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] École Polytechnique Fédérale de Lausanne
[2] Microsoft Research

October 28, 2020

Overview

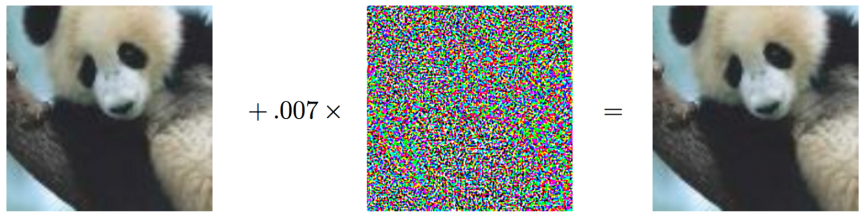
Content

Definition (Robustness Problem)

Given a classification model $f(\theta, \mathbf{x}) : \Theta \times \mathbb{R}^H \to \mathbb{R}^K$ parameterized by θ, data points drawn from the distribution $(\mathbf{x}, y) \sim \mathcal{D}$ and loss function $\mathcal{L}$, robustness problem is formulation as follows:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}'\in\mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta, \mathbf{x}'), y) \tag{1}$$

where $\mathcal{S}_\epsilon(\mathbf{x})$ is called the adversarial budget: $\mathcal{S}_\epsilon(\mathbf{x}) = \{\mathbf{x}' | \|\mathbf{x} - \mathbf{x}'\|_\infty \leq \epsilon\}$.



"panda"

57.7% confidence

noise

"gibbon"

99.3% confidence

$$\max_{\|\mathbf{x}-\mathbf{x}'\|_\infty \leq \epsilon} \mathcal{L}(f(\theta, \mathbf{x}'), y) \tag{2}$$

[1] "Explaining and harnessing adversarial examples." ICLR 2014.

[2] "Towards deep learning models resistant to adversarial attacks." ICLR 2018.

$$\max_{\|\mathbf{x}-\mathbf{x}'\|_\infty \leq \epsilon} \mathcal{L}(f(\theta, \mathbf{x}'), y) \tag{2}$$

- Fast Gradient Sign Method (FGSM) [1].

$$\mathbf{x}' \leftarrow \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f(\theta, \mathbf{x}'), y)) \tag{3}$$

[1] "Explaining and harnessing adversarial examples." ICLR 2014.

[2] "Towards deep learning models resistant to adversarial attacks." ICLR 2018.

$$\max_{\|\mathbf{x}-\mathbf{x}'\|_\infty \leq \epsilon} \mathcal{L}(f(\theta, \mathbf{x}'), y) \tag{2}$$

- Fast Gradient Sign Method (FGSM) [1].

$$\mathbf{x}' \leftarrow \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f(\theta, \mathbf{x}'), y)) \tag{3}$$

- Projected Gradient Descent (PGD) [2] $\sim$ iterative fast gradient sign method.

$$\mathbf{x}^{(t+1)} \leftarrow \Pi_{\{\mathbf{x}' | \|\mathbf{x}'-\mathbf{x}\| \leq \epsilon\}} \left[\mathbf{x}^{(t)} + \alpha \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f(\theta, \mathbf{x}^{(t)}), y)) \right] \tag{4}$$

[1] "Explaining and harnessing adversarial examples." ICLR 2014.

[2] "Towards deep learning models resistant to adversarial attacks." ICLR 2018.

$$\max_{\|\mathbf{x}-\mathbf{x}'\|_\infty \leq \epsilon} \mathcal{L}(f(\theta, \mathbf{x}'), y) \tag{2}$$

- Fast Gradient Sign Method (FGSM) [1].

$$\mathbf{x}' \leftarrow \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f(\theta, \mathbf{x}'), y)) \tag{3}$$

- Projected Gradient Descent (PGD) [2] $\sim$ iterative fast gradient sign method.

$$\mathbf{x}^{(t+1)} \leftarrow \Pi_{\{\mathbf{x}' | \|\mathbf{x}'-\mathbf{x}\| \leq \epsilon\}} \left[\mathbf{x}^{(t)} + \alpha \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f(\theta, \mathbf{x}^{(t)}), y)) \right] \tag{4}$$

- Adversarial training: first generate (optimal) adversarial examples $\mathbf{x}'$ and then train model parameters based on $\mathbf{x}'$.

[1] "Explaining and harnessing adversarial examples." ICLR 2014.

[2] "Towards deep learning models resistant to adversarial attacks." ICLR 2018.

Introduction

Adversarial Training

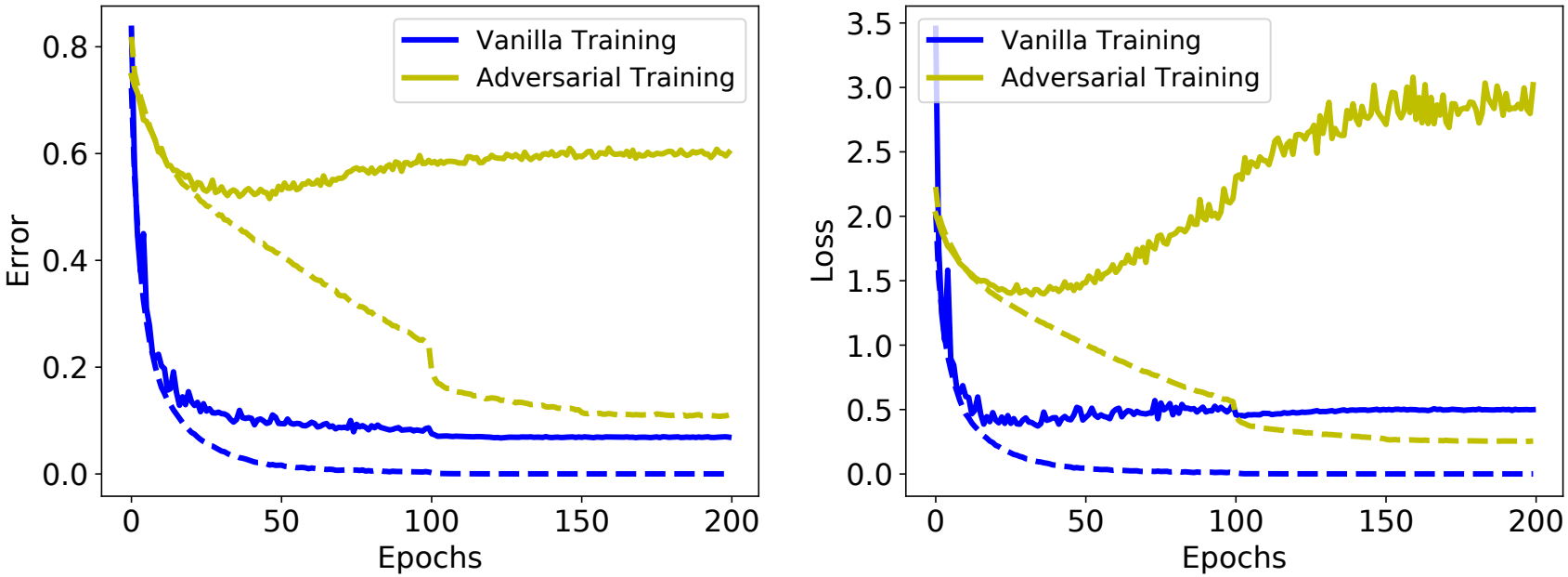


Figure: The training (dashed line) and test (solid line) curve in error (left) and loss (right). The model is ResNet18; the dataset is CIFAR10.

