EFFICIENT REGULARIZATION FOR ROBUST DEEP RELU NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a regularization functional for deep neural networks with ReLU activations, and propose regularizers that encourage networks which are smooth not only in their predictions but also their decision boundaries. We evaluate the stability of our networks against the standard set of ℓ_2 and ℓ_∞ norm-bounded adversaries, as well as several recently proposed perception-based adversaries, including spatial, recoloring, JPEG, and a learned neural threat model. Crucially, our models are simultaneously robust against multiple state-of-the-art adversaries, suggesting that the robustness generalizes well to *unseen* adversaries. Furthermore, our techniques do not rely on adversarial training and are thus very efficient, incurring overhead on par with two additional parallel passes through the network. On CIFAR-10, we obtain our results after training for only 4 hours, while the next-best performing baseline requires nearly 25 hours of training. To the best of our knowledge, this work presents the first technique to achieve robustness against adversarial perturbations *without* adversarial training.

1 INTRODUCTION

Recent results in deep learning highlight the remarkable ability of deep neural networks to perform on real world on tasks ranging from object detection to game playing. However, such models suffer widely from adversarial examples (Szegedy et al., 2013), revealing that the current generation of deep models often rely on unexpectedly brittle or spurious features. This behavior not only presents an attack surface in sensitive applications, but can also have unexpected consequences for performance-critical settings, e.g., models that are deployed in the physical world.

The standard approach to defending against such adversarial examples has been to formally specify a set of perturbations, then augment the training procedure with access to the examples generated by the adversary; such techniques form a broad category called *adversarial training* (Goodfellow et al., 2014; Kurakin et al., 2016; Madry et al., 2017). While these methods have been highly successful in defending against the specific adversaries for which they are tailored, it has been observed that this strategy confers negligible robustness against other forms of perturbations–in many cases even degrading such robustness (Kang et al., 2020). Assuming access to the adversary during training is also unrealistic for delivering the type of real-world robustness necessary for either security- or performance-critical applications. General robustness against unseen adversaries thus remains a major open problem.

Our main observation is that achieving such *generalizeable* robustness requires redefining what it means to be robust. In particular, instead of indexing our goal against a specific adversary, we propose an approach to robustness which eschews the adversarial training paradigm entirely. Our results support the conclusion that the models trained using our methods are generally robust. As an added benefit, because the adversary is often formulated as an inner optimization in the training procedure, we also avoid the significant computational overheads that adversarial training often entails.

Contributions. We develop regularizers for training locally stable deep neural networks. Our goal is to learn a function which does not change much in the neighborhoods of natural inputs, i.e., unperturbed data drawn from the input distribution, independently of whether the network classifies correctly. To that end, we design a regularization functional that exploits the continuous piecewise

linear nature of ReLU networks to induce a function which varies smooth over the data manifold in not only its predictions, but also its decision boundaries.

We evaluate our approach by training neural networks with our regularizers for the task of image classification on CIFAR-10 (Krizhevsky and Hinton, 2009) and MNIST (LeCun et al., 2010). Empirically, a single network trained using our regularizers exhibits robustness against multiple norm-bounded adversarial models implementing ℓ_2 and ℓ_∞-based attacks. We also report state-of-the-art *verified* robust accuracy for CIFAR-10 under ℓ_∞ perturbations of size $\epsilon = 8/255$. Furthermore, we also achieve strong robustness against a suite of recently-proposed perception-based adversaries. In both cases, we outperform adversarial training methods when evaluated on the unseen adversaries, whereas our regularizers achieve such performance without access to *any* adversaries during training.

Furthermore, our regularizers are cheap: we simply evaluate the network at two additional random points for each training sample, so the total computational cost is on par with three parallel passes through the network. Our techniques thus present a novel, regularization-only approach to learning robust neural networks against *unseen* adversaries, which achieves performance comparable to existing tailored defenses while also being an order of magnitude more efficient.

2 BACKGROUND

ReLU networks Our development focuses on a standard architecture for deep neural networks: fully-connected feedforward networks with ReLU activations. In general, we can write the function represented by such a network with n layers and parameters $\theta = \{A_i, b_i\}_{i=1,...,n-1}$ as

$$z_0 = x \tag{1}$$
$$\hat{z}_i = A_i \cdot z_{i-1} + b_i \qquad \text{for } i = 1, ..., n-1 \tag{2}$$
$$z_i = \sigma(\hat{z}_i) \qquad \text{for } i = 1, ..., n-2 \tag{3}$$
$$f(x; \theta) = \hat{z}_{n-1} \tag{4}$$

where the A_i are the weight matrices and the b_i are the bias vectors. We call the z_i "hidden activations", or more simply, activations, and the $\hat{z}_i$ "pre-activations".

In this work, we consider networks in which $\sigma(\cdot)$ in (3) is the Rectified Linear Unit (ReLU)

$$z_i = \sigma(\hat{z}_i) := \max(0, \hat{z}_i) \tag{5}$$

It is clear from this description that ReLU networks are a family of continuous piecewise linear functions.

Adversarial robustness One common measure of robustness for neural networks is against a norm-bounded adversary. In this model, the adversary is given an input budget ϵ over a norm $||\cdot||_{in}$, and asked to produce an output perturbation δ over a norm $|\cdot|_{out}$. A point x' is an ϵ-δ adversarial example for an input pair (x, y) if

$$||x' - x||_{in} \leq \epsilon \tag{6}$$
$$|f(x'; \theta) - y|_{out} \geq \delta \tag{7}$$

When the specific norm is either unimportant or clear from context, we also write the first condition as $x' \in N_\epsilon(x)$, where $N_\epsilon(x)$ refers to the ϵ-ball or neighborhood about x. If such an adversarial example does not exist, we say that the network is ϵ-δ robust at x. Standard examples of $||\cdot||_{in}$ include the ℓ_2 and ℓ_∞ norm, defined for vectors as $||x||_\infty := \max_i |x_i|$. For classification tasks, the adversary is successful if it produces an example in the ϵ-neighborhood of x which causes the network to misclassify. In this case, we drop δ and say that the network is ϵ-robust at x. Note that if $f(x; \theta)$ is already incorrect, then x suffices as an adversarial example.

