Evasion Attacks

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology
Computer Engineering Department (CE)
Trustworthy and Secure AI Lab

October 24, 2025

Today's Agenda

Fast Gradient Sign Method(FGSM) Attack

Abstract

- The primary cause of neural networks' vulnerability to adversarial perturbation is their **linear nature**.
- Giving the first explanation of the most intriguing fact about them: their **generalization** across architectures and training sets.
- In this lecture, we introduce a **simple and fast method of generating adversarial examples**.

Smoothness Prior with L_∞

- For problems with well-separated classes, we expect the classifier to assign the same class to x and $x' = x + \eta$ so long as $\|\eta\|_\infty \leq \epsilon$, where ϵ is small.
  - For $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$, $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

The Linear Explanation of Adversarial Examples

- Let $\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$ and $\boldsymbol{x}' = \boldsymbol{x} + \eta$, the dot product between weight vector $\boldsymbol{w}$ and adversarial example x'

The Linear Explanation of Adversarial Examples

- Let $\hat{y} = \boldsymbol{w}^T\boldsymbol{x}$ and $\boldsymbol{x}' = \boldsymbol{x} + \eta$, the dot product between weight vector $\boldsymbol{w}$ and adversarial example x' is as follows

$$\hat{y}' = \boldsymbol{w}^T\boldsymbol{x}' = \boldsymbol{w}^T(\boldsymbol{x} + \eta) = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{w}^T\eta \Rightarrow \hat{y}' - \hat{y} = \boldsymbol{w}^T\eta$$

The adversarial perturbation causes the activation to grow by $\boldsymbol{w}^T\eta$.

The Linear Explanation of Adversarial Examples

- Let $\hat{y} = \boldsymbol{w}^T\boldsymbol{x}$ and $\boldsymbol{x}' = \boldsymbol{x} + \eta$, the dot product between weight vector $\boldsymbol{w}$ and adversarial example x' is as follows

$$\hat{y}' = \boldsymbol{w}^T\boldsymbol{x}' = \boldsymbol{w}^T(\boldsymbol{x} + \eta) = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{w}^T\eta \Rightarrow \hat{y}' - \hat{y} = \boldsymbol{w}^T\eta$$

  The adversarial perturbation causes the activation to grow by $\boldsymbol{w}^T\eta$.
- To generate adversarial example for x, we should maximize $\boldsymbol{w}^T\eta$, such that $\|\eta\|_\infty \leq \epsilon$. Therefore, we have the following maximization problem.

$$\begin{aligned} &\underset{\eta}{\text{argmax}} < \boldsymbol{w}, \eta > \\ &s.t. \quad \|\eta\|_\infty \leq \epsilon \end{aligned}$$

The Linear Explanation of Adversarial Examples

- Let $\hat{y} = \boldsymbol{w}^T\boldsymbol{x}$ and $\boldsymbol{x}' = \boldsymbol{x} + \eta$, the dot product between weight vector $\boldsymbol{w}$ and adversarial example x' is as follows

$$\hat{y}' = \boldsymbol{w}^T\boldsymbol{x}' = \boldsymbol{w}^T(\boldsymbol{x} + \eta) = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{w}^T\eta \Rightarrow \hat{y}' - \hat{y} = \boldsymbol{w}^T\eta$$

  The adversarial perturbation causes the activation to grow by $\boldsymbol{w}^T\eta$.
- To generate adversarial example for x, we should maximize $\boldsymbol{w}^T\eta$, such that $\|\eta\|_\infty \leq \epsilon$. Therefore, we have the following maximization problem.

$$\underset{\eta}{\text{argmax}} < \boldsymbol{w}, \eta >$$
$$s.t. \quad \|\eta\|_\infty \leq \epsilon$$

  The solution to the above problem is $\eta^* = \epsilon.\text{sign}(\boldsymbol{w})$, we have

$$\hat{y}' - \hat{y} = \boldsymbol{w}^T\eta^* = \boldsymbol{w}^T\epsilon.\text{sign}(\boldsymbol{w}) = \epsilon\|\boldsymbol{w}\|_1$$

The Linear Explanation of Adversarial Examples

- Let $\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$ and $\boldsymbol{x}' = \boldsymbol{x} + \eta$, the dot product between weight vector $\boldsymbol{w}$ and adversarial example x' is as follows

$$\hat{y}' = \boldsymbol{w}^T \boldsymbol{x}' = \boldsymbol{w}^T (\boldsymbol{x} + \eta) = \boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{w}^T \eta \Rightarrow \hat{y}' - \hat{y} = \boldsymbol{w}^T \eta$$

