On the Loss Landscape of Adversarial Training: Identifying Challenges and How to Overcome Them

Chen Liu[1] **Mathieu Salzmann**[1] **Tao Lin**[1] **Ryota Tomioka**[2] **Sabine Süsstrunk**[1]
[1] EPFL, Lausanne, Switzerland, {chen.liu, mathieu.salzmann, tao.lin, sabine.susstrunk}@epfl.ch
[2] Microsoft Research, Cambridge, UK, ryoto@microsoft.com

Abstract

We analyze the influence of adversarial training on the loss landscape of machine learning models. To this end, we first provide analytical studies of the properties of adversarial loss functions under different adversarial budgets. We then demonstrate that the adversarial loss landscape is less favorable to optimization, due to increased curvature and more scattered gradients. Our conclusions are validated by numerical analyses, which show that training under large adversarial budgets impede the escape from suboptimal random initialization, cause non-vanishing gradients and make the model find sharper minima. Based on these observations, we show that a periodic adversarial scheduling (PAS) strategy can effectively overcome these challenges, yielding better results than vanilla adversarial training while being much less sensitive to the choice of learning rate.

1 Introduction

State-of-the-art deep learning models have been found to be vulnerable to adversarial attacks [18, 34, 45]. Imperceptible perturbations of the input can make the model produce wrong predictions with high confidence. This raises concerns about deep learning's deployment in safety-critical applications.

Although many training algorithms have been proposed to counter such adversarial attacks, most of them were observed to fail when facing stronger attacks [4, 10]. Adversarial training [33] is one of the few exceptions, so far remaining effective and thus popular. It uses adversarial examples generated with the attacker's scheme to update the model parameters. However, adversarial training and its variants [2, 6, 24, 42, 53] have been found to have a much larger generalization gap [37] and to require larger model capacity for convergence [49]. Although recent works [6, 40] show that the adversarial training error reduces to almost 0% with a large enough model and that the generalization gap can be narrowed by using more training data, convergence in adversarial training remains much slower than in vanilla training on clean data. This indicates discrepancies in the underlying optimization landscapes. While much work has studied the loss landscape of deep networks in vanilla training [12, 13, 14, 15, 31], such an analysis in adversarial training remains unaddressed.

Here we study optimization in adversarial training. Vanilla training can be considered as a special case where no perturbation is allowed, i.e., zero adversarial budget. Therefore, we focus on the impact of the adversarial budget size on the loss landscape. In this context, we investigate from a theoretical and empirical perspective how different adversarial budget sizes affect the loss landscape to make optimization more challenging. Our analyses start with linear models and then generalize to nonlinear deep learning ones. We study the whole training process and identify different behaviors in the early and final stages of training. Based on our observations, we then introduce a scheduling strategy for the adversarial budget during training. We empirically show this scheme to yield better performance and to be less sensitive to the learning rate than vanilla adversarial training.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Contributions. Our contributions can be summarized as follows. 1) From a theoretical perspective, we show that, for linear models, adversarial training under a large enough budget produces a constant classifier. For general nonlinear models, we identify the existence of an abrupt change in the adversarial examples, which makes the loss landscape less smooth. This causes severe *gradient scattering* and slows down the convergence of training. 2) Our numerical analysis shows that training under large adversarial budgets hinders the model to escape from suboptimal initial regions, while also causing large non-vanishing gradients in the final stage of training. Furthermore, by Hessian analysis, we evidence that the minima reached in the adversarial loss landscape are sharper when the adversarial budget is bigger. 3) We show that a periodic adversarial scheduling (PAS) strategy, corresponding to a cyclic adversarial budget scheduling scheme with warmup, addresses these challenges. Specifically, it makes training less sensitive to the choice of learning rate and yields better robust accuracy than vanilla adversarial training without any computational overhead.

Notation and Terminology. We use plain letters, bold lowercase letters and bold uppercase letters to represent scalars, vectors and matrices, respectively. $\|\mathbf{v}\|$ represents the Euclidean norm of vector $\mathbf{v}$ and $[K]$ is an abbreviation of the set $\{0, 1, 2, ..., K-1\}$. In a classification problem $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $(\mathbf{x}_i, y_i) \in \mathbb{R}^m \times [K]$, the classifier consists of a logit function $f : \mathbb{R}^m \to \mathbb{R}^k$, which is usually a neural network, and a risk function $\ell : \mathbb{R}^k \times [K] \to \mathbb{R}$, which is the softmax cross-entropy loss. The adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$ of a data point $\mathbf{x}$, whose size is ϵ, is defined based on an l_p norm-based constraint $\{\mathbf{x}' | \|\mathbf{x} - \mathbf{x}'\|_p \leq \epsilon\}$, and we use $\mathcal{S}_\epsilon(\mathbf{x})$ to denote the l_∞ constraint for simplicity.

Given the model parameters $\theta \in \Theta$, we use $g(\mathbf{x}, \theta) : \mathbb{R}^m \times \Theta \to \mathbb{R}$ to denote the loss function for an individual data point, ignoring the label y for simplicity. If we use $\mathcal{L}_\epsilon(\theta)$ to denote the adversarial loss function under adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$, adversarial training solves the min-max problem

$$\min_\theta \mathcal{L}_\epsilon(\theta) := \frac{1}{N} \sum_{i=1}^N g_\epsilon(\mathbf{x}_i, \theta) \text{ where } \mathrm{g}_\epsilon(\mathbf{x}_\mathrm{i}, \theta) := \max_{\mathbf{x}_\mathrm{i}' \in \mathcal{S}_\epsilon^{(\mathrm{p})}(\mathbf{x}_\mathrm{i})} \mathrm{g}(\mathbf{x}_\mathrm{i}', \theta) \,. \tag{1}$$

$\mathcal{L}(\theta) := \mathcal{L}_0(\theta)$ is the vanilla loss function. If $\epsilon \neq 0$, the adversarial example $\mathbf{x}_i'$, i.e., the worst-case input in $\mathcal{S}_\epsilon^{(p)}(\mathbf{x}_i)$, depends on the model parameters. We call the landscape of functions $\mathcal{L}(\theta)$ and $\mathcal{L}_\epsilon(\theta)$ the vanilla and adversarial loss landscape, respectively. Similarly, we use $\mathcal{E}(\theta)$ and $\mathcal{E}_\epsilon(\theta)$ to represent the clean error and robust error under adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$. In this paper, we call a function smooth if it is C^1-continuous. We use θ_0 to denote the initial parameters. "Initial plateau" or "suboptimal region in the early stage of training" indicate the parameters that are close to the initial ones and have similar performance. "Vanilla training" means training based on clean input data, while "vanilla adversarial training" represents the popular adversarial training method in [33].

2 Related Work

Adversarial Robustness. In this work, we focus on white-box attacks, in which the attackers have access to the model parameters. Compared with black-box attacks, white box attacks better solve the inner maximization problem in (1). In this context, [18] proposes the fast gradient sign method (FGSM) to perturb the input in the direction of its gradient: $\mathbf{x}' = \mathbf{x} + \epsilon \operatorname{sign}(\nabla_\mathbf{x} \mathcal{L}(\theta))$. Projected gradient descent (PGD) [33] extends FGSM by iteratively running it with a smaller step size and projecting the perturbation back to the adversarial budget. Furthermore, PGD introduces randomness by starting at a random initial point inside the adversarial budget. As a result, PGD generates much stronger adversarial examples than FGSM and is believed to be the strongest attack utilizing the network's first order information [33].

When it comes to robustness against attacks, some methods have been proposed to train provably robust models by linear approximation [5, 29, 48], semi-definite programming [36], interval bound propagation [20] or randomized smoothing [8, 39]. However, these methods either only apply to a specific type of network, have a significant computational overhead, or are unstable. Furthermore, compared with adversarial training, they have been found to over-regularize the model and significantly decrease the clean accuracy [54].

As a result, we focus on PGD-based adversarial training, which first generates adversarial examples $\mathbf{x}'$ by PGD and then uses $\mathbf{x}'$ to optimize the model parameters θ. In all our experiments, the adversarial loss landscape is approximated by the loss of adversarial examples found by PGD.

Loss Landscape of Deep Neural Networks. Many existing works focus on the vanilla loss landscape of the objective function in deep learning. It is challenging, because the objective $\mathcal{L}(\theta)$ of a deep neural network is a high-dimensional nonconvex function, of which we only know very few properties. [26] proves the nonexistence of poor local minima for general deep nonlinear networks. [30] shows that stochastic gradient descent (SGD) can almost surely escape the saddle points and converge to a local minimum. For over-parameterized ReLU networks, SGD is highly likely to find a monotonically decreasing trajectory from the initialization point to the global optimum [38].

Furthermore, some works have studied the geometric properties of local minima in the loss landscape of neural networks. In this context, [27, 35] empirically show that sharp minima usually have larger generalization gaps than flat ones. Specifically, to improve generalization, [51] uses adversarial training to avoid converging to sharp minima in large batch training. However, the correspondence between sharp minima and poor generalization is based on empirical findings and sometimes controversial. For example, [11] shows counterexamples in ReLU networks by rescaling the parameters and claims that sharp minima can generalize as well as flat ones. Moreover, different minima of the loss function have been found to be well-connected. That is, there exist hyper-curves connecting different minima that are flat in the loss landscape [12, 14]. [55] further shows that the learned path connection can help us to effectively repair models that are vulnerable to backdoor or error-injection attacks. Recently, some methods have been proposed to visualize the loss landscape [31, 44], leading to the observation that networks of different architectures have surprisingly different landscapes. Compared with chaotic landscapes, smooth and locally near-convex landscapes make gradient-based optimization much easier.

All of the above-mentioned works, however, focus on networks that have been optimized with vanilla training. Here, by contrast, we study the case of adversarial training.

3 Theoretical Analysis

In this section, we conduct an analytical study of the difference between $\mathcal{L}_\epsilon(\theta)$ and $\mathcal{L}(\theta)$. We start with linear classification models and then discuss general nonlinear ones.

3.1 Linear Classification Models

For the simple but special case of logistic regression, i.e., $K = 2$, we can write the analytical form of $\mathcal{L}_\epsilon(\theta)$. We defer the detailed discussion of this case to Appendix A.1, and here focus on linear multi-class classification, i.e., $K \geq 3$. We parameterize the model by $\mathbf{W} := \{\mathbf{w}_i\}_{i=1}^K \in \mathbb{R}^{m\times K}$ and use $f(\mathbf{W}) = [\mathbf{w}_1^T\mathbf{x}, \mathbf{w}_2^T\mathbf{x}, ..., \mathbf{w}_K^T\mathbf{x}]$ as the logit function. Therefore, the vanilla loss function is convex as $g(\mathbf{x}, \mathbf{W}) = \log\Big(1 + \sum_{j\neq y} \exp^{(\mathbf{w}_j - \mathbf{w}_y)^T\mathbf{x}}\Big)$. Although $g_\epsilon(\mathbf{x}, \mathbf{W})$ is also convex, it is no longer smooth everywhere. It is then difficult to write a unified expression of $g_\epsilon(\mathbf{x}, \mathbf{W})$. So we start with the *version space* $\mathcal{V}_\epsilon$ of $g_\epsilon(\mathbf{x}, \mathbf{W})$ defined as $\mathcal{V}_\epsilon = \left\{\mathbf{W}\middle|(\mathbf{w}_i - \mathbf{w}_y)\mathbf{x}' \leq 0, \forall i \in [K], \mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x})\right\}$.

By definition, $\mathcal{V}_\epsilon$ is the smallest convex closed set containing all solutions robust under the adversarial budget $\mathcal{S}_\epsilon(\mathbf{x})$. The proposition below states that the version space $\mathcal{V}_\epsilon$ shrinks with larger values of ϵ.

Proposition 1. *Given the definition of the version space $\mathcal{V}_\epsilon$, then $\mathcal{V}_{\epsilon_2} \subseteq \mathcal{V}_{\epsilon_1}$ when $\epsilon_1 \leq \epsilon_2$.*

The proof of Proposition 1 is very straightforward, we put it in Appendix B.1.

In addition to $\mathcal{V}_\epsilon$, we define the set $\mathcal{T}_\epsilon$ as $\mathcal{T}_\epsilon = \left\{\mathbf{W}\middle|0 \in \arg\min_\gamma g_\epsilon(\mathbf{x}, \gamma\mathbf{W})\right\}$. $\mathcal{T}_\epsilon$ is the set of all directions in which the optimal point is the origin; that is, the corresponding models in this direction are all no better than a constant classifier. Although we cannot write the set $\mathcal{T}_\epsilon$ in roster notation, we show in the theorem below that $\mathcal{T}_\epsilon$ becomes larger as ϵ increases.