3 RELATED WORK

Adversarial examples were introduced by Szegedy et al. (2013), who found that naively trained neural networks suffer almost complete degradation of performance on natural images under slight perturbations which are imperceptible to humans. A standard class of defenses is *adversarial training*,

which is characterized by training on adversarially generated input points (Goodfellow et al., 2014). In particular, the Projected Gradient Descent (PGD) attack (Kurakin et al., 2016; Madry et al., 2017) is widely considered to be an empirically sound algorithm for both training and evaluation of robust models. However, such training methods rely on solving an inner optimization via an iterative method, effectively increasing the number of epochs by a multiplicative factor (e.g., an overhead of 5–10x for standard PGD).

Achieving robustness against multiple adversarial models has also been explored previously (Schott et al., 2018; Tramer and Boneh, 2019; Maini et al., 2019; Croce and Hein, 2019), though in most cases these works use weaker variants of the subset of standard adversaries we consider (e.g., a smaller ϵ or the single-step version of PGD), and also require training against specific adversaries. The only work to study adversarial robustness against multiple *unseen* adversaries is a recent approach by Laidlaw et al. (2021), which proposes to first learn a neural perceptual threat model that approximates a distance between images based on human perception. However, they ultimately employ the threat model as an adversary during training, incurring overhead on par with standard adversarial training, and achieve lower robustness against the standard norm-bounded adversaries than our methods.

A related goal is to train models which are *provably robust*. One method is to use an exact verification method, such as an MILP solver, to prove that the network is robust on given inputs (Tjeng et al., 2017). In particular, Xiao et al. (2019) use a similar loss based on ReLU pre-activations to learn stable ReLUs for efficient verification. However, their loss does not induce a metric and so does not enjoy the same approximation guarantees as ours; thus they require training against a PGD adversary to produce a robust model. Certification methods modify models to work directly with neighborhoods instead of points (Dvijotham et al., 2018; Gowal et al., 2018; Mirman et al., 2018; Wong et al., 2018). In practice, the inference algorithms must overapproximate the neighborhoods to preserve soundness while keeping the representation compact as it passes through the network. This strategy can be interpreted as solving a convex relaxation of the exact verification problem. Though certification thus far has produced better lower bounds, verification as a technique is fully general and can be applied to any model (given sufficient time); recent work also suggests that methods using layerwise convex relaxations may face an inherent barrier to tight verification (Salman et al., 2019).

Finally, several prior works explore regularizing for robustness based on similar loss functionals (Ross and Doshi-Velez, 2017; Jakubovitz and Giryes, 2018), though they only report results using the weaker single-step PGD adversary. Hein and Andriushchenko (2017) propose a conceptually similar regularizer to minimize the difference between logits and show improved ℓ_2 certified robustness. Another recent work by Zhai et al. (2020) uses randomized smoothing for ℓ_2 certified robustness, and claim similar computational advantage due to avoiding adversarial training, but still take 61 hours to train, compared to only 4 hours in our approach. Finally, we also note that a similar decomposition between accuracy and stability forms the basis for the TRADES algorithm (Zhang et al., 2019), though their procedure ultimately still relies on adversarial training.

4 SETTING

We begin by reframing the goal of learning functions that are robust using a perspective which decouples stability from accuracy. The key observation is that we would like to train networks that are locally stable around natural inputs, even if the network output is incorrect. This approach contrasts with adversarial training, which attempts to train the network to classify correctly on worst-case adversarial inputs. In particular, recall that a network is ϵ-robust at x if no point in the ϵ-neighborhood of x causes the network to misclassify. We consider the related property of ϵ-stability:

Definition 4.1. *A function f is ϵ-δ* stable *at an input x if for all $x' \in N_\epsilon(x)$, $|f(x) - f'(x)| \leq \delta$. Similarly, a classifier f is ϵ-*stable *at an input x if for all $x' \in N_\epsilon(x)$, $f(x) = f(x')$.*

As ϵ-stability does not depend on the correct label for x, we will seek to induce ϵ-stability as a property of the learned function independent of its predictions. For completeness, we state the following connection between robustness and stability:

Proposition 4.1. *A function f is ϵ-δ* robust *at an input x if and only if f is ϵ-δ stable at x and $f(x) = y$. Similarly, a classifier f is ϵ-*robust *at an input x if and only if f is ϵ-stable at x and f correctly classifies x.*

We emphasize that this notion of stability is not new, and can be understood as a local Lipschitz property. For instance, Zheng et al. (2016) propose to regularize deep neural networks for ϵ-stability directly with the following objective:

$$L_{stab}(x, x'; \theta) = |f(x; \theta) - f(x'; \theta)| \tag{8}$$

They incorporate this objective as an additional loss term by setting x' as perturbations of the input data x, sampled from a Gaussian. While their results demonstrate stability under perturbations such as JPEG compression, this type of randomized smoothness is well known to deliver negligible robustness against a more targeted PGD adversary (see, e.g., Jiang et al. (2020)), the reason being that randomly sampling points to evaluate the stability objective is insufficient to deliver the worst-case guarantees needed for ϵ-robustness.

5 HAMMING REGULARIZATION

In this section, we develop a novel regularizer based on the Hamming distance between activation patterns of nearby points. Our main insight is a connection between the stability of a ReLU network and the distribution of its decision boundaries around natural inputs.

5.1 HAMMING DISTANCES AND RELU NETWORKS

Let f be a ReLU network. Because f is a continuous, piecewise linear function, for any input point x (minus a set of measure zero), f restricted to a sufficiently small neighborhood of x is a linear function. We will denote the linear function induced in this way by the input x as $f_x(\cdot; \theta)$, i.e., the analytic extension of the local linear component about x over the input domain.