  The adversarial perturbation causes the activation to grow by $\boldsymbol{w}^T \eta$.
- To generate adversarial example for x, we should maximize $\boldsymbol{w}^T \eta$, such that $\|\eta\|_\infty \leq \epsilon$. Therefore, we have the following maximization problem.

$$\underset{\eta}{\text{argmax}} < \boldsymbol{w}, \eta >$$
$$s.t. \quad \|\eta\|_\infty \leq \epsilon$$

  The solution to the above problem is $\eta^* = \epsilon.\text{sign}(\boldsymbol{w})$, we have

$$\hat{y}' - \hat{y} = \boldsymbol{w}^T \eta^* = \boldsymbol{w}^T \epsilon.\text{sign}(\boldsymbol{w}) = \epsilon \|\boldsymbol{w}\|_1$$

- If $\boldsymbol{w}$ has n dimensions and the average magnitude of an element of the weight vector is m, then the **activation will grow by** ϵmn**.** Thereby, as the dimension of the input increases, the value of $\hat{y}' - \hat{y}$ will grow.
- This explanation shows that a simple **linear model can have adversarial examples** if its input has **sufficient dimensionality**.

Linear Perturbation for Non-linear Models

The **linear view** of adversarial examples suggests a **fast** way of generating them.

- It is hypothesized that deep nets are **too linear** to resist adversarial perturbations (ReLU activation function).
- More nonlinear models such as **sigmoid or tanh** networks are carefully tuned to spend most of their time in the **non-saturating, more linear regime**.

Hence, we suppose Deep nets have **linear behavior** in the **vicinity** of each data point.

Linear Perturbation for Non-linear Models

The **linear view** of adversarial examples suggests a **fast** way of generating them.

- It is hypothesized that deep nets are **too linear** to resist adversarial perturbations (ReLU activation function).
- More nonlinear models such as **sigmoid or tanh** networks are carefully tuned to spend most of their time in the **non-saturating, more linear regime**.

Hence, we suppose Deep nets have **linear behavior** in the **vicinity** of each data point.

Recall: Taylor Series (Expansion)

Suppose n is a positive integer and $f : \mathbb{R} \to \mathbb{R}$ is n times differentiable at a point x_0. Then

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + R_n(x, x_0)$$

$$= f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \dots$$

where the remainder R_n satisfies

$$R_n(x, x_0) = o(|x - x_0|^n) \text{ as } x \to x_0.$$

Definition: A sequence of numbers X_n is said to be $o(r_n)$ if $\frac{X_n}{r_n} \to 0$ as $n \to \infty$.

Linear Perturbation for Non-linear Models

The **linear view** of adversarial examples suggests a **fast** way of generating them.

- It is hypothesized that deep nets are **too linear** to resist adversarial perturbations (ReLU activation function).
- More nonlinear models such as **sigmoid or tanh** networks are carefully tuned to spend most of their time in the **non-saturating, more linear regime**.

Hence, we suppose Deep nets have **linear behavior** in the **vicinity** of each data point.

Consequently, we can linearly approximate classifier $f : \mathbb{R}^d \to \mathbb{R}$ around data point x_0 by **Taylor expansion**. We have:

$$f(x) = f(x_0) + (x - x_0)^T \nabla_x f(x)$$

Linear Perturbation for Non-linear Models

The **linear view** of adversarial examples suggests a **fast** way of generating them.

- It is hypothesized that deep nets are **too linear** to resist adversarial perturbations (ReLU activation function).
- More nonlinear models such as **sigmoid or tanh** networks are carefully tuned to spend most of their time in the **non-saturating, more linear regime**.

Hence, we suppose Deep nets have **linear behavior** in the **vicinity** of each data point.

Consequently, we can linearly approximate classifier $f : \mathbb{R}^d \to \mathbb{R}$ around data point x_0 by **Taylor expansion**. We have:

$$f(x) = f(x_0) + (x - x_0)^T \nabla_x f(x)$$

Let $x' = x_0 + \eta$, we get

$$f(x') = f(x + \eta) = f(x_0) + (\eta)^T \nabla_x f(x) \Rightarrow f(x') - f(x_0) = (\eta)^T \nabla_x f(x)$$

To maximize difference between $f(x)$ and $f(x')$, we should maximize $< \eta^T, \nabla_x f(x) >$. Given $\|\eta\|_\infty \leq \epsilon$, we have

$$\eta = \epsilon.sign(\nabla_x f(x))$$

Linear Perturbation for Non-linear Models

The **linear view** of adversarial examples suggests a **fast** way of generating them.