Introduction

Adversarial Training

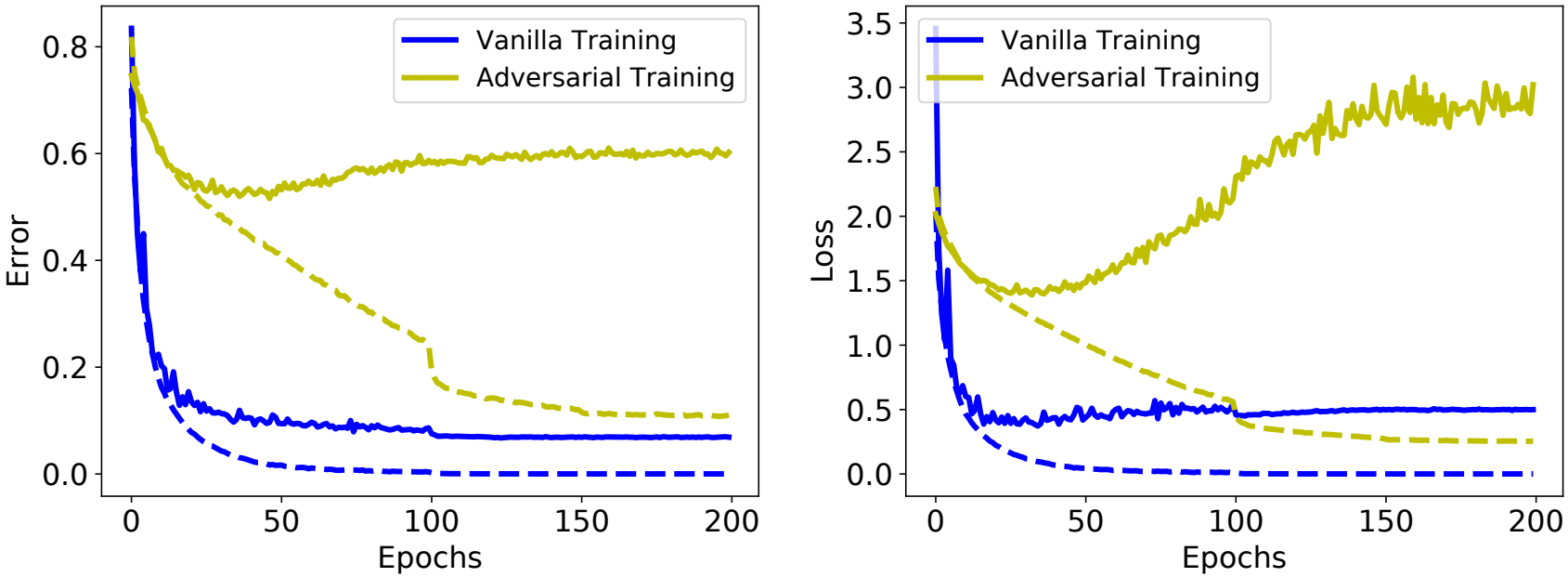


Figure: The training (dashed line) and test (solid line) curve in error (left) and loss (right). The model is ResNet18; the dataset is CIFAR10.

Compared with vanilla training, adversarial training has

- Slower convergence
- Larger generalization gap

Content

Theoretical Analysis

Notation

- $\{\mathbf{x}_i, y_i\}_{i=1}^N$: the given dataset.
- $\mathcal{L}(\theta)$: vanilla loss of parameter θ.
- $\mathcal{L}_\epsilon(\theta)$: adversarial loss of parameter θ when the adversarial budget size is ϵ.
- $g(\theta, \mathbf{x})$, $g_\epsilon(\theta, \mathbf{x})$: vanilla and adversarial loss for an individual data point.

$$\begin{aligned} \min_\theta \mathcal{L}_\epsilon(\theta) &:= \frac{1}{N} \sum_{i=1}^N g_\epsilon(\mathbf{x}_i, \theta) \\ g_\epsilon(\mathbf{x}_i, \theta) &:= \max_{\mathbf{x}_i' \in \mathcal{S}_\epsilon^{(p)}(\mathbf{x}_i)} g(\mathbf{x}_i', \theta) \ . \end{aligned} \tag{5}$$

Theoretical Analysis

Linear Model

For multiclass classification, $K \geq 3$ and the logit function is $f(\mathbf{W}) = [\mathbf{w}_1^T\mathbf{x}, \mathbf{w}_2^T\mathbf{x}, ..., \mathbf{w}_K^T\mathbf{x}]$ where $\mathbf{W} := \{\mathbf{w}_i\}_{i=1}^K \in \mathbb{R}^{m\times K}$ are the trainable parameters.

- $g(\mathbf{x}, \mathbf{W}) = \log\left(1 + \sum_{j\neq y} \exp^{(\mathbf{w}_j - \mathbf{w}_y)\mathbf{x}}\right)$.
- $g_\epsilon(\mathbf{x}, \mathbf{W})$: no analytical form, but convex.

Theoretical Analysis

Linear Model

For multiclass classification, $K \geq 3$ and the logit function is $f(\mathbf{W}) = [\mathbf{w}_1^T\mathbf{x}, \mathbf{w}_2^T\mathbf{x}, ..., \mathbf{w}_K^T\mathbf{x}]$ where $\mathbf{W} := \{\mathbf{w}_i\}_{i=1}^K \in \mathbb{R}^{m\times K}$ are the trainable parameters.

- $g(\mathbf{x}, \mathbf{W}) = \log\left(1 + \sum_{j\neq y} \exp^{(\mathbf{w}_j - \mathbf{w}_y)\mathbf{x}}\right)$.
- $g_\epsilon(\mathbf{x}, \mathbf{W})$: no analytical form, but convex.

Definition (Version Space)

The version space of $g_\epsilon(\mathbf{x}, \mathbf{W})$ is defined as follows:

$$\mathcal{V}_\epsilon = \left\{ \mathbf{W} \middle| (\mathbf{w}_i - \mathbf{w}_y)\mathbf{x}' \leq 0, \forall i \in [K], \mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x}) \right\} . \tag{6}$$

Theoretical Analysis

Linear Model

For multiclass classification, $K \geq 3$ and the logit function is $f(\mathbf{W}) = [\mathbf{w}_1^T\mathbf{x}, \mathbf{w}_2^T\mathbf{x}, ..., \mathbf{w}_K^T\mathbf{x}]$ where $\mathbf{W} := \{\mathbf{w}_i\}_{i=1}^K \in \mathbb{R}^{m\times K}$ are the trainable parameters.

- $g(\mathbf{x}, \mathbf{W}) = \log\left(1 + \sum_{j\neq y} \exp^{(\mathbf{w}_j - \mathbf{w}_y)\mathbf{x}}\right)$.
- $g_\epsilon(\mathbf{x}, \mathbf{W})$: no analytical form, but convex.

Definition (Version Space)

The version space of $g_\epsilon(\mathbf{x}, \mathbf{W})$ is defined as follows:

$$\mathcal{V}_\epsilon = \left\{ \mathbf{W} \middle| (\mathbf{w}_i - \mathbf{w}_y)\mathbf{x}' \leq 0, \forall i \in [K], \mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x}) \right\} . \tag{6}$$

- $\mathcal{V}_\epsilon$ is a convex set of 'good directions'.
- $\forall \mathbf{W} \in \mathcal{V}_\epsilon$ and $\mathbf{W} \neq \mathbf{0}$, $\lim_{\gamma\to\infty} g_\epsilon(\mathbf{x}, \gamma\mathbf{W}) = 0$.

Definition (Version Space)

The version space of $g_\epsilon(\mathbf{x}, \mathbf{W})$ is defined as follows:

$$\mathcal{V}_\epsilon = \left\{ \mathbf{W} \middle| (\mathbf{w}_i - \mathbf{w}_y)\mathbf{x}' \leq 0, \forall i \in [K], \mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x}) \right\} . \tag{7}$$

Proposition

Given the version space $\mathcal{V}_\epsilon$ defined in (7), then $\mathcal{V}_{\epsilon_2} \subseteq \mathcal{V}_{\epsilon_1}$ when $\epsilon_1 \leq \epsilon_2$.

