percent of the test dataset that the smoothed classifer

classifies correctly **and** has a certified area at or below the specified value. As the certified area increases, fewer and fewer points are certified. Also note that there is a trade-off occurring here: the large the σ becomes, the less accurate the model becomes. However, it can certify the points it does predict correctly with a larger area.

To produce these plots, we set the hyperparameters as follows: $n_0 = 100$, $n = 1000$, $\alpha = 0.001$. In practice, we took our σ vectors from the procedure that produced the clean accuracy vs. objectives plots. Also note that the σ initialization for the non-isotropic vectors often differs from the σ used in the isotropic method it is compared to. This is because the σ vector's mean can change over time, so the σ's used to generate the isotropic lines were approximately the means of the non-isotropic σ vectors after training plotted in the same graph.

Our method outperforms previous works on the MNIST, Fashion-MNIST, KITTI, and CIFAR-10 datasets. We are able to achieve greater certified accuracy when compared against an isotropic smoothed classifier that can certify up to the same area. For the MNIST datasets this result was expected, as these two datasets have images with black backgrounds that are presumably not at all useful to a machine learning model. As such, the model is likely much more robust to perturbations on these black backgrounds. For KITTI and CIFAR-10 we are using natural images at a higher resolution, so the benefit of our method is not as obvious. However, like the MNIST datasets, it it still likely that the corners of the images contain less information about the image label. And this is precisely what we observed with our experiments. For the KITTI dataset we were able to achieve massive improvements, and for CIFAR-10 the improvements are more modest but it still outperforms previous methods. All these certified accuracy plots can be viewed in figure 1. The non-isotropic σ vectors of figure 1 are visualized in figure 3.

5.2 Clean Accuracy vs. Certified Area

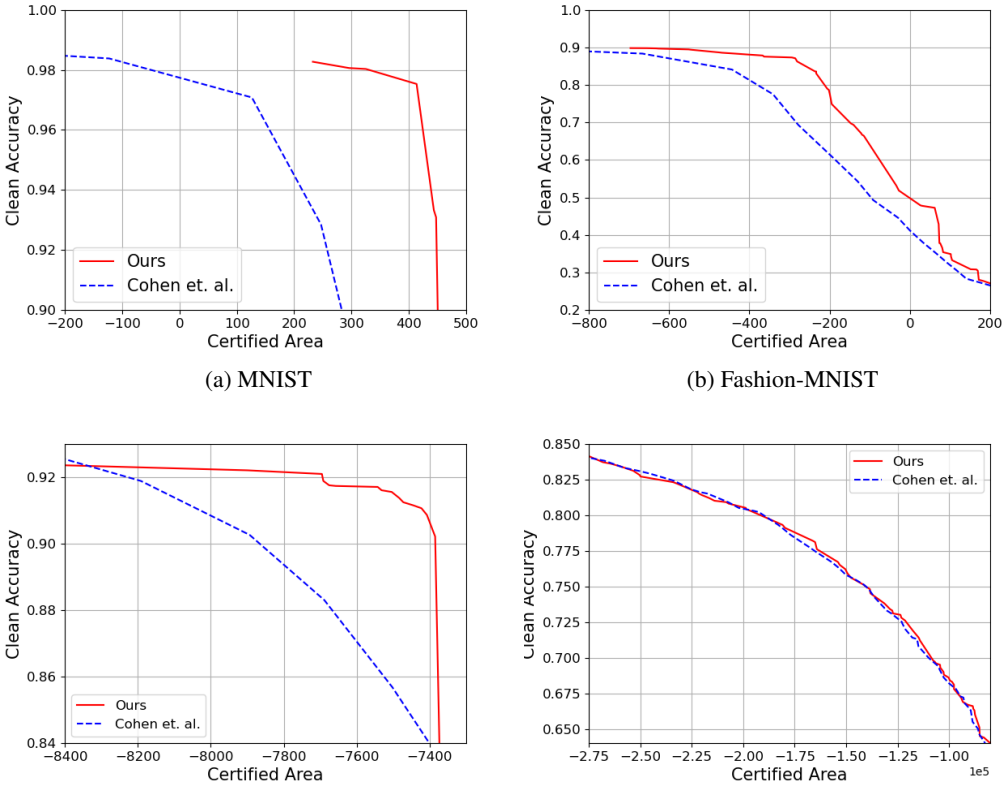
To produce each clean accuracy plot, we optimized σ for certified area when the vector was initialized to have all elements equal a specific constant, as detailed in subsection 4.2. We repeated this optimization procedure for a variety of initialized constants. We took the σ vector we were left with at the end of each epoch, and calculated the accuracy and the average certified area value of the smoothed classifier over the test dataset. Additionally, we calculated the same metrics for the method of [5] with a variety of σ values equal to the same range of constants we used as initialization for our method. Then we plotted each of these accuracy-objective points for both methods.