Theorem 1. *Given the definition of $\mathcal{T}_\epsilon$, then $\mathcal{T}_{\epsilon_2} \subseteq \mathcal{T}_{\epsilon_1}$ when $\epsilon_1 \geq \epsilon_2$. In addition, $\exists\bar{\epsilon}$ such that $\forall\epsilon \geq \bar{\epsilon}, \mathcal{T}_\epsilon = \mathbb{R}^{m\times K}$. In this case, $\mathbf{0} \in \arg\min_{\mathbf{W}} g_\epsilon(\mathbf{x}, \mathbf{W})$.*

We defer the proof of Theorem 1 to Appendix B.2, where we also provide a lower bound for $\bar{\epsilon}$. Theorem 1 indicates that when the adversarial budget is large enough, the optimal point is the origin. In this case, we will get a constant classifier, and training completely fails.

$\mathcal{L}_\epsilon(\mathbf{W})$ is the average of $g_\epsilon(\mathbf{x}, \mathbf{W})$ over the dataset, so Theorem 1 and Proposition 1 still apply if we replace g_ϵ with $\mathcal{L}_\epsilon$ in the definition of $\mathcal{V}_\epsilon$ and $\mathcal{T}_\epsilon$. For nonlinear models like deep neural networks, these conclusions will not hold because $g_\epsilon(\mathbf{x}, \theta)$ is no longer convex. Nevertheless, our experiments in Section 4.1 evidence the same phenomena as indicated by the theoretical analysis above. Larger ϵ make it harder for the optimizer to escape the initial suboptimal region. In some cases, training fails, and we obtain a constant classifier in the end.

3.2 General Nonlinear Classification Models

For deep nonlinear neural networks, we cannot write the analytical form of $g(\mathbf{x}, \theta)$ or $g_\epsilon(\mathbf{x}, \theta)$. To analyze such models, we follow [43] and assume the smoothness of the function g.

Assumption 1. *The function g satisfies the following Lipschitzian smoothness conditions:*

$$\begin{aligned} \|g(\mathbf{x}, \theta_1) - g(\mathbf{x}, \theta_2)\| &\leq L_\theta \|\theta_1 - \theta_2\| \ , \\ \|\nabla_\theta g(\mathbf{x}, \theta_1) - \nabla_\theta g(\mathbf{x}, \theta_2)\| &\leq L_{\theta\theta} \|\theta_1 - \theta_2\| \ , \\ \|\nabla_\theta g(\mathbf{x}_1, \theta) - \nabla_\theta g(\mathbf{x}_2, \theta)\| &\leq L_{\theta\mathbf{x}} \|\mathbf{x}_1 - \mathbf{x}_2\|_p \ . \end{aligned} \tag{2}$$

Based on this, we study the smoothness of $\mathcal{L}_\epsilon(\theta)$.

Proposition 2. *If Assumption 1 holds, then we have* [1]

$$\begin{aligned} \|\mathcal{L}_\epsilon(\theta_1) - \mathcal{L}_\epsilon(\theta_2)\| &\leq L_\theta \|\theta_1 - \theta_2\| \ , \\ \|\nabla_\theta \mathcal{L}_\epsilon(\theta_1) - \nabla_\theta \mathcal{L}_\epsilon(\theta_2)\| &\leq L_{\theta\theta} \|\theta_1 - \theta_2\| + 2\epsilon L_{\theta\mathbf{x}} \ . \end{aligned} \tag{3}$$

The proof is provided in Appendix B.3, in which we can see the upper bound in Proposition 2 is tight and can be achieved in the worst cases. Proposition 2 shows that the first-order smoothness of the objective function is preserved under adversarial attacks, but the second-order smoothness is not. That is to say, gradients in arbitrarily small neighborhoods in the θ-space can change discontinuously.

The unsatisfying second-order property arises from the maximization operator defined in the functions g_ϵ and $\mathcal{L}_\epsilon$. For function $g_\epsilon(\mathbf{x}, \theta)$, the non-smooth points are those where the optimal adversarial example $\mathbf{x}'$ changes abruptly in a sufficiently small neighborhood. Formally, we use θ_1 and $\mathbf{x}'_1$ to represent the model parameters and the corresponding optimal adversarial example. We assume different gradients of the model parameters for different inputs. If there exists a positive number $a > 0$ such that, $\forall \delta > 0$, we can find $\theta_2 \in \{\theta | \|\theta - \theta_1\| \leq \delta\}$, and the corresponding optimal adversarial example $\mathbf{x}'_2$ satisfies $\|\mathbf{x}'_1 - \mathbf{x}'_2\|_p > a$, then $\lim_{\theta \to \theta_1} \nabla_\theta g_\epsilon(\mathbf{x}, \theta) \neq \nabla_\theta g_\epsilon(\mathbf{x}, \theta_1)$. $\mathcal{L}_\epsilon(\theta)$ is the aggregation of $g_\epsilon(\mathbf{x}, \theta)$ over the dataset, so it also has such non-smooth points. In addition, as the $2\epsilon L_{\theta\mathbf{x}}$ term in the second inequality of (3) indicates, the adversarial examples can change more under a larger adversarial budget. As a result, the (sub)gradients $\nabla_\theta \mathcal{L}_\epsilon(\theta)$ can change more abruptly in the neighborhood of the parameter space. That is, the (sub)gradients are more *scattered* in the adversarial loss landscape.

Figure 1 provides a 2D sketch diagram showing the non-smoothness introduced by adversarial training. The red curve represents the vanilla loss function $g(\mathbf{x}, \theta)$. Under adversarial perturbation, the loss landscape fluctuates within the light blue band. Then, the blue curve represents the worst case we can encounter in the adversarial setting, i.e., $g_\epsilon(\mathbf{x}, \theta)$. We can see that the blue curve is not smooth any more at the point where $\theta = 0$. Importantly, as the light blue band becomes wider under a larger adversarial budget, the corresponding non-smooth point becomes sharper, which means that the difference between the gradients on both sides of the non-smooth point becomes larger.

Based on Proposition 2, we show in the following theorem that the non-smoothness introduced by adversarial training makes the optimization by stochastic gradient descent (SGD) more difficult.

Theorem 2. *Let Assumption 1 hold, the stochastic gradient $\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)$ be unbiased and have bounded variance, and the SGD update $\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)$ use a constant step size $\alpha_t = \alpha = \frac{1}{L_{\theta\theta}\sqrt{T}}$ for T iterations. Given the trajectory of the parameters during optimization $\{\theta_t\}_{t=1}^T$, then we can bound the asymptotic probability of large gradients for a sufficient large value of T as*

$$\forall \gamma \geq 2, P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > \gamma \epsilon L_{\theta\mathbf{x}}) < \frac{4}{\gamma^2 - 2\gamma + 4} \ . \tag{4}$$

[1] Strictly speaking, $\mathcal{L}_\epsilon(\theta)$ is not differentiable at some point, so $\nabla_\theta \mathcal{L}_\epsilon(\theta)$ might be ill-defined. In this paper, we use $\nabla_\theta \mathcal{L}_\epsilon(\theta)$ for simplicity. Nevertheless, the inequality holds for any subgradient $\mathbf{v} \in \partial_\theta \mathcal{L}_\epsilon(\theta)$.

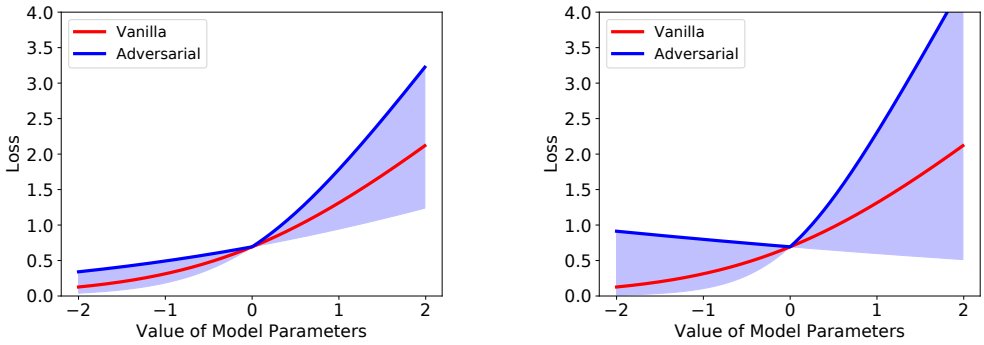


Figure 1: 2D sketch diagram showing the vanilla and adversarial loss landscapes. The clean input data x is 1.0 and loss function $g(x, \theta) = \log(1 + \exp(\theta x))$. The landscape is shown in the parameter interval $\theta \in [-2, 2]$ under a small adversarial budget (left, $\epsilon = 0.6$) and a large adversarial budget (right, $\epsilon = 1.2$). The function $g_\epsilon(\mathbf{x}, \theta)$ is not smooth at $\theta = 0$.

We provide the proof in Appendix B.4. In vanilla training, ϵ is 0 and $\mathcal{L}(\theta)$ is smooth, and (4) implies that $\lim_{t\to+\infty} \|\nabla_\theta g_\epsilon(\theta_t)\| = 0$ almost surely. This is consistent with the fact that SGD converges to a critical point with non-convex smooth functions. By contrast, in adversarial training, i.e., $\epsilon > 0$, we cannot guarantee convergence to a critical point. Instead, the gradients are non-vanishing, and we can only bound the probability of obtaining gradients whose magnitude is larger than $2\epsilon L_{\theta\mathbf{x}}$. For a fixed value of $C := \gamma\epsilon L_{\theta\mathbf{x}}$ larger than $2\epsilon L_{\theta\mathbf{x}}$, the inequality (4) indicates that the probability $P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > C)$ increases quadratically with ϵ.

In deep learning practice, activation functions like sigmoid, tanh and ELU [7] satisfy the second-order smoothness in Assumption 1, but the most popular ReLU function does not. Nevertheless, adversarial training still causes *gradient scattering* and makes the optimization more difficult. That is, the bound of $\|\nabla_\theta \mathcal{L}_\epsilon(\theta_1) - \nabla_\theta \mathcal{L}_\epsilon(\theta_2)\|$ still increases with ϵ, and the parameter gradients change abruptly in the adversarial loss landscape. We provide a more detailed discussion of this phenomenon in Appendix A.2, which shows that our analysis and conclusions easily extend to the ReLU case.

The second-order Lipchitz constant indicates the magnitude of the gradient change for a unit change in parameters. Therefore, it is a good quantitative metric of gradient scattering. In practice, we are more interested in the effective local Lipschitz constant, which only considers the neighborhood of the current parameters, than in the global Lipschitz constant. In this case, the effective local second-order Lipschitz constant can be estimated by the top eigenvalues of the Hessian matrix $\nabla^2_\theta \mathcal{L}_\epsilon(\theta)$.

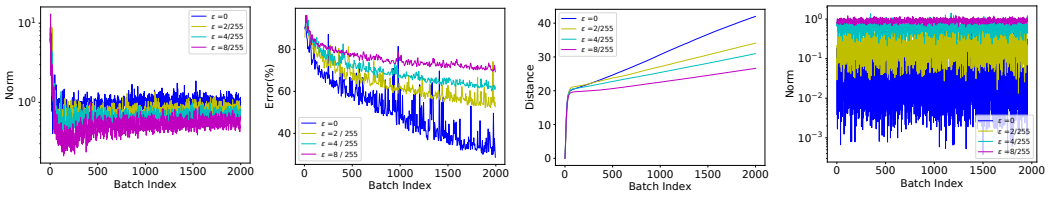
4 Numerical Analysis

In this section, we conduct experiments on MNIST and CIFAR10 to empirically validate the theorems in Section 3. Detailed experimental settings are provided in Appendix C.1. Unless specified, we use LeNet models on MNIST and ResNet18 models on CIFAR10 in this and the following sections. Our code is available on https://github.com/liuchen11/AdversaryLossLandscape.

4.1 Gradient Magnitude

In Section 3.1, we have shown that the training algorithm will get stuck at the origin and yield a constant classifier for linear models under large ϵ. For deep nonlinear models, the initial value of the parameters is close to the origin under most popular initialization schemes [17, 22]. Although Theorem 1 is not applicable here, we are still interested in investigating how effective gradient-based optimization is at escaping from the suboptimal initial parameters. To this end, we track the norm of the stochastic gradient $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$, the robust error $\mathcal{E}_\epsilon(\theta)$ in the training set and the distance from the initial point $\|\theta - \theta_0\|$ during the first 2000 mini-batch updates for CIFAR10 models. Figure 2a, 2b, 2c evidence a clear difference between the models trained with different values of ϵ. When ϵ is small, the gradient magnitude is larger, and the model parameters move faster. Correspondingly, the training error decreases faster, which means that the model quickly escapes the initial suboptimal region. By contrast, when ϵ is large, the gradients are small, and the model gets stuck in the initial region.