Definition (Version Space)

The version space of $g_\epsilon(\mathbf{x}, \mathbf{W})$ is defined as follows:

$$\mathcal{V}_\epsilon = \left\{ \mathbf{W} \middle| (\mathbf{w}_i - \mathbf{w}_y)\mathbf{x}' \leq 0, \forall i \in [K], \mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x}) \right\} . \tag{7}$$

Proposition

Given the version space $\mathcal{V}_\epsilon$ defined in (7), then $\mathcal{V}_{\epsilon_2} \subseteq \mathcal{V}_{\epsilon_1}$ when $\epsilon_1 \leq \epsilon_2$.

- The set of 'good directions' will shrink with increase of ϵ.

Theoretical Analysis

Linear Model

- Now, let's talk about 'bad directions'.

Definition

We define set $\mathcal{T}_\epsilon$ based on $g_\epsilon(\mathbf{x}, \mathbf{W})$:

$$\mathcal{T}_\epsilon = \left\{ \mathbf{W} \middle| 0 \in \arg\min_{\gamma} g_\epsilon(\mathbf{x}, \gamma \mathbf{W}) \right\} . \tag{8}$$

Definition

We define set $\mathcal{T}_\epsilon$ based on $g_\epsilon(\mathbf{x}, \mathbf{W})$:

$$\mathcal{T}_\epsilon = \left\{ \mathbf{W} \middle| 0 \in \arg\min_{\gamma} g_\epsilon(\mathbf{x}, \gamma \mathbf{W}) \right\} . \tag{8}$$

Theorem

Given $\mathcal{T}_\epsilon$ defined in (8), 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 $\mathbf{0} \in \arg\min_{\mathbf{W}} g_\epsilon(\mathbf{x}, \mathbf{W})$.

Theoretical Analysis

Linear Model

Definition

We define set $\mathcal{T}_\epsilon$ based on $g_\epsilon(\mathbf{x}, \mathbf{W})$:

$$\mathcal{T}_\epsilon = \left\{ \mathbf{W} \middle| 0 \in \arg\min_{\gamma} g_\epsilon(\mathbf{x}, \gamma \mathbf{W}) \right\} . \tag{8}$$

Theorem

Given $\mathcal{T}_\epsilon$ defined in (8), 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 $\mathbf{0} \in \arg\min_{\mathbf{W}} g_\epsilon(\mathbf{x}, \mathbf{W})$.

- The set of 'bad directions' will expand with increase of ϵ.
- For suffiently large ϵ, all non-zero vectors are 'bad directions', then the model gets stuck at the orgin.
- Similar obvervations in nonlinear models, although the theorem is not applicable in this case.

For nonlinear models, there is no analytical form of $g(\mathbf{x}, \theta)$ or $g_\epsilon(\mathbf{x}, \theta)$. Instead, we assume the Lipschitzian smoothness of g.

Assumption (Lipschitzian Smoothness)

The function g satisfies the Lipschitzian smoothness conditions:

$$\begin{aligned} \|g(\mathbf{x}, \theta_1) - g(\mathbf{x}, \theta_2)\| &\leq L_\theta \|\theta_1 - \theta_2\| \,, \\ \|\nabla_\theta g(\mathbf{x}, \theta_1) - \nabla_\theta g(\mathbf{x}, \theta_2)\| &\leq L_{\theta\theta} \|\theta_1 - \theta_2\| \,, \\ \|\nabla_\theta g(\mathbf{x}_1, \theta) - \nabla_\theta g(\mathbf{x}_2, \theta)\| &\leq L_{\theta\mathbf{x}} \|\mathbf{x}_1 - \mathbf{x}_2\|_p \,. \end{aligned} \tag{9}$$

L_θ, $L_{\theta\theta}$ and $L_{\theta\mathbf{x}}$ are called Lipschitz constants.

Assumption (Lipschitzian Smoothness of g)

The function g satisfies the Lipschitzian smoothness conditions:

$$\begin{aligned} \|g(\mathbf{x}, \theta_1) - g(\mathbf{x}, \theta_2)\| &\leq L_\theta \|\theta_1 - \theta_2\| \,, \\ \|\nabla_\theta g(\mathbf{x}, \theta_1) - \nabla_\theta g(\mathbf{x}, \theta_2)\| &\leq L_{\theta\theta} \|\theta_1 - \theta_2\| \,, \\ \|\nabla_\theta g(\mathbf{x}_1, \theta) - \nabla_\theta g(\mathbf{x}_2, \theta)\| &\leq L_{\theta\mathbf{x}} \|\mathbf{x}_1 - \mathbf{x}_2\|_p \,. \end{aligned} \tag{10}$$

L_θ, $L_{\theta\theta}$ and $L_{\theta\mathbf{x}}$ are called Lipschitz constants.

Assumption (Lipschitzian Smoothness of g)

The function g satisfies the Lipschitzian smoothness conditions:

$$\begin{aligned} \|g(\mathbf{x}, \theta_1) - g(\mathbf{x}, \theta_2)\| &\leq L_\theta \|\theta_1 - \theta_2\| , \\ \|\nabla_\theta g(\mathbf{x}, \theta_1) - \nabla_\theta g(\mathbf{x}, \theta_2)\| &\leq L_{\theta\theta} \|\theta_1 - \theta_2\| , \\ \|\nabla_\theta g(\mathbf{x}_1, \theta) - \nabla_\theta g(\mathbf{x}_2, \theta)\| &\leq L_{\theta\mathbf{x}} \|\mathbf{x}_1 - \mathbf{x}_2\|_p . \end{aligned} \tag{10}$$

L_θ, $L_{\theta\theta}$ and $L_{\theta\mathbf{x}}$ are called Lipschitz constants.

Theorem (Smoothness of g_ϵ)

If the Lipschitzian smoothness of g holds, then we have:

$$\begin{aligned} \|g_\epsilon(\mathbf{x}, \theta_1) - g_\epsilon(\mathbf{x}, \theta_2)\| &\leq L_\theta \|\theta_1 - \theta_2\| , \\ \|\nabla_\theta g_\epsilon(\theta_1) - \nabla_\theta g_\epsilon(\mathbf{x}, \theta_2)\| &\leq L_{\theta\theta} \|\theta_1 - \theta_2\| + 2\epsilon L_{\theta\mathbf{x}} . \end{aligned} \tag{11}$$

Theoretical Analysis

Nonlinear Model

- The first order smoothness of g is preserved in g_ϵ.
- The second order smoothness of g is, however, broken in g_ϵ. It is because (optimal) adversarial examples $\mathbf{x}'$ depends on model parameters θ.

Theoretical Analysis

Nonlinear Model

- The first order smoothness of g is preserved in g_ϵ.
- The second order smoothness of g is, however, broken in g_ϵ. It is because (optimal) adversarial examples $\mathbf{x}'$ depends on model parameters θ.