We produced these plots for the MNIST, Fashion-MNIST, CIFAR-10, and Imagenet datasets. (We also produced one for KITTI, but as the dataset is dominated by two classes it doesn't provide a very useful illustration of the accuracy-objective tradeoff. As such we deferred this plot to the appendix.) For these plots, we set the hyperparameters as follows: $n_0 = 32$, $n = 256$, $\alpha = 0.001$.

Our results here follow a similar trend to those of subsection 5.1. For MNIST and Fashion-MNIST, we were able to achieve significantly better robustness guarantees than [5]. For CIFAR-10 our method achieves improvements but they are not as dramatic as they are for the MNIST datasets.

We can observe that for Imagenet, our method performs very close to the isotropic randomized smoothing method, and further we observe that the σ vectors learnt by our algorithm are close to uniform. This indicates that the pixels are uniformly important for the model trained on Imagenet, so it is not beneficial (although not harmful) to apply our method. This is probably due to the rich diversity of ImageNet dataset. Furthermore, for Imagenet we used the Pytorch pre-trained model, which may have used some regularization techniques during training. (Unlike our other models which were trained only by using an ADAM optimizer on accuracy). We have observed with previous experiments that such techniques can make models more uniformly robust, so this could be another cause for why we see minimal gains from our method. Comparing the KITTI and ImageNet datasets, they are both natural high resolution images, but the main difference is that most of the images in KITTI datasets are under a similar scene while ImageNet dataset has more diverse scenes, and our method performs better in the former case. This indicates that although it doesn't hurt to perform our method on any dataset, the benefit will be more for those dataset with more monotonous scenes such as self-driving datasets.

All of the clean accuracy plots can be viewed in figure 2.



(a) MNIST

(b) Fashion-MNIST

(c) CIFAR10

(d) ImageNet

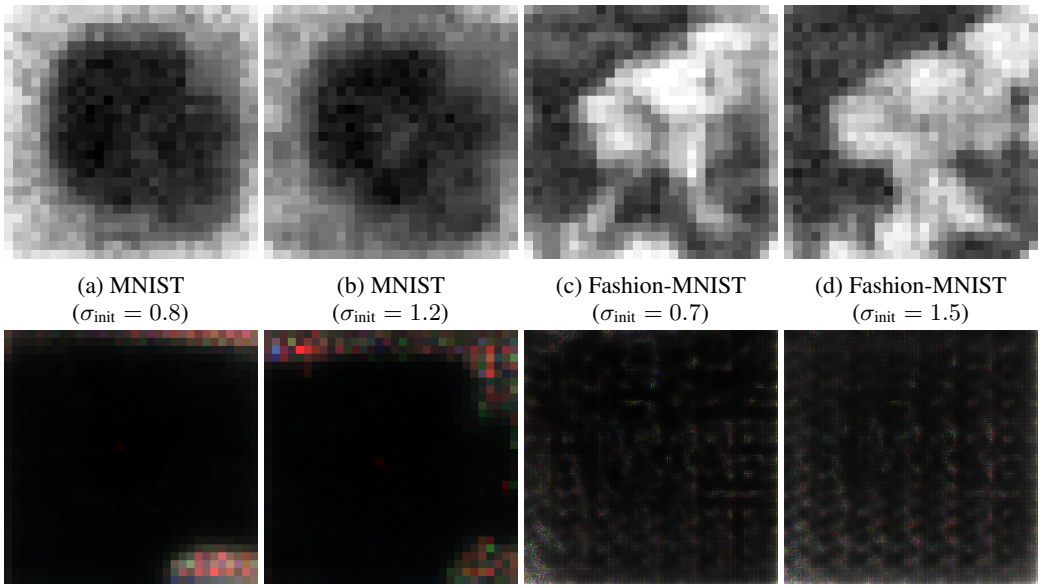
Figure 2: Clean accuracy vs. certified area plots. Clean accuracy is the percent of the test dataset that the smoothed classifier classifies correctly. Certified area is the optimization objective $\mathcal{O}_{CA}$ given in subsection 4.1. Note that this objective is a logarithm, and therefore can be negative.