This implies that the loss landscape under a large adversarial budget impedes the escape from initial suboptimal plateaus in the early stage of training.



(a) $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$, first 2000. (b) $\mathcal{E}_\epsilon(\theta)$, first 2000. (c) $\|\theta - \theta_0\|$, first 2000. (d) $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$, last 2000.

Figure 2: Norm of the stochastic gradient $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$, robust training error $\mathcal{E}_\epsilon(\theta)$, and distance from the initial point $\|\theta - \theta_0\|$ during the first or last 2000 mini-batch updates for CIFAR10 models.

For ReLU networks, adversarially-trained models have been found to have sparser weights and intermediate activations [9], i.e., they have more dead neurons. Dead neurons are implicitly favored by adversarial training, because the output is independent of the input perturbation. Note that training fails when all the neurons in one layer are dead for all training instances. The model is then effectively broken into two parts by this dead layer: the preceding layers will no longer be trained because the gradients are all blocked; the following layers do not depend on the input and thus give constant outputs. In essence, training is then stuck in a parameter space that only includes constant classifiers. In practice, this usually happens when the model has small width and the value of ϵ is large. This is consistent with previous findings that adversarial training needs higher model capacity [33] and too strong adversarial examples are harmful in the early stage of training [46].

Theorem 2 indicates that the gradients are non-vanishing in adversarial training and more likely to have large magnitude under large values of ϵ. This is validated by Figure 2d, in which we report the norm of the stochastic gradient $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$ in the last 2000 mini-batch updates for CIFAR10 models. In vanilla training, the gradient is almost zero in the end, indicating that the optimizer finds a critical point. In this case $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$ is dominated by the variance introduced by stochasticity. However, $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$ increases with ϵ. When ϵ is larger, $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$ is also larger and non-vanishing, indicating that the model is still bouncing around the parameter space at the end of training.

The decreased gradient magnitude in the initial suboptimal region and the increased gradient magnitude in the final near-minimum region indicate that the adversarial loss landscape is not favorable to optimization when we train under large adversarial budgets. Additional results on MNIST models are provided in Figure 8 of Appendix C.2.1, where the same observations can be made.

4.2 Hessian Analysis

To study the effective local Lipschitz constant of $\mathcal{L}_\epsilon(\theta)$, we analyze the Hessian spectrum of models trained under different values of ϵ. It is known that the curvature in the neighborhood of model parameters is dominated by the top eigenvalues of the Hessian matrix $\nabla^2 \mathcal{L}_\epsilon(\theta)$. To this end, we use the power iteration method as in [51] to iteratively estimate the top 20 eigenvalues and the corresponding eigenvectors of the Hessian matrix. Furthermore, to discard the effect of the scale of function $\mathcal{L}_\epsilon(\theta)$ for different ϵ, we estimate the scale of $\mathcal{L}_\epsilon(\theta)$ by randomly sampling θ. We then normalize the top Hessian eigenvalues by the average value of $\mathcal{L}_\epsilon(\theta)$ on these random samples. In addition, we show the learning curve of $\mathcal{L}_\epsilon(\theta)$ on the training set during training in Figure 11 of Appendix C.2.2. It clearly show similar magnitude of $\mathcal{L}_\epsilon(\theta)$ for different values of ϵ.

In Figure 3, we show the top 20 Hessian eigenvalues, both before and after normalization, of CIFAR10 models under different adversarial budgets. We also provide 3D visualizations of the neighborhood in the directions of the top 2 eigenvectors in Figure 12 of Appendix C.2.2. It is clear that the local effective second-order Lipschitz constant of the model obtained consistently increases with the value of ϵ. That is, the minima found in $\mathcal{L}_\epsilon(\theta)$ are sharper under larger ϵ.

To validate the claim in Section 3.2 that non-smoothness arises from abrupt changes of the adversarial examples, we study the similarity of adversarial perturbations generated by different model parameter values in a small neighborhood. Specifically, we perturb the model parameters θ in opposite directions

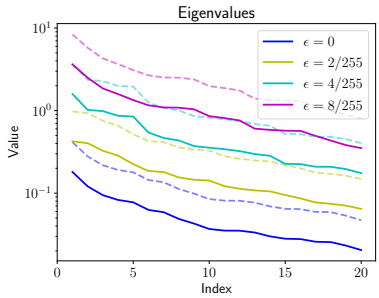


Figure 3: Top 20 eigenvalues of the Hessian matrix for ResNet18 models. We show the normalized (solid) and original (dashed) values.

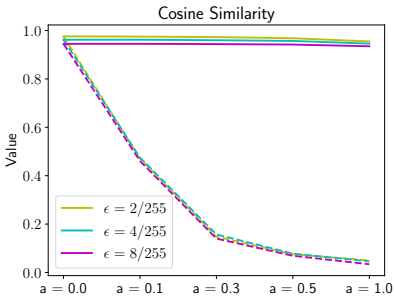


Figure 4: Cosine similarity between perturbations $\mathbf{x}'_{a\mathbf{v}} - \mathbf{x}$ and $\mathbf{x}'_{-a\mathbf{v}} - \mathbf{x}$. $\mathbf{v}$ is either the top eigenvector (dashed) or random (solid).

to $\theta + a\mathbf{v}$ and $\theta - a\mathbf{v}$, where $\mathbf{v}$ is a unit vector and a is a scalar. Let $\mathbf{x}'_{a\mathbf{v}}$ and $\mathbf{x}'_{-a\mathbf{v}}$ represent the adversarial examples generated by the corresponding model parameters. We then calculate the average cosine similarity between the perturbation $\mathbf{x}'_{a\mathbf{v}} - \mathbf{x}$ and $\mathbf{x}'_{-a\mathbf{v}} - \mathbf{x}$ over the training set.

The results on CIFAR10 models are provided in Figure 4. To account for the random start in PGD, we run each experiment 4 times and report the average value. The variances of all experiments are smaller than 0.005 and thus not shown in the figure. Note that, when $\mathbf{v}$ is a random unit vector, the robust error $\mathcal{E}_\epsilon(\theta)$ of the parameters $\theta \pm a\mathbf{v}$ on both the training and test sets remains unchanged for different values of a, indicating a flat landscape in the direction $\mathbf{v}$. The adversarial examples in this case are mostly similar and have very high cosine similarity. By contrast, if $\mathbf{v}$ is the top eigenvector of the Hessian matrix, i.e., the most curvy direction, then we see a sharp increase in the robust error $\mathcal{E}_\epsilon(\theta)$ when we increase a. Correspondingly, the cosine similarity between the adversarial perturbations is much lower, which indicates dramatic changes of the adversarial examples. We perform the same experiments on MNIST models in Appendix C.2.2 with the same observations.

5 Periodic Adversarial Scheduling

In Sections 3 and 4, we have theoretically and empirically shown that the adversarial loss landscape becomes less favorable to optimization under large adversarial budgets. In this section, we introduce a simple adversarial budget scheduling scheme to overcome these problems.

Inspired by the learning rate warmup heuristic used in deep learning [19, 25], we introduce warmup for the adversarial budget. Let d be the current epoch index and D be the warmup period's length. We define a cosine scheduler ϵ_{cos} and a linear scheduler ϵ_{lin}, parameterized by ϵ_{max} and ϵ_{min}, as

$$\epsilon_{cos}(d) = \frac{1}{2}(1 - \cos\frac{d}{D}\pi)(\epsilon_{max} - \epsilon_{min}) + \epsilon_{min},\ \epsilon_{lin}(d) = (\epsilon_{max} - \epsilon_{min})\frac{d}{D} + \epsilon_{min}\ . \quad (5)$$

We clip $\epsilon_{cos}(d)$ and $\epsilon_{lin}(d)$ between 0 and ϵ_{target}, the target value of ϵ. If $\epsilon_{min} \leq 0$ and $\epsilon_{max} > \epsilon_{target}$, the value of ϵ starts from 0, gradually increases to ϵ_{target} and remains constant then.

This warmup strategy allows us to overcome the fact, highlighted in the previous sections, that adversarial training is more sensitive to the learning rate under a large budget because the gradients are more scattered. This is evidenced by Figure 5, which compares the robust test error of MNIST models relying on different adversarial budget scheduling schemes. For all models, we used $\epsilon = 0.4$, and report results after 100 epochs with different but constant learning rates in Adam [28]. Our linear and cosine schedulers perform better than using a constant value of ϵ during training and yield good performance for a broader range of learning rates: in the small learning rate regime, they speed up training; in the large learning rate regime, they stabilize training and avoid divergence. Note that, as shown in Appendix C.2.3, warmup of the learning rate does not yield similar benefits.

As shown in [25], periodic learning rates enable model ensembling to improve the performance. Here, we can follow the same strategy but also for the adversarial budget. To this end, we divide the training phase into several periods and store one model at the end of each period. We make final predictions based on the ensemble of these models. This periodic scheme has no computational overhead. We call it periodic adversarial scheduling (PAS).

As before, we run experiments on MNIST and CIFAR10. For MNIST, we train each model for 100 epochs and do not use a periodic scheduling for the learning rate, which we found not to improve the results even if we use a constant adversarial budget. For CIFAR10, we train each model for 200 epochs. When there are no learning rate resets, our results indicate the final model after 200 epochs. When using a periodic learning rate, we divide the 200 epochs into 3 periods, i.e., we reset the learning rate and the adversarial budget after 100 and 150 epochs, and compute the results using an ensemble of these 3 models. The value of learning rate and the adversarial budget size are calculated based on the ratio of the current epoch index to the current period length. We provide more details about hyper-parameter settings in Appendix C.1.

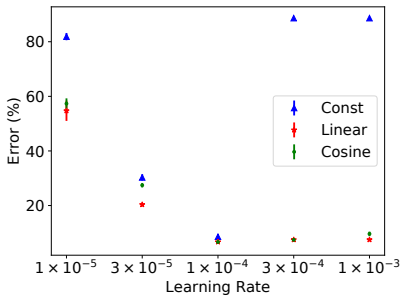


Figure 5: Mean and standard deviation of the test error under different learning rates with Adam and adversarial budget scheduling.

Task	Periodic Learning Rate	ϵ Scheduler	Clean Error (%)	Robust Error (%)				
				PGD (%)	PGD100 (%)	APGD100 CE (%)	APGD100 DLR (%)	Square5K (%)
MNIST LeNet $\epsilon = 0.4$	No	Constant	1.56(17)	8.58(89)	10.86(143)	15.18(155)	14.70(136)	19.58(45)
		Cosine	1.08(2)	6.64(70)	8.46(82)	14.36(134)	13.46(129)	16.78(25)
		Linear	1.06(6)	6.69(59)	8.79(116)	13.91(150)	13.17(120)	17.05(47)
CIFAR10 VGG $\epsilon = 8/255$	No	Constant	28.25(47)	56.22(43)	56.19(32)	58.18(46)	58.65(69)	54.37(29)
		Cosine	25.06(19)	56.06(48)	56.00(42)	57.83(45)	58.88(16)	53.95(15)
		Linear	23.56(95)	56.09(14)	55.88(5)	57.74(16)	58.39(18)	53.66(24)
	Yes	Constant	28.33(81)	54.24(28)	54.16(26)	55.45(26)	56.56(4)	52.85(18)
		Cosine	23.91(21)	53.18(21)	53.10(18)	54.44(16)	55.80(24)	51.41(37)
		Linear	21.88(33)	53.03(14)	52.97(17)	54.32(17)	55.63(17)	51.28(4)
CIFAR10 ResNet18 $\epsilon = 8/255$	No	Constant	18.62(6)	55.00(8)	54.97(9)	57.26(13)	56.60(25)	50.59(19)
		Cosine	18.43(26)	53.95(23)	53.85(21)	56.16(18)	55.77(24)	49.60(18)
		Linear	18.55(14)	53.46(20)	53.41(10)	55.69(17)	55.45(22)	49.66(28)
	Yes	Constant	21.00(5)	48.98(25)	48.87(25)	50.29(27)	50.98(6)	46.84(9)
		Cosine	19.90(18)	48.57(25)	48.49(27)	49.71(22)	50.54(9)	46.19(11)
		Linear	20.26(28)	48.60(13)	48.52(13)	49.73(9)	50.68(11)	46.47(26)

Table 1: Comparison between different adversarial budget schedulers under different adversarial attacks. *Cosine / Linear schedulers* are consistently better than *constant schedulers*. The number between brackets indicate the standard deviation across different runs. Specifically, for example, $1.56(17)$ stands for 1.56 ± 0.17.