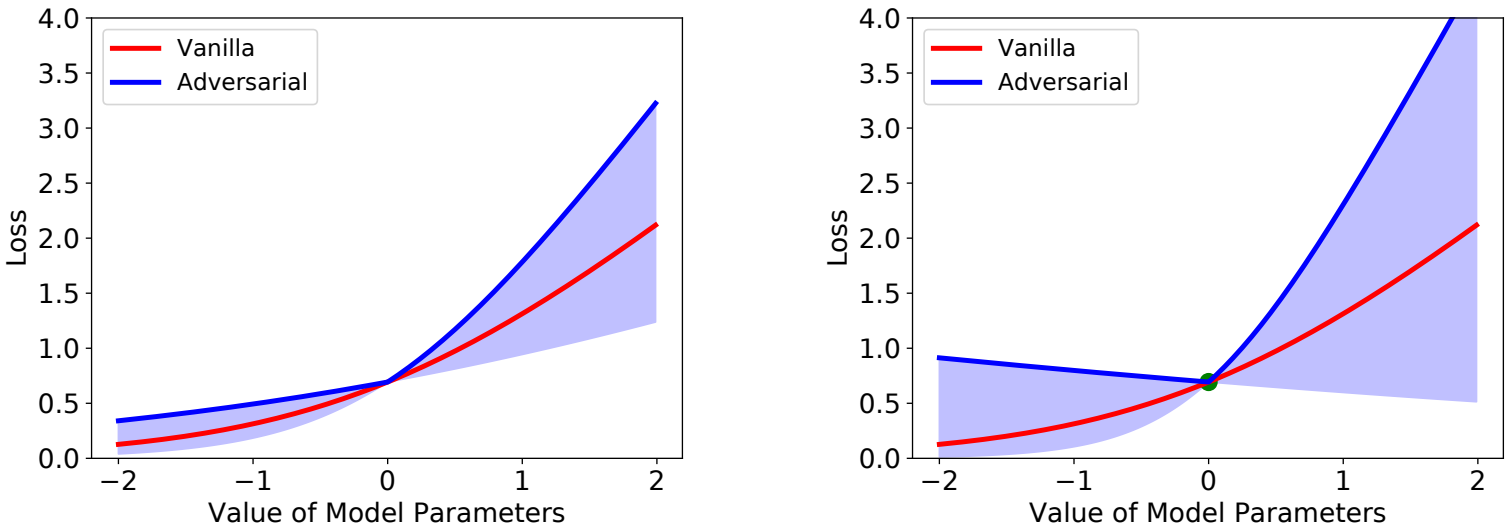


Figure: 2D sketch diagram showing the vanilla and the adversarial loss landscape. The clean input data x is 1.0 and loss function $g(x, \theta) = \log(1 + \exp(\theta x))$ (left: $\epsilon = 0.6$, right: $\epsilon = 1.2$).

Theoretical Analysis

Nonlinear Model

- The first order smoothness of g is preserved in g_ϵ.
- The second order smoothness of g is, however, broken in g_ϵ. It is because (optimal) adversarial examples $\mathbf{x}'$ depends on model parameters θ.

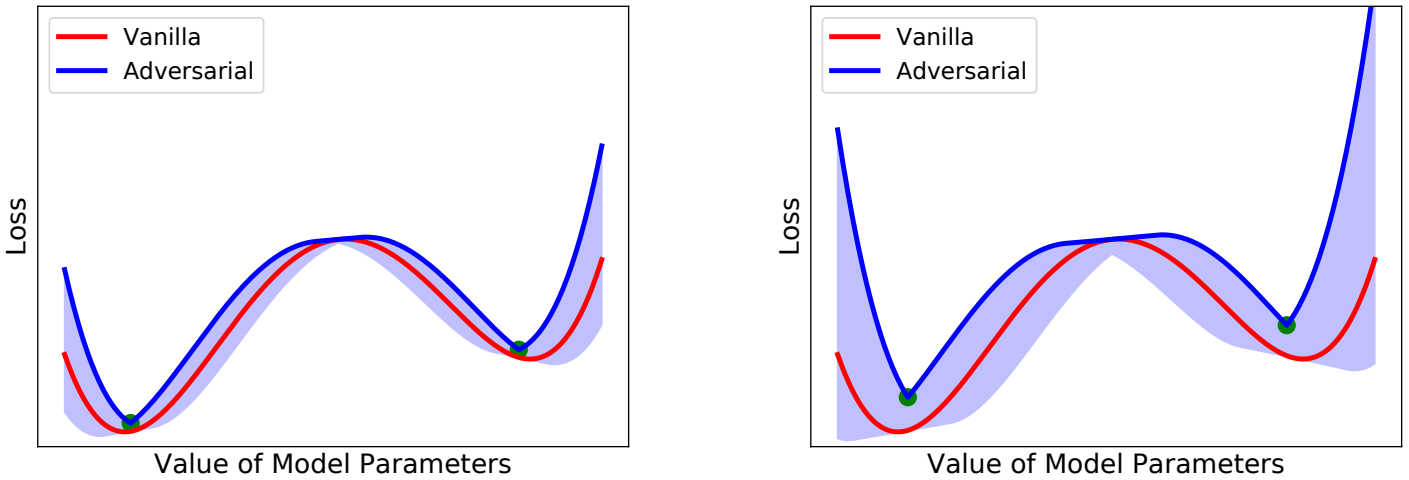


Figure: 2D sketch diagram in the nonconvex cases. The adversarial budget in the left part is smaller than the right part. The minima are sharper when the adversarial budget is bigger.

Why smoothness matters?

Theorem (Convergence)

Let Lipschitzian assumption holds, 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*

$$\forall \gamma \geq 2, P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > \gamma \epsilon L_{\theta \mathbf{x}}) < \frac{4}{\gamma^2 - 2\gamma + 4} \,. \tag{12}$$

Theoretical Analysis

Nonlinear Model

Why smoothness matters?

Theorem (Convergence)

Let Lipschitzian assumption holds, 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*

$$\forall \gamma \geq 2, P(\|\nabla_\theta \mathcal{L}_\epsilon(\theta_t)\| > \gamma \epsilon L_{\theta\mathbf{x}}) < \frac{4}{\gamma^2 - 2\gamma + 4} \,. \tag{12}$$

- In adversarial training, i.e., $\epsilon > 0$, we cannot guarantee convergence to a critical point.
- The gradients are non-vanishing, and we can only bound the probability of obtaining gradients whose magnitude is larger than $2\epsilon L_{\theta\mathbf{x}}$.

Theoretical Analysis

Nonlinear Model

Why smoothness matters?

[3] "On large-batch training for deep learning: Generalization gap and sharp minima." ICLR 2017.

Theoretical Analysis

Nonlinear Model

Why smoothness matters?

- Non-smooth function usually has sharp minima. Sharp minima is shown to generalize worst for deep learning models empirically. [3]

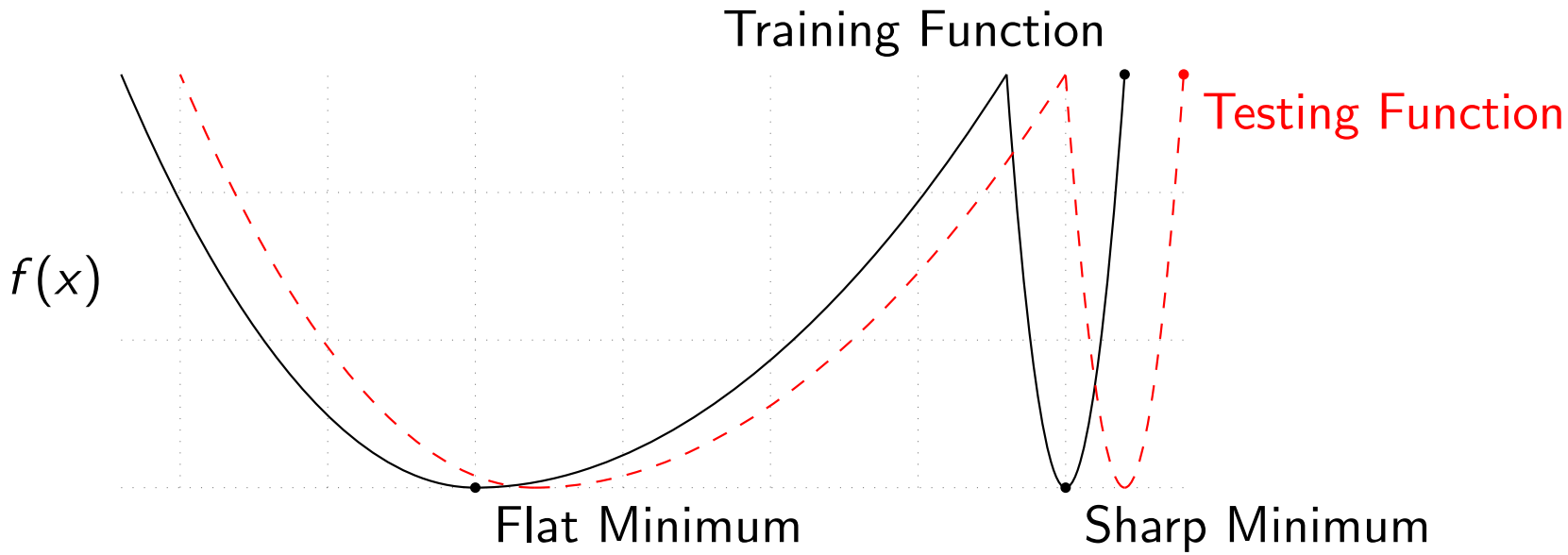


Figure: A Conceptual Sketch of Flat and Sharp Minima. The Y-axis indicates value of the loss function and the X-axis the variables (parameters)

[3] "On large-batch training for deep learning: Generalization gap and sharp minima." ICLR 2017.