Clearly, every local linear component $f_x(\cdot; \theta)$ can be identified with a unique "activation pattern", i.e., the sequence of branches taken at each ReLU by the input x. We will write $f_H(x; \theta)$ for the map that takes inputs x to their activation patterns, which live on the unit hypercube $\{0, 1\}^N$ (where N is the number of ReLUs in the network). If we endow the hypercube with the standard Hamming distance d_H, this induces a pullback (psuedo-)metric in the input space: $d^*_H(x, y) = d_H(f_H(x; \theta), f_H(y; \theta))$. We note that the metric identification is exactly the set of local linear components $f_x(\cdot; \theta)$.

We are now ready to make the connection between local stability and the Hamming distances so defined. We will argue that d^*_H yields good bounds on the local stability of f. We show this in two steps. Consider the objective of reducing the number of distinct local linear components of the learned function f in the ϵ-neighborhood of inputs x. Our first result says that minimizing d^*_H over a neighborhood of points is tractable:

Lemma 5.1. *Let $X = \{x_1, ..., x_n\}$ be a set of input points and let $\bar{X}$ denote the convex polytope defined by X. If there exists $\hat{x} \in \bar{X}$ such that $d^*_H(x_i, \hat{x}; \theta) = 0$ for all $x_i \in X$, then the same also holds for all pairs of points in $\bar{X}$, i.e. $x, x' \in \bar{X} \implies d^*_H(x, x') = 0$.*

The result follows immediately from the previous observation that d^*_H is a pseudo-metric. The next lemma says that the standard stability objective (Equation 8) enjoys the same property when combined with previous statement:

Lemma 5.2. *Let $X = \{x_1, ..., x_n\}$ be a set of input points such that Lemma 5.1 holds. Define the maximum pairwise difference over X of the function f to be $\delta := \max_{x_i, x_j \in X} |f(x_i; \theta) - f(x_j; \theta)|$. Then for any ϵ, $x \in \bar{X}$ such that $N_\epsilon(x) \subseteq \bar{X}$, we have that f is ϵ-δ stable at x.*

Proof. By the assumption that Lemma 5.1 holds, f is a linear function when restricted to $\bar{X}$. Furthermore, $\bar{X}$ is a closed, convex set and thus f achieves its maximum and minimum over $\bar{X}$ at the vertices of the polytope. The result then follows from the fact that $N_\epsilon(x) \subseteq \bar{X}$ and the definition of ϵ-δ stability. □

The primary significance of these results is that optimizing for stability becomes much more tractable when viewed in terms of the Hamming regularizer, which provides the primary theoretical justification for regularizing based on activation patterns. In particular, rather than sampling random perturbations from, e.g., a Gaussian, we can take them to be the corners of an appropriately chosen polytope; then we do not depend on any concentration bounds for the quality of our stability guarantees either.

5.2 THE HAMMING REGULARIZATION FUNCTIONAL

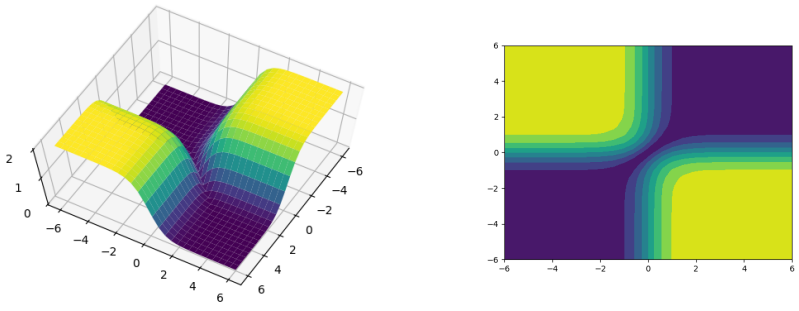


Figure 1: Surface and contour plots for H_α ($\alpha = 1$).

Lemmas 5.1 and 5.2 yield natural objectives for inducing stability. Note that a simple corollary is that if the set X forms a hypercube, it is both necessary and sufficient to check the assumptions of either lemma on the opposing vertices to apply their conclusions. We thus propose the following regularization functionals, which are evaluated pairwise between input points:

$$L_S(x_i, x_j; \theta) = |f(x_i; \theta) - f(x_j; \theta)|_2^2 \tag{9}$$

$$L_H(x_i, x_j; \theta) = d_H^*(x_i, x_j; \theta)^2 \tag{10}$$

To induce ϵ-stability at a point x, we can simply minimize $L_S(x-\rho, x+\rho; \theta)$ and $L_H(x-\rho, x+\rho; \theta)$ for $\rho \in \{\pm\epsilon\}^d$, where d is the input dimension.

However, the loss term L_H is not continuous in the inputs, and furthermore, the gradients vanish almost everywhere, so it does not generate good training signals. We thus also propose the following continuous relaxation of the Hamming distance d_H:

$$H_\alpha(\hat{z}, \hat{y}; \theta) := \frac{1}{2}\text{abs}(\tanh(\alpha * \hat{z}) - \tanh(\alpha * \hat{y})) \tag{11}$$

This form is differentiable everywhere except when $\hat{z} = \hat{y}$, and recovers the Hamming distance when we take α to infinity. Qualitatively, sensitive activations (i.e., small $|\hat{z}|$ and $|\hat{y}|$) are permitted so long as they are precise. Figure 1 presents the surface and contour plots of H_α. Note that this idea can be extended more generally to other activation functions by penalizing differing pre-activations more when the second derivative of the activation function is large (and so the first-order Taylor approximation has larger errors).

5.3 CONVERGENCE OF HAMMING FUNCTIONAL IN THE LIMIT

In this section, we develop a theoretical understanding of the proposed Hamming regularizer functional by showing convergence in the infinite sample limit to the Laplacian of a related function using the theory of manifold regularization.