We compare different scheduler in adversarial budget under different tasks and settings. We evaluate the robustness of our trained models by different kinds of attacks. First we evaluate the models under the PGD attack used in training (PGD), i.e., 50-iteration PGD for MNIST models and 10-iteration PGD for CIFAR10 models. Then, we increase the number of iterations in PGD and compute the robust error under 100-iteration PGD. To solve the issue of suboptimal step size, we also evaluate our models using the state-of-the-art AutoPGD attack [10], which search for the optimal step sizes. We run AutoPGD for 100 iterations for evaluation, based on either cross-entropy loss (APGD100 CE) or the difference of logit ratio loss (APGD100 DLR). To avoid gradient masking, we also run the state-of-the-art black-box SquareAttack [3] for 5000 iterations (Square5K). The hyperparameter details are defered in Appendix C.1.

The results are summarized in Table 1, where we compare the clean and robust accuracy under different adversarial attacks on the test set. It is clear that our proposed cosine or linear schedulers yield better performance, in both clean accuracy and robust accuracy, than using a constant adversarial budget in all cases. For MNIST, warmup not only makes training robust to different choices of learning rate, but also improves the final robust accuracy. For CIFAR10, model ensembling enabled by the periodic scheduler improves the robust accuracy.

6 Discussion

Model capacity. In addition to the size of the adversarial budget, the capacity of the model also greatly affects the adversarial loss landscape and thus the performance of adversarial training.

Adversarial training needs higher model capacity in two aspects: if we decrease the model capacity, adversarial training will fail to converge while vanilla training still works [33]; if we increase the model capacity, the robust accuracy of adversarial training continues to rise while the clean accuracy of normal training saturates [50]. Furthermore, we show in Appendix C.2.4 that smaller models are more likely to have dead layers because of their lower dimensionality. As a result, warmup in adversarial budget is also necessary for small models. In many cases, the parameter space of small models has good minima in terms of robustness, but adversarial training with a constant value of ϵ fails to find them. For example, one can obtain small but robust models by pruning large ones [21, 52].

Architecture. The network architecture encodes the parameterization of the model, so it greatly affects the adversarial loss landscape. For example, in Table 1, ResNet18 has fewer trainable parameters but better performance than VGG on CIFAR10, indicating that ResNet18 has a better parameterization in terms of robustness. Since the optimal architecture for adversarial robustness is not necessarily the same as the one for clean accuracy, we believe that finding architectures inherently favorable to adversarial training is an interesting but challenging topic for future research.

Connectivity of minima. Local minima in the vanilla loss landscape are well-connected [12, 14]: there exist flat hyper curves connecting them. In Appendix C.2.5, we study the connectivity of converged model parameters in the adversarial setting. We find that the parameters of two adversarially trained models are less connected in the adversarial loss landscape than in the vanilla setting. That is, the path connecting them needs to go over suboptimal regions.

Adversarial example generation We approximate the adversarial loss using adversarial examples generated by PGD, which is a good estimate of the inner maximization in (1). PGD-based adversarial training updates model parameters by near-optimal adversarial examples. However, recent works [41, 47] have shown that robust models can also be trained by suboptimal adversarial examples, which are faster to obtain. The formulation of these methods differs from (1), because the inner maximization problem is not approximately solved. Understanding why models (partially) trained on suboptimal adversarial examples are resistant to stronger adversarial examples needs more investigation.

7 Conclusion

We have studied the properties of the loss landscape under adversarial training. We have shown that the adversarial loss landscape is non-smooth and not favorable to optimization, due to the dependency of adversarial examples on the model parameters. Furthermore, we have empirically evidenced that large adversarial budgets slow down training in the early stages and impedes convergence in the end. Finally, we have demonstrated the advantages of warmup and periodic scheduling of the adversarial budget size during training. They make training more robust to different choices of learning rate and yield better performance than vanilla adversarial training.

8 Broader Impact

The existence of adversarial examples has raised serious concerns about the deployment of deep learning models in safety-sensitive domains, such as medical imaging [32] and autonomous navigation [1]. In these domains, as in many others, adversarial training remains the most popular, effective, and general method to train robust models. By studying the nature of optimization in adversarial training and proposing solutions to overcome the underlying challenges, our work has potential for high societal impact in these fields. Although the robust accuracy is much lower than the clean accuracy so far, the intrinsic properties of adversarial training we have discovered open up future research directions to improve its performance. From an ecological perspective, however, we acknowledge that the higher computational cost of adversarial training translates to higher carbon footprint than vanilla training. Nevertheless, we believe that the potential societal benefits of robustness to attacks outweigh this drawback.

9 Acknowledgements

We thankfully acknowledge the support of the Hasler Foundation (Grant No. 16076) for this work.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

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

[3] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. *arXiv preprint arXiv:1912.00049*, 2019.

[4] 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.

[5] Mislav Balunovic and Martin Vechev. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2020.

[6] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 11190–11201, 2019.

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[8] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[9] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. *arXiv preprint arXiv:1810.07481*, 2018.

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

[11] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

[12] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*, 2018.

[13] Stanislav Fort and Stanislaw Jastrzebski. Large scale structure of neural network loss landscapes. *arXiv preprint arXiv:1906.04724*, 2019.

[14] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pages 8789–8798, 2018.

[15] Mario Geiger, Stefano Spigler, Stéphane d'Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, and Matthieu Wyart. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115, 2019.

[16] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. *arXiv preprint arXiv:1901.10159*, 2019.

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

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

[19] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint arXiv:1810.13243*, 2018.

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

[21] Shupeng Gui, Haotao N Wang, Haichuan Yang, Chen Yu, Zhangyang Wang, and Ji Liu. Model compression with adversarial robustness: A unified optimization framework. In *Advances in Neural Information Processing Systems*, pages 1283–1294, 2019.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. *arXiv preprint arXiv:1901.09960*, 2019.

[25] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[26] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

[27] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] J Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 1(2):3, 2017.

[30] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

[31] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

[32] Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition*, page 107332, 2020.

[33] 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.

[34] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[35] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

[36] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[37] Leslie Rice, Eric Wong, and J Zico Kolter. Overfitting in adversarially robust deep learning. *arXiv preprint arXiv:2002.11569*, 2020.

[38] Itay Safran and Ohad Shamir. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, pages 774–782, 2016.

[39] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *arXiv preprint arXiv:1906.04584*, 2019.

[40] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pages 5014–5026, 2018.

[41] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

[42] Abhishek Sinha, Mayank Singh, Nupur Kumari, Balaji Krishnamurthy, Harshitha Machiraju, and Vineeth N Balasubramanian. Harnessing the vulnerability of latent layers in adversarially trained models. *arXiv preprint arXiv:1905.05186*, 2019.

[43] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018.

[44] Ivan Skorokhodov and Mikhail Burtsev. Loss surface sightseeing by multi-point optimization. *arXiv preprint arXiv:1910.03867*, 2019.

[45] 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.

[46] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *ICML*, volume 1, page 2, 2019.

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

[48] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, pages 8410–8419, 2018.

[49] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training. *arXiv preprint arXiv:1906.03787*, 2019.

[50] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *International Conference on Learning Representations*, 2020.

[51] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, pages 4949–4959, 2018.

[52] Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 2, 2019.

[53] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems*, pages 227–238, 2019.

[54] 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.

[55] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. In *International Conference on Learning Representations*, 2020.

A Theoretical Analysis

A.1 Binary Logistic Regression

In this section, we discuss binary logistic regression. In this case, $K = 2$ and the logit function is $f(\mathbf{w}) = \left[\mathbf{w}^T\mathbf{x}, -\mathbf{w}^T\mathbf{x}\right]$, where $\mathbf{w} \in \mathbb{R}^m$ is the only trainable parameter. If we use $+1$ and -1 to label both classes, then the overall loss function for a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ is $\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^N \log\Big(1 + e^{-y_i\mathbf{w}^T\mathbf{x}_i}\Big)$. Under the adversarial budget $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$, the corresponding adversarial loss function is $\mathcal{L}_\epsilon(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^N \log\Big(1 + e^{-y_i\mathbf{w}^T\mathbf{x}_i + \epsilon\|\mathbf{w}\|_q}\Big)$, where l_q is the dual norm of l_p. Since the magnitude of $\mathbf{w}$ does not change the results of the classifier, we can assume $\|\mathbf{w}\|_q = 1$ without loss of generality. As a result, the adversarial loss function is

$$\mathcal{L}_\epsilon(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^N \log\Big(1 + e^{-y_i\mathbf{w}^T\mathbf{x}_i + \epsilon}\Big) \ . \tag{6}$$

The following theorem describes the properties of $\mathcal{L}_\epsilon(\mathbf{w})$ for different values of ϵ.

Theorem 3. *If the dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ is linearly separable under the adversarial budget $\mathcal{S}_{\widehat{\epsilon}}(\mathbf{x})$, then for any unit vector $\mathbf{m} \in \mathbb{R}^m$ and values ϵ_1, ϵ_2 such that $\epsilon_1 \le \epsilon_2 \le \widehat{\epsilon}$, we have $\mathbf{m}^T\nabla^2_{\mathbf{w}}\mathcal{L}_{\epsilon_1}(\mathbf{w})\mathbf{m} \le \mathbf{m}^T\nabla^2_{\mathbf{w}}\mathcal{L}_{\epsilon_2}(\mathbf{w})\mathbf{m}$. More specifically, both the largest and smallest eigenvalue of $\nabla^2_{\mathbf{w}}\mathcal{L}_{\epsilon_1}(\mathbf{w})$ are no greater than those of $\nabla^2_{\mathbf{w}}\mathcal{L}_{\epsilon_2}(\mathbf{w})$.*

We provide the proof of Theorem 3 in Appendix B.5. Since $\mathbf{m}^T\nabla^2_{\mathbf{w}}\mathcal{L}_\epsilon(\mathbf{w})\mathbf{m}$ is the curvature of $\mathcal{L}_\epsilon(\mathbf{w})$ in the direction of $\mathbf{m}$, Theorem 3 shows that the curvature of $\mathcal{L}_\epsilon(\mathbf{w})$ increases with ϵ in any direction if the whole dataset is linearly separable. For an individual data point $\mathbf{x}$, if $\forall \mathbf{x}' \in \mathcal{S}_{\widehat{\epsilon}}(\mathbf{x})$, $\mathbf{x}'$ is correctly classified, then the curvature of $g_\epsilon(\mathbf{x}, \mathbf{w})$ also increases with ϵ in any direction as long as $\epsilon \le \widehat{\epsilon}$. The assumption for an individual point here is much weaker than the one in Theorem 3. If the overwhelming majority of the data points are correctly classified under the adversarial budget $\mathcal{S}_{\widehat{\epsilon}}$, the conclusion still holds in practice.

A.2 Discussions of ReLU Networks

Unlike sigmoid or tanh, ReLU is not a smooth function. However, it is smooth *almost everywhere*, except at 0. As a result, we can make the following assumptions for the function g represented by a ReLU network.

Assumption 2. *The function g satisfies the following conditions:*

$$\begin{aligned} \|g(\mathbf{x}, \theta_1) - g(\mathbf{x}, \theta_2)\| &\le L_\theta\|\theta_1 - \theta_2\| \ , \\ \|\nabla_\theta g(\mathbf{x}, \theta_1) - \nabla_\theta g(\mathbf{x}, \theta_2)\| &\le L_{\theta\theta}\|\theta_1 - \theta_2\| + D_{\theta\theta} \ , \\ \|\nabla_\theta g(\mathbf{x}_1, \theta) - \nabla_\theta g(\mathbf{x}_2, \theta)\| &\le L_{\theta\mathbf{x}}\|\mathbf{x}_1 - \mathbf{x}_2\|_p + D_{\theta\mathbf{x}} \ . \end{aligned} \tag{7}$$

We adjust the second-order smoothness assumption by adding two constants $D_{\theta\theta}$, $D_{\theta\mathbf{x}}$. They are the upper bound of the gradient difference in the neighborhood of non-smooth points. Therefore, they measures how abruptly the (sub)gradients can change in a sufficiently small region in the parameter space and can be considered as a quantitative measure of *gradient scattering*.

The following corollary states the properties of g_ϵ under Assumption 2.

Corollary 1. *If Assumption 2 is satisfied, then we have*

$$\begin{aligned} \|\mathcal{L}_\epsilon(\theta_1) - \mathcal{L}_\epsilon(\theta_2)\| &\le L_\theta\|\theta_1 - \theta_2\| \\ \|\nabla_\theta\mathcal{L}_\epsilon(\theta_1) - \nabla_\theta\mathcal{L}_\epsilon(\theta_2)\| &\le L_{\theta\theta}\|\theta_1 - \theta_2\| + 2\epsilon L_{\theta\mathbf{x}} + D_{\theta\theta} + D_{\theta\mathbf{x}} \ . \end{aligned} \tag{8}$$

The proof directly follows the one of Proposition 2. As in Proposition 2, the additional $2\epsilon L_{\theta\mathbf{x}}$ term in Corollary 1 evidences more severe *gradient scattering* under adversarial training in the context of ReLU networks, which harms optimization.