Content

Numerical Analysis

Experimental Settings

- Models and datasets: LeNet on MNIST, VGG16 and ResNet18 on CIFAR10.
- We use w to denote the width of the network, standard LeNet [LeCun 95], VGG16 [Zhang 15], ResNet18 [He 16] are marked as $w = 16$.
- Each experiment is run for 3 times.

Numerical Analysis

Gradient Analysis

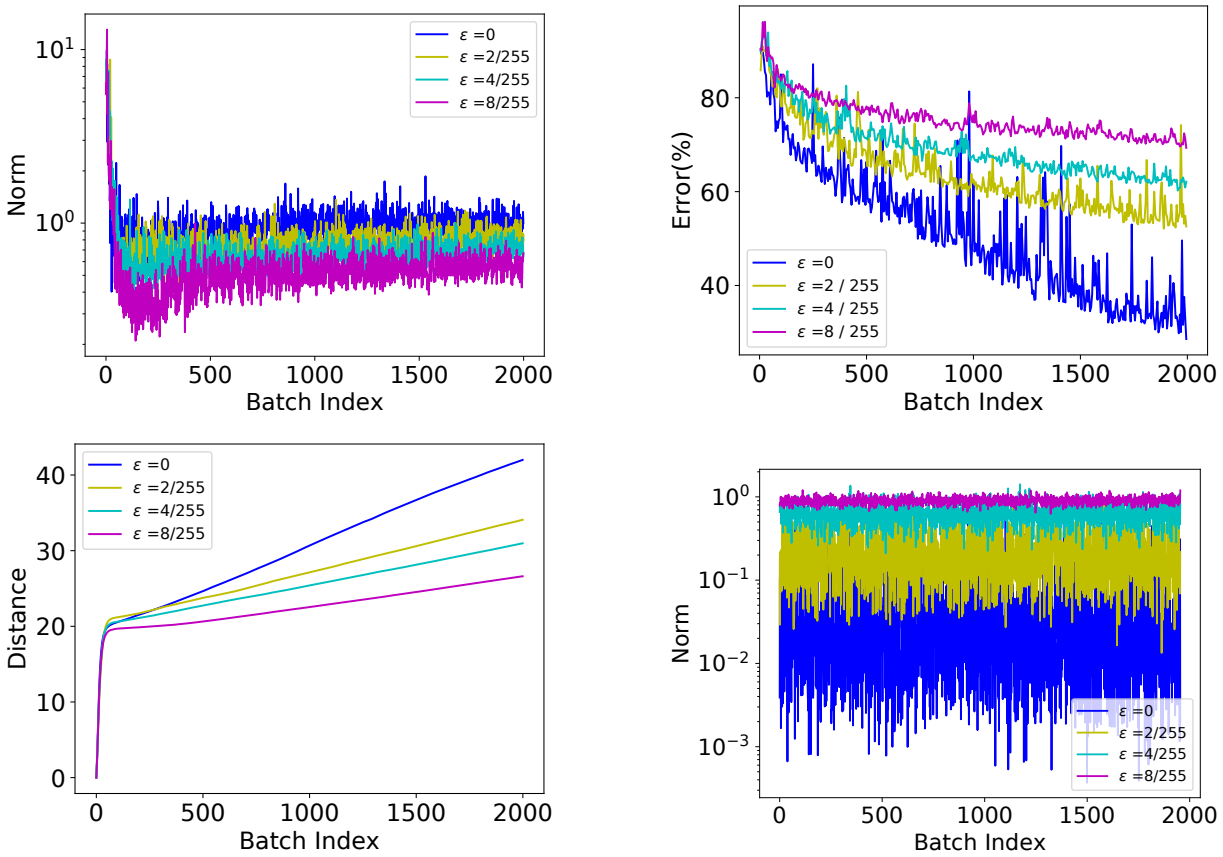


Figure: Norm of the stochastic gradient $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$ (Upper Left, Bottom Right), robust training error $\mathcal{E}_\epsilon(\theta)$ (Upper Right), and distance from the initial point $\|\theta - \theta_0\|$ (Bottom Left) during the first or last 2000 mini-batch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

Numerical Analysis

Gradient Analysis

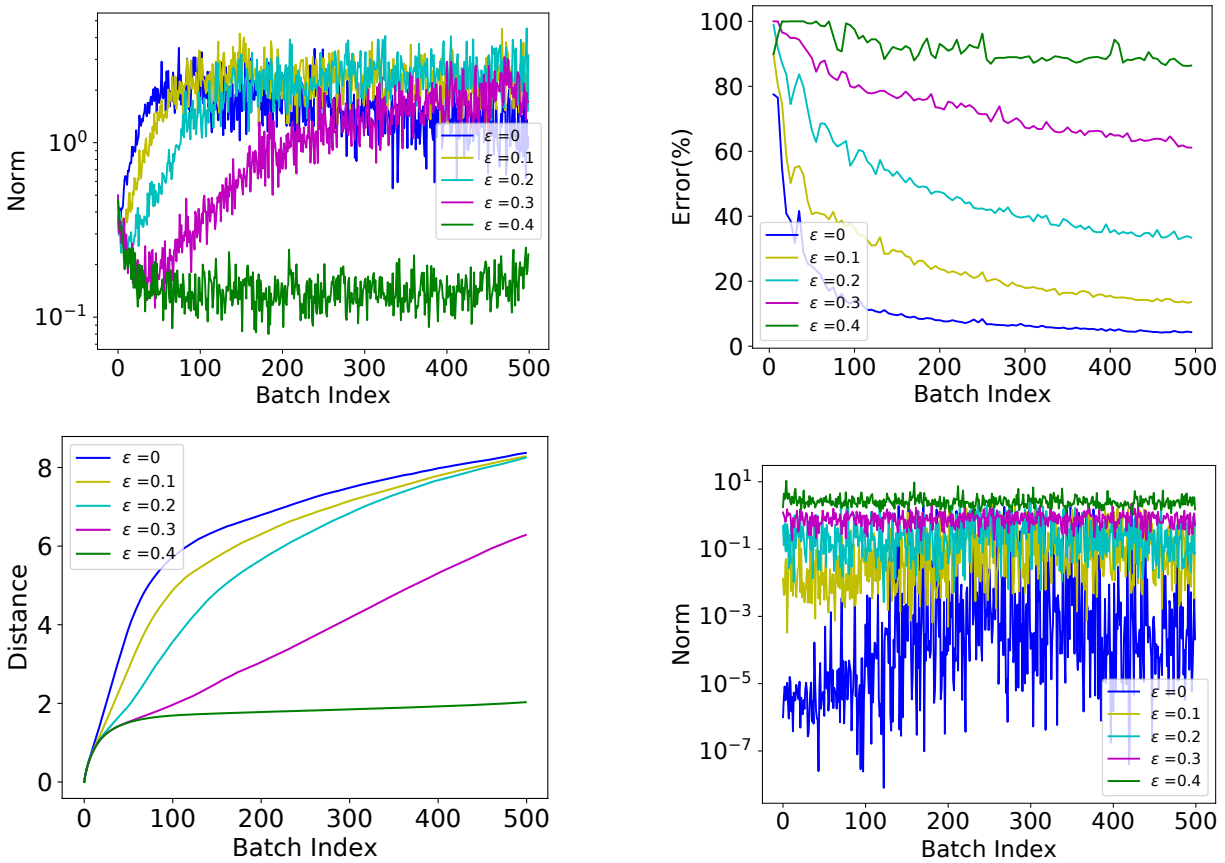


Figure: Norm of the stochastic gradient $\|\nabla_\theta \widehat{\mathcal{L}}_\epsilon(\theta)\|$ (Upper Left, Bottom Right), robust training error $\mathcal{E}_\epsilon(\theta)$ (Upper Right), and distance from the initial point $\|\theta - \theta_0\|$ (Bottom Left) during the first or last 500 mini-batch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