We begin with some brief background. The manifold assumption states that input data is not drawn uniformly from the input domain $\mathcal{X}$, also know as the *ambient space*, but rather is supported on a submanifold $\mathcal{M} \subset \mathcal{X}$, called the *intrinsic space*. There is thus a distinction between regularizing on the ambient space, where the learned function is smooth with respect to the entire input domain (e.g., Tikhonov regularization (Phillips, 1962; Tikhonov et al., 2013)), and regularizing over the intrinsic space, which uses the geometry of the input submanifold to determine the regularization norm.

Let μ be the probability measure of the input distribution, which has support $\mathcal{M}$. We will define μ_ϵ to be the probability measure produced by convolving μ with a uniform distribution over the ℓ_∞ ball of radius ϵ; by definition, one can simulate drawing from the distribution of μ_ϵ given a sample from μ simply by adding some random noise from ℓ_∞ ball of radius ϵ. We write the support of the perturbed input distribution as $\mathcal{M}_\epsilon$.

To apply manifold regularization, we consider the function

$$g(\cdot;\theta) = (\gamma_S^{1/2} f(\cdot;\theta), \gamma_H^{1/2} f_H(\cdot;\theta)) \tag{12}$$

which is simply the output of f concatentated with its activation pattern, with some weighting factors γ_S, γ_H. Then clearly,

$$|g(x_i;\theta) - g(x_j;\theta)|_2^2 = \gamma_S L_S(x_i, x_j;\theta) + \gamma_H L_H(x_i, x_j;\theta) \tag{13}$$

Thus, the regularization functional $|g(x_i;\theta) - g(x_j;\theta)|_2^2$ is simply the sum of our proposed regularizers, so characterizing the behavior of g under regularization suffices for understanding the regularization effects of L_S and L_H.

The corresponding regularizer is as follows:

$$\frac{1}{N^2} \sum_{i,j=1}^{N} |g(x_i) - g(x_j)|_2^2 L_{i,j} \tag{14}$$

Here, the $x_1, ..., x_N$ are samples drawn, by assumption, from the perturbed input manifold $\mathcal{M}_\epsilon$ according to μ_ϵ. The matrix L is known as the discrete graph Laplacian, and is composed of weights measuring the similarity between samples. A common choice of weights is a heat kernel: $L_{i,j} = L(x_i, x_j) := \exp(-||x_i - x_j||^2/s)$. Under these conditions, it can be shown (see, for instance, Belkin et al. (2006)) that the discrete sum converges to the following integral as the number of sample grows to infinity:

$$||g||_I^2 := \int_{\mathcal{M}_\epsilon} ||\nabla_{\mathcal{M}_\epsilon} g(x)||^2 d\mu_\epsilon(x) \tag{15}$$

Hence, the proposed functionals in Equations 9 and 10 have the effect in the limit of inducing the change in network outputs and decision boundaries to be smooth where the probability of drawing a perturbed sample is large, as initially claimed.

5.4 Training with Hamming regularization

In this section, we propose a regularization technique based on the Hamming regularization functional. As in stability training, our perturbations are randomly sampled, which allows our approach to be very efficient. However we use the corners of a hypercube for our perturbations in order to take advantage of the properties developed in the previous sections.

More explicitly, for every sample x, we generate a new random maximal perturbation $\rho \in \{\pm\epsilon\}^d$ and produce the pair of perturbed inputs $x^+ := x + p$ and $x^- := x - p$. For a batch of points $\{x_i\}_{i=1}^N$, we compute the standard stability regularization term as

$$||f(\cdot,\theta)||_S^2 = \sum_i |f(x_i^+) - f(x_i^-)|_2^2 \tag{16}$$

For the Hamming regularization term, we normalize the value for each pair of perturbations to a value between 0 and 1. More explicitly, let N be the number of samples, n be the number of layers, and m_j be the number of activations at the j^{th} layer. Denote the k^{th} activation in the j^{th} layer of the i^{th} sample as $\hat{z}_{ijk}$. Then we use the following (empirical) regularizer on the activation map f_H:

$$||f_H(\cdot,\theta)||_H^2 = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n} \sum_{j=1}^{n} \frac{1}{m_j} \Big| \sum_k H_\alpha(\hat{z}_{ijk}^+, \hat{z}_{ijk}^-;\theta) \Big|_2 \tag{17}$$

The final optimization objective is thus

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} V(f(x_i;\theta), y_i) + \gamma_K ||f(\cdot;\theta)||_K^2 + \gamma_S ||f(\cdot;\theta)||_S^2 + \gamma_H ||f_H(\cdot;\theta)||_H^2 \tag{18}$$

where V is the loss function, $||f(\cdot,\theta)||_K^2$ is the ambient regularizer (e.g., ℓ_2 regularization), and the $\gamma_K, \gamma_S, \gamma_H$ are hyperparameters which control the relative contributions of the different regularizers.

Table 1: CIFAR-10 robust accuracy against various adversaries and perturbation models. Results for the adversarially trained models are taken from Laidlaw et al. (2021). Training times use a single RTX 1080 GPU.

Defense	clean	ℓ_∞	ℓ_2	Spatial	ReColor	JPEG	NPTM	minutes
AT ℓ_∞	86.8	49.0	19.2	4.8	54.5	3.2	0.0	1620
AT ℓ_2	85.0	39.5	47.8	7.8	53.5	25.5	0.3	1537
AT Spatial	86.2	0.1	0.2	53.9	5.1	0.0	0.0	1607
AT ReColor	93.4	8.5	3.9	0.0	65.0	1.4	0.0	3053
AT NPTM	71.6	28.7	33.3	64.5	67.5	30.9	9.8	1481
Ours	72.1	36.8	43.4	28.4	63.1	34.2	9.1	252

6 Experimental results

We report results a variety of adversarial settings on CIFAR-10 and MNIST image classification our regularization techniques. For CIFAR-10, we additionally a smaller CNN for verified robustness, following the setup in Xiao et al. (2019). We also include several ablation studies in the standard CIFAR-10 setting to better understand the individual contributions of the proposed method to robustness. Appendix A gives full experimental details and training hyperparameters.