- It is hypothesized that deep nets are **too linear** to resist adversarial perturbations (ReLU activation function).
- More nonlinear models such as **sigmoid or tanh** networks are carefully tuned to spend most of their time in the **non-saturating, more linear regime**.

Hence, we suppose Deep nets have **linear behavior** in the **vicinity** of each data point.

Consequently, we can linearly approximate classifier $f : \mathbb{R}^d \to \mathbb{R}$ around data point x_0 by **Taylor expansion**. We have:

$$f(x) = f(x_0) + (x - x_0)^T \nabla_x f(x)$$

Let $x' = x_0 + \eta$, we get

$$f(x') = f(x + \eta) = f(x_0) + (\eta)^T \nabla_x f(x) \Rightarrow f(x') - f(x_0) = (\eta)^T \nabla_x f(x)$$

To maximize difference between $f(x)$ and $f(x')$, we should maximize $< \eta^T, \nabla_x f(x) >$. Given $||\eta||_\infty \leq \epsilon$, we have

$$\eta = \epsilon.sign(\nabla_x f(x))$$

We can replace classifier output with cost function J

$$\eta = \epsilon.sign(\nabla_x J(\theta, x, y))$$

Fast Gradient Sign Method (FGSM)

Let $\boldsymbol{\theta}$ be the parameters of a model, $\boldsymbol{x}$ the input to the model, y the label associated with $\boldsymbol{x}$ and $J(\boldsymbol{\theta}, \boldsymbol{x}, y)$ be the cost used to train the neural network.

We can linearize the cost function around the current value of $\boldsymbol{\theta}$, obtaining an optimal max-norm constrained perturbation of

$$\boldsymbol{\eta} = \epsilon \, sign(\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, y))$$

We refer to this as the "**fast gradient sign method**" of generating adversarial examples.

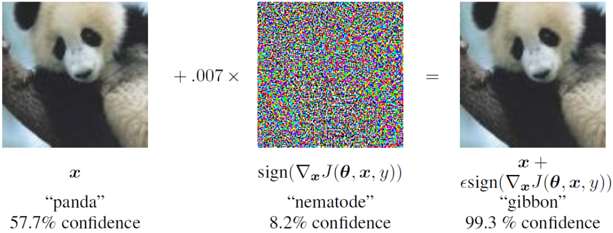


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

Targeted and Untargeted FGSM

Untargetted attack

- The adversary wants to change the predication of the classifier to a wrong class.
  - Untargeted FGSM attack on clean data $(\boldsymbol{x}, y)$

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon.sign(\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, y))$$

Targeted attack

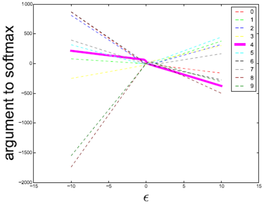
- The adversary wants to change the predication of the classifier to a given target class.
  - Targeted FGSM attack on clean data $(\boldsymbol{x}, y)$ for given target class t

$$\boldsymbol{x}_{adv} = \boldsymbol{x} - \epsilon.sign(\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, t))$$

Why Do Adversarial Examples Generalize?

By tracing out different values of ϵ we see that adversarial examples occur in **contiguous regions** of the 1-D subspace defined by the fast gradient sign method, **not in fine pockets**.

- This explains why adversarial examples are abundant and why an example misclassified by one classifier has a fairly high prior probability of being misclassified by another classifier.



