Deterministic Gaussian Averaged Neural Networks

Ryan Campbell[*1] **Chris Finlay**[1] **Adam M Oberman**[1]

[1]Department of Mathematics and Statistics, McGill University, Montréal, Canada

Abstract

We present a deterministic method to compute the Gaussian average of neural networks used in regression and classification. Our method is based on an equivalence between training with a particular regularized loss, and the expected values of Gaussian averages. We use this equivalence to certify models which perform well on clean data but are not robust to adversarial perturbations. In terms of certified accuracy and adversarial robustness, our method is comparable to known stochastic methods such as randomized smoothing, but requires only a single model evaluation during inference.

1 Introduction

Neural networks are very accurate on image classification tasks, but they are vulnerable to adversarial perturbations, i.e. small changes to the model input leading to misclassification (Szegedy et al., 2014). Adversarial training Madry et al. (2018) improves robustness, but does not solve the problem completely. A complementary approach is *certification*: find the greatest amount of perturbation that can be added to an input example before the model's performance is compromised (Lécuyer et al., 2019; Raghunathan et al., 2018; Cohen et al., 2019). This means that, given a model f and an input x, we wish to certify a minimum distance so that the classification of $f(x+\eta)$ is constant for all perturbations below the certified distance.

One promising approach to certification is to certify models within a given Gaussian region (Cohen et al., 2019; Salman et al., 2019). That is, a model f should satisfy

$$f(x) \approx \mathbb{E}_\eta \left[f(x+\eta) \right] \tag{1}$$

where $\eta \sim \mathcal{N}(0, \sigma^2 I)$ and x is a given example. While this is not easily attainable in practice, it is possible to obtain a new "smoothed" model v that is the expected Gaussian average of our initial model f at a given input example x,

$$v(x) \approx \mathbb{E}_\eta \left[f(x+\eta) \right] \tag{2}$$

The works of Cohen et al. and Salman et al. both obtain this Gaussian smoothed model (2) stochastically. The idea is that by defining the classification of the stochastic model to be the mode (most popular prediction) of the classifications given by ensembles $f(x+\eta_1), f(x+\eta_2), \ldots$, we achieve a more robust ensemble. While ensemble models are generally more robust, the main advantage of the Gaussian ensemble is that we can obtain certified bounds based on probabilistic arguments. However, like all ensemble models, it requires multiple inferences, which can be undesirable when computation is costly.

In this work we present a *deterministic* model whose outputs are equivalent to the outputs of the Gaussian smoothed model $v(x)$ given by (2). While it is generally not possible to

[*] Correspondance to `ryan.campbell2@mail.mcgill.ca`

Preprint. Under review.

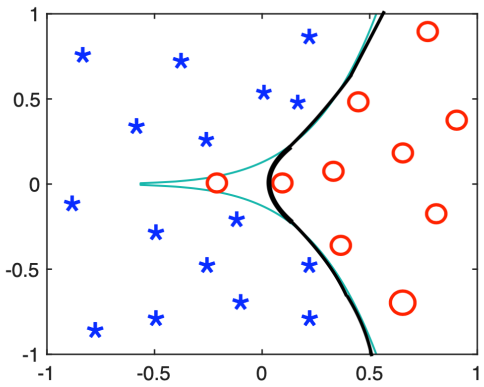


Figure 1: *Illustration of gradient regularization in the binary classification setting. The lighter line represents classification boundary for original model with large gradients, and the darker line represents classification boundary of the smoothed model. The symbols indicate the classification by the original model: a single red circle is very close to many blue stars. The smoothed model has a smoother classification boundary which flips the classification of the outlier.*

capture an ensemble with a single model, in the special case of Gaussian smoothing, we can do so, due to specific properties of Gaussian averaging. Our deterministic smoothed model arises by training a regularized loss, which we call HeatSmoothing. Our work relies on the insight, known since the early days of neural networks (Bishop, 1995; LeCun et al., 1998), that gradient regularization is equivalent to Gaussian smoothing.

Formally this is stated as follows.

Theorem 1. *(Bishop, 1995) Training a model using the quadratic loss, with added Gaussian noise of variance σ^2 to the inputs, is equivalent to training with*

$$\mathbb{E}_x \left[\|f(x) - y\|^2 + \sigma^2 \|\nabla f(x)\|^2 \right] \tag{3}$$

up to higher order terms.

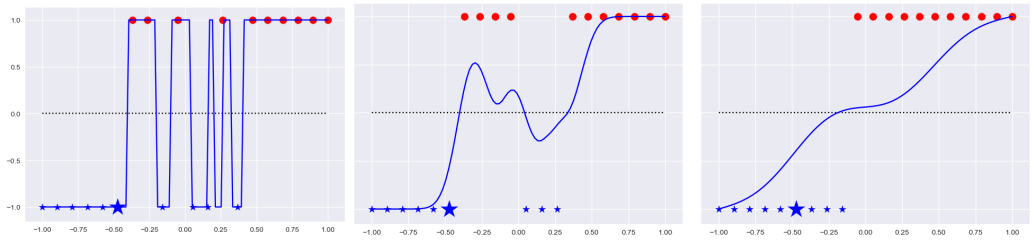
The proof is deferred to Appendix A; a similar result is available with the cross-entropy loss.