6.1 Robustness against unseen adversaries

Table 1 reports our main experimental results. All adversaries are given white-box access to the trained models. We separate the evaluation into two categories. The first category consists of norm-bounded adversaries, with the ℓ_2 and ℓ_∞ norm-bounded adversaries being the most common in the literature. These adversaries are allowed to arbitrarily perturb the input images within an ℓ_p ball in order to cause the network to misclassify. We report results at the standard bounds of $\ell_\infty = 8/255$ and $\ell_2 = 1$, using the state-of-the-art AutoAttack ensemble (Croce and Hein, 2020) as the adversary.

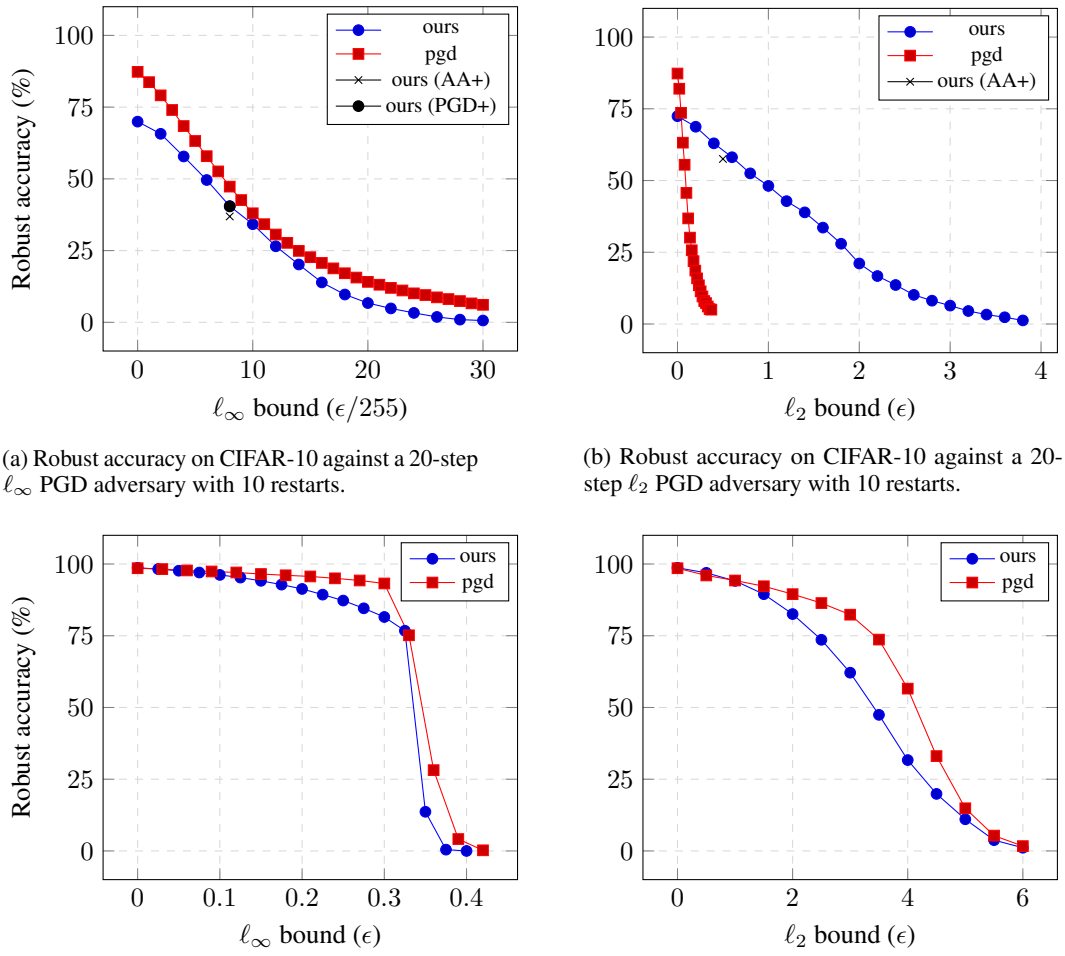
The second class consists of perception-based adversaries. These attacks aim to reduce the visual impact of their perturbations, often relying on heuristic methods to bound the attack. In particular, the Spatial (Xiao et al., 2018) attack uses a distance similar to optimal flow; ReColor (Laidlaw and Feizi, 2019) is applied to the color space of the image, rather than pixels; JPEG (Kang et al., 2020) perturbs the image within the JPEG-compressed space; and NPTM (Laidlaw et al., 2021) uses a learned perceptual distance which is captured by a separate neural module.

In both cases, the experiments indicate that our Hamming regularizer outperforms adversarial training on the unseen class. More explicitly, adversarial training against the ℓ_∞ and ℓ_2 norm-bounded adversaries imparts negligible robustness against many of the perceptual-based attacks, with the spatial and NPTM adversaries being particularly challenging; conversely, the three methods which train against the perceptual attacks all achieve lower robustness against the norm-bounded adversaries than our defense. Interestingly, the strongest threat model empirically is the NPTM attack, and our defense achieves nearly the same performance against the NPTM adversary as does the adversarially trained model (AT NPTM, which uses an approximate version of the NPTM adversary in order to be efficient enough for training). Note also that, though our method performs worse on clean data compared with standard AT, the drop is comparable with AT NPTM, which is the only other method to achieve comparable robust accuracy against all perceptual adversaries.

Finally, as shown in the last column of Table 1, our defense takes drastically less time to train than every other baseline, being almost 6 times faster than the next best defense.

6.2 Robustness curves against standard PGD

We provide detailed robustness curves for the most common settings for adversarial robustness in the literature, using the standard 20-step PGD adversary with 10 random restarts. Figure 2 reports the results on both MNIST and CIFAR-10, for both ℓ_2 and ℓ_∞ bounds. We also plot for reference the robustness against the stronger AutoAttack+ adversary used for the results in Table 1 as "ours (AA+)";



(a) Robust accuracy on CIFAR-10 against a 20-step ℓ_∞ PGD adversary with 10 restarts.