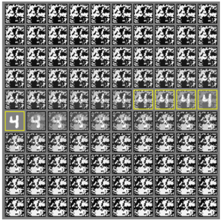
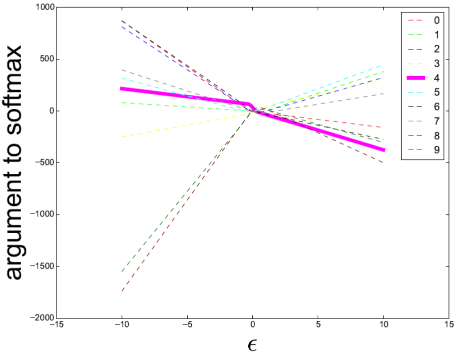


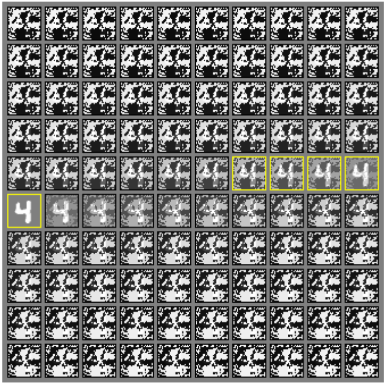
Figure 4: By tracing out different values of ϵ, we can see that adversarial examples occur reliably for almost any sufficiently large value of ϵ provided that we move in the correct direction. Correct classifications occur only on a thin manifold where $\boldsymbol{x}$ occurs in the data. Most of $\mathbb{R}^n$ consists of adversarial examples and *rubbish class examples* (see the appendix). This plot was made from a naively trained maxout network. Left) A plot showing the argument to the softmax layer for each of the 10 MNIST classes as we vary ϵ on a single input example. The correct class is 4. We see that the unnormalized log probabilities for each class are conspicuously piecewise linear with ϵ and that the wrong classifications are stable across a wide region of ϵ values. Moreover, the predictions become

Why Do Adversarial Examples Generalize?

By tracing out different values of ϵ we see that adversarial examples occur in **contiguous regions** of the 1-D subspace defined by the fast gradient sign method, **not in fine pockets**.

- This explains why adversarial examples are abundant and why an example misclassified by one classifier has a fairly high prior probability of being misclassified by another classifier.





Observations

- Adversarial examples can be explained as a property of high-dimensional dot products. They are a result of models being **too linear, rather than too nonlinear**.
- The **direction of perturbation**, rather than the specific point in space, matters most.
- Because it is the direction that matters most, adversarial perturbations **generalize** across different clean examples.

L_P Norm

L_P Norm

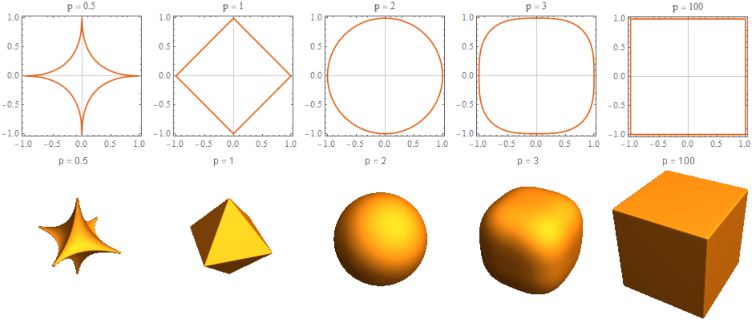
Let $p \geq 1$ be a real number, the P-norm (also called L_P-norm) of vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$ is

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

L_P Norm

Let $p \geq 1$ be a real number, the P-norm (also called L_P-norm) of vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$ is

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$



The boundary of $\|\boldsymbol{x}\|_P = 1$
(Source).

Is L_P with $P < 1$ really a norm? The answer is no, because it violates the triangle inequality (See Convex Optimization by Stephen

L_P Norm

The L_P distance is written $\|x - x'\|_P$, where $x, x' \in \mathbb{R}^n$ and the P-norm $\|.\|_P$ is defined as

$$\|x - x'\|_p = \left(\sum_{i=1}^{n} |x_i - x'_i|^p \right)^{1/p}$$

L_P Norm

The L_P distance is written $\|x - x'\|_P$, where $x, x' \in \mathbb{R}^n$ and the P-norm $\|.\|_P$ is defined as

$$\|x - x'\|_p = \left(\sum_{i=1}^{n} |x_i - x'_i|^p \right)^{1/p}$$

- $\boldsymbol{L_0}$ **distance** measures the number of coordinates i such that $x_i \neq x'_i$.
  - Thus, the L_0 distance corresponds to the number of pixels that have been altered in an image.

L_P Norm

The L_P distance is written $\|x - x'\|_P$, where $x, x' \in \mathbb{R}^n$ and the P-norm $\|.\|_P$ is defined as

$$\|x - x'\|_p = \left(\sum_{i=1}^{n} |x_i - x'_i|^p \right)^{1/p}$$

- $\boldsymbol{L_0}$ **distance** measures the number of coordinates i such that $x_i \neq x'_i$.
  - Thus, the L_0 distance corresponds to the number of pixels that have been altered in an image.
- $\boldsymbol{L_2}$ **distance** measures the standard Euclidean (root mean-square) distance between x and x'.
  - The L_2 distance can remain small when there are many small changes to many pixels.