The result from Theorem 1 gives an equivalence which is normally used to go from models augmented with Gaussian noise to regularized models. In our case, we use the result in the other direction: we train a regularized model in order to produce a model which is equivalent to evaluating with noise. In practice, this means that rather than adding noise to regularize models for certifiable robustness, we explicitly perform a type of gradient regularization, *in order to produce a model which performs as if Gaussian noise was added.* See Figure 1 for an illustration of the effect of this gradient regularization.

To our knowledge, our method is the first deterministic Gaussian smoothing certification technique. The main appeal of our approach is a large decrease in the number of function evaluations for computing certifiably robust models, and a corresponding decrease in compute time at inference. Rather than stochastically sampling many times from a Gaussian at inference time, our method is certifiably robust with only a single model query. Our improvement in inference time is achieved by iterative retraining of the model so that model predictions are equal to the local Gaussian average of the model. Although this iterative retraining procedure can be time consuming, we argue that trading inference time for training time will be beneficial in many use case scenarios.

2 Related work

The issue of adversarial vulnerability arose in the works of Szegedy et al. (2014) and Goodfellow et al. (2015), and has spawned a vast body of research. In terms of certification, early work by Cheng et al. (2017) provided a method of computing maximum perturbation bounds for neural networks, and reduced to solving a mixed integer optimization problem.



(a) f^0 (b) f^1 (c) f^5

Figure 2: *Illustration of performing the iterative model update (4) for 5 timesteps in the binary classification setting. The dashed black line represents our decision boundary. The blue line represents our current classification model. The blue stars and red circles represent our predicted classes using the current model iteration. Consider the datapoint at $x = -0.5$. In the initial model f^0, the adversarial distance is ≈ 0.10. In model f^5, the adversarial distance is increased to ≈ 0.35.*

Weng et al. (2018a) introduced non-trivial robustness bounds for fully connected networks, and provided tight robustness bounds at low computational cost. Weng et al. (2018b) proposed a metric that has theoretical grounding based on Lipschitz continuity of the classifier model and is scaleable to state-of-the-art ImageNet neural network classifiers. Zhang et al. (2018) proposed a general framework to certify neural networks based on linear and quadratic bounding techniques on the activation functions, which is more flexible than its predecessors.

Training a neural network with Gaussian noise has been shown to be equivalent to gradient regularization (Bishop, 1995). This helps improve robustness of models; however, there have been recent using noise for certification purposes. Lécuyer et al. (2019) first considered adding random Gaussian noise as a certifiable defense in a method called *PixelDP*. In their method, they take a known neural network architecture and add a layer of random noise to make the model's output random. The expected classification is in turn more robust to adversarial perturbations. Furthermore, their defense is a certified defense, meaning they provide a lower bound on the amount of adversarial perturbations for which their defense will always work. In a following work, Li et al. (2018) provided a defense with improved certified robustness. The certification guarantees given in these two papers are loose, meaning the defended model will always be more robust than the certification bound indicates.

In contrast, Cohen et al. (2019) provided a defense utilizing randomized Gaussian smoothing that leads to *tight* robustness guarantees under the ℓ_2 norm. Moreover Cohen et al. used Monte Carlo sampling to compute the radius in which a model's prediction is unchanged; we refer to this method as RandomizedSmoothing. In work building on Cohen et al., Salman et al. (2019) developed an adversarial training framework called SmoothAdv and defined a Lipschitz constant of averaged models. Yang et al. (2020) generalize previous randomized smoothing methods by providing robustness guarantees in the ℓ_1, ℓ_2, and ℓ_∞ norms for smoothing with several non-Gaussian distributions.

3 Deterministic Smoothing

Suppose we are given a dataset consisting of paired samples $(x, y) \in \mathcal{X} \times \mathcal{Y}$ where x is an example with corresponding true classification y. The supervised learning approach trains a model $f : \mathcal{X} \longrightarrow \mathbb{R}^{Nc}$ which maps images to a vector whose length equals the number of classes. Suppose f is the initial model, and let v be the averaged model given by equation (2). Cohen et al. (2019) find a Gaussian smoothed classification model v by sampling $\eta \sim \mathcal{N}(0, \sigma^2 I)$ independently n times, performing n classifications, and then computing the most popular classification. In the randomized smoothing method, the initial model f is trained on data which is augmented with Gaussian noise to improve accuracy on noisy images.

We take a different approach to Gaussian smoothing. Starting from an accurate pretrained model f, we now discard the training labels, and iteratively retrain a new model, f^k using

a quadratic loss between the present iterate f^k and the new model's predictions, with an additional gradient regularization term. We have found that discarding the original one-hot labels and instead using model predictions helps make the model smoother.