5.3 Variance Images

In order to help visualize what was occurring, during producing these graphs we would save and display each σ vector as an image. This revealed an interesting side effect of our method; it allows you to gain insight into what input perturbations your model is more robust to.

We included a sampling of these σ visualizations in figure 3. Each σ image is linearly scaled down to the image color range (the lowest value in the picture being the darkest color, and the highest value being the brightest color). Note that for color images, we normalize globally instead of per-channel, to preserve any bias a model might have to a specific color. The σ images in figure 3 correspond to the σ vectors who's smoothed classifiers are plotted the figure 1.

The MNIST, CIFAR-10, and KITTI images are essentially as expected. For MNIST the model largely ignores the black background on the borders. For CIFAR-10 and KITTI the model ignores certain corners of the image more than it does the interior. However, Fashion-MNIST provides a fascinating result. The left side and bottom right corner have very low variances, whereas the center and top right have high variances. This indicates that the model mostly relies on the bottom right corner and left side of the image. After examining the dataset, we suspect that the model is using the bottom right corner to identify shoes via their heel shape, and the left side to identify clothing via a sleeve or pant leg. Whether this is truly the case or not, it is very interesting to observe that the model effectively does not rely on the center of the image nearly as much as it does specific edges and corners.



(a) MNIST ($\sigma_{\text{init}} = 0.8$) (b) MNIST ($\sigma_{\text{init}} = 1.2$) (c) Fashion-MNIST ($\sigma_{\text{init}} = 0.7$) (d) Fashion-MNIST ($\sigma_{\text{init}} = 1.5$)

(e) CIFAR10 ($\sigma_{\text{init}} = 0.325$) (f) CIFAR10 ($\sigma_{\text{init}} = 0.35$) (g) KITTI ($\sigma_{\text{init}} = 0.05$) (h) KITTI ($\sigma_{\text{init}} = 0.075$)

Figure 3: σ images. Each image is a trained σ vector with its elements linearly normalized to fit the entire image color space. (Darker is a lower value, brighter is a higher value.) Each image corresponds to a smoothed classifier trained on the dataset listed in the sub-caption. The given σ value indicates what constant the σ vector was initialized to during training. Each image here corresponds to a smoothed classifier shown in figure 1.

6 Conclusion

This work furthers the research community's efforts to find large formal verification bounds that are computable in tractable times. The modifications provided here allow for randomized smoothing methods to have more flexibility in which input features are perturbed more aggressively than others, allowing for much more optimal robustness bounds to be obtained. It is our hope that this method can be used to achieve larger certification and better robustness for machine learning models where safety is critical.

Furthermore, we have introduced a method to find optimal variance matrices for a smoothed classifier via backpropagation and gradient descent. And as an interesting side effect, these optimized matrices help give insight into the models underlying the smoothed classifiers.

References

[1] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017.

[2] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[3] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, JinFeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2019.