Extreme cases ('dead layer') for ReLU networks. (small model, large ϵ)

Extreme cases ('dead layer') for ReLU networks. (small model, large ϵ)

- Pre-activation of one layer is all negative for all training instances.
- The network is effectively broken into two parts.

Extreme cases ('dead layer') for ReLU networks. (small model, large ϵ)

- Pre-activation of one layer is all negative for all training instances.
- The network is effectively broken into two parts.
  - Layers before the 'dead layer' will never be updated.
  - Layers after the 'dead layer' don't depend on the input.

Extreme cases ('dead layer') for ReLU networks. (small model, large ϵ)

- Pre-activation of one layer is all negative for all training instances.
- The network is effectively broken into two parts.
  - Layers before the 'dead layer' will never be updated.
  - Layers after the 'dead layer' don't depend on the input.
- The model gets stuck in a space consisting of only constant classifiers.

- The sharpness of $\mathcal{L}_\epsilon(\theta)$ w.r.t. θ is captured by the top Eigenvalues of its Hessian matrix $\nabla^2_\theta \mathcal{L}_\epsilon(\theta)$.

Numerical Analysis

Hessian Analysis

- The sharpness of $\mathcal{L}_\epsilon(\theta)$ w.r.t. θ is captured by the top Eigenvalues of its Hessian matrix $\nabla^2_\theta \mathcal{L}_\epsilon(\theta)$.
- Theorem about the smoothness of $\mathcal{L}_\epsilon(\theta)$ indicates the numerical results of Hessian top eigenvalues will increase with ϵ.

- Theorem about the smoothness of $\mathcal{L}_\epsilon(\theta)$ indicates the numerical results of Hessian top eigenvalues will increase with ϵ.

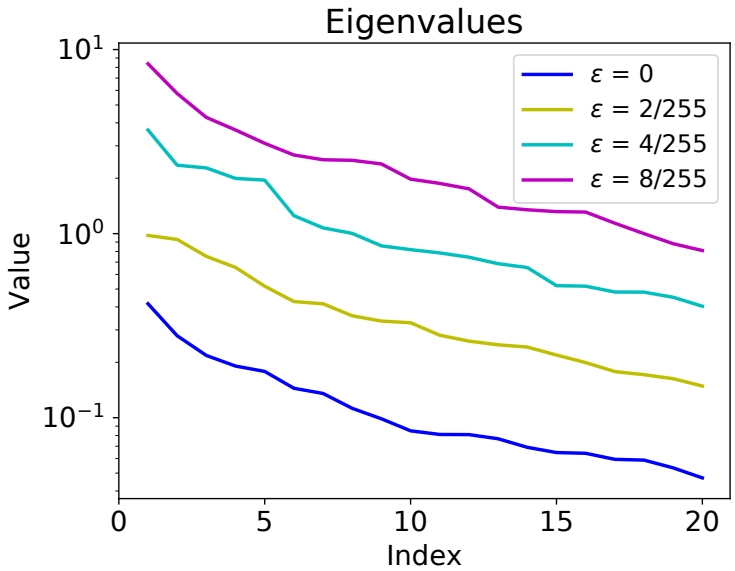


Figure: Top Hessian eigenvalues of ResNet18 models on CIFAR10.

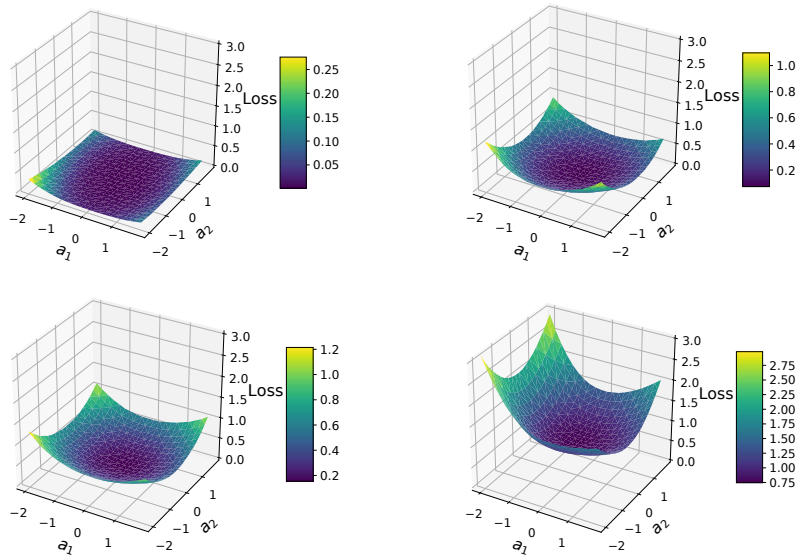


Figure: Visualization of landscape in the neighborhood of minima. (top left: $\epsilon = 0$, top right: $\epsilon = 2/255$, bottom left: $\epsilon = 4/255$, bottom right: $\epsilon = 8/255$.)

- Theorem about the smoothness of $\mathcal{L}_\epsilon(\theta)$ indicates the numerical results of Hessian top eigenvalues will increase with ϵ.

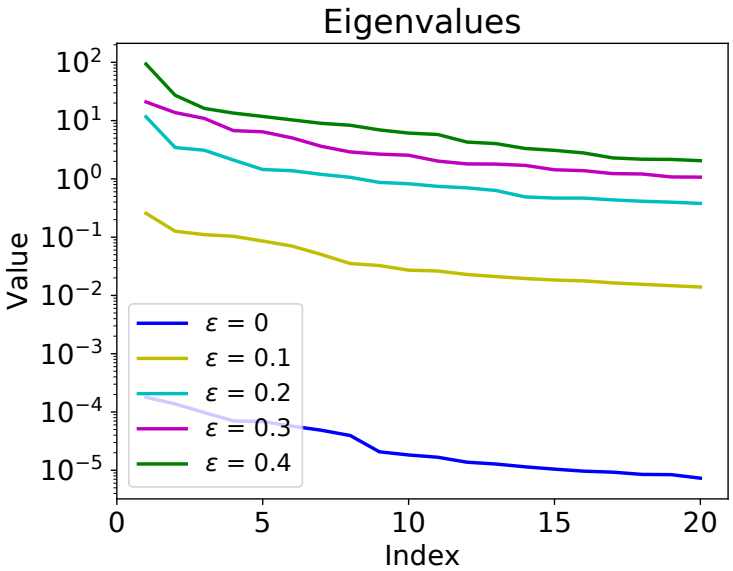


Figure: Top Hessian eigenvalues of LeNet models on MNIST.