To be precise, our iterative retraining procedure is given by

$$f^{k+1} = \underset{v}{\operatorname{argmin}}\ \mathbb{E}_x \left[\frac{1}{2} \left\| v(x) - f^k(x) \right\|_2^2 + \lambda \frac{h\sigma^2}{2} \left\| \nabla_x v(x) \right\|_2^2 \right] \tag{4}$$

We set $f^0(x) = f(x)$, the initial model, and generate an iterative sequence of retrained models for $k = 0, \ldots, n_T - 1$, where n_T is the number of retraining steps. In (4), $h = 1/n_T$ and $\lambda > 0$ is a scaling hyperparameter. At each retraining step, the retrained model is progressively smoothed by gradient regularization, while being encouraged to remain faithful to the underlying base model via the quadratic loss. This is illustrated in Figure 2.

3.1 Algorithmic details

Note that the $\|\nabla_x v(x)\|_2^2$ term in (4) requires the computation of a Jacobian matrix norm. In high dimensions this is computationally expensive. To approximate this term, we make use of the *Johnson-Lindenstrauss lemma* (Johnson and Lindenstrauss, 1984; Vempala, 2005) followed by the finite difference approximation from Finlay and Oberman (2019). We are able to approximate $\|\nabla_x v(x)\|_2^2$ by taking the average of the product of the Jacobian matrix and Gaussian noise vectors. Jacobian-vector products can be easily computed via reverse mode automatic differentiation, by moving the noise vector w inside:

$$w \cdot (\nabla_x v(x)) = \nabla_x \left(w \cdot v(x) \right) \tag{5}$$

Further computation expense is reduced by using finite-differences to approximate the norm of the gradient. Once the finite-difference is computed, we detach this term from the automatic differentiation computation graph, further speeding training. More details of our implementation of these approximation techniques are presented in Appendix C.

We have found that early on in training, the distance value $\frac{1}{2} \left\| v(x) - f^k(x) \right\|_2^2$ may be far greater than the $\frac{h\sigma^2}{2} \left\| \nabla_x v(x) \right\|_2^2$ term. We introduce a scaling term $\lambda > 0$ to correct this. We perform the training minimization of (4) n_T times for $k = 0, \ldots, n_T - 1$, for a set number epochs at each timestep k. The pseudo-code[1] for our neural network weight update is given by Algorithm 1.

3.2 Certified Radii

To assess how well our model approximates the Gaussian average of the initial model, we compute the certified ℓ_2 radius introduced in Cohen et al. (2019). A larger radius implies a better approximation of the Gaussian average of the initial model. We compare our models with stochastically averaged models via *certified accuracy*. This is the fraction of the test set which a model correctly classifies at a given radius while ignoring abstained classifications. Throughout, we always use the same σ value for certification as for training.

In conjunction with the certification technique of Cohen et al., we also provide the following theorem, which describes a bound based on the Lipschitz constant of a Gaussian averaged model. We refer to this bound as the L-bound.

Theorem 2 (L-bound). *Suppose v is the convolution (average) of $f : \mathbb{R}^d \to [0,1]^{Nc}$ with a Gaussian kernel of variance $\sigma^2 I$,*

$$v(x) = \left(f * \mathcal{N}(0, \sigma^2 I) \right)(x)$$

Then any perturbation δ which results in a change of rank of the k-th component of $v(x)$ must have norm bounded as follows:

$$\|\delta\|_2 \geq \sigma(\pi/2)^{1/2} (v(x)_{(k)} - v(x)_{(k+1)}) \tag{6}$$

where $v(x)_{(i)}$ is the i^{th} largest value in the vector $v(x) \in [0,1]^{Nc}$.

See Appendix D for proof. This bound is equally applicable to deterministic or stochastically averaged models. In stochastically averaged models $v(x)$ is replaced by $\mathbb{E}_{\eta \sim \mathcal{N}(0,\sigma^2 I)} [f(x+\eta)]$.

[1] Code is publicly available at https://github.com/ryancampbell514/HeatSmoothing.

Algorithm 1: HEATSMOOTHING Neural Network Weight Update

Input : Minibatch of input examples $\boldsymbol{x}^{(mb)} = \left(x^{(1)}, \ldots, x^{(Nb)}\right)$
A model v set to "train" mode
Current model f^k set to "eval" mode
σ, standard deviation of Gaussian smoothing
κ, number of Gaussian noise replications (default= 10)
δ, finite difference step-size (default= 0.1)
n_T, total number of timesteps being executed (default= 5)
$\lambda \geq 1$, scaling hyperparameter

Update : learning-rate according to a pre-defined scheduler.

for $i \in \{1, \ldots Nb\}$ **do**

  Compute : $v(x^{(i)}), f^k(x^{(i)}) \in \mathbb{R}^{Nc}$

    $J_i = \frac{1}{2}\left\|v(x^{(i)}) - f^k(x^{(i)})\right\|_2^2 \in \mathbb{R}$

    for $j \in \{1, \ldots \kappa\}$ **do**

      Generate $w = \frac{1}{\sqrt{Nc}}(w_1, \ldots, w_{Nc})$, $w_1, \ldots, w_{Nc} \in \mathcal{N}(0,1)$

      Compute l via (19), detach $x^{(i)}$ from the computation graph

      $J_i \leftarrow J_i + \lambda \frac{\sigma^2}{2n_T}\left(\frac{w \cdot v(x^{(i)} + \delta l) - w \cdot v(x^{(i)})}{\delta}\right)^2$

    end

  $J \leftarrow \frac{1}{Nb}\sum_{i=1}^{Nb} J_i$

end

Update the weights of v by running backpropagation on J at the current learning rate.