Similarly, we can easily extend the study of the asymptotic gradient magnitude of Theorem 2 to the account for Assumption 2.

Corollary 2. *Let Assumption 2 hold, the stochastic gradient $\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)$ be unbiased and have bounded variance, and the SGD update $\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)$ use a constant step size $\alpha_t = \alpha = \frac{1}{L_{\theta\theta}\sqrt{T}}$ for T iterations. Given the trajectory of the parameters during optimization $\{\theta_t\}_{t=1}^T$, then we can bound the asymptotic probability of large gradients for a sufficiently large T as*

$$\forall \gamma \geq 2, P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > \gamma(\epsilon L_{\theta\mathbf{x}} + \frac{1}{2} D_{\theta\theta} + \frac{1}{2} D_{\theta\mathbf{x}})) < \frac{4}{\gamma^2 - 2\gamma + 4} \ . \tag{9}$$

B Proofs

B.1 Proof of Proposition 1

Proof. For arbitrary $\mathbf{W} \in \mathcal{V}_{\epsilon_2}$, we have $\forall i \in [K], \mathbf{x}' \in \mathcal{S}_{\epsilon_2}(\mathbf{x})$, $(\mathbf{w}_i - \mathbf{w}_y)\mathbf{x}' \leq 0$ based on the definition of $\mathcal{V}_\epsilon$.

Since $\epsilon_1 \leq \epsilon_2$, we have $\mathcal{S}_{\epsilon_1}(\mathbf{x}) \subseteq \mathcal{S}_{\epsilon_2}(\mathbf{x})$. As a result, $\forall i \in [K], \mathbf{x}' \in \mathcal{S}_{\epsilon_1}(\mathbf{x})$, $(\mathbf{w}_i - \mathbf{w}_y)\mathbf{x}' \leq 0$. That is to say, $\mathbf{W} \in \mathcal{V}_{\epsilon_1}$. $\mathbf{W}$ is arbitrarily picked, so $\mathcal{V}_{\epsilon_2} \subseteq \mathcal{V}_{\epsilon_1}$.

□

B.2 Proof of Theorem 1

Proof. In multi-class logistic regression, as discussed in Section 3.1, the function $g(\mathbf{x}, \mathbf{W}) = \log\Big(1 + \sum_{j \neq y} \exp^{(\mathbf{w}_j - \mathbf{w}_y)\mathbf{x}}\Big)$ is a convex function w.r.t. the parameters $\mathbf{W}$, and so is $g_\epsilon(\mathbf{x}, \mathbf{W})$. Based on convexity, for any $\mathbf{W} \in \mathcal{T}_\epsilon$, the statement $0 \in \arg\min_\gamma g_\epsilon(\mathbf{x}, \gamma\mathbf{W})$ is equivalent to the following statement:

$$\forall \Delta\gamma > 0, g_\epsilon(\mathbf{x}, \Delta\gamma\mathbf{W}) \geq g_\epsilon(\mathbf{x}, \mathbf{0}) \text{ and } g_\epsilon(\mathbf{x}, -\Delta\gamma\mathbf{W}) \geq g_\epsilon(\mathbf{x}, \mathbf{0}) \ . \tag{10}$$

Note that $g_\epsilon(\mathbf{x}, \mathbf{0}) \equiv \log K$, which means that the loss of the model is independent of both the input and the adversarial budget when $\mathbf{W} = \mathbf{0}$. Given $\epsilon_1 \geq \epsilon_2$, we have, $\forall \mathbf{x}, \mathbf{W}, g_{\epsilon_1}(\mathbf{x}, \mathbf{W}) \geq g_{\epsilon_2}(\mathbf{x}, \mathbf{W})$. Therefore, for an arbitrary $\mathbf{W} \in \mathcal{T}_{\epsilon_2}$, we have the following inequality:

$$\forall \Delta\gamma > 0, g_{\epsilon_1}(\mathbf{x}, \Delta\gamma\mathbf{W}) \geq g_{\epsilon_2}(\mathbf{x}, \Delta\gamma\mathbf{W}) \geq g_{\epsilon_2}(\mathbf{x}, \mathbf{0}) = g_{\epsilon_1}(\mathbf{x}, \mathbf{0}) \ . \tag{11}$$

The first inequality is based on $\epsilon_1 \geq \epsilon_2$, the second one is based on (10) and the last one arises from the fact that, $\forall \epsilon, g_\epsilon(\mathbf{x}, \mathbf{0})$ is a constant. Similarly, we also have $g_{\epsilon_1}(\mathbf{x}, -\Delta\gamma\mathbf{W}) \geq g_{\epsilon_1}(\mathbf{x}, \mathbf{0})$. Therefore, we have $\mathbf{W} \in \mathcal{T}_{\epsilon_1}$, which means $\mathcal{T}_{\epsilon_2} \subseteq \mathcal{T}_{\epsilon_1}$.

To prove the second half of Theorem 1, one barrier is that we do not have an analytical form for $g_\epsilon(\mathbf{x}, \mathbf{W})$. Instead, we introduce a lower bound $\underline{g_\epsilon}(\mathbf{x}, \mathbf{W})$ of $g_\epsilon(\mathbf{x}, \mathbf{W})$, which has an analytical form. We consider the perturbation $\mathbf{x}' = \mathbf{x} + \epsilon \frac{(\mathbf{w}_m - \mathbf{w}_y)^{\frac{q}{p}}}{\|\mathbf{w}_m - \mathbf{w}_y\|_q^{\frac{q}{p}}}$ [2], where $m = \arg\max_j \|\mathbf{w}_j - \mathbf{w}_y\|_q$. It can be verified that $\mathbf{x}' \in \mathcal{S}_\epsilon^{(p)}(\mathbf{x})$. Therefore, we set $\underline{g_\epsilon}(\mathbf{x}, \mathbf{W}) = g(\mathbf{x}', \mathbf{W})$, which is a valid lower bound of $g_\epsilon(\mathbf{x}, \mathbf{W})$. Then, the analytical expression of $\underline{g_\epsilon}(\mathbf{x}, \mathbf{W})$ can be written as

$$\underline{g_\epsilon}(\mathbf{x}, \mathbf{W}) = \log\left(1 + \exp^{(\mathbf{w}_m - \mathbf{w}_y)\mathbf{x} + \epsilon\|\mathbf{w}_m - \mathbf{w}_y\|_q} + \sum_{j \neq y, j \neq m} \exp^{(\mathbf{w}_j - \mathbf{w}_y)\mathbf{x} + \epsilon(\mathbf{w}_j - \mathbf{w}_y)\frac{(\mathbf{w}_m - \mathbf{w}_y)^{\frac{q}{p}}}{\|\mathbf{w}_m - \mathbf{w}_y\|_q^{\frac{q}{p}}}}\right) . \tag{12}$$

Since $m = \arg\max_j \|\mathbf{w}_j - \mathbf{w}_y\|_q$, then $(\mathbf{w}_j - \mathbf{w}_y)\frac{(\mathbf{w}_m - \mathbf{w}_y)^{\frac{q}{p}}}{\|\mathbf{w}_m - \mathbf{w}_y\|_q^{\frac{q}{p}}} \leq \|\mathbf{w}_m - \mathbf{w}_y\|_q$. As a result, if ϵ is large enough, the second term inside the logarithm of (12) will dominate the summation and

[2] l_q is the dual norm of l_p, i.e., $\frac{1}{p} + \frac{1}{q} = 1$

$\lim_{\epsilon\to\infty} \underline{g_\epsilon}(\mathbf{x}, \mathbf{W}) = \infty$. More specifically, we can find $\bar{\epsilon} = \frac{\log(K-1)-(\mathbf{w}_m-\mathbf{w}_y)\mathbf{x}}{\|\mathbf{w}_m-\mathbf{w}_y\|_q}$, such that, $\forall \epsilon > \bar{\epsilon}$, $\mathbf{W}$, then $\underline{g_\epsilon}(\mathbf{x}, \mathbf{W}) \geq \log K = g_\epsilon(\mathbf{x}, \mathbf{0})$.

Now, $\forall \mathbf{W} \in \mathbb{R}^{m\times K}, \Delta\gamma > 0, \epsilon \geq \bar{\epsilon}$, we have $g_\epsilon(\mathbf{x}, \Delta\gamma\mathbf{W}) \geq \underline{g_\epsilon}(\mathbf{x}, \Delta\gamma\mathbf{W}) \geq g_\epsilon(\mathbf{x}, \mathbf{0})$. Similarly, we have $g_\epsilon(\mathbf{x}, -\Delta\gamma\mathbf{W}) \geq g_\epsilon(\mathbf{x}, \mathbf{0})$. As a result, we have, $\forall \mathbf{W} \in \mathbb{R}^{m\times K}, g_\epsilon(\mathbf{x}, \mathbf{W}) \geq g_\epsilon(\mathbf{x}, \mathbf{0})$, so $\mathbf{0} \in \arg\min_{\mathbf{W}} g_\epsilon(\mathbf{x}, \mathbf{W})$. Based on (10), we have $\mathcal{T}_\epsilon = \mathbb{R}^{m\times K}$.

□

B.3 Proof of Proposition 2

Proof. Recall that $\mathcal{L}_\epsilon(\theta)$ is the average of $g_\epsilon(\mathbf{x}, \theta)$ over the dataset. Therefore, to prove Proposition 2, we only need to prove the following inequalities for any data point $\mathbf{x}$:

$$\begin{aligned} \|g_\epsilon(\mathbf{x}, \theta_1) - g_\epsilon(\mathbf{x}, \theta_2)\| &\leq L_\theta \|\theta_1 - \theta_2\| \ , \\ \|\nabla_\theta g_\epsilon(\mathbf{x}, \theta_1) - \nabla_\theta g_\epsilon(\mathbf{x}, \theta_2)\| &\leq L_{\theta\theta}\|\theta_1 - \theta_2\| + 2\epsilon L_{\theta\mathbf{x}} \ . \end{aligned} \tag{13}$$

To prove the first inequality, we introduce the adversarial examples for parameter θ_1 and θ_2:

$$\begin{aligned} \mathbf{x}_1 &= \mathop{\arg\max}_{\mathbf{x}'\in\mathcal{S}_\epsilon^{(p)}(\mathbf{x})} g(\mathbf{x}', \theta_1) \ , \\ \mathbf{x}_2 &= \mathop{\arg\max}_{\mathbf{x}'\in\mathcal{S}_\epsilon^{(p)}(\mathbf{x})} g(\mathbf{x}', \theta_2) \ . \end{aligned} \tag{14}$$

Therefore, $g_\epsilon(\mathbf{x}, \theta_1) = g(\mathbf{x}_1, \theta_1)$ and $g_\epsilon(\mathbf{x}, \theta_2) = g(\mathbf{x}_2, \theta_2)$.

By definition, we have $g(\mathbf{x}_1, \theta_1) \geq g(\mathbf{x}_2, \theta_1)$ and $g(\mathbf{x}_2, \theta_2) \geq g(\mathbf{x}_1, \theta_2)$. As a result, $\|g_\epsilon(\mathbf{x}, \theta_1) - g_\epsilon(\mathbf{x}, \theta_2)\| = \|g(\mathbf{x}_1, \theta_1) - g(\mathbf{x}_2, \theta_2)\|$. If $g(\mathbf{x}_1, \theta_1) - g(\mathbf{x}_2, \theta_2) \leq 0$, we have

$$\|g_\epsilon(\mathbf{x}, \theta_1) - g_\epsilon(\mathbf{x}, \theta_2)\| = g(\mathbf{x}_2, \theta_2) - g(\mathbf{x}_1, \theta_1) \leq g(\mathbf{x}_2, \theta_2) - g(\mathbf{x}_2, \theta_1) \leq L_\theta\|\theta_1 - \theta_2\| \ . \tag{15}$$

Similarly, if $g(\mathbf{x}_1, \theta_1) - g(\mathbf{x}_2, \theta_2) \geq 0$, we have

$$\|g_\epsilon(\mathbf{x}, \theta_1) - g_\epsilon(\mathbf{x}, \theta_2)\| = g(\mathbf{x}_1, \theta_1) - g(\mathbf{x}_2, \theta_2) \leq g(\mathbf{x}_1, \theta_1) - g(\mathbf{x}_1, \theta_2) \leq L_\theta\|\theta_1 - \theta_2\| \ . \tag{16}$$

This proves the first inequality in (13). The bound is tight, and equality is achieved when, for example, $\mathbf{x}_1 = \mathbf{x}_2$.