(b) Robust accuracy on CIFAR-10 against a 20-step ℓ_2 PGD adversary with 10 restarts.

(c) Robust accuracy on MNIST against a 20-step ℓ_∞ PGD adversary with 10 restarts.

(d) Robust accuracy on MNIST against a 20-step ℓ_2 PGD adversary with 10 restarts.

Figure 2: Robustness curves against a 20-step PGD adversary with 10 random restarts.

for CIFAR-10, we additionally ran a 500-step ℓ_∞ PGD adversary with 20 restarts at $\epsilon = 8/255$, yielding 40.4% robust accuracy, labelled "ours (PGD+)". For comparison, we also plot the robustness of models trained using vanilla adversarial training against an ℓ_∞-bounded PGD adversary at the standard levels for each dataset ($\epsilon = 8/255$ for CIFAR-10, and $\epsilon = 0.3$ for MNIST), taken from Madry et al. (2017). For CIFAR-10, our findings mirror those in Schott et al. (2018), namely, that a model trained using PGD on CIFAR-10 for ℓ_∞ performs poorly against ℓ_2 attacks. The results on MNIST shows a much less dramatic difference in performance, which we attribute to the lower complexity of the task. These results show that, compared with models trained using standard PGD against ℓ_∞ perturbations, the standard setting in the adversarial training literature, our methods perform comparably (or substantially better, in the case of the ℓ_2 adversary on CIFAR-10) in a variety of settings, despite not using the adversary during training.

6.3 Verified Robustness

We also report the *verified robustness* of our method on CIFAR-10 against ℓ_∞ perturbations at $\epsilon = 8/255$ in Table 2. We provide this metric primarily as a baseline for establishing a provable lower bound on the robustness achieved by our defense, compared to empirical robustness which is often a brittle measure of true robustness, particularly for newly proposed defenses (Athalye et al., 2018; Carlini et al., 2019; Tramer et al., 2020).

Table 2: CIFAR-10 verified and robust accuracies for ℓ_∞-bounded perturbations at $\epsilon = 8/255$.

Defense	Test Accuracy (%)		
	Verified	Robust	Clean
RS Loss (Xiao et al. (2019))	20.27	26.78	40.45
Hamming Regularization	21.04	25.56	36.66

Table 3: CIFAR-10 ablation studies against an ℓ_∞ PGD-adversary at $\epsilon = 8/255$.

Defense	Test Accuracy (%)	
	Robust	Clean
PGD (Madry et al. (2017))	**45.8**	87.3
FGSM (Madry et al. (2017))	0.0	90.3
Hamming Regularization	40.54	72.1
stability regularizer only	20.11	34.74
Hamming regularizer only	24.87	90.24

We achieve state-of-the-art results in the same setting as in Xiao et al. (2019), using an MILP solver to directly verify the outputs of the neural network as a program. The main added challenge in this setting is ensuring that the solver does not time out for a significant fraction of inputs. The RS Loss proposes to regularize the pre-activations of ReLU networks so as to encourage functions that are easier to verify by reducing the number of active ReLUs (branches) in the neural network (program). Our performance suggests that the Hamming regularizer achieves a similar effect. However, we note a crucial difference between the two approach is that our regularization functional is based on a pre-metric, and thus can effectively regularize over large perturbation neighborhood (as developed in Section 5.1). In contrast, Xiao et al. (2019) pair the RS Loss with a standard PGD adversary to find strong perturbations; thus the approach suffers from the same problems as standard adversarial training, namely, substantially longer training times as well as likely overfitting to a specific adversary.

6.4 Ablation Studies

Table 3 presents the results of ablation studies using the individual terms of our proposed regularization scheme on CIFAR-10 against a 20-step, ℓ_∞ bound PGD-adversary with 10 random restarts at $\epsilon = 8/255$. Our results indicate that both terms contribute jointly and individually to our performance. Intriguingly, applying the Hamming regularizer alone seems to provide a significant amount of robustness with only minimal degradation in clean accuracy.

7 Conclusion

We design regularizers that encourage piecewise linear neural networks to learn locally stable functions. We demonstrate this stability by showing that a single model trained using our regularizers is resilient against both norm-bounded and perception based adversaries, achieving a new state-of-the-art for robust accuracy against unseen classes of adversaries. We also achieve state-of-the-art verified robustness of 21% against ℓ_∞-bounded perturbations of size $\epsilon = 8/255$ on CIFAR-10.

Critically, computing our regularizers relies only on random sampling, and thus does not require running an inner optimization loop to find strong perturbations at training time. As such, our techniques exhibit strong scaling, since they increase batch sizes rather than epochs during training, allowing us to train an order of magnitude faster than standard adversarial training. This work thus presents the first regularization-only approach to achieve comparable results to standard adversarial training against a variety of unseen perturbation models.

REFERENCES

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7 (Nov):2399–2434, 2006.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2017.

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

Francesco Croce and Matthias Hein. Provable robustness against all adversarial l_p-perturbations for $p \geq 1$. *arXiv preprint arXiv:1905.11213*, 2019.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.

Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*, 2018.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library). 2019. URL `https://github.com/MadryLab/robustness`.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2266–2276, 2017.

Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529, 2018.

Haoming Jiang, Zhehui Chen, Yuyang Shi, Bo Dai, and Tuo Zhao. Learning to defense by learning to attack. *arXiv preprint arXiv:1811.01213*, 2020.

Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016*, 2020.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. *arXiv preprint arXiv:1906.00001*, 2019.

Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. *arXiv preprint arXiv:2006.12655*, 2021.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

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.

Pratyush Maini, Eric Wong, and J Zico Kolter. Adversarial robustness against the union of multiple perturbation models. *arXiv preprint arXiv:1909.04068*, 2019.

Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pages 3578–3586, 2018.

David L Phillips. A technique for the numerical solution of certain integral equations of the first kind. *Journal of the ACM (JACM)*, 9(1):84–97, 1962.

Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *arXiv preprint arXiv:1711.09404*, 2017.

Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems 32*, pages 9835–9846. Curran Associates, Inc., 2019.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on mnist. *arXiv preprint arXiv:1805.09190*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Andrei Nikolaevich Tikhonov, AV Goncharsky, VV Stepanov, and Anatoly G Yagola. *Numerical methods for the solution of ill-posed problems*, volume 328. Springer Science & Business Media, 2013.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. 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 5866–5876. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/8821-adversarial-training-and-robustness-for-multiple-perturbations.pdf`.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

Eric Wong, Frank R. Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514*, 2018.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

Kai Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. *ICLR*, 2019.

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. *arXiv preprint arXiv:2001.02378*, 2020.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016.

A EXPERIMENTAL METHODS AND HYPERPARAMETERS

We use a PreActResNet18 model (He et al., 2016) for the CIFAR-10 robustness experiments. We train using SGD with a batch size of 128 and weight decay γ_K of 5e-4. We follow Maini et al. (2019) for our learning rate schedule, which is piecewise linear and starts at 0 and goes to 0.1 over first 40 epochs; 0.005 over the next 40 epochs; and 0 over the final 20 epochs. We increase epsilon from 2 to 8 over epochs 10 to 35. We start the weight γ_S of the manifold regularization at 0.8 and the weight γ_H of the Hamming regularization at 2,400; these increase linearly up to a factor of 10 from epochs 20 to 80. We set the hyperparameter $\alpha = 8$. We use this set of hyperparamaters for all the CIFAR-10 ablation studies (except when setting γ_S or γ_H to 0 for the ablation studies involving the individual regularizers).

We use a two-layer convolutional neural network for the CIFAR-10 verification experiments, consisting of 2x2 strided convolutions with 16 and 32 filters, then a 128 hidden unit fully connected layer. This is the same model as used in Wong et al. (2018) and Xiao et al. (2019), except those works use a 100 hidden unit fully connected layer. We use the same schedule for the learning rate and ϵ as in the PreActResNet18 model. The weight γ_S starts at 0.4 and the weight γ_H starts at 9000; these increase linearly up to a factor of 10 from epochs 20 to 80. We use the same hyperparameter α as for the PreActResNet18 model.

We use the CNN with four convolutional layers plus three fully-connected layers from Carlini and Wagner (2017) for the MNIST robustness experiments. We use the same schedule for the learning rate, ϵ, γ_S, and γ_H as in the PreActResNet18 model (except that ϵ scales to 0.3). We use the same hyperparameter α as for the PreActResNet18 model.

To set the hyperparameters γ_S and γ_H on the PreActResNet18 model, we ran a grid search for $\gamma_S = 0.2, ..., 1.0$ and $\gamma_H/\gamma_S = 100, 200, ..., 500$ and selected the settings which yielded the best robust accuracy against a 20-step ℓ_∞ PGD adversary with 10 restarts for $\epsilon = 8/255$ on the full training set; the range of γ_H/γ_S was set such that the corresponding losses were roughly equal for a randomly initialized, untrained network. For reporting results, we train five models using the selected hyperparameters and report results using the one with the median performance on the test set against a 20-step ℓ_∞ PGD adversary at $\epsilon = 8/255$ on CIFAR-10 or 0.3 on MNIST. For the PreActResNet18 model, robust accuracy over the 5 runs ranged from 40.1% to 41.5% with median 40.5%; clean accuracy ranged from 66.9% to 72.4% with median 70.0%.

For our stability results, we use the full version of AutoAttack+, an ensemble of attacks proposed by Croce and Hein (2020), for the ℓ_2 and ℓ_∞ perturbations. We choose the attack because it is parameter-free, which reduces the possibility of misconfiguration; empirically it is at least as strong as standard PGD, and has been successful in breaking many proposed defenses. For the Spatial, Recolor, JPEG, and NPTM adversaries, we used the public implementation by Laidlaw et al. (2021). For comparing with standard PGD, we use a 20-step PGD adversary with 10 random restarts and a step size of $2.5 \cdot \epsilon/20$ as implemented by Engstrom et al. (2019). For verification, we adopt the setup of Xiao et al. (2019), using the MIP verifier of Tjeng et al. (2017), with solves parallelized over 8 CPU cores and the timeout set to 120 seconds.

B ADDITIONAL THEORETICAL PERSPECTIVE ON THE PROPOSED REGULARIZERS

Section 5.3, and in particular, Equation 14, suggests computing the regularization term between all *pairs of (perturbed) input points*. We emphasize that this development is intended to provide a theoretically rigorous characterization of the behavior of our regularization functional in the ideal case when the number of samples goes to infinity. These results provide justification to the intuition that our proposed regularizers indeed penalize functions with respect to both the outputs and decision boundaries via a *precise* formulation as the gradient of an auxilliary function over the input manifold.

In contrast, the proposed regularizers (given in Equations 16 and 17) take the sum over *pairs of perturbations*. In this section, we present some arguments that, in the setting of our experiments, our regularizers can be viewed as a sparsified version of the dense regularizer considered in the previous section. The main observation is that, if the data is sufficiently sparse, the resampled Laplacian will consist of two types of edges: very short edges between points resampled from the

same ϵ-neighborhood, and very long edges between points sampled from different ϵ-neighborhoods. For an appropriate choice of the scale factor s, the exponential form of the heat kernel causes the weights on the long edges to fall off very quickly compared to the short edges. In fact, it is standard when computing such graph-based quantities to sparsify the Laplacian by truncating the weights using the k-nearest neighbors, or within some ϵ-ball.