3.3 Theoretical Details

We appeal to partial differential equations (PDE) theory for explaining the equivalence between gradient regularization and Gaussian convolution (averaging) of the model[2]. The idea is that the gradient term which appears in the loss leads to a smoothing of the new function (model). The fact that the exact form of the smoothing corresponds to Gaussian convolution is a mathematical results which can be interpreted probabilistically or using techniques from analysis. Briefly, we detail the link as follows.

Einstein (1906) showed that the function value of an averaged model under Brownian motion is related to the heat equation (a PDE); the theory of stochastic differential equations makes this rigorous (Karatzas and Shreve, 1998). Moreover, solutions of the heat equation are given by Gaussian convolution with the original model. Crucially, in addition solutions of the heat equation can be interpreted as iterations of a regularized loss problem (called a variational energy) like that of Equation 4. The minimizer of this variational energy (4) satisfies an equation which is formally equivalent to the heat equation (Gelfand et al., 2000). Thus, taking these facts together, we see that a few steps of the minimization of the loss in (4) yield a model which approximately satisfies the heat equation, and corresponds to a model smoothed by Gaussian convolution. See Figure 1 for an illustration of a few steps of the training procedure. This result is summarized in the following theorem.

Theorem 3. *(Strauss, 2007) Let f be a bounded function, $x \in \mathbb{R}^d$, and $\eta \sim \mathcal{N}\left(0, \sigma^2 I\right)$. Then the following are equivalent:*

1. $\mathbb{E}_\eta\left[f(x+\eta)\right]$, *the expected value of Gaussian averages of f at x.*
2. $\left(f * \mathcal{N}(0, \sigma^2 I)\right)(x)$, *the convolution of f with the density of the $\mathcal{N}(0, \sigma^2 I)$ distribution evaluated at x.*

[2]We sometimes interchange the terms Gaussian averaging and Gaussian convolution; they are equivalent, as shown in Theorem 3.

3. The functions defined in 1. and 2. are solutions of the heat equation,

$$\frac{\partial}{\partial t} f(x,t) = \frac{\sigma^2}{2} \Delta_x f(x,t) \tag{7}$$

at time $t = 1$, with initial condition $f(x,0) = f(x)$.

In Appendix B, we use Theorem 3 to show the equivalence of training with noise and iteratively training (4).

3.4 Adversarial Attacks

To test how robust our model is to adversarial examples, we calculate the minimum ℓ_2 adversarial via our L-bound and we attack our models using the *projected gradient descent (PGD)* (Kurakin et al., 2017; Madry et al., 2018) and *decoupled direction and norm (DDN)* (Rony et al., 2019) methods. These attacks are chosen because there is a specific way they can be applied to stochastically averaged models (Salman et al., 2019). In both attacks, it is standard to take the step

$$g = \alpha \frac{\nabla_{\delta_t} L\left(f(x+\delta_t), y\right)}{\|\nabla_{\delta_t} L\left(f(x+\delta_t), y\right)\|_2} \tag{8}$$

in the iterative algorithm. Here, x is an input example with corresponding true class y; δ_t denotes the adversarial perturbation at its current iteration; L denotes the cross-entropy Loss function (or KL Divergence); ε is the maximum perturbation allowed; and α is the step-size. In the stochastically averaged model setting, the step is given by

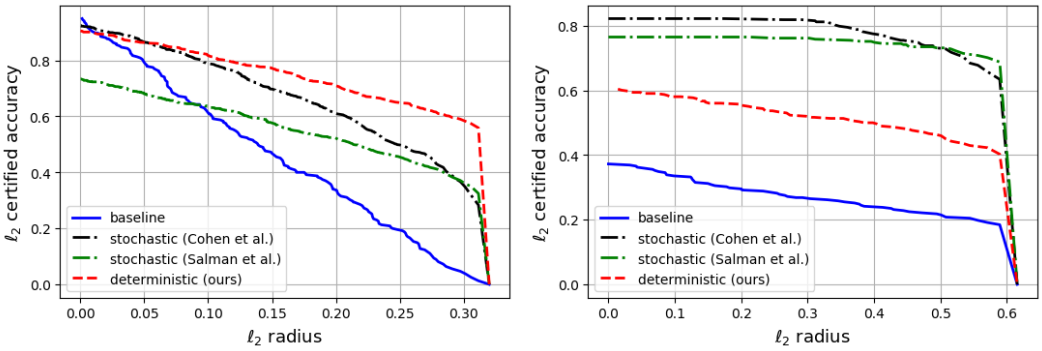
$$g_n = \alpha \frac{\sum_{i=1}^{n} \nabla_{\delta_t} L\left(f(x+\delta_t+\eta_i), y\right)}{\left\| \sum_{i=1}^{n} \nabla_{\delta_t} L\left(f(x+\delta_t+\eta_i), y\right) \right\|_2} \tag{9}$$