The second inequality in (13) is more straightforward. We have

$$\begin{aligned} \|\nabla_\theta g_\epsilon(\mathbf{x}, \theta_1) - \nabla_\theta g_\epsilon(\mathbf{x}, \theta_2)\| &= \|\nabla_\theta g(\mathbf{x}_1, \theta_1) - \nabla_\theta g(\mathbf{x}_2, \theta_2)\| \\ &= \|\nabla_\theta g(\mathbf{x}_1, \theta_1) - \nabla_\theta g(\mathbf{x}_1, \theta_2) + \nabla_\theta g(\mathbf{x}_1, \theta_2) - \nabla_\theta g(\mathbf{x}_2, \theta_2)\| \\ &\leq \|\nabla_\theta g(\mathbf{x}_1, \theta_1) - \nabla_\theta g(\mathbf{x}_1, \theta_2)\| + \|\nabla_\theta g(\mathbf{x}_1, \theta_2) - \nabla_\theta g(\mathbf{x}_2, \theta_2)\| \\ &\leq L_{\theta\theta}\|\theta_1 - \theta_2\| + L_{\theta\mathbf{x}}\|\mathbf{x}_1 - \mathbf{x}_2\|_p \\ &\leq L_{\theta\theta}\|\theta_1 - \theta_2\| + 2\epsilon L_{\theta\mathbf{x}} \ . \end{aligned} \tag{17}$$

The last inequality in (17) is satisfied because both $\mathbf{x}_1$ and $\mathbf{x}_2$ belong to $\mathcal{S}_\epsilon^{(p)}(\mathbf{x})$. This bound is tight, and equality is reached only when $\|\mathbf{x}_1 - \mathbf{x}_2\|_p = 2\epsilon$.

□

B.4 Proof of Theorem 2

Proof. Let σ^2 to denote the variance of stochastic gradient $\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)$. Based on the assumption that $\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)$ is unbiased, we have

$$\begin{aligned} \mathbb{E}[\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)] &= \nabla_\theta \mathcal{L}_\epsilon(\theta) \ , \\ \mathbb{E}\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|^2 &= \|\nabla_\theta \mathcal{L}_\epsilon(\theta)\|^2 + \sigma^2 \ . \end{aligned} \tag{18}$$

Proposition 2 shows that $\mathcal{L}_\epsilon(\theta)$ is continuous. Therefore, we introduce $\tilde{\theta}_t(u) = \theta_t + u(\theta_{t+1} - \theta_t)$ and derive an upper bound of $\mathcal{L}_\epsilon(\theta_{t+1}) - \mathcal{L}_\epsilon(\theta_t)$ by first order Taylor expansion and using the update

rule $\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)$. This yields

$$\begin{aligned}
\mathcal{L}_\epsilon(\theta_{t+1}) - \mathcal{L}_\epsilon(\theta_t) &= \int_0^1 \langle \theta_{t+1} - \theta_t, \nabla_\theta \mathcal{L}_\epsilon(\tilde{\theta}_t(u)) \rangle d_u \\
&= \int_0^1 \langle -\alpha_t \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t), \nabla_\theta \mathcal{L}_\epsilon(\tilde{\theta}_t(u)) \rangle d_u \\
&= \int_0^1 \langle -\alpha_t \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t), \nabla_\theta \mathcal{L}_\epsilon(\tilde{\theta}_t(u)) - \nabla_\theta \mathcal{L}_\epsilon(\theta_t) \rangle d_u + \langle -\alpha_t \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t), \nabla_\theta \mathcal{L}_\epsilon(\theta_t) \rangle \\
&\leq \int_0^1 \alpha_t \|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)\| \|\nabla_\theta \mathcal{L}_\epsilon(\tilde{\theta}_t(u)) - \nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| d_u - \alpha_t \langle \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t), \nabla_\theta \mathcal{L}_\epsilon(\theta_t) \rangle \\
&\leq \int_0^1 \alpha_t \|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)\| (L_{\theta\theta} \|\tilde{\theta}_t(u) - \theta_t\| + 2\epsilon L_{\theta \mathbf{x}}) d_u - \alpha_t \langle \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t), \nabla_\theta \mathcal{L}_\epsilon(\theta_t) \rangle \\
&= \frac{1}{2} \alpha_t^2 L_{\theta\theta} \|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)\|^2 + 2\epsilon L_{\epsilon \mathbf{x}} \alpha_t \|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)\| - \alpha_t \langle \nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t), \nabla_\theta \mathcal{L}_\epsilon(\theta_t) \rangle \ .
\end{aligned} \tag{19}$$

Here, the first inequality comes from Hölder's Inequality; the second one follows the conclusion of Proposition 2.

By taking the expectation over the noise introduced by SGD, we have

$$\begin{aligned}
\mathbb{E}[\mathcal{L}_\epsilon(\theta_{t+1})] - \mathbb{E}[\mathcal{L}_\epsilon(\theta_t)] &\leq \frac{1}{2} \alpha_t^2 L_{\theta\theta} (\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|^2 + \sigma^2) + 2\epsilon L_{\theta \mathbf{x}} \alpha_t \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| - \alpha_t \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|^2 \\
&= (\frac{1}{2} \alpha_t^2 L_{\theta\theta} - \alpha_t) \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|^2 + 2\epsilon L_{\theta \mathbf{x}} \alpha_t \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| + \frac{1}{2} \alpha_t^2 \sigma^2 L_{\theta\theta} \\
&\leq -\frac{1}{2} \alpha_t \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|^2 + 2\epsilon L_{\theta \mathbf{x}} \alpha_t \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| + \frac{1}{2} \alpha_t^2 \sigma^2 L_{\theta\theta} \ .
\end{aligned} \tag{20}$$

We use the approximation $\mathbb{E}\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\| \simeq \|\nabla_\theta \mathcal{L}_\epsilon(\theta)\|$ because the variance arises mainly from the term $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta_t)\|^2$. The last inequality is based on the fact that $\alpha_t = \alpha = \frac{1}{L_{\theta\theta}\sqrt{T}}$, so $\alpha_t L_{\theta\theta} = \frac{1}{\sqrt{T}} \leq 1$.

Let us now sum (20) over $t \in [T]$. This gives

$$\begin{aligned}
\sum_{t=0}^{T} \left[\frac{1}{2} \alpha_t \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|^2 - 2\epsilon L_{\theta \mathbf{x}} \alpha_t \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| \right] &\leq \mathcal{L}_\epsilon(\theta_0) - \mathbb{E}[\mathcal{L}_\epsilon(\theta_T)] + \frac{T}{2} \alpha_t^2 \sigma^2 L_{\theta\theta} \\
&\leq \mathcal{L}_\epsilon(\theta_0) - \mathcal{L}_\epsilon(\theta^*) + \frac{T}{2} \alpha_t^2 \sigma^2 L_{\theta\theta} \ .
\end{aligned} \tag{21}$$

We use θ^* to denote the global minimum since $\mathcal{L}_\epsilon(\theta)$ is lower bounded. By introducing $\alpha_t = \alpha = \frac{1}{L_{\theta\theta}\sqrt{T}}$ into the formulation, we obtain

$$\begin{aligned}
\frac{1}{T} \sum_{t=0}^{T} \left[\frac{1}{2} \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|^2 - 2\epsilon L_{\theta \mathbf{x}} \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| \right] &\leq \frac{1}{\alpha T} \left[\mathcal{L}_\epsilon(\theta_0) - \mathcal{L}_\epsilon(\theta^*) \right] + \frac{1}{2} \alpha \sigma^2 L_{\theta\theta} \\
&= \frac{1}{\sqrt{T}} \left[L_{\theta\theta} (\mathcal{L}_\epsilon(\theta_0) - \mathcal{L}_\epsilon(\theta^*)) + \frac{1}{2} \sigma^2 \right] \ .
\end{aligned} \tag{22}$$

Since the righthand side of (22) converges to 0 as $T \to +\infty$, we have

$$\lim_{T \to +\infty} \frac{1}{T} \sum_{t=0}^{T} \left[\frac{1}{2} \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|^2 - 2\epsilon L_{\theta \mathbf{x}} \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| \right] \leq 0 \ . \tag{23}$$

Let us define $h(\theta_t) = \frac{1}{2} \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|^2 - 2\epsilon L_{\theta \mathbf{x}} \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\|$ for notation simplicity. Then, inequality (23) shows that $\mathbb{E}_t[h(\theta_t)] \leq 0$ when T is large enough.

We define $\widehat{\gamma} := \|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| / (\epsilon L_{\theta \mathbf{x}})$, then we have $h(\theta_t) = (\frac{1}{2} \widehat{\gamma}^2 - 2\widehat{\gamma}) \epsilon^2 L_{\theta \mathbf{x}}^2$. $h(\theta)$ is monotonically increasing when $\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| \geq 2\epsilon L_{\theta \mathbf{x}}$, so when $\widehat{\gamma} \geq 2$, $h(\theta_t) \geq (\frac{1}{2} \widehat{\gamma}^2 - 2\widehat{\gamma}) \epsilon^2 L_{\theta \mathbf{x}}^2$. Considering

$h(\theta_t) \geq -2\epsilon^2 L_{\theta\mathbf{x}}^2$ always holds, we can then bound the probability of $P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > \gamma\epsilon L_{\theta\mathbf{x}})$ when $\gamma > 2$ as follows:

$$\mathbb{E}_t[h(\theta_t)] > -2\epsilon^2 L_{\theta\mathbf{x}}^2(1 - P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > \gamma\epsilon L_{\theta\mathbf{x}})) + (\frac{1}{2}\gamma^2 - 2\gamma)\epsilon^2 L_{\theta\mathbf{x}}^2 P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > \gamma\epsilon L_{\theta\mathbf{x}}) \ . \tag{24}$$

Finally, by rearranging (24) and using $\mathbb{E}_t[h(\theta_t)] \leq 0$, we obtain

$$\forall \gamma > 2,\ P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > \gamma\epsilon L_{\theta\mathbf{x}}) < \frac{4}{\gamma^2 - 2\gamma + 4} \ . \tag{25}$$

□

B.5 Proof of Theorem 3

To prove Theorem 3, let us first introduce the following lemma.

Lemma 1. *Given a vector set $\{\mathbf{x}_i\}_{i=1}^N$ and scalar sets $\{a_i\}_{i=1}^N$, $\{b_i\}_{i=1}^N$, we define $\mathbf{A} = \sum_{i=1}^N a_i \mathbf{x}_i \mathbf{x}_i^T$ and $\mathbf{B} = \sum_{i=1}^N b_i \mathbf{x}_i \mathbf{x}_i^T$. If, $\forall i,\ a_i \geq b_i$, then $\forall \mathbf{m} \in \mathbb{R}^m$, $\mathbf{m}^T\mathbf{A}\mathbf{m} \geq \mathbf{m}^T\mathbf{B}\mathbf{m}$. Furthermore, the largest and the smallest eigenvalues of $\mathbf{A}$ are no smaller than those of $\mathbf{B}$.*

Proof. Because $\forall\ i, a_i \geq b_i$, we have $\forall \mathbf{m} \sum_{i=1}^N (a_i - b_i)(\mathbf{x}_i^T\mathbf{m})^2 \geq 0$, which can be re-organized into $\mathbf{m}^T\mathbf{A}\mathbf{m} \geq \mathbf{m}^T\mathbf{B}\mathbf{m}$.