L_P Norm

The L_P distance is written $\|x - x'\|_P$, where $x, x' \in \mathbb{R}^n$ and the P-norm $\|.\|_P$ is defined as

$$\|x - x'\|_p = \left(\sum_{i=1}^{n} |x_i - x'_i|^p \right)^{1/p}$$

- $\boldsymbol{L_0}$ **distance** measures the number of coordinates i such that $x_i \neq x'_i$.
  - Thus, the L_0 distance corresponds to the number of pixels that have been altered in an image.
- $\boldsymbol{L_2}$ **distance** measures the standard Euclidean (root mean-square) distance between x and x'.
  - The L_2 distance can remain small when there are many small changes to many pixels.
- $\boldsymbol{L_\infty}$ **distance** measures the maximum change to any of the coordinates

$$\|x - x'\|_\infty = max(|x_1 - x'_1|, ..., |x_n - x'_n|).$$

  - For images, we can imagine there is a maximum budget, and each pixel is allowed to be changed by up to this limit, with no limit on the number of pixels that are modified.

FGSM Expansion on different L_P Norms

Let $\boldsymbol{g} := \nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)$. We choose $\boldsymbol{\eta}$ to maximize the first-order increase in loss:

$$\arg\max_{\boldsymbol{\eta}} \ \langle \boldsymbol{g}, \boldsymbol{\eta} \rangle \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon_p.$$

By Hölder's inequality, with dual norm q such that $\frac{1}{p} + \frac{1}{q} = 1$,

$$\max_{\|\boldsymbol{\eta}\|_p \leq \varepsilon_p} \langle \boldsymbol{g}, \boldsymbol{\eta} \rangle = \varepsilon_p \, \|\boldsymbol{g}\|_q,$$

and one optimal perturbation is

$$\boxed{\boldsymbol{\eta} = \varepsilon_p \, \frac{|\boldsymbol{g}|^{\,q-1} \odot \operatorname{sign}(\boldsymbol{g})}{\|\boldsymbol{g}\|_q^{q-1}}}$$

(where $\odot$ is elementwise multiplication and $|\boldsymbol{g}|^{\,q-1}$ is elementwise power).

Closed forms for $p \in \{1, 2, \infty\}$ & budget matching

Let $\boldsymbol{g} = \nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)$.

$$L_\infty \text{ (FGSM):} \qquad \boldsymbol{\eta}^\star = \varepsilon_\infty \ \mathrm{sign}(\boldsymbol{g}).$$

$$\Delta J = \varepsilon_\infty \ \|\boldsymbol{g}\|_1 \, .$$

$$L_2 \text{ (FGM-}L_2\text{):} \qquad \boldsymbol{\eta}^\star = \varepsilon_2 \, \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}.$$

$$\Delta J = \varepsilon_2 \ \|\boldsymbol{g}\|_2 \, .$$

$$L_1 \text{ (sparse FGM):} \qquad i^\star \in \arg\max_i |g_i|.$$

$$\boldsymbol{\eta}^\star = \varepsilon_1 \ \mathrm{sign}(g_{i^\star}) \, \boldsymbol{e}_{i^\star} \, .$$

$$\Delta J = \varepsilon_1 \ \|\boldsymbol{g}\|_\infty \, .$$

Matching budgets (dimension d): $\qquad \varepsilon_2 = \varepsilon_\infty \sqrt{d}, \qquad \varepsilon_1 = \varepsilon_\infty d.$

$\boldsymbol{\eta}_{L_2} = \varepsilon_\infty \sqrt{d} \, \dfrac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}, \qquad \boldsymbol{\eta}_{L_1}$: allocate total budget $\varepsilon_\infty d$ to the largest $|g_i|$.

References

- C. Szegedy, W. Zaremba, I. Sutskever, et al., "Intriguing properties of neural networks," in 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014.
- I. Goodfellow, J. Shlens, C. Szegedy, "Explaining and Harnessing Adversarial Examples" in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015.
- D. Baehrens, T. Schroeter, S. Harmeling, et al., "How to explain individual classification decisions." The Journal of Machine Learning Research 11 (2010): 1803-1831.