where $\eta_1, \ldots, \eta_n \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2 I)$. For our deterministically averaged models, we implement the update (8). This is because our models are deterministic, meaning there is no need to sample noise at evaluation time. For stochastically averaged models (Cohen et al., 2019; Salman et al., 2019), we implement the update (9).

4 Experiments & Results

4.1 CIFAR-10

We begin by testing our method on the CIFAR-10 dataset (Krizhevsky et al., 2009) with the ResNet-34 model architecture. The initial model f was trained for 200 epochs with the cross-entropy loss function. Our smoothed model v was computed by setting $f^0 = f$ and running Algorithm 1 with $\sigma = 0.1$ and $\lambda = 5$ for $n_T = 5$ timesteps at 200 epochs each timestep. The training of our smoothed model took 5 times longer than the baseline model. We compare our results to a ResNet-34 model trained with $\sigma = 0.1$ noisy examples as stochastically averaged model using RandomizedSmoothing (Cohen et al., 2019). We also trained a SmoothAdv model (Salman et al., 2019) for 4 steps of PGD with the maximum perturbation set to $\varepsilon = 0.5$. To assess certified accuracy, we run the Certify algorithm from Cohen et al. (2019) with $n_0 = 100, n = 10,000, \sigma = 0.1$ for the stochastically trained models. For the HeatSmoothing model, we run the same certification algorithm, but without running SamplingUnderNoise to compute $\hat{c}_A$. For completeness, we also certify the baseline model f^0. Certification plots are presented in Figure 3a. In this plot, we see that our model's ℓ_2 certified accuracy outperforms the stochastic models. Next, we attack our four models using PGD and DDN. We run both attacks with 20 steps and maximum perturbation $\varepsilon = 4.0$ to force top-1 misclassification. Results are presented in Table 1 and Figures 4a and 4c. In Table 1, we see that HeatSmoothing outperforms the stochastic models in terms of robustness. The only exception is robustness to mean PGD perturbations. This is shown in Figures 4a. Our model performs well up to an ℓ_2 PGD perturbation of just above 1.0.



(a) *CIFAR-10 top-1 certified accuracy* **(b)** *ImageNet-1k top-5 certified accuracy*

Figure 3: *Certified accuracy as a function of ℓ_2 radius.*

Table 1: *ℓ_2 adversarial distance metrics on CIFAR-10. A larger distance implies a more robust model.*

Model	L-bound		PGD		DDN	
	median	mean	median	mean	median	mean
HeatSmoothing	0.094	0.085	0.7736	0.9023	0.5358	0.6361
SmoothAdv	0.090	0.078	0.7697	1.3241	0.4812	0.6208
RandomizedSmoothing	0.087	0.081	0.7425	1.2677	0.4546	0.5558
Undefended baseline	-	-	0.7088	0.8390	0.4911	0.5713

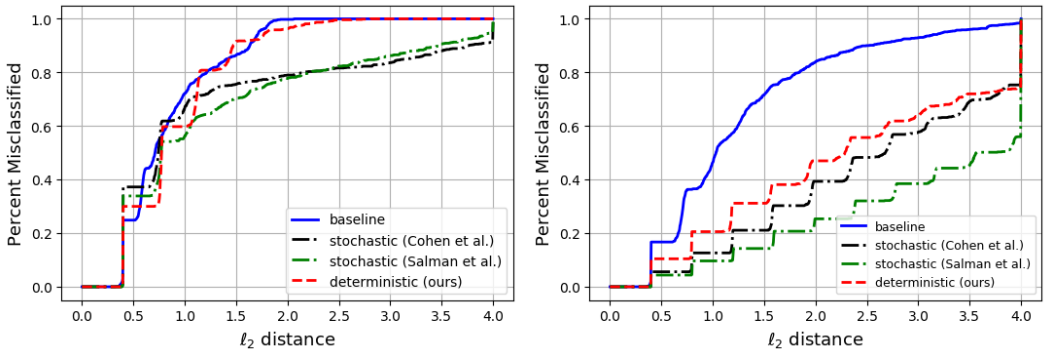
4.2 ImageNet-1k

We now execute our method on the ImageNet-1k dataset (Deng et al., 2009) with the ResNet-50 model architecture. The initial model f was trained for 29 epochs with the cross-entropy loss function. Our smoothed model v was computed by setting $f^0 = f$ and running Algorithm 1 with $\sigma = 0.25$ and $\lambda = 100$ for $n_T = 5$ timesteps at 15 epochs each timestep. This took roughly 2 days per timestep. We note that the $\frac{1}{2}\left\|v(x) - f^k(x)\right\|_2^2$ term is not suitable in the ImageNet-1k setting. This is due to the fact that the output vectors are too large, so the ℓ_2 distance blows up early in training. To remedy this, we replace the ℓ_2 distance with the Kullback-Leiblerd divergence between vectors $v(x)$ and $f^k(x)$. Due to time constraints, we had to terminate training early; however, our model still attains a good top-5 accuracy.