[4] Ping-Yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studor, and Tom Goldstein. Certified defenses for adversarial patches. *arXiv preprint arXiv:2003.06693*, 2020.

[5] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *CoRR*, abs/1902.02918, 2019.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] Krishnamurthy (Dj) Dvijotham, Jamie Hayes, Borja Balle, Zico Kolter, Chongli Qin, Andras Gyorgy, Kai Xiao, Sven Gowal, and Pushmeet Kohli. A framework for robustness certification of smoothed classifiers using f-divergences. In *ICLR*, 2020.

[8] Ruediger Ehlers. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. *arXiv e-prints*, page arXiv:1705.01320, May 2017.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

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

[11] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy A. Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *CoRR*, abs/1810.12715, 2018.

[12] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *Lecture Notes in Computer Science*, page 97–117, 2017.

[13] J. Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[15] Aounon Kumar, Alexander Levine, Tom Goldstein, and Soheil Feizi. Curse of dimensionality on randomized smoothing for certifiable robustness. In *ICML*, 2020.

[16] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[17] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy, 2019.

[18] Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi Jaakkola. Tight certificates of adversarial robustness for randomly smoothed classifiers. In *Advances in Neural Information Processing Systems*, pages 4911–4922, 2019.

[19] Alexander Levine and Soheil Feizi. Robustness certificates for sparse adversarial attacks by randomized ablation. In *AAAI*, 2020.

[20] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise, 2019.

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[22] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,

Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[24] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya P. Razenshteyn, and Sébastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *CoRR*, abs/1906.04584, 2019.

[25] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10802–10813. Curran Associates, Inc., 2018.

[26] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL):41:1–41:30, January 2019.

[27] Sahil Singla and Soheil Feizi. Second-order provable defenses against adversarial attacks. In *ICML*, 2020.

[28] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

[29] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[30] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[31] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Mixtrain: Scalable training of formally robust neural networks. *arXiv preprint arXiv:1811.02625*, 2018.

[32] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. *CoRR*, abs/1809.08098, 2018.

[33] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018.

[34] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[35] Runtian Zhai, Chen Dan, Di He, Huan Zhang, Boqing Gong, Pradeep Ravikumar, Cho-Jui Hsieh, and Liwei Wang. Macer: Attack-free and scalable robust training via maximizing certified radius. In *International Conference on Learning Representations*, 2020.

[36] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*, 2019.

[37] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *CoRR*, abs/1811.00866, 2018.

A Theorem Proofs

A.1 Proof of Theorem 3.1

Note that much of this work will be based off proofs by [5], and as such the work will look similar. (Some parts will be identical as they require no modification, but we included them to try to be as self-contained and clear as possible.) We will note which of their lemmas or theorems correspond to the ones we introduce here. Please refer to their paper for all lemmas and theorem numbers we mention in this section.

First we need to introduce a lemma that corresponds with lemma 4 of [5].

Lemma A.1 *Let $X \sim \mathcal{N}(x, \Sigma)$ and $Y \sim \mathcal{N}(x+\delta, \Sigma)$. Let $h : \mathbb{R}^d \to \{0, 1\}$ be any deterministic or random function. Then:*

1. *If $S = \{z \in \mathbb{R}^d : \lambda^T z \leq \beta\}$ for some β and $\mathbb{P}(h(X) = 1) \geq P(X \in S)$, then $\mathbb{P}(h(Y) = 1) \geq P(Y \in S)$*
2. *If $S = \{z \in \mathbb{R}^d : \lambda^T z \geq \beta\}$ for some β and $\mathbb{P}(h(X) = 1) \leq P(X \in S)$, then $\mathbb{P}(h(Y) = 1) \leq P(Y \in S)$*

Where $\lambda = \delta \oslash \sigma^{\circ 2}$. ($\oslash$ is hadamard division, $\cdot^{\circ 2}$ is element-wise square.)