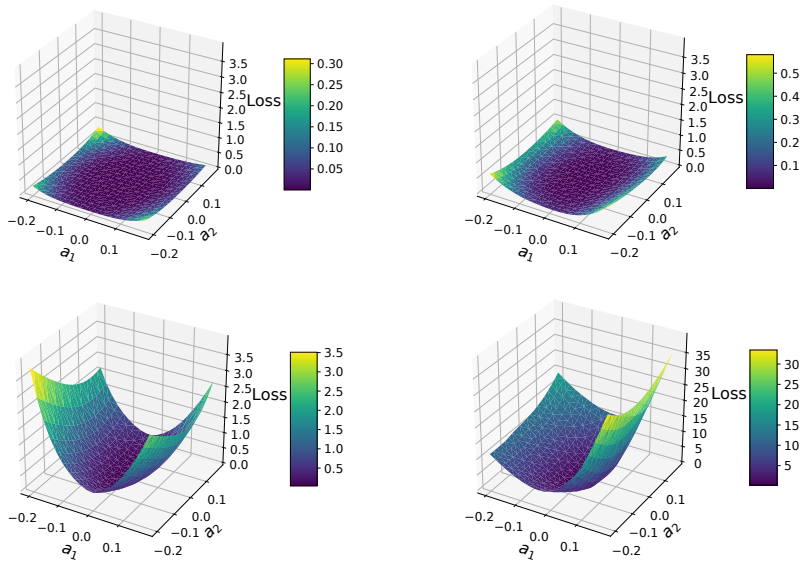


Figure: Visualization of landscape in the neighborhood of minima. (top left: $\epsilon = 0$, top right: $\epsilon = 0.3$, bottom left: $\epsilon = 0.3$, bottom right: $\epsilon = 0.4$.)

- To confirm the reasons behind the increased curvature of $\mathcal{L}_\epsilon(\theta)$ for larger ϵ, we check the similarity of adversarial examples generated by parameter $\theta + a\mathbf{v}$ and $\theta - a\mathbf{v}$.
- $\mathbf{v}$ can be a random unit vector or the top eigenvector of the Hessian matrix.

Numerical Analysis

Hessian Analysis

- To confirm the reasons behind the increased curvature of $\mathcal{L}_\epsilon(\theta)$ for larger ϵ, we check the similarity of adversarial examples generated by parameter $\theta + a\mathbf{v}$ and $\theta - a\mathbf{v}$.
- $\mathbf{v}$ can be a random unit vector or the top eigenvector of the Hessian matrix.

$\mathbf{v}$		$\epsilon = 2/255$		$\epsilon = 4/255$		$\epsilon = 8/255$	
		Cosine	Test(%)	Cosine	Test(%)	Cosine	Test(%)
Eigenvector	a = 0.0	0.976	27.0	0.962	42.1	0.945	58.6
	a = 0.1	0.471	28.0	0.475	43.7	0.461	61.3
	a = 0.3	0.148	34.2	0.157	49.9	0.140	70.2
	a = 0.5	0.075	44.1	0.078	63.1	0.069	83.9
	a = 1.0	0.050	79.0	0.046	88.4	0.034	89.0
Random	a = 0.0	0.976	27.0	0.962	42.1	0.945	58.6
	a = 0.1	0.975	27.1	0.962	42.1	0.945	58.6
	a = 0.3	0.973	27.1	0.960	42.1	0.944	58.6
	a = 0.5	0.968	27.1	0.957	42.1	0.942	58.7
	a = 1.0	0.955	27.1	0.946	42.1	0.935	58.6

Table: Average cosine similarity of perturbations in the training set and its corresponding test error between ResNet18 models with parameters $\theta + a\mathbf{v}$ and $\theta - a\mathbf{v}$ on CIFAR10.

Numerical Analysis

Hessian Analysis

- To confirm the reasons behind the increased curvature of $\mathcal{L}_\epsilon(\theta)$ for larger ϵ, we check the similarity of adversarial examples generated by parameter $\theta + a\mathbf{v}$ and $\theta - a\mathbf{v}$.
- $\mathbf{v}$ can be a random unit vector or the top eigenvector of the Hessian matrix.

$\mathbf{v}$		$\epsilon = 0.1$		$\epsilon = 0.2$		$\epsilon = 0.3$		$\epsilon = 0.4$	
		Cosine	Test(%)	Cosine	Test(%)	Cosine	Test(%)	Cosine	Test(%)
Eigenvector	a = 0.00	0.861	3.34	0.837	6.72	0.918	6.94	0.928	8.74
	a = 0.01	0.847	3.36	0.823	6.83	0.241	7.48	0.188	15.9
	a = 0.03	0.812	3.42	0.789	7.07	0.236	9.19	0.282	31.4
	a = 0.05	0.779	3.53	0.757	7.66	0.274	11.2	0.375	44.6
	a = 0.10	0.705	4.20	0.683	10.4	0.367	19.1	0.455	99.3
Random	a = 0.00	0.861	3.34	0.837	6.72	0.918	6.94	0.928	8.74
	a = 0.01	0.860	3.34	0.837	6.71	0.913	6.99	0.906	8.80
	a = 0.03	0.861	3.34	0.837	6.72	0.888	6.99	0.818	8.65
	a = 0.05	0.860	3.35	0.837	6.69	0.863	6.94	0.745	8.49
	a = 0.10	0.860	3.35	0.837	6.67	0.780	6.82	0.653	8.90

Table: The average cosine similarity of perturbations in the training set and its corresponding test error between LeNet models with parameter $\theta + a\mathbf{v}$ and $\theta - a\mathbf{v}$ on MNIST.

Content

Periodic Adversarial Scheduling

Formulation

- Instead of using constant ϵ, we introduce warmup in ϵ.
- We use periodic scheduling in both learning rate and adversarial budget size.

Periodic Adversarial Scheduling

Formulation

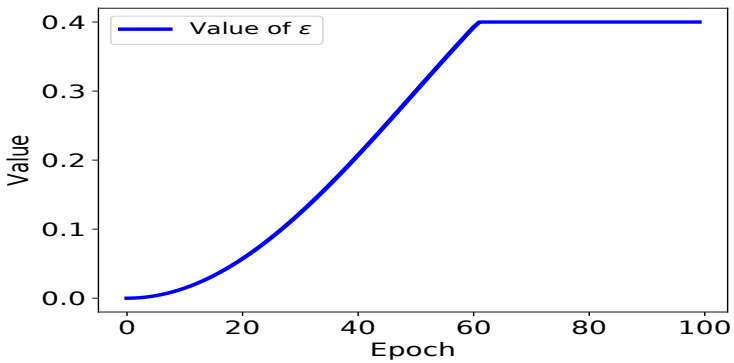
- Instead of using constant ϵ, we introduce warmup in ϵ.
- We use periodic scheduling in both learning rate and adversarial budget size.

Let the increasing list $\{T_0, T_1, ..., T_M\}$ be the epoch indices when the learning rate and adversarial budget is reset. If the current epoch index d satisfy $T_{i-1} \leq d < T_i$ and $p = \frac{d - T_{i-1}}{T_i - T_{i-1}}$, we introduce two different functions to schedule the value of ϵ.