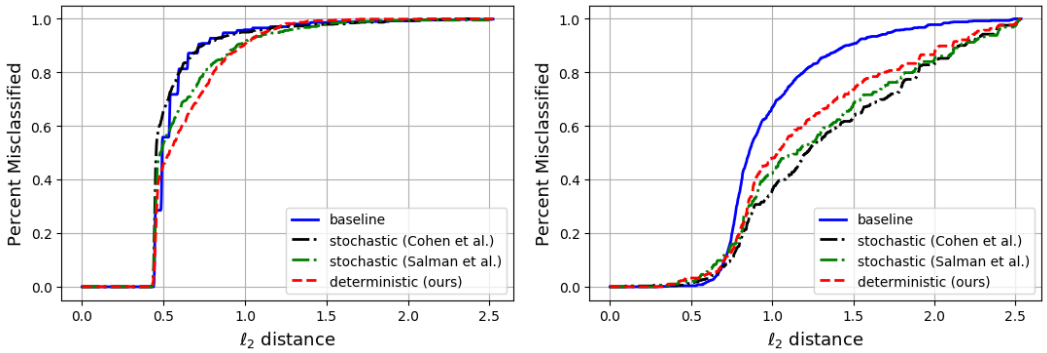
We compare our results to a pretrained RandomizedSmoothing ResNet-50 model with $\sigma = 0.25$ provided by Cohen et al. (2019). We also compare to a pretrained SmoothAdv ResNet-50 model trained with 1 step of PGD and with a maximum perturbation of $\varepsilon = 0.5$ with $\sigma = 0.25$ provided by Salman et al. (2019). To assess certified accuracy, we run the Certify algorithm from Cohen et al. (2019) with $n_0 = 25, n = 1,000, \sigma = 0.25$ for the stochastically trained models. As in the CIFAR-10 setting, for the HeatSmoothing model, we run the same certification algorithm but without running SamplingUnderNoise to compute $\hat{c}_A$. For completeness, we also certify the baseline model f^0. Certification plots are

Table 2: *ℓ_2 adversarial distance metrics on ImageNet-1k*

Model	L-bound		PGD		DDN	
	median	mean	median	mean	median	mean
HeatSmoothing	0.076	0.110	2.2857	2.3261	1.0313	1.2102
SmoothAdv	0.160	0.160	3.5643	3.0244	1.1537	1.2850
RandomizedSmoothing	0.200	0.180	2.6787	2.5587	1.2114	1.3412
Undefended baseline	-	-	1.0313	1.2832	0.8573	0.9864



(a) *CIFAR-10 top-1 PGD* **(b)** *ImageNet-1k top-5 PGD*



(c) *CIFAR-10 top-1 DDN* **(d)** *ImageNet-1k top-5 DDN*

Figure 4: *Attack curves: % of images successfully attacked as a function of ℓ_2 adversarial distance.*

presented in Figure 3b. We see in this plot that due to the fact that we had to terminate our training early, our model has a lower certified accuracy compared to the pretrained stochastic models. Next, we attack our four models using PGD and DDN. We run both attacks until top-5 misclassification or until 20 steps are reached. Results are presented in Table 2 and Figures 4b and 4d. We see that our model is comparable to the stochastic models, but does not outperform them. We expect that with additional training iterations, we will be able to outperform the stochastic models.

5 Conclusion

Randomized smoothing is a well-known method to achieve a Gaussian average of some initial neural network. This is desirable to guarantee that a model's predictions are unchanged given perturbed input data. In this work, we used a regularized loss to obtain deterministic Gaussian averaged models. By computing ℓ_2 certified radii, we showed that our method is comparable to previously-known stochastic methods. By attacking our models, we showed that our method is comparable to known stochastic methods in terms of adversarial robustness of the resulting averaged model. We also developed a new lower bound on perturbations necessary to throw off averaged models, and used it as a measure of model robustness. Lastly, our method is less computationally expensive in terms of inference time (see Appendix E).

Future work includes longer training time of our ImageNet-1k model. We are confident that once this is done, we will achieve comparable results to randomized smoothing methods on larger datasets. Secondly, adversarial training can be combined with our training method. This should result in deterministic averaged models that are also more robust to adversarial perturbations. Since our method is deterministic, we can adversarially train with any attack we choose without having to modify them to a stochastic averaging framework.

References

C. M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7 (1):108–116, 1995. doi: 10.1162/neco.1995.7.1.108. URL https://doi.org/10.1162/neco.1995.7.1.108.