Proof. This lemma is the special case of Lemma 3 in [5] when X and Y are Gaussians with means x and $x + \delta$. By Lemma 3 in [5] it suffices to simply show that for any β, there is some $t > 0$ for which:

$$\{z : \delta^T z \leq \beta\} = \{z : \frac{\mu_Y(z)}{\mu_X(z)} \leq t\} \quad \text{and} \quad \{z : \delta^T z \geq \beta\} = \{z : \frac{\mu_Y(z)}{\mu_X(z)} \geq t\}$$

The likelihood ratio for this choice of X and Y turns out to be:

$$\begin{aligned}
\frac{\mu_Y(z)}{\mu_X(z)} &= \frac{\exp(\sum_{i=1}^d \frac{-1}{2\sigma_i^2}(z_i - (x_i + \delta_i))^2)}{\exp(\sum_{i=1}^d \frac{-1}{2\sigma_i^2}(z_i - x_i)^2)} \\
&= \exp(\sum_{i=1}^d \frac{1}{2\sigma_i^2}(2z_i\delta_i - \delta_i^2 - 2x_i\delta_i)) \\
&= \exp(\lambda^\top z + b)
\end{aligned}$$

Where $\lambda = \delta \oslash \sigma^{\circ 2}$ and $b = \frac{-1}{2}\|\delta \odot \lambda\|_1 - \|x_i \odot \lambda\|_1$. ($\oslash$ is hadamard division, $\cdot^{\circ 2}$ is element-wise square.)

Therefore, given any β we may take $t = \exp(\lambda^\top z + b)$, noticing that:

$$\begin{aligned}
\lambda^\top z \leq \beta &\iff \exp(\lambda^\top z + b) \leq t \\
\lambda^\top z \geq \beta &\iff \exp(\lambda^\top z + b) \geq t
\end{aligned}$$

Finally, we can prove Theorem 3.1.

To show that $g(x + \delta) = c_A$, it follows from the definition of g that we need to show that:

$$\mathbb{P}(f(x + \delta + \epsilon) = c_a) \geq \max_{c_B \neq c_a} \mathbb{P}(f(x + \delta + \epsilon) = c_B)$$

We will show that $\mathbb{P}(f(x + \delta + \epsilon) = c_a) \geq \mathbb{P}(f(x + \delta + \epsilon) = c_B)$ for every class $c_B \neq c_A$. Fix one class c_B without loss of generality. And for convenience we'll define the random variables:

$$\begin{aligned}
X &:= x + \epsilon = \mathcal{N}(x, \Sigma) \\
Y &:= x + \delta + \epsilon = \mathcal{N}(x + \delta, \Sigma)
\end{aligned}$$

By Definition 3.2 we know that:

$$\begin{aligned}
\mathbb{P}(f(X) = c_a) &\geq \underline{p_A} \\
\mathbb{P}(f(X) = c_B) &\leq \overline{p_B}
\end{aligned}$$

and our goal is to show that

$$\mathbb{P}(f(Y) = c_a) > \mathbb{P}(f(Y) = c_B)$$

To prove this, we define the half-spaces:

$$A := \{z : \lambda^\top (z - x) \leq \|\sigma \odot \lambda\| \Phi^{-1}(\underline{p_A})\}$$
$$B := \{z : \lambda^\top (z - x) \geq \|\sigma \odot \lambda\| \Phi^{-1}(1 - \overline{p_B})\}$$

Where $\lambda = \delta \oslash \sigma^{\circ 2}$, as per Lemma A.1.

Algebra (deferred to the end, see the "Deferred Algebra" section) shows that $\mathbb{P}(X \in A) = \underline{p_A}$. Therefore, combined with Definition 3.2 we know that $\mathbb{P}(f(X) = c_a) \geq \mathbb{P}(X \in A)$. Hence we may apply Lemma A.1 with $h := \mathbf{1}[f(z) = c_a]$ to conclude:

$$\mathbb{P}(f(Y) = c_A) \geq \mathbb{P}(Y \in A)$$

Similarly, algebra shows that $\mathbb{P}(X \in B) = \overline{p_B}$. Therefore, combined with Definition 3.2 we know that $\mathbb{P}(f(X) = c_b) \leq \mathbb{P}(X \in B)$. Hence we may apply Lemma A.1 with $h := \mathbf{1}[f(z) = c_B]$ to conclude:

$$\mathbb{P}(f(Y) = c_B) \leq \mathbb{P}(Y \in B)$$

Now all we need to do is show that $\mathbb{P}(Y \in A) > \mathbb{P}(Y \in B)$, as this produces a chain of inequalities:

$$\mathbb{P}(f(Y) = c_A) \geq \mathbb{P}(Y \in A) > \mathbb{P}(Y \in B) \geq \mathbb{P}(f(Y) = c_B)$$

We can compute that (also shown in the "Deferred Algebra" section):

$$\mathbb{P}(Y \in A) = \Phi(\Phi^{-1}(\underline{p_A}) - \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|})$$
$$\mathbb{P}(Y \in B) = \Phi(\Phi^{-1}(\overline{p_B}) + \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|})$$

Where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors.

So now all that is left is algebra to determine when $\mathbb{P}(Y \in A) > \mathbb{P}(Y \in B)$, and $g(x + \delta) = c_a$ for all δ vectors that satisfy this condition.

$$\begin{aligned}
\mathbb{P}(Y \in B) &< \mathbb{P}(Y \in A) \\
\Phi(\Phi^{-1}(\overline{p_B}) + \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|}) &< \Phi(\Phi^{-1}(\underline{p_A}) - \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|}) \\
2\frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|} &< \Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B}) \\
\langle \lambda, \delta \rangle &< \frac{\|\sigma \odot \lambda\|}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) \\
\langle \delta, \delta \oslash \sigma^{\circ 2} \rangle &< \frac{\|\delta \oslash \sigma\|}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) \\
\|\delta \oslash \sigma\|^2 &< \frac{\|\delta \oslash \sigma\|}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) \\
\|\delta \oslash \sigma\| &< \frac{1}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B}))
\end{aligned}$$

A.2 Deferred Algebra

Claim. $\mathbb{P}(X \in A) = \underline{p_A}$

Proof. Recall that σ is a vector of standard deviations for each element, and Σ is a diagonal matrix with $\sigma^{\circ 2}$ as the diagonal. Additionally, note that $X \sim \mathcal{N}(x, \Sigma)$ and $A := \{z : \lambda^\top (z - x) \leq$

$\|\sigma \odot \lambda\|\Phi^{-1}(\underline{p_A})\}$

$$\begin{aligned}
\mathbb{P}(X \in A) &= \mathbb{P}(\lambda^\top(X - x) \leq \|\sigma \odot \lambda\|\Phi^{-1}(\underline{p_A})) \\
&= \mathbb{P}(\lambda^\top \mathcal{N}(0, \Sigma) \leq \|\sigma \odot \lambda\|\Phi^{-1}(\underline{p_A})) \\
&= \mathbb{P}(\|\sigma \odot \lambda\|\mathcal{N}(0, 1) \leq \|\sigma \odot \lambda\|\Phi^{-1}(\underline{p_A})) \\
&= \mathbb{P}(\mathcal{N}(0, 1) \leq \Phi^{-1}(\underline{p_A})) \\
&= \Phi(\Phi^{-1}(\underline{p_A})) \\
&= \underline{p_A}
\end{aligned}$$

Claim. $\mathbb{P}(X \in B) = \overline{p_B}$

Proof. Recall that σ is a vector of standard deviations for each element, and Σ is a diagonal matrix with $\sigma^{\circ 2}$ as the diagonal. Additionally, note that $X \sim \mathcal{N}(x, \Sigma)$ and $B := \{z : \lambda^\top(z - x) \geq \|\sigma \odot \lambda\|\Phi^{-1}(1 - \overline{p_B})\}$