$\mathbf{A}$ is a symmetric matrix, so the largest eigenvalue $\lambda_1(\mathbf{A})$ is $\max_{\|\mathbf{m}\|_2=1} \mathbf{m}^T\mathbf{A}\mathbf{m} = \max_{\|\mathbf{m}\|_2=1} \sum_{i=1}^N a_i(\mathbf{x}_i^T\mathbf{m})^2$. Similarly, we have $\lambda_1(\mathbf{B}) = \max_{\|\mathbf{m}\|_2=1} \sum_{i=1}^N b_i(\mathbf{x}_i^T\mathbf{m})^2$. Let $\mathbf{m}_\mathbf{B} \in \arg\max_{\|\mathbf{m}\|_2=1} \sum_{i=1}^N b_i(\mathbf{x}_i^T\mathbf{m})^2$. Then we have

$$\lambda_1(\mathbf{B}) = \sum_{i=1}^N b_i(\mathbf{x}_i^T\mathbf{m}_\mathbf{B})^2 \leq \sum_{i=1}^N a_i(\mathbf{x}_i^T\mathbf{m}_\mathbf{B})^2 \leq \max_{\|\mathbf{m}\|_2=1} \sum_{i=1}^N a_i(\mathbf{x}_i^T\mathbf{m})^2 = \lambda_1(\mathbf{A}) \ . \tag{26}$$

In the same way as for the largest eigenvalue, the smallest eigenvalue of $\mathbf{A}$ and $\mathbf{B}$ are $\lambda_m(\mathbf{A}) = \min_{\|\mathbf{m}\|_2=1} \sum_{i=1}^N a_i(\mathbf{x}_i^T\mathbf{m})^2$ and $\lambda_m(\mathbf{B}) = \min_{\|\mathbf{m}\|_2=1} \sum_{i=1}^N b_i(\mathbf{x}_i^T\mathbf{m})^2$, respectively. Let $\mathbf{m}_\mathbf{A} \in \arg\min_{\|\mathbf{m}\|_2=1} \sum_{i=1}^N a_i(\mathbf{x}_i^T\mathbf{m})^2$. Then we have

$$\lambda_m(\mathbf{A}) = \sum_{i=1}^N a_i(\mathbf{x}_i^T\mathbf{m}_\mathbf{A})^2 \geq \sum_{i=1}^N b_i(\mathbf{x}_i^T\mathbf{m}_\mathbf{A})^2 \geq \min_{\|\mathbf{m}\|_2=1} \sum_{i=1}^N b_i(\mathbf{x}_i^T\mathbf{m})^2 = \lambda_m(\mathbf{B}) \ . \tag{27}$$

□

Let us now go back to Theorem 3.

Proof. We first calculate the first and second derivatives of $\mathcal{L}_\epsilon(\mathbf{w})$ in Equation 6, as

$$\begin{aligned} \nabla_\mathbf{w} \mathcal{L}_\epsilon(\mathbf{w}) &= \frac{1}{N} \sum_{i=1}^N -\frac{1}{1 + e^{y_i \mathbf{w}^T \mathbf{x}_i - \epsilon}} y_i \mathbf{x}_i \ , \\ \nabla_\mathbf{w}^2 \mathcal{L}_\epsilon(\mathbf{w}) &= \frac{1}{N} \sum_{i=1}^N \frac{e^{y_i \mathbf{w}^T \mathbf{x}_i - \epsilon}}{(1 + e^{y_i \mathbf{w}^T \mathbf{x}_i - \epsilon})^2} y_i^2 \mathbf{x}_i \mathbf{x}_i^T = \frac{1}{N} \sum_{i=1}^N \frac{e^{y_i \mathbf{w}^T \mathbf{x}_i - \epsilon}}{(1 + e^{y_i \mathbf{w}^T \mathbf{x}_i - \epsilon})^2} \mathbf{x}_i \mathbf{x}_i^T \ . \end{aligned} \tag{28}$$

The second equality of $\nabla_\mathbf{w}^2 \mathcal{L}_\epsilon(\mathbf{w})$ is satisfied because y_i is either $+1$ or -1. The dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ is linearly separable under adversarial budget $\mathcal{S}_{\hat{\epsilon}}^{(p)}(\mathbf{x})$, so, $\forall i,\ y_i \mathbf{w}^T \mathbf{x}_i \geq \hat{\epsilon}$. When $\epsilon \leq \hat{\epsilon}$, $e^{y_i \mathbf{w}^T \mathbf{x}_i - \epsilon} > 1$ and monotonically decreases with ϵ. As a result, $\frac{e^{y_i \mathbf{w}^T \mathbf{x}_i - \epsilon}}{(1 + e^{y_i \mathbf{w}^T \mathbf{x}_i - \epsilon})^2}$ monotonically increases with ϵ in the range $[0, \hat{\epsilon}]$.

Based on Lemma 1, $\forall \mathbf{m} \in \mathbb{R}^m$, $\mathbf{m}^T \nabla_\mathbf{w}^2 \mathcal{L}_\epsilon(\mathbf{w}) \mathbf{m}$ increases with ϵ, and so do the largest and the smallest eigenvalues of the Hessian matrix $\nabla_\mathbf{w}^2 \mathcal{L}_\epsilon(\mathbf{w})$.

□

C Additional Experiments

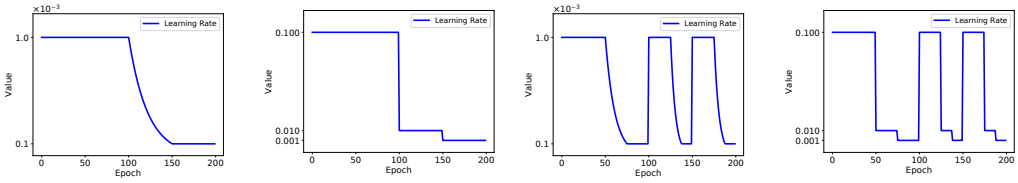
C.1 Experimental Details

For MNIST, we set the step size of PGD to 0.01 and the number of iterations to $\epsilon/0.01 + 10$. For CIFAR10, we set the number of PGD iterations to 10 and the step size to $\epsilon/4$. The network architectures we use are the same as the ones in [52]. We provide the details in Table 2 and use a factor w to control the width of the network. Unless specified, the LeNet models on MNIST have a width factor of 16, the VGG and ResNet18 models on CIFAR10 have a width factor of 8.

Name	Architecture
MNIST, LeNet	Conv($2w$), Conv($4w$), FC($196w$, $64w$), FC($64w$, 10)
CIFAR, VGG	Conv($4w$) $\times$ 2, M, Conv($8w$) $\times$ 2, M, Conv($16w$) $\times$ 3, M Conv($32w$) $\times$ 3, M, Conv($32w$) $\times$ 3, M, A, FC($32w$, 10)
CIFAR, ResNet18	ResNet18 in [23], which uses a width $w = 16$

Table 2: Network architectures. Conv, FC, M and A represent convolutional layers, fully-connected layers, max-pooling layers and average pooling layers, respectively. The parameter of the convolutional layers indicates the number of output channels. The parameters of the fully-connected layers indicate the number of input and output neurons. The kernel sizes of the max-pooling layers and average pooling layers are always 2. w corresponds to the width factor mentioned in Section 4 of the main paper.

We train the models for 100 epochs on MNIST and 200 epochs on CIFAR10. Unless explicitly mentioned, for LeNet models on MNIST, we use Adam [28] with a learning rate of 1×10^{-4}. For VGG models on CIFAR10, we also use Adam, with an initial learning rate of 1×10^{-3}, decreased exponentially to 1×10^{-4} between the 100th epoch and the 150th epoch, and then fixed to 1×10^{-4} after 150 epochs. For ResNet18 models on CIFAR10, we use accelerated SGD with a momentum factor of 0.9. The initial learning rate is 0.1 and is divided by 10 after 100 and 150 epochs.



(a) CIFAR10, VGG, Vanilla (b) CIFAR10, ResNet18, Vanilla (c) CIFAR10, VGG, Periodic (d) CIFAR10, ResNet18, Periodic

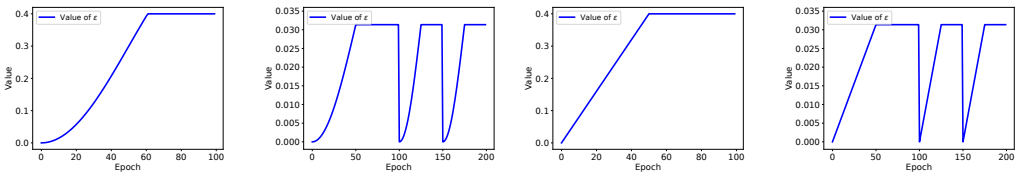
Figure 6: Learning rate scheduling for VGG-8 and ResNet18-8 for CIFAR10 classification.

Experiments in Section 5. The details of the adversarial attacks used in this section are demonstrated below:

- PGD: for MNIST models, PGD with 50 iterations, the step size is 0.1; for CIFAR10 models, PGD with 10 iterations, the step size is $2/255$.
- PGD100: PGD with 100 iterations, the step size is 0.1 for MNIST models and $1/255$ for CIFAR10 models.
- APGD100-CE: AutoPGD with 100 iterations and cross-entropy loss. We use the default settings in [10], i.e., $\rho = 0.75$, $\alpha = 0.75$.
- APGD100-DLR: AutoPGD with 100 iterations and difference-of-logit-ratio loss. We use the default settings in [10], i.e., $\rho = 0.75$, $\alpha = 0.75$.
- Square5K: SquareAttack with 5000 iterations. We use the default settings in [3], i.e., we use the *margin loss*.

For the results in Table 1, we fine-tune the weight-decay factor, choosing 1×10^{-3} as the optimal value. In periodic settings, the learning rate and the adversarial budget are reset after 100 and 150 epochs. The scheduling in each period is scaled proportionally. We plot the learning rate scheduling

curves for VGG-8 and ResNet18-8 in Figure 6 for both the vanilla and periodic settings. Regarding the scheduling of ϵ, we do not fully explore the value range of the hyper-parameters in the cosine and linear schedulers. We use $\epsilon_{min} = 0$ for all experiments. For the MNIST experiments, we set $\epsilon_{max} = 0.6$ for the cosine scheduler and $\epsilon_{max} = 0.8$ for the linear one. For the CIFAR10 experiments, we set $\epsilon_{max} = 16/255$ for both the cosine and linear schedulers. We plot the curves for $\epsilon_{cos}(d)$ and $\epsilon_{lin}(d)$ in Figure 7.



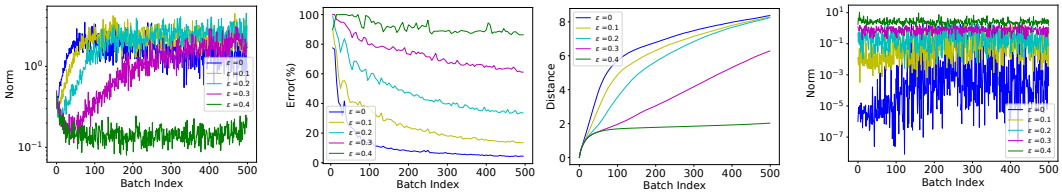
(a) MNIST, Cosine (b) CIFAR10, Cosine (c) MNIST, Linear (d) CIFAR10, Linear

Figure 7: Adversarial budget scheduling for MNIST and CIFAR10 models.

C.2 Additional Experimental Results

C.2.1 Additional Results for Section 4.1

To complement the results on CIFAR10 models in Section 4.1, in Figure 8, we provide a numerical analysis on the first and last 500 training mini-batches of MNIST under different values of ϵ. We observe the same phenomenon: in the early stages of training, a large adversarial budget leads to smaller gradient magnitudes and slows down the training; in the final stages of training, a large adversarial budget yields severe gradient scattering, indicated by larger gradient magnitudes.



(a) $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$, first 500. (b) $\mathcal{E}_\epsilon(\theta)$, first 500. (c) $\|\theta - \theta_0\|$, first 500. (d) $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$, last 500.

Figure 8: Norm of the stochastic gradient $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$, robust training error $\mathcal{E}_\epsilon(\theta)$, distance from the initial point $\|\theta - \theta_0\|$ during the first or last 500 mini-batch updates for MNIST models.

C.2.2 Additional Results for Section 4.2

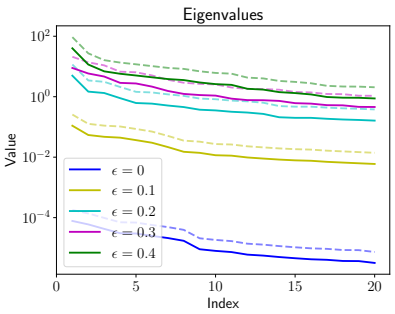


Figure 9: Top 20 eigenvalues of the Hessian matrix for LeNet models. Both normalized (solid) and original (dashed) values are shown.

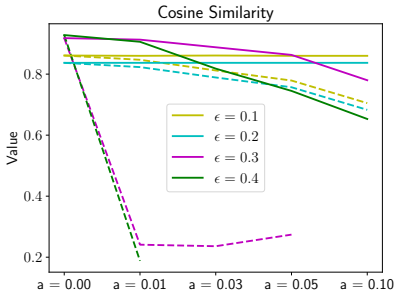


Figure 10: Cosine similarity between perturbations $\mathbf{x}'_{a\mathbf{v}} - \mathbf{x}$ and $\mathbf{x}'_{-a\mathbf{v}} - \mathbf{x}$. $\mathbf{v}$ can be either the top eigenvector (dashed) or randomly picked (solid).