C. Cheng, G. Nührenberg, and H. Ruess. Maximum resilience of artificial neural networks. In D. D'Souza and K. N. Kumar, editors, *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings*, volume 10482 of *Lecture Notes in Computer Science*, pages 251–268. Springer, 2017. doi: 10.1007/978-3-319-68167-2_18. URL https://doi.org/10.1007/978-3-319-68167-2_18.

J. M. Cohen, E. Rosenfeld, and J. Z. Kolter. Certified adversarial robustness via randomized smoothing. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 2019. URL http://proceedings.mlr.press/v97/cohen19c.html.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

A. Einstein. On the theory of the Brownian movement. *Ann. Phys*, 19(4):371–381, 1906.

C. Finlay and A. M. Oberman. Scaleable input gradient regularization for adversarial robustness. *CoRR*, abs/1905.11468, 2019. URL http://arxiv.org/abs/1905.11468.

I. M. Gelfand, R. A. Silverman, et al. *Calculus of Variations*. Courier Corporation, 2000.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572.

W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

I. Karatzas and S. E. Shreve. Brownian motion. In *Brownian Motion and Stochastic Calculus*, pages 47–127. Springer, 1998.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=BJm4T4Kgx.

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

M. Lécuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 656–672. IEEE, 2019. doi: 10.1109/SP.2019.00044. URL https://doi.org/10.1109/SP.2019.00044.

B. Li, C. Chen, W. Wang, and L. Carin. Second-order adversarial attack and certifiable robustness. *CoRR*, abs/1809.03113, 2018. URL http://arxiv.org/abs/1809.03113.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

A. Raghunathan, J. Steinhardt, and P. Liang. Certified defenses against adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Bys4ob-Rb.

J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger. Decoupling direction and norm for efficient gradient-based L2 adversarial attacks and defenses. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4322–4330. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00445. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Rony_Decoupling_Direction_and_Norm_for_Efficient_Gradient-Based_L2_Adversarial_Attacks_CVPR_2019_paper.html.

H. Salman, J. Li, I. P. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 11289–11300, 2019. URL http://papers.nips.cc/paper/9307-provably-robust-deep-learning-via-adversarially-trained-smoothed-classifiers.

W. A. Strauss. *Partial differential equations: An introduction*. John Wiley & Sons, 2007.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

S. S. Vempala. *The random projection method*, volume 65. American Mathematical Soc., 2005.

T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. S. Boning, and I. S. Dhillon. Towards fast computation of certified robustness for ReLU networks. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5273–5282. PMLR, 2018a. URL http://proceedings.mlr.press/v80/weng18a.html.

T. Weng, H. Zhang, P. Chen, J. Yi, D. Su, Y. Gao, C. Hsieh, and L. Daniel. Evaluating the robustness of neural networks: An Extreme Value Theory approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018b. URL https://openreview.net/forum?id=BkUHlMZ0b.

G. Yang, T. Duan, J. E. Hu, H. Salman, I. P. Razenshteyn, and J. Li. Randomized smoothing of all shapes and sizes. *CoRR*, abs/2002.08118, 2020. URL https://arxiv.org/abs/2002.08118.

H. Zhang, T. Weng, P. Chen, C. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 4944–4953, 2018. URL http://papers.nips.cc/paper/7742-efficient-neural-network-robustness-certification-with-general-activation-functions.

A Proof of Theorem 1

Proof. For clarity we treat the function as one dimensional. A similar calculation can be done in the higher dimensional case. Apply the Taylor expansion to the quadratic loss $(f(x+v)-y)^2$, to give

$$f(x+\eta) = f(x) + \eta f_x + \frac{1}{2}\eta^2 f_{xx} + O(\eta^3)$$

Keeping only the lowest order terms, in η and the lowest order derivatives, we obtain

$$(f(x+\eta)-y)^2 = (f(x)-y)^2 + 2(f_x\eta + \frac{1}{2}\eta^2 f_{xx})(f(x)-y) + \text{ higher order terms} \tag{10}$$

Take expectation

$$\mathbb{E}\left[(f(x+\eta)-y)^2\right] = \mathbb{E}\left[(f(x)-y)^2\right] + \sigma^2(f_x)^2 + \text{ higher order terms} \tag{11}$$

to drop the terms with odd powers of v gives (3). □

B Solving the heat equation by training with a regularized loss function

Theorem 3 tells us that training a model with added Gaussian noise is equivalent to training a model to solve the heat equation. We can discretize the heat equation (7) to obtain

$$\frac{f^{k+1}-f^k}{h} = \frac{\sigma^2}{2}\Delta f^{k+1} \tag{12}$$

for $k = 0, \ldots, n_T - 1$, where n_T is the fixed number of timesteps, $h = 1/n_T$, and $f^0 = f$, our initial model. Notice how, using the Euler-Lagrange equation, we can express f^{k+1} in (12) as the variational problem

$$f^{k+1} = \underset{v}{\operatorname{argmin}} \frac{1}{2}\int_{\mathbb{R}^d} \left(\left|v(x) - f^k(x)\right|^2 + \frac{h\sigma^2}{2}\left\|\nabla_x v(x)\right\|_2^2 \right) \rho(x)dx \tag{13}$$

where ρ is the density from which our clean data comes form. Therefore, this is equivalent to solving

$$f^{k+1} = \underset{v}{\operatorname{argmin}}\, \mathbb{E}_x \left[\left|v(x) - f^k(x)\right|^2 + \frac{h\sigma^2}{2}\left\|\nabla_x v(x)\right\|_2^2 \right] \tag{14}$$