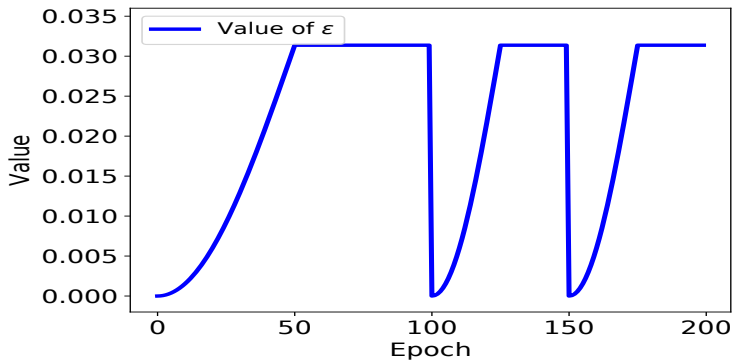
$$\begin{aligned} \epsilon_{cos}(d) &= \frac{1}{2}(1 - \cos p\pi)(\epsilon_{max} - \epsilon_{min}) + \epsilon_{min} \ , \\ \epsilon_{lin}(d) &= (\epsilon_{max} - \epsilon_{min})p + \epsilon_{min} \ . \end{aligned} \tag{13}$$

Periodic Adversarial Scheduling

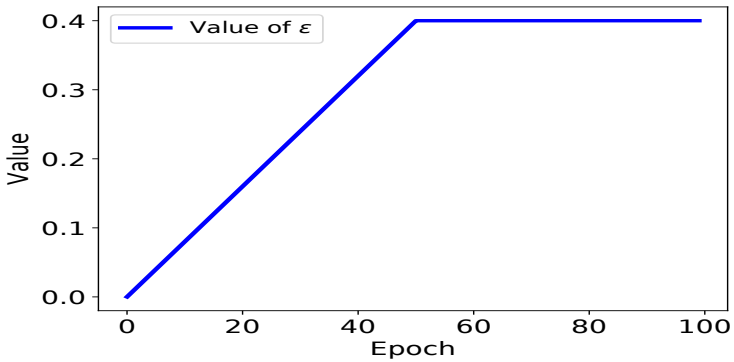
Formulation



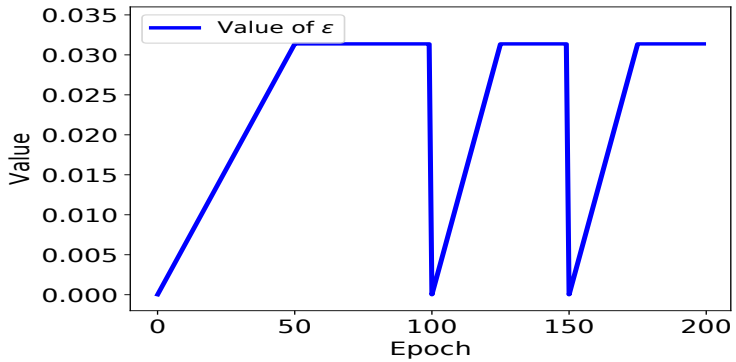
(a) MNIST, LeNet, Cosine



(b) CIFAR10, VGG & ResNet18, Cosine



(c) MNIST, LeNet, Linear

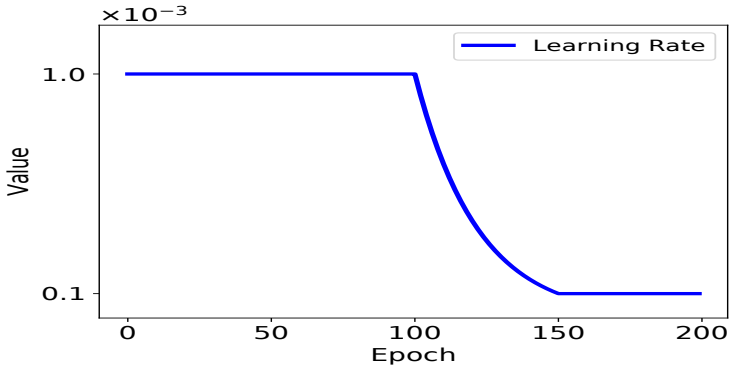


(d) CIFAR10, VGG & ResNet18, Linear

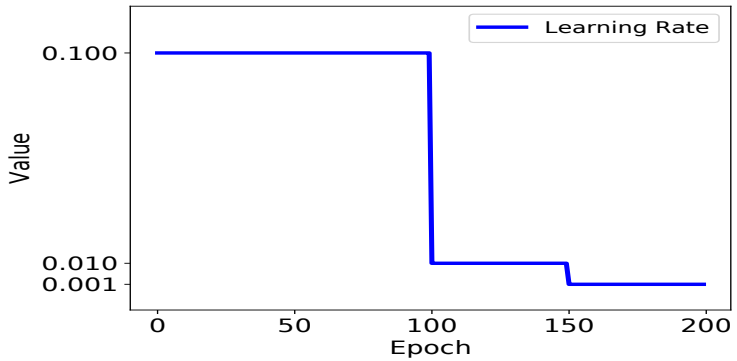
Figure: Adversarial budget scheduling for MNIST and CIFAR10 models.

Periodic Adversarial Scheduling

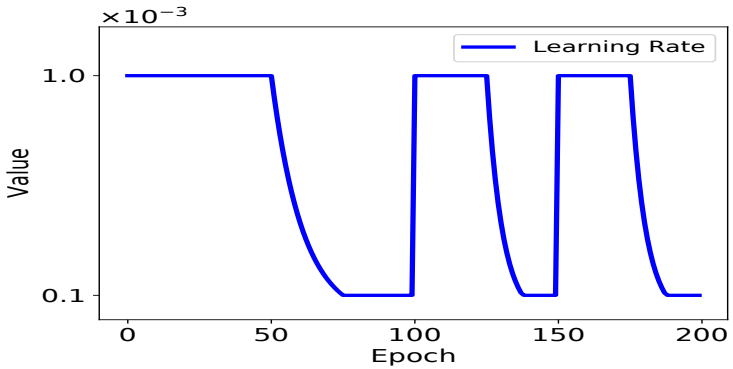
Formulation



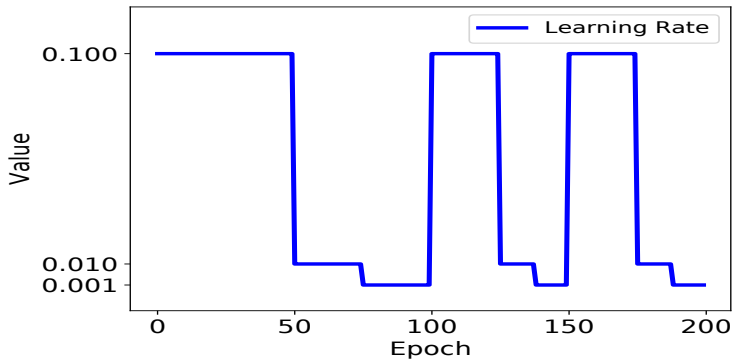
(a) CIFAR10, VGG, Vanilla



(b) CIFAR10, ResNet18, Vanilla



(c) CIFAR10, VGG, Periodic



(d) CIFAR10, ResNet18, Periodic

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

Periodic Adversarial Scheduling

Advantages

- Insensitive to learning rate choice.

Learning Rate	Const	Cosine	Linear
Adam, 1×10^{-4}	8.58 ± 0.89	6.64 ± 0.70	6.69 ± 0.59
Adam, 3×10^{-4}	88.65 ± 0.00	7.44 ± 0.52	7.48 ± 0.11
Adam, 1×10^{-3}	88.65 ± 0.00	9.62 ± 0.83	7.48 ± 0.15

Table: Mean robust error and standard derivation of LeNet models with different adversarial budget scheduling schemes and learning rates when $\epsilon = 0.4$.

Periodic Adversarial Scheduling

Advantages

- Improved final results.

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

Table: 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.

Content

Conclusion

Some take away messages

- With the increase of the adversarial budget ϵ, the loss landscape becomes unfavorable to optimization.
  - Non-smoothness.
  - Less-connected minima.
  - Harder to escape initial suboptimal regions.
- Periodic scheduling and model ensembling can help ease the training difficulty and improve robustness.

Discussion

- Ease the overfitting problem in adversarial training.
  - Entropy-SGD [4], Add fisher noise [5] e.t.c.
- Design new activation functions.
  - ReLU is the key ingredients of the 'dead layer' phenomenon.

[4] "Entropy-SGD: Biasing Gradient Descent Into Wide Valleys". ICLR 2017.

[5] "An Empirical Study of Large-Batch Stochastic Gradient Descent with Structured Covariance Noise". 2020.

Thank You!