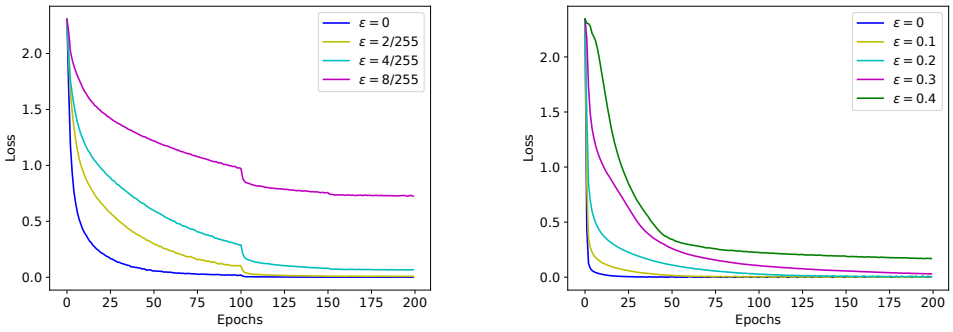
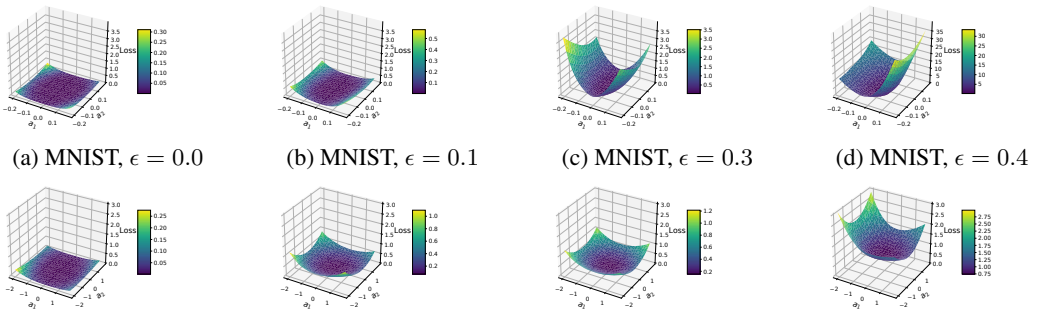


Figure 11: The learning curves of training loss for CIFAR10 models (left) and MNIST models (right) under different values of ϵ.

In Figure 9, we provide the Hessian spectrum analysis for LeNet models on MNIST under various adversarial budgets. As in Figure 3, the top eigenvalues, both the original and normalized values, of the Hessian matrix of our trained models are larger in the presence of larger adversarial budgets.

Note that the magnitudes of $\mathcal{L}_\epsilon(\theta)$ with different ϵ are similar. To show this, we randomly sample 10 θ and calculate the value of $\mathcal{L}_\epsilon(\theta)$ over the training set. For CIFAR10 models, the mean values are 2.3034, 2.3044, 2.3053, 2.3071 when ϵ is 0, $2/255$, $4/255$ and $8/255$, respectively. For MNIST models, the mean values are 2.3029, 2.3414, 2.3424, 2.3429, 2.3432 when ϵ is 0, 0.1, 0.2, 0.3 and 0.4, respectively. In Figure 11, we plot the learning curves of the training loss; this clearly shows that the magnitudes of $\mathcal{L}_\epsilon(\theta)$ during training with different ϵ values are similar. Furthermore, the range of values of $\mathcal{L}_\epsilon(\theta)$ during training is smaller under large values of ϵ. As a result, the increased curvature under large adversarial budgets is not caused by the magnitudes of the function $\mathcal{L}_\epsilon(\theta)$, and we empirically observed that optimization in adversarial training cannot be facilitated by tuning the learning rate.

Ideally, the sharpness of the minima is depicted by the condition number of its Hessian matrix. However, in the context of deep neural networks, the eigenvalue with the smallest absolute value of the Hessian matrix is almost zero, which renders the computation of the condition number both algorithmically and numerically unstable [16]. Instead, the spectral norm and the nuclear norm of the Hessian matrix are typically used as quantitative metrics for the sharpness of the minima [11]. Figure 3 and Figure 9 thus demonstrate that the obtained minima are shaper when ϵ is larger.



(a) MNIST, $\epsilon = 0.0$ (b) MNIST, $\epsilon = 0.1$ (c) MNIST, $\epsilon = 0.3$ (d) MNIST, $\epsilon = 0.4$

(e) CIFAR10, $\epsilon = 0$ (f) CIFAR10, $\epsilon = 2/255$ (g) CIFAR10, $\epsilon = 4/255$ (h) CIFAR10, $\epsilon = 8/255$

Figure 12: Loss landscape $\mathcal{L}_\epsilon(\theta + a_1\mathbf{v}_1 + a_2\mathbf{v}_2)$ under different adversarial budgets. θ, $\mathbf{v}_1$, $\mathbf{v}_2$ are the parameter, and the first and second unit eigenvectors of the Hessian matrix. (Note that the z-scale for $\epsilon = 0.4$ in the MNIST case differs from the others.)

In Figure 10, we report the cosine similarity of input perturbations when we move the model parameter θ in two opposite directions. As in Figure 4, we see high similarity of the perturbations and high robust accuracy when $\mathbf{v}$ is a random direction. By contrast, when $\mathbf{v}$ is the first eigenvector of the Hessian matrix, we see a sharp decrease in both the perturbation similarity and robust accuracy as the

value of a increases. In Figure 10, we only plot the perturbation similarity when the robust accuracy on the training set is higher than 70%; otherwise the model parameters can no longer be considered to be in a small neighborhood of the original ones.

Figure 12 shows 3D visualizations of $\mathcal{L}_\epsilon(\theta)$ under different values of ϵ in the parameter neighborhood of our obtained MNIST and CIFAR10 models on the training set. We study the curvature in the directions of the top 2 eigenvectors. The curvature clearly increases with ϵ and the corresponding minima become sharper.

C.2.3 Additional Results for Section 5

Here, we compare the performance of warmup in the adversarial budget and warmup in the learning rate. As in Figure 5, we use the LeNet model on MNIST and set the target adversarial budget size to 0.4. Our warmup period consists of the first 10 epochs: the learning rate starts at 0 and linearly increases to the final value in the warmup period; the learning rate remains constant after the warmup period. In Figure 13, we show the robust accuracy on the test set when the final learning rate is set to 1×10^{-4}, 3×10^{-4} and 1×10^{-3}. For comparison, we show the best performance obtained when using warmup in the adversarial budget with a blue line. We run each experiment 5 times.

When the final learning rate is 1×10^{-4}, the learning rate wamup performance is not as good as warmup in the adversarial budget. When the final learning rate is 3×10^{-4} or 1×10^{-3}, the variance of the performance becomes large. Learning rate warmup can sometimes yield good performance but sometimes fails to converge.

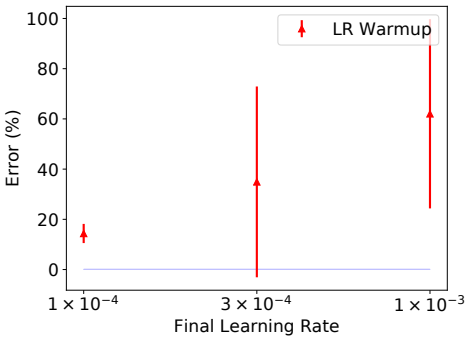


Figure 13: Mean and standard deviation of the test error on MNIST models when we use learning rate warmup but constant adversarial budget. The best performance by constant learning rate but adversarial budget warmup is depicted by a blue line.

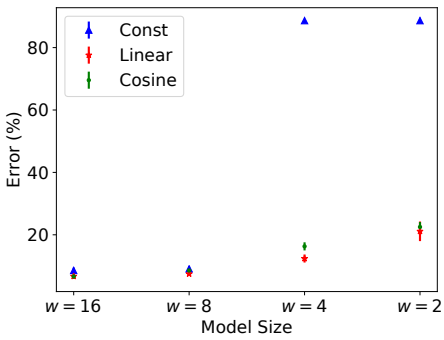


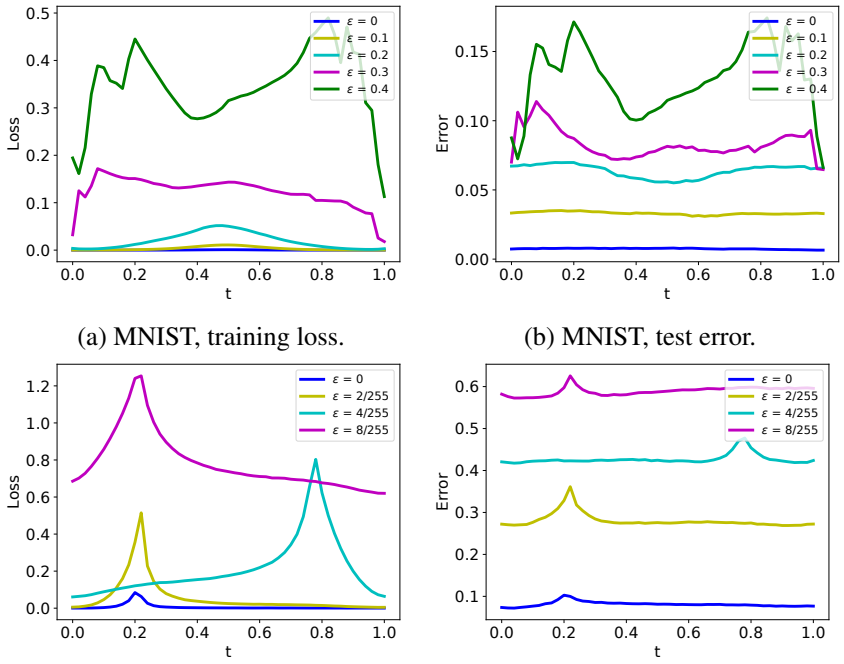
Figure 14: Mean and standard deviation of the test error with LeNet-16 models of different sizes on MNIST, using different adversarial budget scheduling schemes.

C.2.4 Robustness v.s. Model Capacity

In Figure 14, we report the performance of LeNet models of different width factors w using different schedulers for ϵ. The adversarial budget size ϵ at test time is 0.4. We set the learning rate in Adam to be 10^{-4}, because, for constant ϵ during training, it yields the best performance. Both Cosine and Linear schedulers outperform using a constant ϵ in all cases. When the model size is small, e.g., $w = 4$ and $w = 2$, using a constant ϵ during training fails to converge, but the Cosine and Linear schedulers still yield competitive results.

C.2.5 Connectivity of Different Minima

The minima reached in the loss landscape of vanilla training have been found to be well connected [12, 14]. That is, if we train two neural networks under the same settings but different initializations, there exists a path connecting the resulting two models in the parameter space such that all points along this path have low loss. In this section, we study the connectivity of different trained models in adversarial training. Similarly to [14], we parameterize the path joining two minima using a *general Bezier curve*. Let θ_0 and θ_n be the parameters of two separately-trained models, and $\{\widehat{\theta}_i\}_{i=1}^{n-1}$ the



(a) MNIST, training loss.

(b) MNIST, test error.

(c) CIFAR10, training loss.

(d) CIFAR10, test error.

Figure 15: Training loss and test error along the path connecting the minima of two independently-trained models.

parameters of $(n-1)$ trainable intermediate models. Then, an n-order Bezier curve is defined as a linear combination of these $(n+1)$ points in parameter space, i.e.,

$$\mathcal{B}(t) = (1-t)^n \theta_0 + t^n \theta_n + \sum_{i=1}^{n-1} \binom{n}{t} (1-t)^{n-i} t^i \widehat{\theta}_i \ . \tag{29}$$

$\mathcal{B}(t)$ is a smooth curve, and $\mathcal{B}(0) = \theta_0$ and $\mathcal{B}(1) = \theta_n$. We train $\{\widehat{\theta}_i\}_{i=1}^{n-1}$ by minimizing the average loss along the path: $\mathbb{E}_{t \sim U[0,1]} \mathcal{L}_\epsilon(\mathcal{B}(t))$, where $U[0,1]$ is the uniform distribution between 0 and 1. We use the Monte Carlo method to estimate the gradient of this expectation-based function and minimize it using gradient-based optimization. We use second-order Bezier curves to connect MNIST model pairs and fourth-order Bezier curves to connect CIFAR10 model pairs. When evaluating the models on the learned curves, we re-estimate the running mean and variance in the batch normalization layer based on the training set. The results are reported based on the evaluation mode of the models, and we turn off data augmentation to avoid stochasticity.

In Figure 15, following [14], we plot the training loss and test error along the learned curve, as a function of t in Equation (29). For vanilla training or when the adversarial budget is small, we can easily find flat curves connecting different minima. However, the learned curves are not flat anymore when the adversarial budget increases. This indicates that the minima are less well-connected under adversarial training, and that it is more difficult for the optimizer to find a minimum.