Note that the minimizer of the objective of (13) satisfies

$$v - f^k = \frac{h\sigma^2}{2}\Delta v \tag{15}$$

which matches (12) if we set $f^{k+1} = v$. In the derivation of (15), we take for granted the fact that empirically, ρ is approximately uniform and is therefore constant. In the end, we iteratively compute (14) and obtain models $f^1, \ldots, f^{n_T}$, setting $v = f^{n_T}$, our smoothed model.

Something to take note of is that our model outputs be vectors whose length corresponds to the total number of classes; therefore, the objective function in (14) will not be suitable for vector-valued outputs $f^k(x)$ and $v(x)$. We instead use the following update

$$f^{k+1} = \underset{v}{\operatorname{argmin}}\, \mathbb{E}_x \left[\frac{1}{2}\left\|v(x) - f^k(x)\right\|_2^2 + \frac{h\sigma^2}{2}\left\|\nabla_x v(x)\right\|_2^2 \right] \tag{16}$$

C Approximating the gradient-norm regularization term

By the Johnson-Lindenstrauss Lemma (Johnson and Lindenstrauss, 1984; Vempala, 2005), $\|\nabla_x v(x)\|_2^2$ has the following approximation,

$$\begin{aligned} \|\nabla_x v(x)\|_2^2 &\approx \sum_{i=1}^{\kappa} \|\nabla_x \left(w_i \cdot v(x)\right)\|_2^2 \\ &\approx \sum_{i=1}^{\kappa} \left(\frac{\left(w_i \cdot v\left(x + \delta l_i\right)\right) - \left(w_i \cdot v(x)\right)}{\delta} \right)^2 \end{aligned} \tag{17}$$

where

$$w_i = \frac{1}{\sqrt{K}} \left(w_{i1}, \ldots, w_{iK}\right)^T \in \mathbb{R}^K \ , \quad w_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0,1) \tag{18}$$

and l is given by

$$l_i = \begin{cases} \frac{\nabla_x (w_i \cdot v(x))}{\|\nabla_x (w_i \cdot v(x))\|_2} & \text{if } \nabla_x \left(w_i \cdot v(x)\right) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

In practice, we set $\delta = 0.1$, $\kappa = 10$, and $K = Nc$, the total number of classes.

D Proof of Theorem 2

Proof. Suppose the loss function ℓ is Lipschitz continuous with respect to model input x, with Lipschitz constant L. Let ℓ_0 be such that if $\ell(x) < \ell_0$, the model is always correct. Then by Proposition 2.2 in Finlay and Oberman (2019), a lower bound on the minimum magnitude of perturbation δ necessary to adversarially perturb an image x is given by

$$\|\delta\|_2 \geq \frac{\max\left\{\ell_0 - \ell(x), 0\right\}}{L} \tag{20}$$

By Lemma 1 of Appendix A in Salman et al. (2019), our averaged model

$$v(x) = \left(f * \mathcal{N}(0, \sigma^2 I)\right)(x)$$

has Lipschitz constant $L = \frac{1}{\sigma}\sqrt{\frac{2}{\pi}}$. Replacing L in (20) with $\frac{1}{\sigma}\sqrt{\frac{2}{\pi}}$ and setting $\ell_0 = v(x)_{(k)}$, $\ell(x) = v(x)_{(k+1)}$ gives us the proof. □

E Model inference computation time

In Tables 3 and 4, we present the average time it takes to perform classification and certification of an image on the CPU and GPU, respectively. For certification, we implement the CERTIFY algorithm from Cohen et al. (2019). On CIFAR-10, we use the top-1 classification setting while on ImageNet-1k, we use top-5 classification.

Table 3: *Average classification & certification time on the CPU (seconds)*

Model	Classification		Certification	
	CIFAR-10	ImageNet-1k	CIFAR-10	ImageNet-1k
HEATSMOOTHING	0.0049	0.0615	13.5994	3.2475
RANDOMIZEDSMOOTHING	0.0480	0.1631	13.7394	3.3938

Table 4: *Average classification & certification time on the GPU (seconds)*

Model	Classification		Certification	
	CIFAR-10	ImageNet-1k	CIFAR-10	ImageNet-1k
HEATSMOOTHING	0.0080	0.0113	0.3786	2.6624
RANDOMIZEDSMOOTHING	0.0399	0.0932	0.4142	2.8690