The following results gives precise bounds for this approximation of the regularizer in Equation 14 under a certain separation assumption in the data:

Proposition B.1. *Consider a function f and a set of points $\{x_i\}_{i=1}^N$ such that the pairwise differences bounded from above and below by $0 < c_1 \leq ||f(x_i) - f(x_j)||_2^2 \leq c_2 < \infty$ for all $i \neq j$. Let s be the parameter of the heat kernel $k_s = \exp(-||x_i - x_j||^2/s)$. Assume further that the points $\{x_i\}$ are separated in the sense that there exist at least $O(n)$ pairs of points $(x_{i1}, x_{i2}), i \in S$ whose distances are bounded from above by $\sqrt{s}$, and the remainder of the points have distances bounded from below by $m\sqrt{s}$ for some constant $m > 1$. Writing $R(L) := \sum_{i,j=1}^N (f(x_i) - f(x_j))^2 L_{i,j}$ for the dense regularizer and $R(L_\epsilon)$ for the regularizer with weights truncated to an ϵ-ball, we have that*

$$R(L) - c_2 N^2 \exp(-m^2) \leq R(L_\epsilon) \leq R(L) \leq (1 + b^{-1}) R(L_\epsilon) \tag{19}$$

where $b = \Theta((c_1/c_2)(\exp(m^2)/N))$.

Proof of Proposition B.1. We introduce $\bar{L}_\epsilon := L - L_\epsilon$, which is just the Laplacian on the complement of the ϵ neighborhood subgraph, i.e., the subgraph whose edges are at least ϵ. Clearly $R(L) = R(\bar{L}_\epsilon) + R(L_\epsilon)$. Then we have the following bounds:

$$R(\bar{L}_\epsilon) = \sum_{i,j \notin S} (f(x_i) - f(x_j))^2 L_{i,j} \leq c_2 N^2 \exp(-m^2) \tag{20}$$

$$R(L_\epsilon) = \sum_{i,j \in S} (f(x_i) - f(x_j))^2 L_{i,j} \geq a c_1 N \exp(-1) \tag{21}$$

where the a is a constant introduced for the $O(N)$ edges in the graph of L_ϵ.

Take $b = (ac_1/c_2)(\exp(m^2 - 1)/N)$. A simple rearrangement gives $c_2 = (b^{-1} a c_1)(\exp(m^2 - 1)/N)$, thus we have that

$$\begin{aligned}
R(L) &= R(L_\epsilon) + R(\bar{L}_\epsilon) && (22)\\
&\leq R(L_\epsilon) + c_2 N^2 \exp(-m^2) && (23)\\
&\leq R(L_\epsilon) + \left[(b^{-1} a c_1)(\exp(m^2 - 1)/N)\right] N^2 \exp(-m^2) && (24)\\
&= R(L_\epsilon) + b^{-1} a c_1 N \exp(-1) && (25)\\
&\leq R(L_\epsilon) + b^{-1} R(L_\epsilon) && (26)\\
&= (1 + b^{-1}) R(L_\epsilon) && (27)
\end{aligned}$$

Then the first inequality follows from (23), the second inequality is trivial, and the third inequality follows from (22)–(27). □

Discussion In this work, we only consider bounded functions f (i.e., the output of a softmax layer, and the 0-1 activations of a neural network), which allows us to apply this proposition. This result gives two bounds on the dense regularization $R(L)$ in terms of $R(L_\epsilon)$. The first inequality says that the *absolute* approximation error is bounded by a term which vanishes unless the *squared* sample count N^2 grows as the exponential of the squared separation m^2. The third inequality gives a tighter bound on the *relative* error in a term that vanishes unless the *linear* sample count N grows as the exponential of the squared separation m^2, but requires that we have good control over the ratio c_1/c_2. Note that the dependence on c_1/c_2 is unavoidable in the sense that a function which is very smooth in local neighborhoods of size s will necessarily have large relative contribution from the longer edges; however, in this case we still have a bound on the absolute contribution (though we trade the ratio c_1/c_2 for a factor of N). Crucially, in both cases the error bound depends on an exponential factor of the squared separation m^2, which, under the curse of the dimensionality, grows with the input dimension. Note that the assumption that there is a linear number of close points is satisfied by our resampling procedure.

Table 4: Dense vs. sparse regularizer, CIFAR-10 against an ℓ_∞ PGD-adversary at $\epsilon = 8/255$.

Regularizer	Test Accuracy (%)	
	Robust	Clean
Sparse	40.54	69.95
Dense	35.04	64.08

Next, to provide some empirical support that the separation m can be made nontrivial, we offer the following statistics from CIFAR-10, the main setting of our experiments. We selected a random subset of 10,000 (20%) training samples. For each of these points, we found the closest point within the remaining 49,999 training samples, and computed the distance. Using the ℓ_2 metric, we found a mean distance of 9.597, a median distance of 9.405, and the distance at the 10th percentile to be 5.867. Conversely, in our experiments, we sample points in an ℓ_∞ ball of radius $\epsilon = 8/255$, which is contained in an ℓ_2 ball of radius $\sqrt{3 \cdot 32 \cdot 32} \cdot 8/255 = 1.739$. In fact, only 21 training samples, or 0.21% of the random subset, have a partner closer than twice the perturbation bounds. We note that these statistics also assume training over the entire dataset, rather than the Stochastic Gradient Descent algorithm commonly used in deep learning, which yields batches of data that are almost certainly sparse. Furthermore, as the dimension of the inputs grows, we would expect the distance between pairs of input points in the training distribution to increase exponentially; indeed, this is one of the manifestations of the oft-cited curse of dimensionality (Hastie et al., 2009).

Finally, we find in our experiments that the dense regularizer of Equation 14 performs worse than the sparse regularizer in practice, which agrees with the intuition that the pairwise distances between input points is too large to provide any additional regularization benefit (and in fact, actually slightly *hurts* performance due to an over-regularization effect). Table 4 presents a summary of these results. Our regularizer is also more efficient to compute by a linear factor, and since all the pairwise distances are constant, we can drop the Laplacian weights entirely.