$$\begin{aligned}
\mathbb{P}(X \in B) &= \mathbb{P}(\lambda^\top(X - x) \geq \|\sigma \odot \lambda\|\Phi^{-1}(1 - \overline{p_B})) \\
&= \mathbb{P}(\lambda^\top \mathcal{N}(0, \Sigma) \geq \|\sigma \odot \lambda\|\Phi^{-1}(1 - \overline{p_B})) \\
&= \mathbb{P}(\|\sigma \odot \lambda\|\mathcal{N}(0, 1) \geq \|\sigma \odot \lambda\|\Phi^{-1}(1 - \overline{p_B})) \\
&= \mathbb{P}(\mathcal{N}(0, 1) \geq \Phi^{-1}(1 - \overline{p_B})) \\
&= 1 - \Phi(\Phi^{-1}(1 - \overline{p_B})) \\
&= \overline{p_B}
\end{aligned}$$

Claim. $\mathbb{P}(Y \in A) = \Phi(\Phi^{-1}(\underline{p_A}) - \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|})$

Proof. Recall that σ is a vector of standard deviations for each element, and Σ is a diagonal matrix with $\sigma^{\circ 2}$ as the diagonal. Additionally, note that $Y \sim \mathcal{N}(x + \delta, \Sigma)$ and $A := \{z : \lambda^\top(z - x) \leq \|\sigma \odot \lambda\|\Phi^{-1}(\underline{p_A})\}$

$$\begin{aligned}
\mathbb{P}(Y \in A) &= \mathbb{P}(\lambda^\top(Y - x) \leq \|\sigma \odot \lambda\|\Phi^{-1}(\underline{p_A})) \\
&= \mathbb{P}(\lambda^\top \mathcal{N}(0, \Sigma) + \langle \lambda, \delta \rangle \leq \|\sigma \odot \lambda\|\Phi^{-1}(\underline{p_A})) \\
&= \mathbb{P}(\|\sigma \odot \lambda\|\mathcal{N}(0, 1) \leq \|\sigma \odot \lambda\|\Phi^{-1}(\underline{p_A}) - \langle \lambda, \delta \rangle) \\
&= \mathbb{P}(\mathcal{N}(0, 1) \leq \Phi^{-1}(\underline{p_A}) - \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|}) \\
&= \Phi(\Phi^{-1}(\underline{p_A}) - \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|})
\end{aligned}$$

Claim. $\mathbb{P}(Y \in B) = \Phi(\Phi^{-1}(\overline{p_B}) + \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|})$

Proof. Recall that σ is a vector of standard deviations for each element, and Σ is a diagonal matrix with $\sigma^{\circ 2}$ as the diagonal. Additionally, note that $Y \sim \mathcal{N}(x + \delta, \Sigma)$ and $B := \{z : \lambda^\top(z - x) \geq \|\sigma \odot \lambda\|\Phi^{-1}(1 - \overline{p_B})\}$

$$\begin{aligned}
\mathbb{P}(Y \in B) &= \mathbb{P}(\lambda^\top(Y - x) \geq \|\sigma \odot \lambda\|\Phi^{-1}(1 - \overline{p_B})) \\
&= \mathbb{P}(\lambda^\top \mathcal{N}(0, \Sigma) + \langle \lambda, \delta \rangle \geq \|\sigma \odot \lambda\|\Phi^{-1}(1 - \overline{p_B})) \\
&= \mathbb{P}(\|\sigma \odot \lambda\|\mathcal{N}(0, 1) \geq \|\sigma \odot \lambda\|\Phi^{-1}(1 - \overline{p_B}) - \langle \lambda, \delta \rangle) \\
&= \mathbb{P}(\mathcal{N}(0, 1) \geq \Phi^{-1}(1 - \overline{p_B}) - \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|}) \\
&= \mathbb{P}(\mathcal{N}(0, 1) \leq \Phi^{-1}(\overline{p_B}) + \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|}) \\
&= \Phi(\Phi^{-1}(\overline{p_B}) + \frac{\langle \lambda, \delta \rangle}{\|\sigma \odot \lambda\|})
\end{aligned}$$

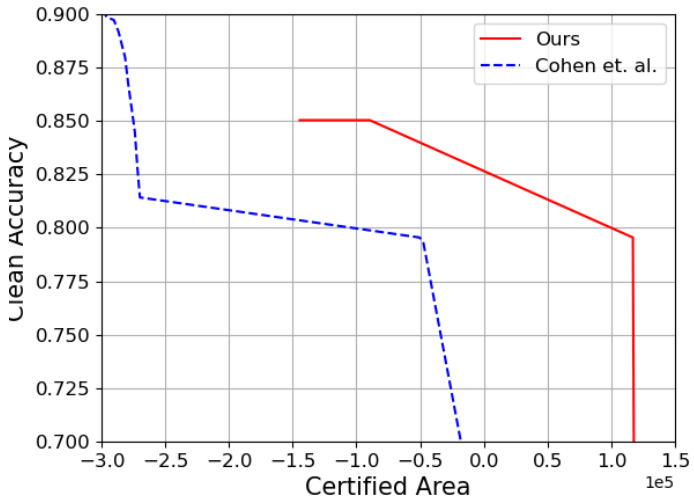


Figure 4: KITTI Clean accuracy vs. certified area plot. Note that the KITTI dataset we generated (by taking the maximum-occurring object in the image as the full image label) is heavily biased towards a single class. As such, any accuracy under approximately 80% is a trivial model (it can be achieved by always predicting a single class). As there is only a narrow window of accuracies that are non-trivial to achieve, only a few initialized σ's produce useful randomized smothers, making the clean accuracy plot not particularly informative for the accuracy-certified area trade-off. As such, we opted to put it here in the appendix instead.

B Experiments

For all runs, we used a batch size of 128 for MNIST and Fashion-MNIST, 32 for CIFAR-10, and 2 for ImageNet plus KITTI. This batch size refers to the number of data points we predicted with the smoothed classifier per optimization step. Because the smoothed classifier uses n samples from the base classifier per prediction, this batch size is essentially a multiplicative factor with n for the number of base classifier samples used in an optimization step. These batch sizes were essentially chosen to be as large as possible without the GPU running out of memory.

For each epoch, the approximate average run time on our system is as follows:

- MNIST : 5 minutes
- Fashion-MNIST : 5 minutes
- CIFAR-10 : 1 hour 20 minutes
- ImageNet : 33 minutes
- KITTI : 24 minutes

This runtime includes optimizing over the entire train dataset, plus gathering accuracy and certified area metrics on the entire test set.

B.1 Clean Accuracy vs. Certified Area Plots

When producing these plots, as detailed in the main section we would re-initialize σ with a different-magnitude constant in all elements and then optimize for a specific number of epochs. For all models, we started with a near-zero constant and continued until the model performance deteriorated to random chance. For MNIST and Fashion-MNIST, we would try initialization constants at an interval of 0.1, and optimize for 5 sub-epochs at every initialization. For CIFAR-10, ImageNet, and KITTI models, we tried initialization constants at an interval of 0.01 and optimized for 10 sub-epochs. For all models we used an Adam optimizer with a learning rate of 0.001 for MNIST, Fashion-MNIST,

and Imagenet, and 0.005 for CIFAR-10 and KITTI. We also multiplied the learning rate by 0.5 after every sub-epoch for MNIST models, and multiplied it by 0.8 for all other models.

To filter out any optimized σ vectors that were sub-optimal compared to others we were able to produce at the end of each sub-epoch, we applied a "blanket" algorithm. This algorithm would sort the points by accuracy, and starting from the highest accuracy it would only add a point if it was outside the certified area range that all points already added provided. Essentially this algorithm removes all points that have a lower accuracy than a point on both the left (lower certified area) and right (higher certified area) of it. We also tried filtering by removing all points that were under a line produced by any other two points, but this ended up creating misleading plots. (If there was a long tail at the end of a plot, it would remove all points between the peak and tail end of the slope.) The exact filtering code can be found in our repository.